Universidade Nova de Lisboa – ISEGI
Lisboa, Portugal

Artificial Intelligence in Geospatial Analysis: applications of Self-Organizing Maps in the context of Geographic Information Science

A thesis submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy in Information Systems
by

Roberto André Pereira Henriques

Supervisors: Fernando Bação, Ph.D.
Victor Lobo, Ph.D.

Lisbon, March 2010

Abstract

The size and dimensionality of available geospatial repositories increases every day, placing additional pressure on existing analysis tools, as they are expected to extract more knowledge from these databases. Most of these tools were created in a data poor environment and thus rarely address concerns of efficiency, dimensionality and automatic exploration. In addition, traditional statistical techniques present several assumptions that are not realistic in the geospatial data domain. An example of this is the statistical independence between observations required by most classical statistics methods, which conflicts with the well-known spatial dependence that exists in geospatial data.

Artificial intelligence and data mining methods constitute an alternative to explore and extract knowledge from geospatial data, which is less assumption dependent. In this thesis, we study the possible adaptation of existing general-purpose data mining tools to geospatial data analysis. The characteristics of geospatial datasets seems to be similar in many ways with other aspatial datasets for which several data mining tools have been used with success in the detection of patterns and relations. It seems, however that GIS-minded analysis and objectives require more than the results provided by these general tools and adaptations to meet the geographical information scientist's requirements are needed. Thus, we propose several geospatial applications based on a well-known data mining method, the self-organizing map (SOM), and analyse the adaptations required in each application to fulfil those objectives and needs. Three main fields of GIScience are covered in this thesis: cartographic representation; spatial clustering and knowledge discovery; and location optimization.

In the cartographic representation field, we propose the use of SOM to build cartograms. We use the standard SOM method for this purpose, although the cartogram construction requires new pre-processing and post-processing phases. We present several cartograms, such as the USA states and counties population cartograms, the Portuguese population cartogram and the world countries population cartogram.

The second field covered is spatial clustering and knowledge discovery from geospatial databases. Two SOM based methods were applied to achieve this goal. GeoSOM,

which is a geospatial-aware variant of SOM, was extended and implemented in the GeoSOM Suite tool, providing a useful and efficient framework for knowledge extraction and spatial clustering tasks. Using a different approach, a hierarchical SOM is proposed to explore and cluster geospatial datasets. Tests are performed using Lisbon's Metropolitan Area 2001 census data.

Finally, concerning a location/allocation problem, a variant of SOM is proposed to manage a network of surveillance agents. This method is an online trajectory predictor, defining at each instant the path each agent should take to maximize the coverage of relevant events. The testing of this tool was performed based on an unmanned aerial vehicles network for maritime surveillance scenario, allowing the tracking of ships in a predefined region.

Resumo

O tamanho e a dimensionalidade dos repositórios de dados geoespaciais aumenta, a cada dia, exigindo das ferramentas de análise existentes um maior esforço para a extracção de conhecimento. A maioria destas ferramentas foram desenvolvidas num ambiente pobre em dados, razão pela qual aspectos como a eficiência, a elevada dimensionalidade e a exploração automática dos dados não são normalmente abordados nestas técnicas. A juntar a este facto, as técnicas estatísticas tradicionais apresentam diversas premissas que geralmente não são reais no contexto geoespacial. Um exemplo é a independência estatística entre os dados que é ponto de partida para a maioria dos métodos estatísticos clássicos, e que no caso dos dados geoespaciais não se verifica devido ao fenómeno de autocorrelação espacial.

Os métodos de inteligência artificial e data mining, que são menos dependentes de modelos, constituem assim uma alternativa na exploração e extracção de conhecimento de dados geoespaciais. Nesta tese, estudamos a possível adaptação de métodos gerais de data mining para analisar dados geoespaciais. As características destes dados parecem semelhantes em diversos aspectos aos dados não espaciais, para os quais diversas ferramentas de data mining têm sido usadas com sucesso na detecção de padrões e relações. Parece, contudo, que as análises típicas e os objectivos na maioria dos problemas da Ciência da Informação Geográfica, exigem uma adaptação dos métodos gerais que possam corresponder às expectativas dos cientistas geoespaciais. Assim, propomos nesta tese, diversas aplicações geoespaciais baseadas num método famoso em data mining, os mapas auto-organizáveis de Kohonen (SOM), e estudamos para cada caso as adaptações necessárias para garantir o cumprimento desses objectivos. Três áreas da Ciência da Informação Geográfica são abordadas nesta tese: a representação cartográfica; a descoberta de conhecimento e clustering espaciais; e a optimização de posicionamento.

No campo da representação cartográfica, propomos o uso dos SOM para a construção de cartogramas. O algoritmo standard do SOM é usado para este caso, embora para a construção de cartogramas seja necessário a introdução de operações específicas de pré e pós processamento. Diversos cartogramas, construídos a partir deste novo método, são apresentados de onde destacamos o cartograma de população dos

Estados Unidos da América (baseado nos estados e nos condados), o cartograma da população portuguesa ou o cartograma da população mundial.

A segunda área da Ciência da Informação Geográfica abordada neste tese é a descoberta de conhecimento e clustering espaciais. Dois métodos são apresentados neste campo. O GeoSOM, que é uma adaptação do método SOM para lidar com dados geoespaciais, foi melhorado e implementado numa ferramenta (GeoSOM Suite) que permite de forma fácil e eficiente a extracção de conhecimento de dados geoespaciais. Usando uma abordagem diferente, é proposto um SOM hierárquico para exploração dos dados e criação de clustering temático. Como exemplo, diferentes análises foram feitas usando os dados censitários para a Área Metropolitana de Lisboa, referentes a 2001.

Finalmente, na área da optimização da localização ou posicionamento, é proposta uma variante do SOM para a gestão de redes móveis de agentes de vigilância. Este método permite em tempo real, a definição do percurso que cada agente deve tomar, permitindo uma maximização da área coberta pela rede. Como exemplo de aplicação é usada uma rede de veículos aéreos não tripulados para vigilância marítima, que permitem a detecção e acompanhamento de navios numa determinada região de estudo.

List of publications

List of published publications resulting from this thesis:

Henriques, R., F. Bacao and V. Lobo (2009). "Carto-SOM: cartogram creation using self-organizing maps." International Journal of Geographical Information Science **23**(4): 483 - 511.

Henriques, R., F. Bacao and V. Lobo (2009). Spatial Clustering with SOM and GeoSOM Case study of Lisbon's Metropolitan Area. The Second International Conference on Advanced Geographic Information Systems, Applications, and Services International. GEOProcessing 2010

Henriques, R., F. Bacao and V. Lobo (2009). GeoSOM Suite: A Tool for Spatial Clustering. Computational Science and Its Applications - ICCSA 2009. 5592: 453-466.

Henriques, R., F. Bacao and V. Lobo (2009). UAV Path Planning Based on Event Density Detection. International Conference on Advanced Geographic Information Systems & Web Services, 2009. GEOWS '09.

Henriques, R., F. Bação and V. Lobo (2009). Cartograms, Self-Organizing Maps, and Magnification Control. Advances in Self-Organizing Maps**:** 89-97.

Henriques, R. and R. M. Rocha (2009). Sensor Network Deployment based on Data Variability. Proceedings of the 7th Conference on Telecommunications CONFTELE 2009, Santa Maria da Feira, Instituto de Telecomunicações.

Henriques, R., F. Bação and V. Lobo (2008). Planeamento de percursos em UAVs baseado em densidades de eventos. Jornadas do Mar 2008. O OCEANO - Riqueza da Humanidade, Escola Naval, Alfeite, Marinha Portuguesa.

Henriques, R., F. Bação and V. Lobo (2008). Self-Organizing Networks of Unmanned Aerial Vehicles. XV Jornadas de Classificação e Análise de Dados - JOCLAD, Setúbal, Escola Superior De Ciências Empresariais de Setúbal.

Publications to be published:

Henriques, R., F. Bacao and V. Lobo "Exploratory geospatial data analysis using the GeoSOM suite." In submission.

Henriques, R., F. Bacao and V. Lobo “Hierarchical SOM and geospatial clustering" In submission.

Acknowledgements

Reaching this stage, I feel that the work presented in this thesis was not possible without the contribution of many persons and institutions though, of course, the final responsibility of this work remains mine. To them, I would like to express my great gratitude:

First, my supervisors, Professor Doutor Fernando Bação and Professor Doutor Victor Lobo, for their guidance, support, patience and friendship. They made me feel part of the team, and working with them was very compensating. May our collaboration and Chinese food dinners continue for many years.

The friendship and support from all my colleagues from LabNT. My special thanks to Paula Curvelo for all the discussions we had about this thesis.

To all the colleagues and Professors from ISEGI-UNL and IST who were somehow involved in this journey.

In addition, the reviewers of the different publications made from this work, which suggestions and comments allowed an improvement on the quality of this work.

Finally, to my family and friends for being my support in this thesis. To my wife Nucha, I must thank all the patience, love and help she gave me in this time. To my parents I must thank all the opportunities and belief they give me allowing the conclusion of this stage. To my brothers, parents in law and friends Paulo, Pedro and Carlos for their support and encouragement. Also, Thomas and Jutta, for their hospitality in Munster at the IFGI Spring School.

This work was financed by the Portuguese Foundation for Science and Technology (FCT) by the FCT fellowship SFRH/BD/30360/2006.

Para os meus pais,

a Nucha,

e o feijoca.

Acronyms

Acronyms are ordered by appearance in the text.

GIS	Geographic Information Systems
GISc	Geographic Information Science
SOM	Self-Organizing Map
U-Mat	Unified Matrix
UAV	Unmanned Aerial Vehicle
HSOM	Hierarchical Self-Organizing Map
SOAP	Simple Object Access Protocol
GPS	Global Positioning System
API	Application Programming Interface
GPX	GPS Exchange Format
KML	Keyhole Markup Language
VGI	Volunteered Geographic Information
UCGIS	University Consortium for Geographic Information Science
NCGIA	National Center for Geographic Information and Analysis
AI	Artificial Intelligence
CI	Computational Intelligence
HPC	High Performance Computing
TFL	Tobler's First Law of Geography
MAUP	Modifiable Areal Unit Problem
NUTS3	Nomenclature of Territorial Units for Statistics (level 3)
ESDA	Exploratory Spatial Data Analysis
EDA	Exploratory Data Analysis
DM	Data Mining
GDM	Geographic Data Mining
BMU	Best Matching Unit
PCA	Principal Components Analysis
MDS	Multidimensional Scaling
PCP	Parallel Coordinate Plots
ESOM	Emergent Self-Organizing Maps
GA	Genetic Algorithms
LVQ	Learning Vector Quantization
AAG	Association of American Geographers
CCA	Chromated Copper Arsenate
SAR	Search And Rescue Operations
SOMSD	Self-Organizing Map for Spatial Data

STFM	Spatial Temporal Feature Map
KDD	Knowledge Discovery and Data Mining
LISA	Local Indicators of Spatial Association
ED	Enumeration Districts
LMA	Lisbon Metropolitan Area
GUI	Graphical User Interface
MLP	Multilayer Perceptron
GHSOM	Growing Hierarchical Self-Organizing Map
TMR	Tension and Mapping Ratio extension
TSTFM	Tree Structured Topological Feature Map

Short index

Index

List of figures

List of tables

1. Introduction

"Such systems [GIS] *are basically concerned with describing the Earth's surface rather than analysing it. Or if you prefer, traditional 19th century geography reinvented and clothed in 20th century digital technology"* (Openshaw, Charlton *et al.* 1987).

"*Techniques are wanted that are able to hunt out what might be considered to be localised patterns or 'database anomalies' in geographically referenced data but without being told either 'where' to look, or 'what' to look for, or 'when' to look*" (Openshaw 1994).

In order to answer these challenges, we believe that new tools from the artificial intelligence field are needed to deal with today's GIScience problems and objectives. Indeed, traditional methods are no longer effective for geospatial analysis, mainly due to changes in the data itself (as we will further explore in this thesis), and in the type of analysis required.

This thesis reports the findings of a three-year study that started in January 2007 and examines the possible connections between Geographic Information Science (GISc) and artificial neural networks, more specifically Self-Organizing Maps (SOM).

1.1. Context

We live in a digital world. Nowadays data acquisition methods continuously record all sort of events occurring in the physical world. Improvements on both hardware and software technologies allow us to collect huge amounts of data, producing, every day, more complete, accurate and detailed pictures of human activity and interaction with the environment. All this torrent of data is being stored in ever increasing data warehouses.

These developments have increased the relevance of information in modern society, which in turn has led to even higher rate of information production developed in this area. It is a generally shared idea that information is an important resource in any organization. Possibly, the answer to many human/world problems may in fact depend on our ability to tap into this digital picture that we have of the world. However, organizations often collect raw data and produce sparse information but fail to create knowledge. A step further is needed, where this data/information is analysed, explored and converted into knowledge and ultimately used to solve problems and create value. Data Mining can be an important tool in bridging the gap between data and knowledge, through automated analysis which enables the extraction of knowledge from these databases (Hand, Smyth *et al.* 2001).

An important evolution has occurred in most databases, related with the global demand for a geospatial context, which *forced* the inclusion of space in these repositories. This demand has been underlined by major technological advances leading to a new paradigm in the creation/use of contents. Major changes started around 2000, and were caused by factors such as:

- the dot-com boom and the increase of broadband use;
- the browsers' improvement in supporting new technologies such as SOAP and XML;
- the fall in prices of data storage;
- the changes in the way software developers and end-users use the Web (known as WEB 2.0 (O'Reilly 2005));
- the creation of Web services and simplified APIs;
- maturity of positioning technology such as GPS, remote sensing, inertial systems, and GSM radio triangulation;

- widespread use of low-cost position-aware devices such as GPS-navigation devices and cell phones.

The combination of these factors led to an increase in the number of people using the Web to create, assemble and disseminate geographic information. These new users have, in general, new needs and objectives in dealing with geospatial data and technologies. To deal with this new perspective, Turner (2006) proposes a new subfield in Geography, which he called *Neogeography* and defines it as the "*set of techniques and tools that fall outside the realm of traditional GIS* [Geographic Information Systems] *(...). Where historically a professional cartographer might use ArcGIS, talk of (...) projections, and resolve land area disputes, a neogeographer uses a mapping API*[1] *like Google Maps, talks about GPX*[2] *versus KML*[3] *and geotags his photos to make a map of his summer vacation"*.

Goodchild called this new paradigm "*volunteered geographic information*" (VGI) and defined it as "*... a special case of the more general Web phenomenon of user generated content*" (Goodchild 2007). Goodchild thinks that, although some quality issues must be thought over carefully, these new sources of data can be useful to several applications such as military and commercial intelligence.

Some examples of the most common geospatial databases come from Earth Observation Satellites, census surveys and climate/environmental monitoring systems. Examples where the geospatial component has surfaced recently can be found in

---

[1] API (application programming interface) is an interface that defines the ways by which an application program may request services from libraries and/or operating systems Wikipedia. (2009). "Application programming interface." Retrieved 19-08-2009, from http://en.wikipedia.org/wiki/Application_programming_interface.

[2] GPX (GPS Exchange Format) is a XML data format for the interchange of GPS data (waypoints, routes, and tracks) between applications and Web services on the Internet Foster, D. (2009). "GPX: the GOS exchange format." Retrieved 19-08-2009, from http://www.topografix.com/gpx.asp.

[3] KML (Keyhole Markup Language) is a file format used to display geographic data in an Earth browser such as Google Earth, Google Maps, and Google Maps for mobile devices Google. (2009). "KML Documentation Introduction." Retrieved 19-08-2008, from http://code.google.com/apis/kml/documentation/., that was adopted by the Open Geospatial Consortium.

customer/supply databases and product transaction repositories. Finally, some examples of Neogeography or VGI paradigm include geo-referenced data collected by position aware devices such as GPS receivers or cell phones or even wireless internet clients and cameras. This data is then uploaded by its creators to web based data repositories such as Google Maps (Google 2005), OpenStreetMap (OpenStreetMap 2004) or Wikimapia (Wikimapia 2006).

1.1.1. GIScience & Geocomputation

Geographic Information Science (GISc or GIScience) is the scientific discipline that deals with the geospatial data. This discipline emerged 30 years after the creation of the first "*modern"* Geographic Information System (GIS) by Tomlinson in 1960, called Canada Geographic Information System (Tomlinson 1984; Tomlinson 1998). The term GISc was introduced by Goodchild (1992) and it is concerned with "*the development and use of theories, methods, technology, and data for understanding geographic processes, relationships, and patterns. The transformation of geographic data into useful information is central to geographic information science*" (UCGIS 2001). Mark (2003) on the other hand, compiled a GISc definition by including the word *geographic* in an Information Science definition due to Shuman (1992). In his proposal, "(*Geographic) Information science is very difficult to define. (...) the field of (geographic) information science, however, may be defined as one that investigates the properties and behaviour of (geographic) information, how it is transferred from one mind to another, and optimal means for making that transfer, in both natural and artificial systems. Finally, (geographic) information science is concerned with the effects of (geographic) information on people and on machines*."

A generally accepted definition of GIS (Geographic Information Systems) is given by the National Center for Geographic Information and Analysis (NCGIA) which proposes *"GIS as a system of hardware*, *software and procedures to facilitate the management*, *manipulation*, *analysis*, *modelling*, *representation and display of georeferenced data to solve complex problems regarding planning and management of resources"* (NCGIA 1990).

Most of the analysis performed by GIS, even today, use techniques from traditional statistics (Openshaw and Openshaw 1997). These techniques are not well suited to deal

with the amount, diversity and characteristics of modern geospatial data. As a reaction to the limits imposed by GIS software, Stan Openshaw proposes the "*artificial intelligence paradigm as a core geographic skill*" (Openshaw and Openshaw 1997).

The problems of using traditional statistical techniques in geospatial data derive from the following points (Atkinson and Martin 2000):

1) assumption of statistical independency of the data;
2) generalization of geography using global measures (*e.g.* average);
3) use of stationary models and;
4) use of model-based statistics for inference instead of letting data *speak for themselves*.

As an answer to these limitations, a new field called GeoComputation emerged. GeoComputation "*is concerned with new computational techniques, algorithms, and paradigms that are dependent upon and can take advantage of high performance computing* (Openshaw 2000). In fact, Openshaw considers that GeoComputation is founded on four technologies: the GIS for gathering data; artificial intelligence (AI) and computational intelligence (CI) providing the tools; computing power provided by high performance computing (HPC); and (geographic) science, which provides the philosophy or "*raison d'etre*" (Openshaw 2000). To Openshaw, combining these factors makes GeoComputation the basis for a new paradigm for doing Geography. In fact, the letters *G* and *C* in GeoComputation are purposely capitalized to distinguish this new field of spatial analysis.

A different view is proposed by Couclelis who believes that "*we have been doing geocomputation for years without realizing it*", under the quantitative geography umbrella (Couclelis 1998). Her vision of GeoComputation is the "*eclectic application of computational methods and techniques to portray spatial properties, to explain geographical phenomena and to solve geographical problems*" (Couclelis 1998). Atkinson and Martin (2000) are more cautious defining GeoComputation as a new approach to geo-information analysis or just as GIS with more powerful computers. To them this is a field that "*will be defined, as time passes, by what geocomputation researchers do*" (Atkinson and Martin 2000).

Three main aspects make GeoComputation unique (Openshaw 2000). First, it is applied to geospatial data, assuming its distinctiveness. Many methods in quantitative geography were brought from other fields assuming no particularity exists in geospatial data. Secondly, GeoComputation uses an unparalleled computing power that provides new solutions and new ways of solving problems. Finally, GeoComputation requires a change in the way of thinking, because it is *data-driven* in the sense that knowledge is deduced from data, instead of predefined or deduced by reasoning.

Longley *et.al.* (2005) assume GeoComputation as a synonym of GIScience, since they both "*suggest a scientific approach to the fundamental issues raised by the use of GIS and related technologies".* However, they adopted the idea that GeoComputation is more focused on the use of high-performance computers and artificial intelligence.

While agreeing in general with most of these views, in this thesis we assume that GeoComputation is a specialized branch of GIScience, which differs from the more general concepts of quantitative geography and GIS, in the sense that it is more focused on taking advantage of artificial intelligence techniques and computing power to develop new methods to solve GIScience problems.

1.2. The Problem

The amount of data in current geospatial repositories along with their high-dimensional nature requires a sophisticated set of analysis capabilities in order to extract new and unexpected patterns, trends, and relationships embedded in that data. General-purpose methods of data mining and knowledge discovery may not be suitable to geospatial data. This lack of suitability results from the fact that the spatial dimension cannot be seen just as two or three extra variables (such as *x*, *y* and *z* coordinates). It has been said that geospatial data is particular and calls for special methods and analysis (Anselin 1989; Goodchild 1992; Openshaw 1999). These particularities fall into four major categories concerning aspects related to:

- the particular characteristics of their attributes;
- the distribution and dimensionality of data;
- the data models and representation used;
- the typical analysis performed.

So, what is special in the attributes of geospatial data? First, the observations, the uncertainty and the error distribution are spatially dependent. This concept of spatial dependency was postulated in Tobler's first law (TFL) which states that "*everything is related to everything else but near things are more related than distant things*" (Tobler 1970). Directly related to the spatial dependency is the concept of spatial autocorrelation (Goodchild 1986). Spatial autocorrelation is the computational expression of spatial dependency. Another characteristic of geospatial data is spatial heterogeneity (Anselin 1988). Spatial heterogeneity is the property that makes each place on Earth unique, making design decisions successfully adopted in one region not always general and applicable in other regions (Goodchild 2008). These characteristics are important obstacles to standard premises used in traditional statistics.

As for distribution and dimensionality, geospatial data has, in general, a non-normal distribution and lies in a high-dimensional data space made up of two or three spatial dimensions and a potentially large number of aspatial dimensions. Also, this high-dimensional structure of data usually comprises redundancy and high correlation of some variables.

Data models and representation are also quite particular in geospatial data. The geographical space is continuous and infinite, but GIS requires the use of discrete representations. The delimitation of crisp boundaries to represent spatial continuous phenomena affects the accuracy and precision of data and consequently the analysis. One of these problems is known as the modifiable areal unit problem (MAUP) (Openshaw 1984). The MAUP consists on the fact that the variation in the spatial units used for aggregation will cause variation in statistical results. The outline of the area over which the description is obtained will influence critically the perception of the phenomena and if this aggregation is obtained at different scales, that perception will be even more biased. As an example, we can consider the criminality rate of Portugal. Assuming different aggregation levels such as Enumeration District, Civil Parish (*Freguesia* in Portuguese), Municipality (*Concelho* in Portuguese) or NUTS3, different criminality rates are calculated. Additionally, different criminal rates will be obtained by using different aggregation schemes at the same scale. Related to the MAUP problem is the ecological fallacy problem (Robinson 1950). The ecological fallacy arises when statistics for groups are incorrectly assumed to apply at the individual level. For instance, if a County has a high percentage of unemployment and high criminality, the ecological

fallacy exists if one assumes that unemployed people are responsible for the crimes. Another particularity of geospatial data is related to its representation, which is usually made through compiled categorical layers, typically associated between them by spatial relationships.

Finally, the typical analysis performed with geospatial data has to consider the geospatial analyst's objectives and needs. For a geospatial analyst, space is the most important element, since the analysis is always spatially contextualized, and insights should primarily come from the specific spatial arrangements found. When exploring spatial data, the GIS scientist is searching for patterns, trends, and relationships spatially relevant. In fact, the most frequent type of analysis in geospatial data is exploratory. Exploratory spatial data analysis (ESDA) is a subset of exploratory data analysis (EDA) focused on the particular characteristics of geographic data (Anselin 1998). This set of techniques is based on user/data interaction allowing the detection of spatial patterns to build hypotheses on the dataset and evaluate its validity (Haining, Wise *et al.* 1998).

Data Mining (DM) is a step in the knowledge discovery process that automatically detects patterns in data (Fayyad, Piatetsky-Shapiro *et al.* 1996). Thus, Geographic Data Mining (GDM) is a special type of data mining that seeks to apply standard data mining tools modified to take into account the special features of geospatial data and particular objectives and needs of GIS (Openshaw 1999; Harvey and Han 2001). Openshaw's perspective is that "*new types of data mining tools that can handle the special nature of spatial information* [are needed to] *capture the spirit and essence of geographicalness that a GIS-minded data miner would expect to have available"* (Openshaw 1999).

A slightly different perspective is held in (Bacao, Lobo *et al.* 2005) about how special spatial data is, and consequently the necessity of new methods. In their view, most of the geospatial characteristics are not unique and may exist in aspatial secondary datasets. Secondary datasets, in opposition to primary datasets, are the sets of data not purposely collected for the analysis. Thus, geospatial data "*can be considered a specific set of secondary data ... obtained with no specific analysis objectives in mind ... [with some] characteristics similar [to] those observed in typical secondary data ... such as the dependency issues ... selection bias, fuzziness, redundancy and even nonstationarity"* (Bacao, Lobo *et al.* 2005). However, although machine learning methods developed for secondary data are able to handle spatial data, adaptations to these

methods need to be made to deal with geographical perspective (Bacao, Lobo *et al.* 2005).

Bacao *et.al.* (2005) propose the adaptation of standard data mining tools whenever possible, in opposition to the original GeoComputation proposal of developing new and specific methods to deal with geospatial data. Anyway, both visions agree that the use of standard *of-the-box* data mining tools in geospatial analysis will produce poor results when compared with spatial aware methods.

The characteristics of geospatial datasets seem to be similar in many ways with other aspatial datasets for which several data mining tools exist to help in the detection of patterns and relations. It seems, however that GIS-minded analysis and objectives require more than the results provided by these general tools and possible adaptations to meet these conditions are needed.

This thesis attempts to answer the following research questions:

- Is the SOM suitable for exploratory spatial data analysis?
- What are the possible applications of SOM in the GISc field?
- What are the possible adaptations to the standard SOM algorithm to include geospatial reasoning in the analysis?

1.3. Objectives

The main objective of this thesis is to prove that data mining tools can be used to deal with geospatial data and provide examples of how they should be adapted to do so. The particular characteristics of this type of data and specific GIS-minded analysis can sometimes require adjustments in the general data mining methods, while in other cases minimal or no changes are necessary. These adjustments are dependent on the problem we are dealing with and on the expected results and analysis.

The thesis presents a very popular method in the Data Mining field, the Self-Organizing Map (SOM) (Kohonen 1982), and its value in handling several GIScience problems. The SOM is a technique used mainly in visualisation, data abstraction and clustering tasks. Its main advantage is that it allows the users to investigate the data structure and thus it

allows an improved understanding of the data. There have been several proposals to use SOM in the exploration of spatial data. One of SOM's most useful features is the density estimation capability. In fact, this ability can be of great usefulness in addressing several long-term problems in GIScience. In this thesis, the SOM is used in three of these GISc problems where density estimation is central: in the cartographic representation, in spatial data mining and in a particular case of the location/allocation problem.

The objectives of this thesis are:

1. to review SOM, focusing on some features that are familiar to GIS scientists like topology and visualisation techniques;
2. to review the use of SOMs in the geospatial context, comparing different approaches, results and available tools;
3. to examine more possibilities of adapting the SOM algorithm to the GIS-minded analysis;
4. to propose the use of the SOM to deal with several GIScience problems.

In line with the last objective, this thesis proposes:

a) a new method to create cartograms based on one geospatial variable;
b) an improved method to perform spatial clustering from high-dimensional datasets;
c) a new method to perform thematic spatial clustering from geospatial databases;
d) a new method to manage unmanned vehicles defining an optimised path to best cover a specific region.

1.4. Methodology

The approach we follow in this thesis starts by an in depth study of an artificial intelligence method, the SOM. This is a method widely used among data analysts with robust and reliable results. Also in the GIScience context, a growing number of proposals exist in the literature, *e.g.* (Agarwal and Skupin 2008). A survey of these approaches using SOM to deal with geospatial data is presented.

In our opinion, the use of SOM in the field of GISc could help in the solution of many GISs challenges and problems. From a review of the work done in the field, we identified three problems, stemming from three different areas in the GIScience field that could benefit from this particular method.

From the cartographic representation area, we test the possibility of using SOM to build cartograms. Concerning spatial analysis, we explored SOM-based methods (GeoSOM, Hierarchical SOM, and standard SOM), developing new techniques, improving others, and developing a user friendly software package that we made freely available. Finally, regarding a particular location/allocation problem, we tested SOM as the core of a method to handle the problem of path planning for a network of mobile sensors.

For each proposed problem we present a review, with special focus on the alternative methods used, benchmark datasets and evaluation measures. A detailed formulation of each problem is also presented, followed by a SOM-based method providing a possible solution. These methods are evaluated and compared with other methods using both fictional and real datasets.

1.5. Thesis organization

This thesis is organized as follows:

Chapter 2 presents the state of the art related to SOM. A review of the method is presented in the first part followed by a survey of SOM applications in the geospatial context.

Chapter 3 presents the first method developed in this thesis, where SOM is used to build cartograms in an approach called Carto-SOM. The basic idea of a cartogram is to distort a geographical map by substituting the geographic area of a region by some other variable of interest. The objective is to rescale each region according to the value of the variable of interest while keeping the map, as much as possible, recognizable.

Chapter 4 presents the second application of SOM in GIScience developed in this thesis. In this chapter, SOM is adapted to perform spatial clustering. Clustering forms one of the most popular and important tasks in data analysis. The goal here is to present a tool (GeoSOM Suite) with an improved method of spatial clustering. GeoSOM Suite is

designed to bridge the gap between clustering and the typical geographic information science objectives and needs, providing a user friendly environment. We believe that this tool can very useful to other researchers and practitioners.

In Chapter 5, we propose the use of Hierarchical SOMs to perform geospatial clustering. As we shall see, several characteristics of geospatial data make Hierarchical SOMs a natural choice for their analysis.

Chapter 6 introduces the fourth SOM based method developed. Here, SOM is used to deal with a real time location/allocation problem. The goal is to present a new method to define patrol paths for individual UAV (unmanned aerial vehicles), that compose a network of mobile sensors.

Chapter 7 summarises the conclusions of our research. Open research questions and future research are also discussed in this chapter.

2. State of art

2.1. Introduction

The aim of this chapter is to provide an overview of the work done in the field of SOM and GeoComputation. We start by making an SOM overview, presenting its main properties and tools for exploration. Section 2.3 presents a survey of methods using SOM in the geospatial context.

2.2. Self Organizing Maps

Artificial Neural Networks (ANNs) are an approach for quantitative data analysis inspired on the way we believe the brain processes information (Silipo 1999). It consists of a processing system composed by a large number of interconnected elements (units or neurons) working together to solve specific problems. Their learning capability makes them very suitable to different specific applications, such as pattern recognition or data classification (Gurney 1997). ANNs represent not a single method but a family of models. One example of a network that performs a non-linear projection of

multidimensional input data onto a lower dimension array of neurons (or units) is the Self-Organizing Map (SOM).

Self-Organizing Maps (SOMs), or Self-organizing feature maps (SOFMs) were first proposed by Tuevo Kohonen in the beginning of the 1980s (Kohonen 1982), and constitute the product of his work on associative memory and vector quantization. Since then there have been many excellent papers and books on SOM, but his book Self Organizing Maps,edited originally as (Kohonen 1995), and later revised in 1997 and 2001 (Kohonen 2001) is generally regarded as the main reference on the subject.

Kohonen's SOMs draw some inspiration from the way we believe the human brain works. Research has shown that the cerebral cortex of the human brain is divided into functional subdivisions and that the neural activity decreases as the distance to the region of initial activation increases (Kohonen 2001).

2.2.1. Overview

SOM's basic idea is to map high-dimensional data onto one or two dimensions, maintaining the topological relations between data patterns. SOM's main objective is to "*extract and illustrate*" the essential structures in a dataset, through a map resulting from an unsupervised learning process (Kaski and Kohonen 1996; Kaski, Nikkilä *et al.* 1998). SOM is normally used as a tool for mapping high-dimensional data onto a one, two, or three dimensional discrete feature map. The grid formed by the units or neurons is what is usually referred to as the output space, as opposed to the input space, which is the original space where the data patterns lie (Figure 1).

The main advantage of SOM is that it allows us to have some idea of the structure of the data by observing the map. This is possible mainly due to preservation of topological relations, *i.e.*, patterns that are close in the input space will be mapped, as far as possible, to units that are close in the output space. The output space is usually 2-dimensional and in most implementations it is a rectangular grid of units (the grid can also be hexagonal) (Kohonen 2001). Single-dimensional SOMs are also common (*e.g.* for solving the travelling salesman problem) (Lobo and Bação 2005), and some authors have used 3-dimensional SOMs (Seiffert and Michaelis 1995; Kim and Cho 2004;

Gorricha and Lobo 2009). Using higher dimensional SOMs is rare because, although they pose no theoretical problem, the output space is difficult to visualise.

Figure 1 shows an example of a two-dimensional SOM with 3 x 3 units adapting to a three-dimensional input dataset.

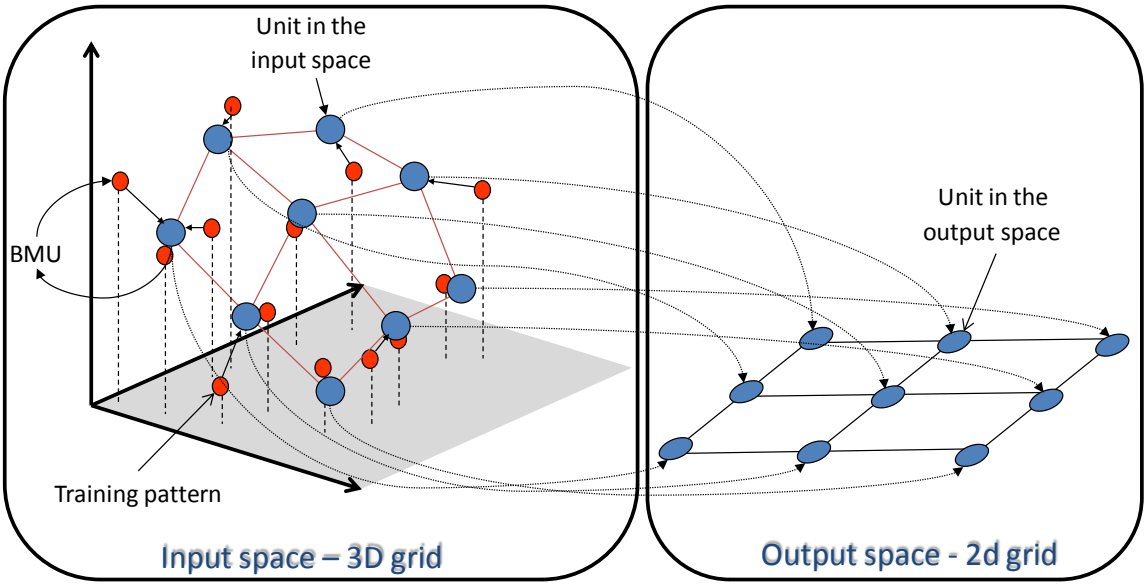


Figure 1 - Self Organizing Map's output space (two-dimensional) and input space (three-dimensional). Blue circles represent the units of the SOM while the red circles represent the input patterns

Each unit of the SOM, is represented by a vector $\boldsymbol{m_i}=[m_{i1}...,m_{in}]$ of dimension *n*, where *n* equals the dimension of the input space. In the training phase, a given training pattern ***x*** is presented to the network, and the closest unit is selected. This unit is called the best-matching unit (BMU) (Figure 1). The unit's vector values (synaptic weights in neural network jargon) and those of its neighbours are then modified in order to get closer to the data pattern ***x***:

$$m_i = m_i + \alpha(t)h_{ci}(t)(x - m_i) \qquad \textbf{Equation I}$$

Where $\alpha(t)$ is the learning rate at time *t*, and $h_{ci}(t)$ is the neighbourhood function centred in unit *c*, and *i* identifies each unit. Both $\alpha(t)$ and $h_{ci}(t)$ decrease with time during the learning phase. Figure 2 shows an iteration of the SOM training phase, where a training pattern is presented to the network and the closest unit is selected (BMU). This unit is then moved towards the input pattern according to the learning rate. Based on the

BMU and on the neighbourhood function, neighbours are selected on the output space. These neighbours will also be updated, albeit less, towards the input pattern.

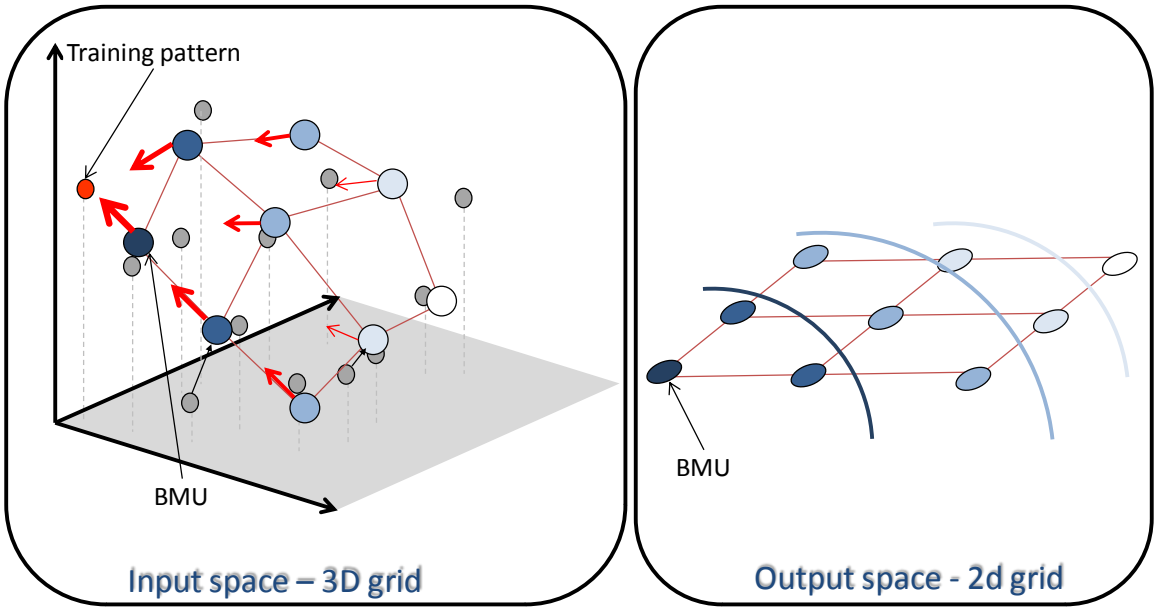


Figure 2 – SOM training phase. A training pattern (red dot) is presented to the network and the closest unit is selected (BMU). Depending on the leaning rate, this unit moves towards the input pattern (represented by the red arrow). Based on the BMU and on the neighbourhood function, neighbours are selected on the output space (blue lightness represents the degree of neighbourhood). Neighbours are also updated towards the input pattern

Independently of the quality of the training phase there will always be some residual distance between the training pattern and its representative unit. This difference is known as quantization error. This value is used to measure the accuracy of the map's representation of data. Section 2.2.5 presents a deeper explanation on the quality of the SOM.

2.2.2. SOM algorithm

The SOM algorithm can be easily described as shown below:

```
For all training patterns
      Compute the distances to all units
      Find the closest unit
      Update that unit and its neighbours
Repeat this process until a given stopping criteria is met
```

The first step is to define the network size, the initial learning rate and initial neighbourhood radius. There are no theoretical results indicating the optimal values for these initial parameters. This way the user's experience plays a major role in the definition of these parameters and can be of paramount importance in the outcome of the method. The next step is the initialisation of the unit's weights. These may be randomly generated, providing they have the same dimensionality as the training patterns. However, a careful choice will generally give better results. The next step is to initialise the training phase of the algorithm. For a number of iterations defined by the user, each pattern from the dataset is selected and presented to the network. The nearest unit (BMU) is found, usually based on Euclidean distance, but different measures can be used (Kohonen 2001; Lourenco, Lobo *et al.* 2004). The update phase consists on the adaptation of the unit weights and depends on the distance in the output space between each unit and the BMU, and the distance in the input space between the unit and the training pattern.

In order to SOM converge to a stable solution, both the learning rate and the neighbourhood radius should converge to zero. Usually these parameters decrease in a linear fashion but other functions can be used. Additionally, the update of both parameters can be done after each individual data pattern is presented to the network (iteration) or after all the data patterns have been presented (epoch). The former case is known as sequential training and the latter is usually known as batch training.

2.2.2.1. Sequential training

The basic SOM algorithm consists in three major phases: competition, cooperation and updating (Kohonen 2001). In the first phase, competition, all units compete in order to find the BMU for a given training pattern. In the cooperation phase, the BMU neighbourhood is defined based on the neighbourhood radius. Finally, in the update phase the BMU's weights along with those of its neighbours are updated in order to be closer to the data pattern. The algorithm can be written as follows:

Let
$\boldsymbol{X}$ be the set of n training patterns $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_n}$
$\boldsymbol{W}$ be a $p \times q$ grid of units $\boldsymbol{w_{ij}}$ where i and j are their coordinates on that grid
α be the learning rate, assuming values in $]0,1[$, initialised to a given initial learning rate
r be the radius of the neighbourhood function $h(\boldsymbol{w_{ij}}, \boldsymbol{w_{mn}}, r)$, initialised to a given initial radius

1 Repeat
2 For $k = 1$ to n
3 For all $\boldsymbol{w_{ij}} \in \boldsymbol{W}$, calculate $d_{ij} = \|\boldsymbol{x_k} - \boldsymbol{x_{ij}}\|$
4 Select the unit that minimizes d_{ij} as the winner $\boldsymbol{w_{winner}}$
5 Update each unit $\boldsymbol{w_{ij}} \in \boldsymbol{W}$: $\boldsymbol{w_{ij}} = \boldsymbol{w_{ij}} + \boldsymbol{\alpha h}(\boldsymbol{w_{winner}}, \boldsymbol{w_{ij}}, r)\|\boldsymbol{x_k} - \boldsymbol{w_{ij}}\|$
6 Decrease the value of α and r
7 Until α reaches 0

In this type of training, for each randomly selected training pattern presented to the network, a BMU, *i.e.* the closest unit, is found. The BMU is then updated according to the weights of the training pattern and the learning rate. Initially this learning rate is high allowing bigger adjustments of the units. The unit's mobility will decrease proportionality with the decrease of the learning rate. Based on the neighbourhood rate, a group of surrounding units is also moved closer to the training pattern.

2.2.2.2. Batch Training

The difference in batch training when compared to sequential training relies on the unit's updating process, and on the non-obligation to randomly present the training patterns to the network. Sometimes, the learning rate may also be omitted (Vesanto 2000). In this algorithm, units are updated only after an epoch, *i.e.* after all training patterns are presented once. In each epoch, the input space is divided according to the distance between the map units. The division of the input space is made using Voronoi regions (Brassel and Reif 1979; Brown 1979), also known as Thiessen polygons. These regions are polygons that include all points that are closer to a unit than to any other (Figure 3).

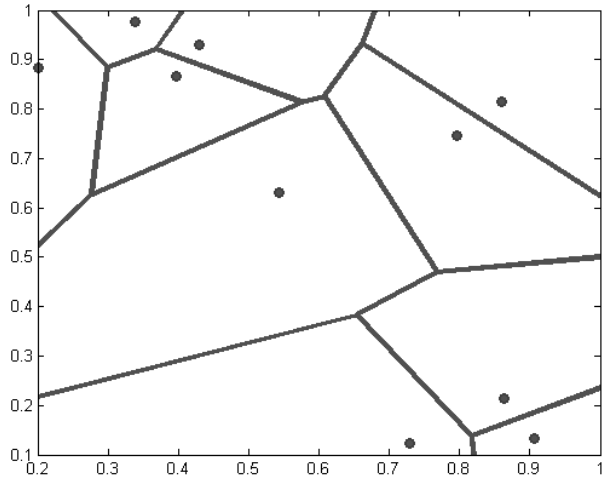


Figure 3 - Voronoi regions. Space division where all the interior points are closer to the corresponding generator than to any other

The new units' weights are in this case calculated according to Vesanto (2000):

$$m_i(t+1) = \frac{\sum_{j=1}^{N} h_{bi}(t) x_j}{\sum_{j=1}^{N} h_{bi}(t)} \qquad \textbf{Equation II}$$

Where t is the time, b is the BMU for the training pattern x_j, and $h_{bi}(t)$ is a neighbourhood kernel centred on the winner unit. The new weight vectors are a weighted average of the training patterns where the weight of each data pattern is the neighbourhood function value $h_{bi}(t)$ to its BMU b (Vesanto 2000).

Another way to get the new units' weights, computationally more efficient, is using the Voronoi set centroids n_j:

$$n_j = \frac{1}{N_j} \sum_{j=1}^{N} x_i \, \epsilon \, V_j x_i \qquad \textbf{Equation III}$$

$$m_j = \frac{\sum_{i=1}^{M} N_i h_{ij} n_i}{\sum_{i=1}^{M} N_i h_{ij}} \qquad \textbf{Equation IV}$$

Where N_j is the number of points in Voronoi set V_j.

2.2.3. Parameterisation of the SOM

Several parameters affecting the result have to be defined prior to the training phase of SOM. These include the definition of the SOM structure (size, topology, shape and initialisation of the map used) and the training parameters (number of iterations, initial learning rate and decrease function and initial neighbourhood radius and decrease function).

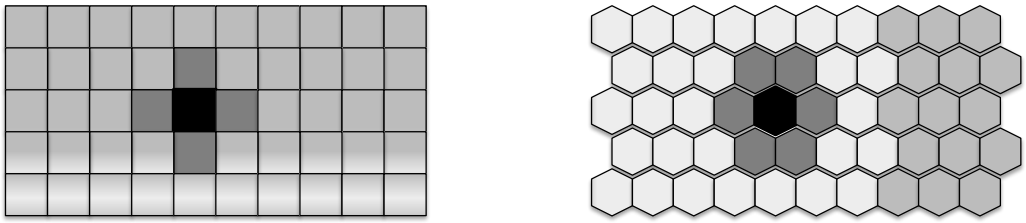
2.2.3.1. Size and dimension of the map

The SOM dimension and size depends on the problem at hand, which makes this definition mainly an empirical process (Kohonen, 2001). The number of units in a SOM is usually selected as big as possible, keeping in mind that as the size of the map increases the training phase becomes computationally inefficient. Depending on the SOM output space dimensionality, the size is given by the product of the number of units used in each dimension (*x, y, z, etc.*). For instance in a three-dimensional SOM, using *x* equal to 10, *y* equal to 10 and *z* equal to 20 produces a network with 2000 units.

The size of the SOM must also take into consideration the size of the dataset of training patterns. Two fundamentally different approaches are possible when choosing the size of the SOM, commonly known as *k*-means *SOM* (Bacao, Lobo *et al.* 2005) and emergent *SOM* (Ultsch 2005). In *k-means SOM*, the number of units should be equal to the expected number of clusters, and thus each cluster should be represented by a single unit. In that case, each unit is a cluster centroid in a similar way as the *k*-means clustering. In *emergent SOM*, a very large number of units is used (sometimes as many or more than the number of training patterns), to obtain very large SOMs. These very large SOM allow for very clear U-Matrices (Ultsch and Siemon 1990) and are useful for detecting quite clearly the underlying structure of the data.

2.2.3.2. Topology, shape and initialisation

The SOM uses, typically, two types of topology: square and hexagonal (Figure 4). In the first case, each unit is connected to its four neighbours (with the exception of the border units). In the second case, each unit is connected to its six neighbours (again, with the exception of the border units).



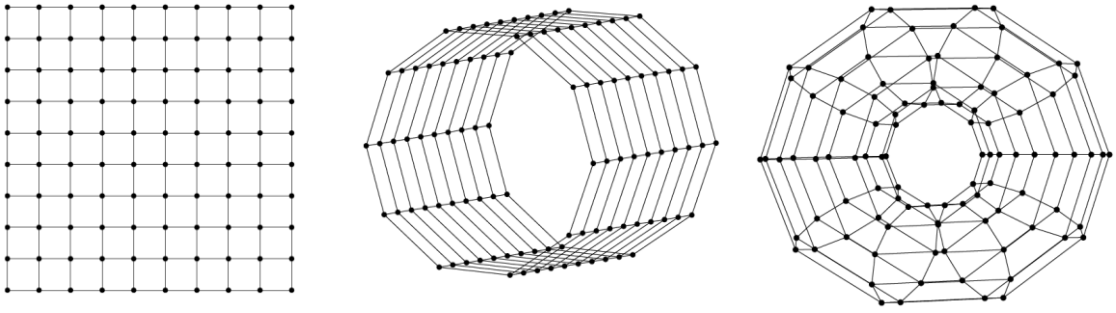
a) **b)**

Figure 4 – SOM topology: a) square topology with four neighbours and; b) hexagonal topology with six neighbours. The units considered neighbours of the black unit are presented in dark grey. All other units are in light grey

Usually, the hexagonal topology is preferred although if training is sufficiently long the network performance is independent of the topology used (Jiang, Berry *et al.* 2009). In practical cases, a hexagonal topology will produce smoother maps.

The SOM can also have different shapes. While the default is the sheet, two other configurations are implemented in the MATLAB™ SOM toolbox (Vesanto, Himberg *et al.* 1999): the cylinder and the toroidal shape. The cylinder and toroidal shapes are very effective in eliminating the edges and creating a continuous two-dimensional surface, although a good distribution of the network to represent data is usually harder to achieve (Ultsch, Hermann *et al.* 2005).

MATLAB™ SOM toolbox supports both hexagonal and rectangular lattices and sheet, cylinder and toroid shapes (Figure 5).



a) **b)** **c)**

Figure 5 – Different SOM shapes implemented in SOM Toolbox using a square topology: a) sheet is the default SOM shape; b) cylinder shape and; c) toroid shape (Vesanto, Himberg *et al.* 2000)

Another possible configuration of the SOM is the spherical shape (Ritter 1999). In this context, Wu *et al.* (2005; 2006) propose a spherical SOM based on a icosahedron geodesic dome (Figure 6). The spherical shape has the advantage of eliminating the edge effect and of great suitability for data without underlying directional structures. In addition, spherical maps are more familiar to most people than the toroid or cylinder.

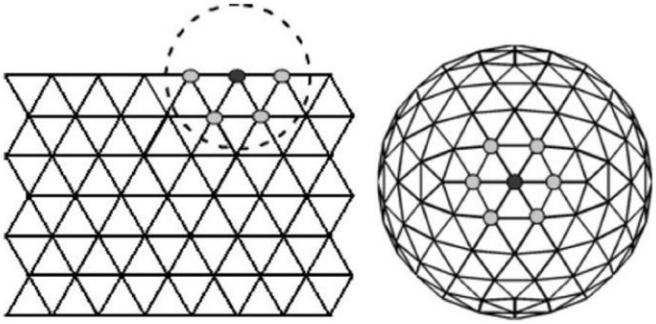


Figure 6 – Comparison of the two-dimensional and spherical SOM (Wu and Takatsuka 2006)

Finally, before the training phase starts each unit vector has to be initialised. Although SOM is very robust with respect to initialisation, a proper initialisation allows the algorithm to converge faster to a good solution (Kohonen 2001). This initialisation can be random or linear. In the case of random initialisation, the unit's weights are randomly selected or randomly drawn from the input training samples. If linear initialisation is used, the units' weights are initialised along a linear subspace that can be defined by the two principal eigenvectors of the input dataset. A particular initialisation of SOM is proposed in (Bação, Lobo *et al.* 2008) called geo initialisation. In this proposal, if using geospatial input data, the geographical units' weights are linearly distributed along the geographical space while aspatial weights are randomly selected.

2.2.3.3. Number of iterations

The number of iterations in the SOM defines how long the training phase will be. Thus, the number of iterations defines the number of input patterns (possibly with repetitions) presented to the network. An alternative, when using the MATLAB™ SOM toolbox, is to define the number of epochs. An epoch is the set of iterations where all the training patterns were presented once to the network. This presentation of the input patterns can be random or follow the order of the dataset. There is no rule about the optimal number

of iterations to use; only that the training phase of the SOM should be large enough to allow a good adaptation to the input patterns. The degree of adaptation can be measured using SOM quality measures explained in section 2.2.5.

2.2.3.4. Learning rate and learning functions

The learning rate α assumes values in [0, 1], having an initial value $\alpha(t_0)$ given by the user. As already discussed, the learning rate decreases to zero during the training phase. Several different functions can be used to control the learning rate behaviour. In MATLAB™ SOM toolbox implementation, the available functions to control the decrease of the learning rate are *linear*, *power* and *inv* as shown in Figure 7.

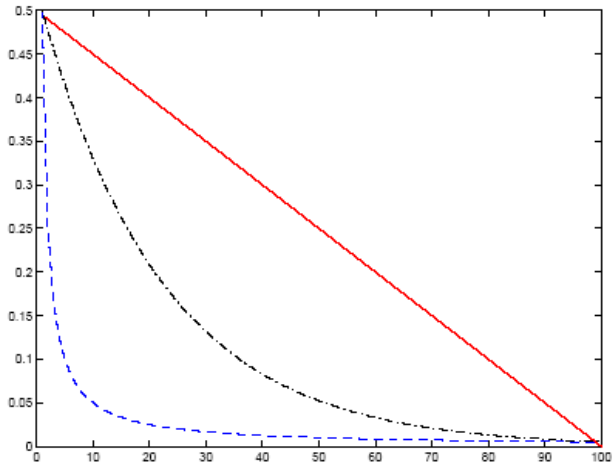


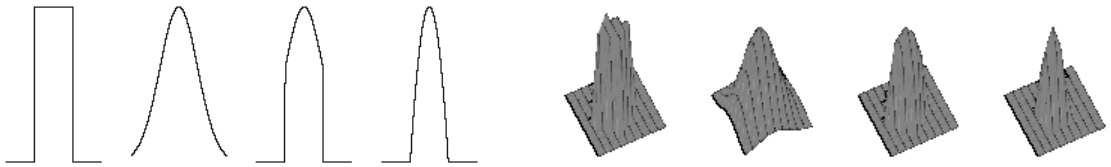
Figure 7 - Learning rate functions.

Linear (in red) $(t) = \alpha_0\left(1 - \frac{t}{T}\right)$ **; Power (in black)** $\alpha(t) = \alpha_0\left(\frac{0.005}{\alpha_0}\right)^{\frac{t}{T}}$ **and; Inv (in blue)** $\alpha(t) = \frac{\alpha_0}{\left(1+\frac{100t}{T}\right)}$ **, where** T **is the training length and** α_0 **is the learning initial rate (Vesanto, Himberg et al. 2000)**

Different decrease functions for the learning rate will influence the network's mobility in adjusting to the input patterns. In the three cases presented, the linear function will allow a constant decrease in the mobility while the *inv* function exhibits a high mobility at the beginning of the training phase and rapidly moves to small adjustments in the network.

2.2.3.5. Neighbourhood radius and neighbourhood functions

The neighbourhood function, *h*, can have values in [0, 1], and is a function of the position of two units (a winner unit and another unit) and a given radius *r*, that usually decreases with time. *h* has a high value for units that are close in the output space, and decreases with distance increase. Usually, the neighbourhood function is a radial function with a maximum at the centre, monotonically decreasing up to a radius *r* (sometimes called the neighbourhood radius) and is zero from there onwards. The neighbourhood radius can have values between 0 (in this case only the BMU will be updated) and the maximum size of the network (in that case, all SOM's units will be updated). MATLAB™ SOM toolbox makes available several neighbourhood functions, such as *bubble*, *gaussian*, *cutgass* and *ep*. Figure 8 exemplifies the four neighbourhood functions in two and three dimensions.



a) **b)**

Figure 8 - Neighbourhood functions.

From the left, *bubble* $h_{ci}(t) = 1(\sigma_i - d_{ci})$; g*aussian* $h_{ci}(t) = e^{\frac{-d_{ci}^2}{2\sigma_i^2}}$; *cutgass* $h_{ci}(t) = e^{\frac{-d_{ci}^2}{2\sigma_i^2}} 1(\sigma_i - d_{ci})$ and *ep* $h_{ci}(t) = \max\{0.1 - (\sigma_t - d_{ci})^2\}$. Where σ_t is the neighbourhood radius at time t, and $d_{ci} = \|r_c - r_i\|$ is the distance between units c and i in the output space. a) Neighbourhood function using a one-dimension output space; b) Neighbourhood function using a two-dimension output space and a neighbourhood radius $\sigma_t = 2$ (Vesanto, Himberg *et al.* 2000)

The neighbourhood function and the number of units affect the granularity of the resulting mapping. High values of the neighbourhood function allied with high neighbourhood radius make maps that are more rigid. Increasing the size of the map, on the other hand, makes the SOM more flexible. This interplay determines the accuracy and generalization capability of the SOM (Vesanto, Himberg *et al.* 1999).

2.2.4. Visualisation of the SOM

There are several ways to visualise the SOM to improve the understanding of the data patterns (Vesanto 1999). In Figure 9, we present a possible taxonomy for the several methods used to visualise SOM.

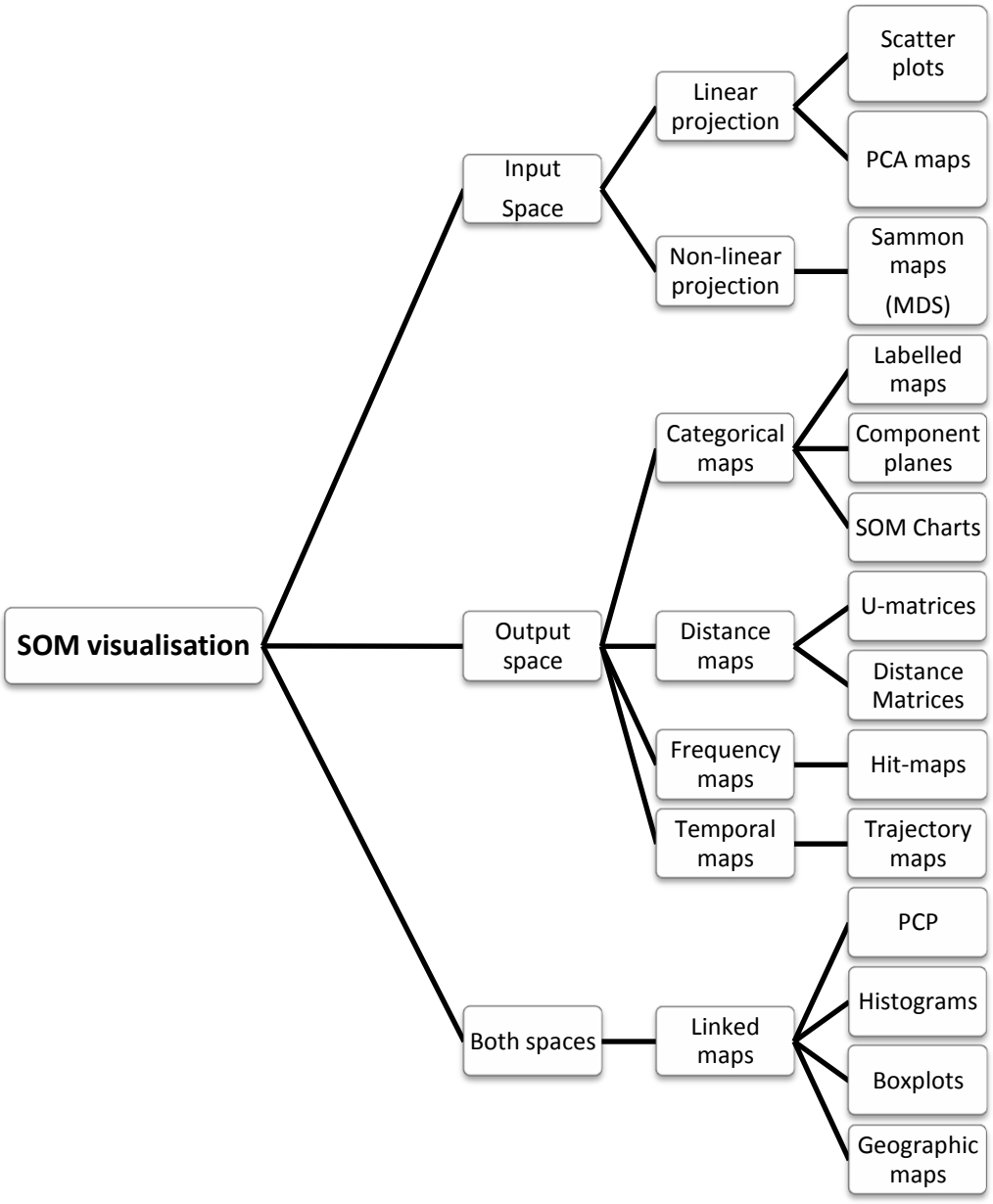


Figure 9 - Taxonomy for SOM visualisation methods

SOM visualisation methods can be divided into three major groups according to the space visualised. Thus, methods can show the input space and/or the output space of

the SOM. To visualise the input space of SOM, which has the same dimensionality of the input data, we can use linear or non-linear projections. Scatter plots of a sub-set of the original dimensions or Principal Components Analysis (PCA) of the units' weights are types of linear projections, while Sammon mapping and other Multidimensional Scaling techniques (MDS) fall in the category of non-linear maps.

In the SOM output space visualisation, the methods can be subdivided into different categories according to the type of information presented. Thus, we can have: 1) categorical maps presenting the distribution of some attributes, like labelled maps, component planes or SOM charts; 2) distance maps presenting the input space's distances in the output space, some examples being the U-matrices or distance matrices; 3) frequency maps which show the number of occurrences in each unit, such as hit-maps and; 4) temporal maps such as the trajectory maps.

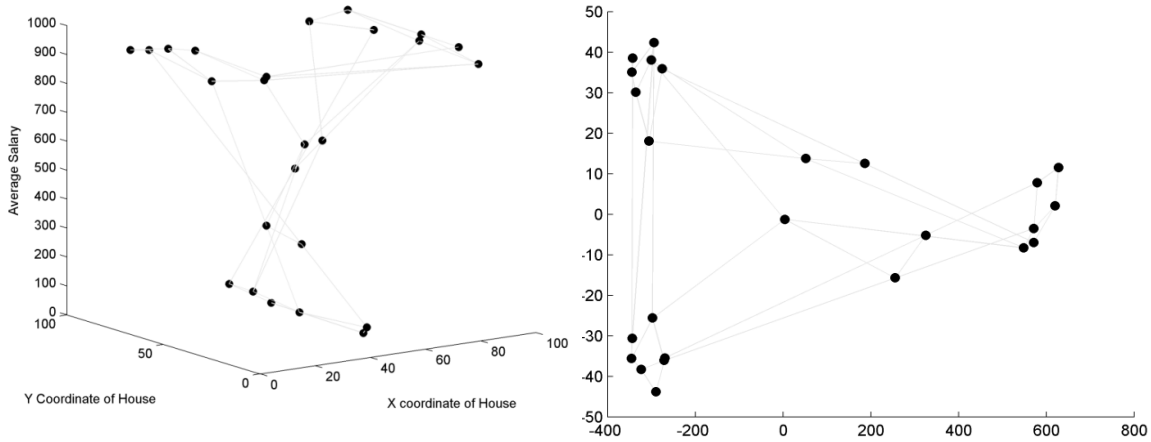
Finally, some methods can use both spaces and present an input space's visualisation method combined with elements from the output space. We called this category linked maps. Linked maps use specific input data attributes to make associations of the SOM's output space with different spaces. Examples of this case are parallel coordinate plots, histograms, box-plots and geographic maps.

In the remaining of this section, we will review each of these methods, using the Squareville dataset (Lobo, Bação *et al.* 2009) as example. Squareville dataset characterizes a small number of houses (100) in terms of location (*x, y* coordinates) and average salary. Further description on this dataset is presented in section 4.4.1.

2.2.4.1. Input space: linear projection

One of the possible representations of SOM is to map the units in the input space (Figure 10a), through scatter plots. The visualisation of the training patterns' positions on the output map is often not enough to see structures, because high dimensional distances are distorted in the projection. Units that are immediate neighbours on the map can still be separated by large distances in the input space. By visualising the units in the input space, we will have a clear view of how they are related. Unfortunately, the input space is usually high-dimensional, and thus we will have to project the units onto two or three of the original dimensions.

Another possibility to visualise the SOM input space, in the case it has more than three dimensions, is to use principal component analysis (PCA) proposed by Pearson (1901) to reduce the dimensionality. PCA will achieve this by transforming the original variables into new ones (the principal components) retaining most of the variance of the whole dataset. Figure 10b shows an example using PCA to visualise the SOM input space.



a) **b)**

Figure 10 – Linear projections of the SOM's input space: a) Scatter plot of the SOM's input space using the unit's weights for the houses location (*x* and *y* coordinates) and the average salary; b) scores from the two principal components obtained from the principal components analysis

A way around to this limitation is to use a non-linear projection of the input space.

2.2.4.2. Input space: non-linear projection

Several methods can be used to perform this projection of multi-dimensional data into lower spaces. Multidimensional scaling (MDS) is a set of techniques that deals with this problem, placing objects represented by points in a two or three-dimensional space, so that the distances between the points correspond in some sense to their dissimilarities (Torgerson 1952; Kruskal 1964). One of these techniques is the Sammon mapping introduced in (Sammon 1969). As far as possible, Sammon's mapping preserves the original distances by minimizing a criterion that penalizes differences in distances. Figure 11 presents a Sammon mapping obtained from the SOM units' weights.

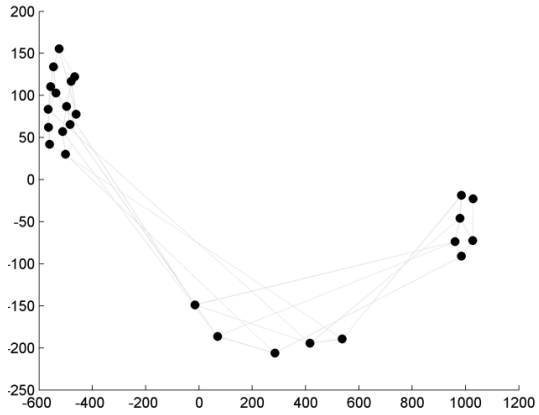


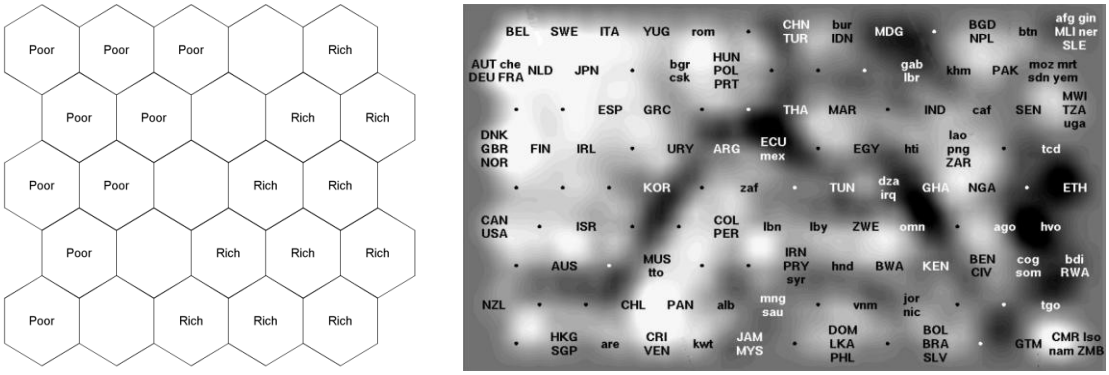
Figure 11 – Non-linear projections of the SOM's input space using Sammon mapping projection of the original three dimensions into a two-dimensional map

Finally, as already described SOM also performs non-linear projection of the input data. Thus, plotting the output space of SOM, despite its distortions, will also give an idea of the distribution of the units in the input space. There are several representations to show the output space of SOM which we present in the following sections.

2.2.4.3. Output space: categorical maps

Categorical maps present the distribution of data using the output space. Some examples of this category are labelled maps, component planes and SOM charts.

Labelled maps use the output space matrix to map the units and their labels. Each unit label is obtained from some attribute or class (from now on called label) of its training patterns in a process called labelling or calibration (Kohonen 2001). Several variants can be used in this process: each unit can get the labels from all its training patterns; each unit can get only a unique label from its training patterns (say, the label of the nearest training pattern) or; each unit can get the most frequent label from its training patterns. This map allows users to visualise the way training patterns were mapped into the SOM. Figure 12 presents two examples of labelled maps. The first example presents a labelled map from Squareville while the second example presents a combination of labelled maps and U-matrix (which we will see later) using economic data from world countries. U-matrices are further explained in the section 2.2.4.4.

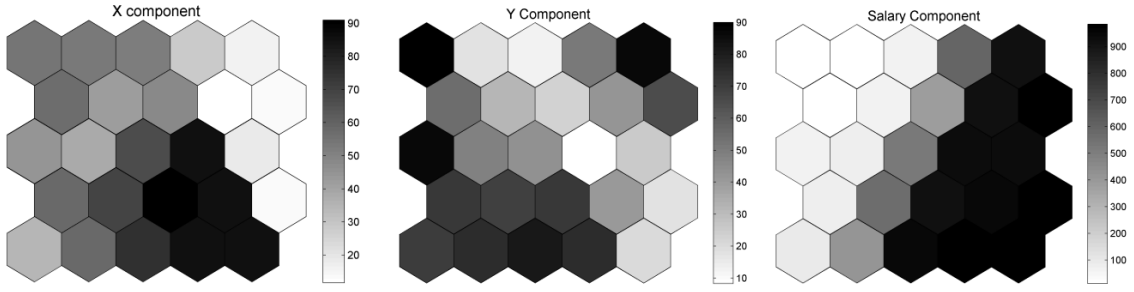


a) **b)**

Figure 12 – Labelled maps: a) labelled map using the Squareville dataset and; b) labelled map combined with U-matrix showing using world countries' economic data (Kaski and Kohonen 1996)

Component planes (Kohonen 2001) are also used for SOM visualisation and exploration. In a component plane, each unit is coloured according to the weight of each variable in the SOM (Figure 13). Through the analysis of the component planes, data distribution can be evaluated. For instance, it is quite simple to identify variables which are correlated (their component planes will have the same shape), and it is also possible to have an improved understanding of the contributions of each variable to the SOM. By comparing two or more component planes, one can visually identify correlations between variables, both globally and at a local scale.

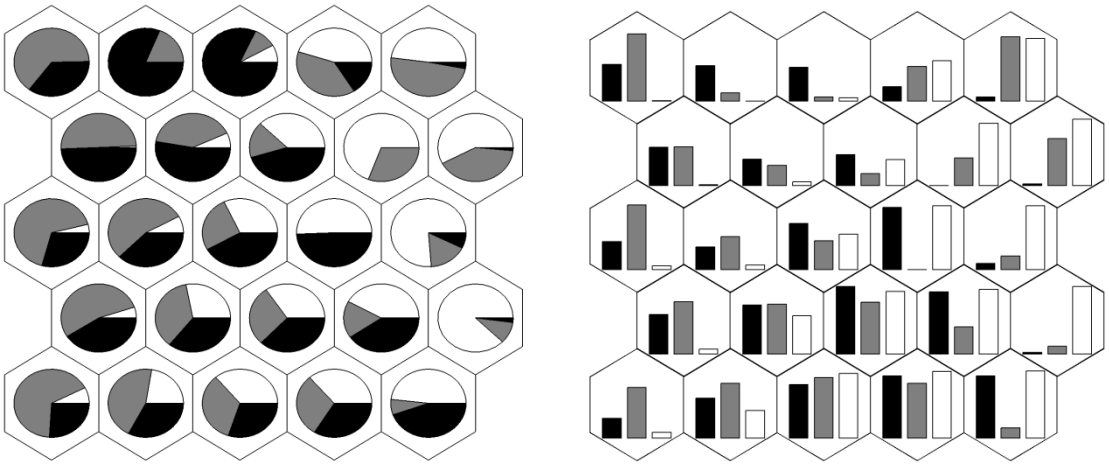
When there are many components, visualising all at the same time can be difficult. One way of conveying some information about each component without producing too much clutter is to use what we called *component plane hotspots*. A component plane hotspot is simply the region where that component has higher values. On a single map, we may plot the hotspots of all components, and thus have an idea of where each variable reaches its maximum.



a) **b)** **c)**

Figure 13 - Component planes showing the Squareville dataset variables: a) *x* coordinate; b) *y* coordinate and; c) average salary

Finally, SOM charts can be used to visualise in the output space, the units' distribution in the input space. Figure 14 presents two examples of SOM charts, in which a pie chart or a bar chart is used to represent each unit of SOM. In the example, each slice of the pie (or bar in the bar chart case) represents the value for each component of SOM.



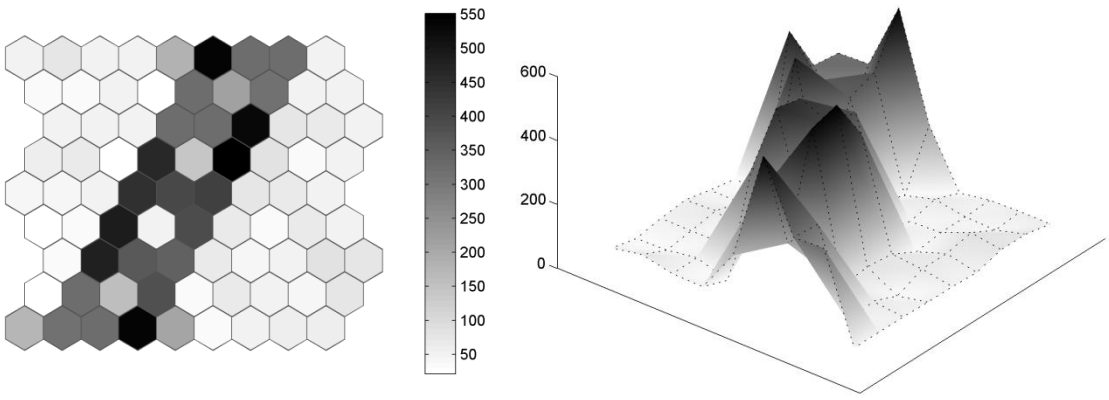
a) **b)**

Figure 14 – Squareville variables histograms plotted on the SOM's output space. Black represents the *x* coordinate; grey represents the *y coordinate* and white represents the average salary: a) SOM pie chart and; b) SOM bar chart

2.2.4.4. Output space: distance maps

Distance maps are representations used to transmit the units' distances in the input space using the output space. Some examples of this category are the U-matrices and distance matrices.

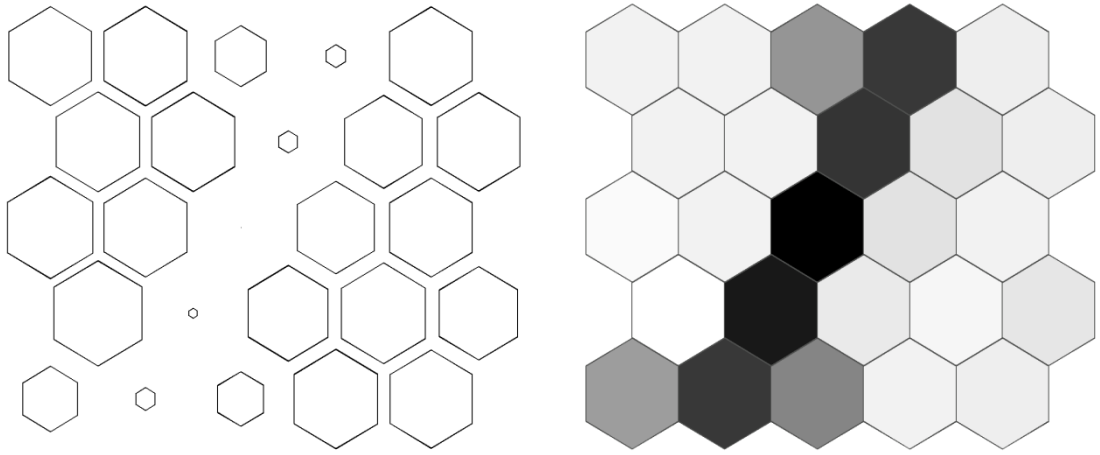
The U-matrix is one of the most used methods to visualise SOM (Ultsch and Siemon 1990). U-matrices are computed by finding the distances in the input space of neighbouring units in the output space. There are two ways to visualise a U-matrix. The most common is to use a colour code to depict distances, corresponding to the values of the U-Mat. Usually a greyscale is used, with the highest value being represented with black and the lowest with white (Figure 15a). Another possibility is to plot these distances in the form of a 3D landscape with mountains and valleys. A mountain region indicates large distances between units, while low distances between the units form valleys (Figure 15b).



a) **b)**

Figure 15 – U-matrix using Squareville data: a) two-dimensional U-matrix and; b) three-dimensional U-matrix

Distance matrices are similar to U-matrices but while the U-matrix uses inter-space elements to represent the space between units, distance matrices condense this information in the positions of the original units, and uses no such "inter unit elements". Thus, in a $n \times m$ SOM the distance matrices will present $n \times m$ elements (squares or hexagons) while the U-matrix present $(n \times 2 - 1) \times (m \times 2 - 1)$ elements. Distance matrices can use colour codes or element's size to represent distances. Figure 16 presents an example of these two types of distance matrices using Squareville data.



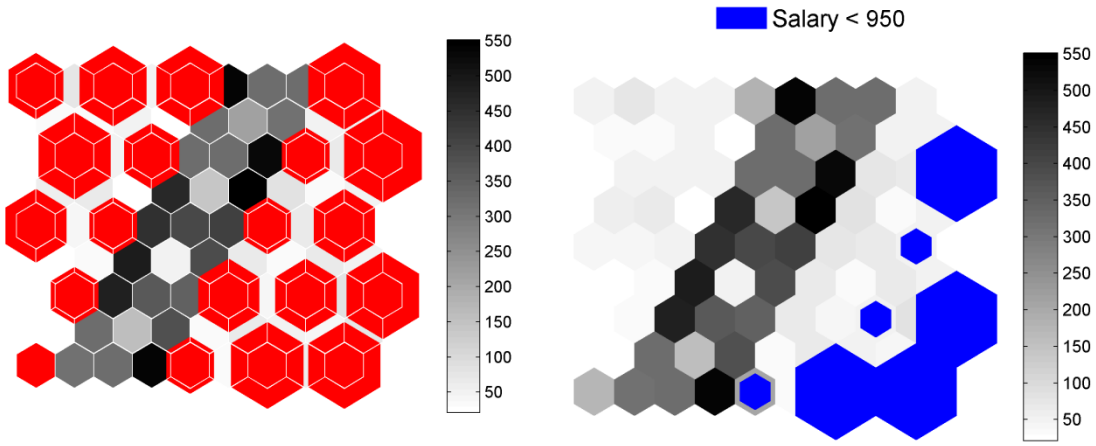
a) **b)**

Figure 16 – Distance matrices using Squareville data: a) size coded distances; b) colour coded distances

2.2.4.5. Output space: frequency maps

Frequency maps present the number of occurrences of data patterns in each unit. An example of this category is the Hit-maps (Kohonen 2001). This representation is usually

superimposed on the U-matrix or on the component planes, and gives information about the number of data items represented by each unit, *i.e.* data items with the same BMU. Hit-maps can be used to see how a certain set of data points are mapped on the SOM, gaining more information about the clustering structure. Figure 17 presents two examples of hit-maps. The first example uses all Squareville data while the second example uses only input patterns where the average salary is less than 950.



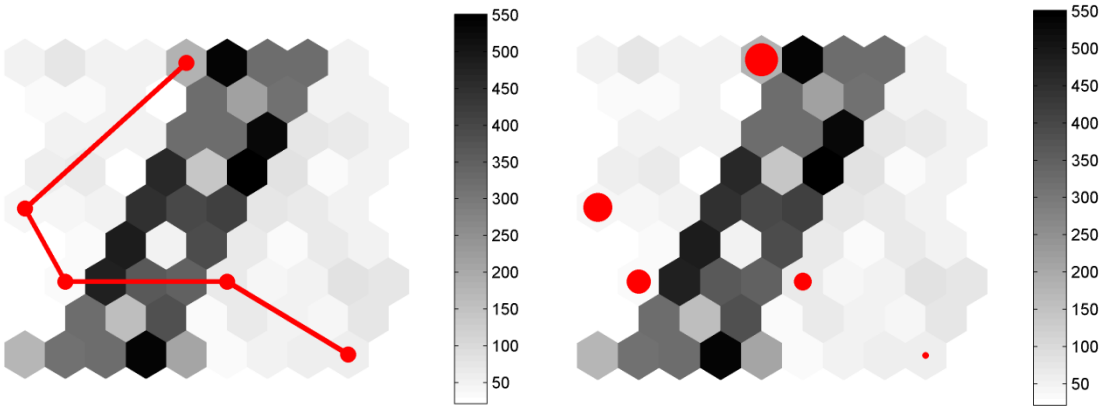
a) **b)**

Figure 17 – Hits-map plot: a) using all data from Squareville, the size of the red hexagons represent the number of input patterns belonging to each unit and; b) using only input patterns where the average salary is less than 950, the size of the blue hexagons represent the number of input patterns belonging to each unit

2.2.4.6. Output space: temporal maps

The standard SOM algorithm does not consider the variable time as a special attribute. However, many approaches have been used so that SOMs may process temporal data. For an extensive and thorough review of the work done in temporal SOMs the reader is referred to (Guimaraes 2000; Lobo 2002).

However, in the case the samples are ordered forming a time-series, it is possible to track the evolution on the map. Figure 20 presents two types of trajectory maps showing the evolution of a family for which the house location and salary for a period of five years is known.



a) **b)**

Figure 18 – Trajectories maps: a) trajectory map using a line to present the evolution and; b) comet map, in this case a comet like drawing presents the evolution (from larger to smaller circles)

2.2.4.7. Both spaces: linked maps

Finally, the SOM can be visualised using a combination of methods presenting both input and output space elements at the same time. Usually, this type of presentation works better on a digital exploration platform, which allows user interaction with the results. Examples of this interaction are capabilities like selection, zooming and brushing. Some of these visualisation methods are parallel coordinate plots (PCP), histograms, box-plots and geographic maps.

A parallel coordinate plot (Inselberg 1985) is a data representation technique for plotting multivariate data. This technique starts by drawing a parallel axis for each variable. A line connecting each variable axis value then maps each data item. This allows the user to visually compare multivariate data vectors. Boxplots are 2-dimensional graphics displaying several statistics for each variable (the smallest observation of a given variable, the lower quartile, the median, the upper quartile, the largest observation and the outliers). Histograms are a graphical display of tabulated frequencies, shown as bars. Figure 19 presents examples of these methods, combined with the SOM output space.

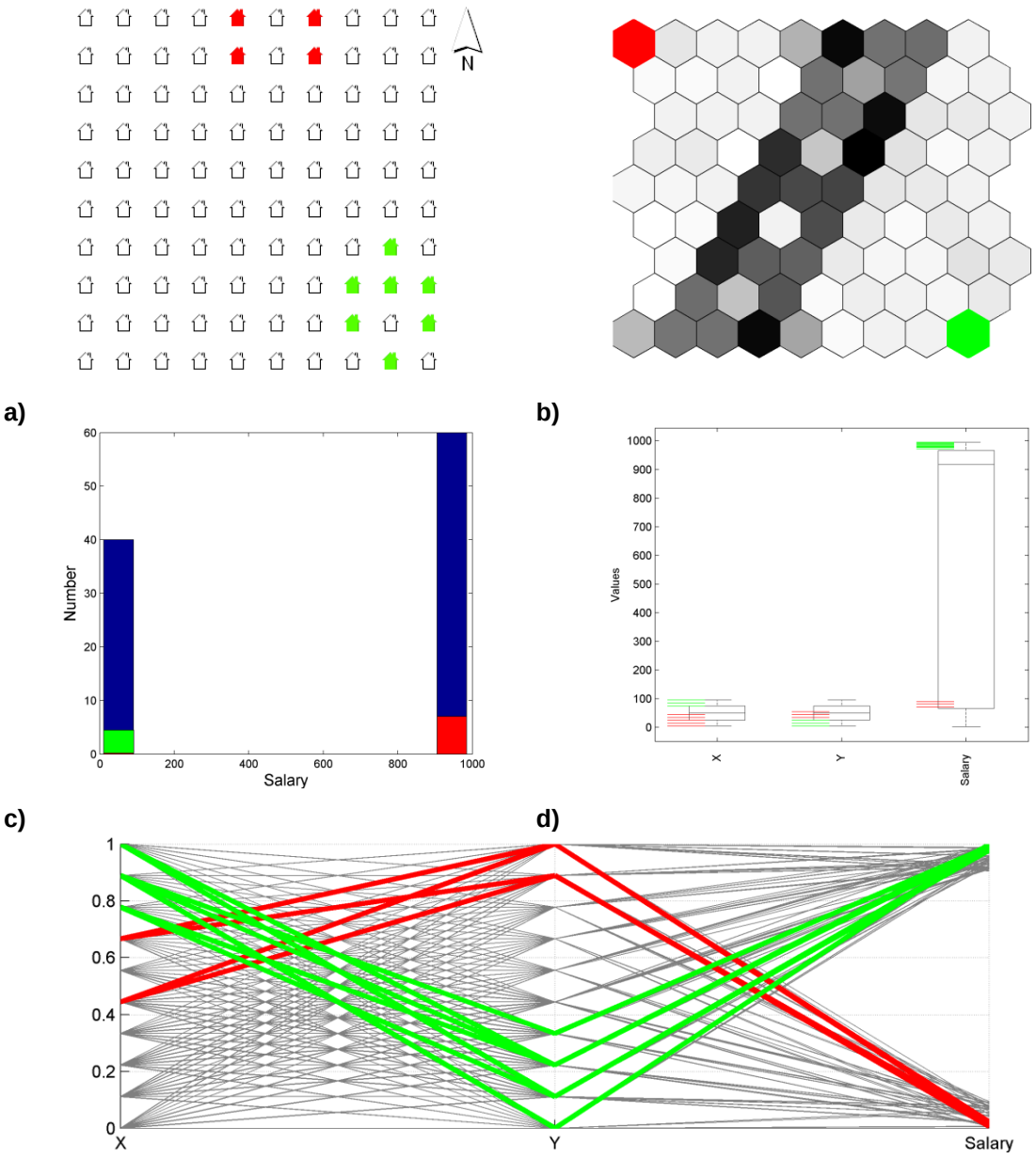


Figure 19 – Several linked space visualisations: a) geographical map, with classes obtained from the SOM; b) U-matrix presenting the same classes; c) combined histogram of the average salary for all the input patterns and input patterns from the selected classes; d) boxplot of the dataset presenting the distribution of the input patterns belonging to the classes and; e) parallel coordinate plot showing the classes' input pattern distribution

Finally, if the input data has a geospatial context, geographic maps can be used to visualise the SOM results. An example that is more expressive for a geographical map than the one in Figure 19a, is given in Figure 20 that presents a SOM based cluster analysis using socio-demographic variables for the Lisbon Metropolitan Area.

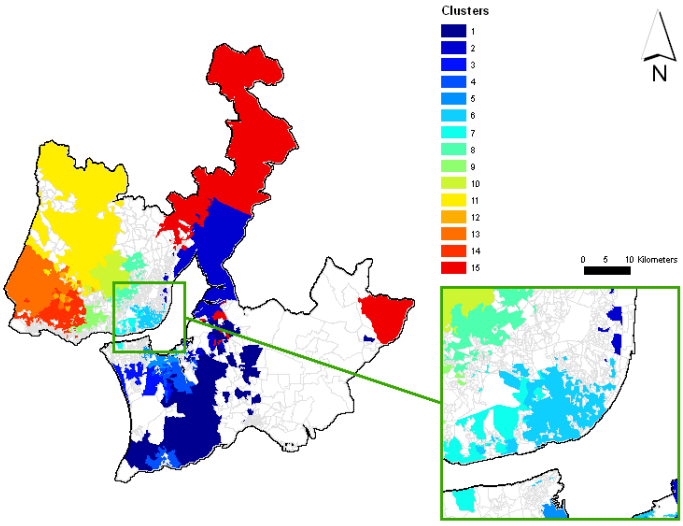


Figure 20 – Visualising the SOM in a geographic map. This example presents SOM based clusters of Lisbon Metropolitan Area, which will be further explained in chapter 4

2.2.5. Quality of the SOM

Assessing the quality of the SOM is a very important aspect in the production of reliable analysis. There are two main types of error in SOM: the quality of SOM adaptation quality and its topographical errors (Kohonen 2001).

In terms of adaptation quality the most usual measures are the quantisation error (q_e) (Kohonen 2001) and the classification error.

In this thesis, we will focus mainly on the quantization error (q_e) (Kohonen 2001). The quantization error is given by the average distance between a unit and the training patterns mapped to it, *i.e.,* all the input data patterns that share it as BMU.

$$q_e = \frac{\sum_{k=1}^{N} \|x_k - w_{BMU}\|}{N} \qquad \textbf{Equation V}$$

Where x_k is the training pattern, w_{BMU} is the best-matching unit for the training pattern x_k and N is the number of existing training patterns.

The classification error can only be calculated if we are in the presence of labelled data and, preferably, use a test dataset. Also, Wu and Takatsuka (2005) propose the Error Entropy to quantitatively analyse the spatial distribution of error in the SOM.

Topographical errors, on the other hand are concerned with the topology preservation of the SOM. Some of the most used measures are the topographic error (t_e) (Kiviluoto 1996), the topographic product (Bauer and Pawelzik 1992), the topographic function (Villmann, Der *et al.* 1997), the *c*-measure (Goodhill and Sejnowski 1997), the minimal path length and minimal wiring (Durbin and Mitchison 1990), the Zrehen's measure (Zrehen and Blayo 1992), kaski's trustworthiness measure (Kaski, Nikkila *et al.* 2003), and the minimal U-ranking (Ultsch, Hermann *et al.* 2005). Vesanto *et.al.* (2003) proposed a SOM distortion measure to compare the quantization and topology preservation errors. For a further review in the SOM's evaluation measurements the reader is recommend to several surveys in the field (Bauer and Pawelzik 1992; Li, Gasteiger *et al.* 1993; Pölzlbauer 2004).

The topographic error (t_e) (Kiviluoto 1996) evaluates the topology preservation of the SOM. Considering that for each data pattern the closest unit will be its BMU, the second closest unit will be called the second BMU, or BMU2. Topographic error measures the proportion of all training patterns for which first and second BMU are not adjacent units.

$$t_e = \frac{\sum_{k=1}^{N} u(x_k)}{N}$$ **Equation VI**

Where x_k is the training pattern, N is the number of data vectors and $u(x_k)$ equals one (1) if the BMU and the BMU2 are not adjacent and zero (0) otherwise.

2.2.6. Available Software

There are several public-domain implementations of SOM, of which we would like to highlight the SOM-PAK (Kohonen, Hynninen *et al.* 1996) and MATLAB™ SOM Toolbox (Vesanto, Himberg *et al.* 2000), both developed by Kohonen's research group and available at http://www.cis.hut.fi/. The SOM Toolbox has, besides the core MATLAB™ routines, an excellent graphic-based user interface, that makes it a very simple tool to use, simplifying the experiments with SOMs.

Another public SOM implementation is the Databionic ESOM Tools (Ultsch and Moerchen 2005). The Databionic ESOM Tools is a suite of programs to perform data mining tasks like clustering, visualisation, and classification with emergent SOM (Ultsch 2005).

Several general data analysis tools include the SOM as a tool for visualisation and multidimensional analysis. A free example is the Orange (Demšar, Zupan *et al.* 2004). Orange is a library of C++ core objects and routines that includes a large variety of machine learning and data mining algorithms.

Several commercial data analysis software packages include SOM as a visualisation and clustering tool. Some of the most known examples are SAS® Enterprise Miner (SAS Institute Inc. 2009), SPSS (SPSS Inc. 2009), Viscovery SOMine® (Viscovery Software GmbH 2009) and VisiSOM© (Visipoint Ltd. 2009).

In the field of GIS some tools have already implemented SOM in their default version or take advantage of user developed add-ins. One example is the GIS and image analysis software IDRISI© (Clark Labs 2008), capable of either a supervised or unsupervised classification of remotely sensed imagery through the SOM. Another example where SOM is an add-in tool in a standard GIS is proposed in (Lacayo and Skupin 2007). This new module is written for the ESRI© ArcGIS© software.

2.2.7. General considerations on SOM

The SOM objective is to adjust the units to the data in the input space, so that the network is (as best as possible) representative of the training dataset. One of the problems related to this adjustment is the fact that the network tends to underestimate high probability regions and overestimate low probability areas (Bishop, Svensen *et al.* 1997).

This problem is usually called magnification factor (Cottrell, Fort *et al.* 1998). A basic theorem for the case of one-dimensional data shows that the unit's density is proportional to the training patterns density with a magnification factor of 2/3 (Ritter and Schulten 1986; Bauer, Der *et al.* 1996).

Another problem of the SOM algorithm lies in the fact that the user to has to choose the values of several parameters heuristically (size of the network, learning rate and neighbourhood radius). Some research has been made on this subject and Haese and Goodhill (2001) proposed an algorithm that estimates the learning parameters during the training of SOMs automatically, using a Kalman filter. Another proposal was the use of

Genetic Algorithms (GA) in order to estimate the training parameters of a SOM (Silva and Rosa 2002).

The training of a SOM is more effective if it is done in two phases: the unfolding phase, and the fine-tuning phase. In the unfolding phase the objective is to spread the units in the region of the input space where the data patterns are located. In this phase the neighbourhood function should have a large initial radius so that all units have high mobility and the map can quickly cover the input space.

The fine tuning phase, as the name implies, is the process of small adjustments in order to reduce the quantization error, and centre the units in the areas where the density of patterns is highest. Usually, in this phase the learning rate and the neighbourhood radius are smaller than the ones used in the unfolding phase. As these two parameters are smaller, the map will need more time to adjust its weights and that is why the number of iterations or epochs is normally higher.

2.2.8. Supervised variants of SOM

As already stated SOM is an unsupervised method, suited for clustering, visualisation and abstraction of data. However, the Learning Vector Quantization (LVQ) neural networks (Kohonen 2001), which are suited for supervised learning are closely related to SOM. The LVQ are more apt for classification, and rely on training algorithms similar to the SOM. Several algorithms exist in the LVQ class, such as LVQ1, LVQ2, LVQ3 and OLVQ1 (Kohonen 2001). As a supervised method, LVQ requires a subset of classified input patterns to allow the classification of all the input data.

All LVQ algorithms are based on the intuitive idea of rewarding correct classifications and punishing incorrect ones. The algorithms use as input data, patterns with a specific assigned class.

The original algorithm, LVQ1, moves the units closer or further away from each data input pattern based on whether they are correctly classified. The algorithm can be written as follows:

Let
$\boldsymbol{X}$ be the set of n training patterns $\boldsymbol{x_1, x_2, \dots x_n}$
$\boldsymbol{x_n}$ be the composed of $\boldsymbol{m}$ variables $\boldsymbol{x_{n1}, x_{n2}, \dots x_{nm}}$
$\boldsymbol{x_{nm}}$ be the class of each input data training
$\boldsymbol{W}$ be a $p \times q$ grid of units $\boldsymbol{w_{ij}}$ where i and j are their coordinates on that grid

1 Repeat
2 For $k = 1$ to n
3 For all $\boldsymbol{w_{ij} \in W}$, calculate $d_{ij} = \|\boldsymbol{x_k - x_{ij}}\|$
4 Select the unit that minimizes d_{ij} as the winner $\boldsymbol{w_{winner}}$
5 If $\boldsymbol{w_{winner}}$ and $\boldsymbol{x_k}$ belong to the same class
6 Update each unit
$\boldsymbol{w_{ij} \in W: w_{ij} = w_{ij} + \alpha h(w_{winner}, w_{ij}, r)\|x_k - w_{ij}\|}$
7 else $\boldsymbol{w_{winner}}$ and $\boldsymbol{x_k}$ belong to different classes
$\boldsymbol{w_{ij} \in W: w_{ij} = w_{ij} - \alpha h(w_{winner}, w_{ij}, r)\|x_k - w_{ij}\|}$
8 Decrease the value of α and r
9 Until α reaches 0

The OLVQ1 is similar to the LVQ1 but in this situation, each unit has its own learning rate. The LVQ2 algorithm uses the same classification decision than the LVQ1. However, the two best-matching units (BMU) are selected and updated if one belongs to the same class the input data pattern and the other does not. Finally the LVQ3 is similar to LVQ2, but in this case the two BMU are only updated if both share the same classification with the input data pattern. In this case, this adaptation uses a given value (ε), to adjust the global learning rate.

2.3. SOM & georeferenced data

SOM is widely used in many different fields. GIScience is no exception, and SOM have been used in many different GIS problems. A recent book named *Applications of Different Self-Organizing Map Variants to Geographical Information Science Problems* (Agarwal and Skupin 2008) is proof of the growing interest of geospatial scientists in this type of methods. As we shall see, SOM has some characteristics that makes it a suitable and friendly method for geospatial analysis.

First, SOM is capable of dealing with different types of input data, such as continuous, categorical or binary data. This is an important feature since GIScience usually makes use of these types of variables to represent spatial phenomena. Some examples of geospatial continuous variables are altitude or precipitation, while soil type or County are categorical variables and the presence/absence of pesticides or endemic species can be represented as binary variables.

Secondly, SOM is an unsupervised method, requiring no prior knowledge about the nature and number of clusters. This means the analysis is data driven instead of assuming any data model. Additionally, SOM makes no assumption about statistical distributions of variables and presents good results when handling missing values and noisy data.

Third, SOM relies heavily on topological relations amongst data, which is a familiar concept in GIScience. If fact, SOM can and is used to spatialize (*i.e.* to represent in space) aspatial variables. Unlike other clustering and data analysis techniques, such as *k*-means (MacQueen 1967) or hierarchical classification trees (Sokal and Sneath 1963), SOM represents the data in a space where topological relations and partial orderings are present. The interpretation of such output space bears many resemblances with the interpretation of geographic space, which is a familiar task to any geospatial scientist.

This third characteristic of SOM has a strong impact on the type of analysis required to evaluate the SOM results. SOM comprehends a set of analysis and visualisation tools, such as U-matrices, component planes and hit maps that are at the same time very useful and familiar to the geospatial scientists. SOM typically represents multidimensional data through two or three dimensions in the output space. This

characteristic fits well the geospatial scientists' skills, used to dealing with geographical space. Additionally, due to this dimensionality reduction, many GIS tools that could not be used with the original variable, are suitable to analyse the SOM results. An example is presented in (Fincke, Lobo *et al.* 2008) where several GIS common surface analysis tools are used in the evaluation of SOM results.

Fourth, SOM preserves in the output space, as far as possible, data's topological relations from the input space. This is an important feature when dealing with geospatial data since the topological relations that exist in geographic space should have an impact on the analysis and thus on the output space of the SOM. This connection between topological relations in geographic space and topological relations in attribute space is expressed by Tobler's first law of Geography (Tobler 1970), or perhaps even better with the First Law of Cognitive Geography which states that ''*People think that closer things are more similar*'' (Montello, Fabrikant *et al.* 2003).

2.3.1. Survey of SOM applied to the GIScience

In the field of GIScience, SOM has shown to be an interesting method dealing with many problems from different areas, such as:

- cartographic generalization (Jiang and Harrie 2003; Jiang and Harrie 2004; Jiang and Harrie 2004; Allouche and Moulin 2005; Sester 2005; Sester 2008),
- analysis of patterns and relations applied to agricultural data (Koua and Kraak 2004),
- health and census data analysis (Koua and Kraak 2004; Oyana, Boppidi *et al.* 2005),
- distribution of species habitat (Céréghino, Santoul *et al.* 2005; Gevrey, Worner *et al.* 2006),
- analysis of soil and geochemical datasets (Penn 2005; Fraser 2006; Mele and Crowley 2008),
- distribution of biophysical characterization data (Hosokawa and Hoshi 2001; Ferentinou and Sakellariou 2007),
- crime data analysis (Chen, Atabakhsh *et al.* 2003),
- urban growth patterns (Franzini 2001),

- location/allocation problems (Lozano, Guerrero *et al.* 1998; Hsieh and Tien 2004; Lobo and Bação 2005; Gomes, Ribeiro *et al.* 2007),
- zone design (Henriques and Bação 2004),
- mapping concepts using cartographic approach (Skupin 2002; Leitich and Topf 2007; Thill, Jr *et al.* 2008)
- map projections and cartograms (Skupin 2003; Henriques 2006)
- remote sensing (Wan and Fraser 1993; Wan and Fraser 1994; Ji 2000; Lee and Lathrop 2006; Li and Eastman 2006; Yun and Uchimura 2007; Ehsani and Quiel 2008)
- climate analysis (Hewitson 2008; Kropp and Schellnhuber 2008)

The SOM is generally well suited to handle two main problems: visualisation and clustering of high-dimensional datasets. Still some methods based on the SOM, already described in section 2.2.8, can be applied in the task of classification. Considering this, and the applications we reviewed, we propose the taxonomy shown in Figure 21 for the applications of SOM to georeferenced data.

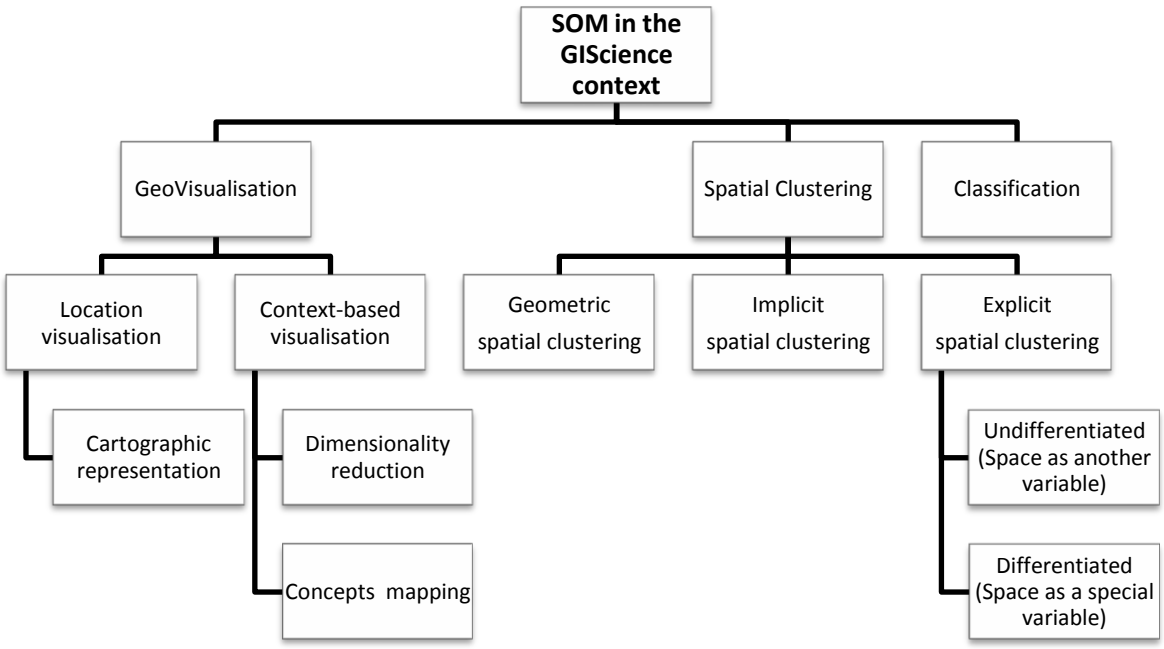


Figure 21 – Taxonomy for Self-Organizing Maps applications in GIScience

The top level (root) of the taxonomy includes all methods using SOM to deal with geospatial data. The second level of the taxonomy classifies applications according to the type of task performed: geovisualisation, spatial clustering and classification.

Geovisualisation deals with methods supporting geospatial data analysis through the use of visualisation techniques. Two sub-groups are considered in this category: 1) methods using only location of the objects as input data; 2) methods using all the attributes (components or context) of the objects as input data.

In the second class, concerning spatial clustering, the focus is to create groups of similar geospatial objects. Three sub-groups are considered in this category: 1) geometric spatial clustering deals only with spatial components, clustering data objects that are close to each other in geographic space; 2) implicit spatial clustering uses only aspatial components clustering data objects that are similar in attribute space; 3) explicit spatial clustering, that clusters objects taking into account all components, thus balancing geographical space and aspatial characteristics.

The third class of the taxonomy includes applications where SOM is used as a supervised classification tool.

In the remaining of this section, we present a detailed survey of these applications of SOM classified according to the proposed taxonomy.

2.3.1.1. Geovisualisation

The SOM, as already described, is a very suitable tool for data visualisation. The increase of dimensionality of geospatial datasets faces a new visualisation challenge in common GIS. GIS makes use of several tools for presenting data, such as maps, statistical plots and tables with matrix data. Maps are the most used and effective in presenting spatial distributions while statistical plots and tables are used as a mean to present statistical and auxiliary data.

There are four major types of maps: choropleth maps, dot maps, proportional symbol maps and isopleths maps (Robinson, Morrison *et al.* 1984). Choropleth maps are representations in which each spatial unit is filled with a uniform colour or pattern and are appropriate for data that have been scaled or normalized in some way. Dot maps are made by placing a dot or some other symbol in the approximate location of one or more instances of the variable being mapped producing a visual representation of density. Proportional symbol maps scale icons according to the data they represent. Proportional

symbol maps are not dependent on the size of the spatial unit associated with their attributes. Finally, isopleths maps differ from choropleth maps in that the data is not aggregated to a pre-defined unit but to the data based unit. This means that new spatial units are built according to data attributes. Typically, GIS have the capability, depending on the experience and creativity of the user, to present several variables on the same map. However, the increase in the number of variables usually causes a decrease in the map readability.

Depending on the type of variables used in the SOM, two groups of methods may be considered. The first one uses only the geographical location of the spatial objects (location visualisation) while the second one uses all the variables of the geospatial dataset (context-based visualisation).

A. Location visualisation

This family of methods uses SOM to automate some tasks in the cartographic representation field. Some examples include the process of map-making and creation of new geospatial projections.

One important and still difficult phase in the process of map-making is cartographic generalisation (Robinson, Morrison *et al.* 1984; Buttenfield and McMaster 1991; McMaster and Shea 1992). Cartographic generalization is the process of simplification and reduction of cartographic elements according to the needs and objectives of map users. Several approaches to this problem using the SOM have been proposed.

One example is the work of Sester (2005; 2008) using the SOM in the process of typification. Typification, which is a cartographic generalization process, is the reduction of the number of spatial objects while preserving the spatial structure. In this approach, SOM estimates the density of the cartographic elements at a particular map scale. The spatial location of the cartographic elements is used as input data. The new cartographic elements are obtained from the SOM's final weights and are therefore fitted to the smaller map scale. The number of units used in the SOM controls the degree of abstraction, *i.e.* the reduction in the number of cartographic elements. This level of abstraction relies on the desired change of map scale.

Allouche and Moulin (2005) also present a SOM based method for typification. In this work, the amount of data represented is reduced by replacing the objects of each region by a polygon that preserves their shape. Using the objects' position as input data, the SOM is used to map high-density areas. Then, two filters are applied to the SOM weights to detect these regions. Finally, using the objects belonging to these regions, polygons are drawn applying a refined convex-hull method.

In the work presented in (Jiang and Harrie 2003; Jiang and Harrie 2004; Jiang and Harrie 2004) the authors make use of the SOM in the selection of cartographic objects. More specifically, the approach presented uses topological, geometric and semantic attributes from street objects as input to an SOM. From the trained SOM it is possible to obtain clusters of streets. As expected, each of the clusters is composed of similar streets, which will, based on user expertise, be selected depending on the final map scale. The authors also propose the application of their method for selection or generalization of other spatial objects based on multiple attributes[4].

SOM can also be used to produce new projections of spatial data (Skupin 2003). In this paper, Skupin uses a world map with an equal area projection to visualise the SOM distortions. In fact, this representation allows a better understanding of some of the already known distortion properties of SOM using a familiar concept to geospatial scientists. Starting from a point grid created on top of the world countries, SOM was applied and the *countries* layer was drawn using this *new projection*. Observing the world configuration in this new projection allowed Skupin to draw some conclusions. First, SOM preserves topological relations in detriment of distances. The *movement* suffered by the continents to areas mainly occupied by ocean illustrates this fact. Second, a stronger space contraction occurs on the edge of the SOM. This property, also known as border effect (Wu and Takatsuka 2006) is well known and causes SOM units close to edges to be drawn to the centre of the SOM, and present a higher quantization error. A third feature observed is that most of the countries and continents maintained continuity. Last, although not explicit in the paper, the SOM magnification effect (Cottrell, Fort *et al.* 1998; Claussen 2003) is also visible in the *countries* layer. This

[4] This approach also contains elements of other classes of the proposed taxonomy namely explicit spatial clustering.

effect makes the distribution of the units more than proportional in lower density areas. Thus, it is possible to see that the smaller countries are oversized in the new projection.

B. Context based visualisation

In this group of methods, SOM is used to simplify the visualisation of multidimensional geospatial data. It does so by reducing data dimensionality. In this abstraction process, the high-dimensional aspatial data is mapped to two or three dimensions, making its visualisation easier. By associating the SOM's output space (the space where data dimensionality has been decreased) with the geographical space, the user can spatially visualise a set of variables.

An example of this type of use is presented in (Penn 2005) where two high-dimensional datasets concerning geochemical and electromagnetic material measurements are visualised using the SOM. In this proposal, the author uses a one-dimensional SOM to give an insight on existing relationships and simultaneously produce a statistical model of the dataset.

Koua and Kraak (2004) used the SOM to represent and visualise underlying dynamics between rice and maize production in African countries over space and time. Several different analyses, using U-matrices, component planes and trajectories over U-matrices are made, focusing on the effective use of visualisation to facilitate knowledge extraction. The use of these techniques allows the exploration and visual detection of geospatial-temporal trends and patterns.

Another example is presented in (Skupin and Hagelman 2005) where some visualisation techniques are used to present temporal demographic changes. Skupin uses standard GIS software to manipulate and present the SOM results and create a spatialization of multi-temporal and multidimensional changes. Based on SOM's results, behaviour trajectories are constructed for each county. Some manipulation of these trajectories is proposed, such as clustering them to find counties with similar changes over time, and mapping other variables onto these trajectories.

Still within this group of geovisualisation, SOM has been used for spatialization of multidimensional data, *i.e.*, to use the cartographic approach to map non-geographical concepts. Examples of this utilization are presented in (Skupin 2002; Leitich and Topf

2007). Skupin (2002) applies the SOM to perform text mining using the keywords and abstracts of the Association of American Geographers' (AAG) 1999 conference. The author then labels the topological ordered output space of SOM with the original keywords. To avoid several labels on the same unit, Skupin creates several *sub-units* in the SOM's output space, positioned near the original units, so that each of the new ones has a single label. Thiessen polygons are created using these new positions and hierarchical clustering is performed using the aspatial attributes. The application of cartographic design principles using base polygons and hierarchical clustering will produce a map with different levels of detail and abstraction of reality.

2.3.1.2. Spatial Clustering

SOM is very frequently used for clustering. In the spatial context of GIScience, clustering will usually involve spatial clustering (Han 2005). Spatial clustering can be defined as the partition of spatial objects into groups so that objects within a cluster are as similar as possible. This wide definition can encompass different approaches. To clarify different objectives of spatial clustering, we propose in our taxonomy a categorization into three types of spatial clustering:

- Geometric spatial clustering
- Implicit spatial clustering
- Explicit spatial clustering

Geometric spatial clustering is concerned only with the spatial attributes of phenomena. Therefore, only the two or three spatial dimensions (relative to the *x*, *y*, and *z* coordinates) are used in this type of clustering. Because clustering is based only on the objects' spatial location, the process can be labelled as geometric clustering.

Implicit spatial clustering includes methods that only use aspatial variables. However, due to the spatial autocorrelation phenomenon, visualisation of the SOM results can be meaningfully be presented in a spatial context. In fact, clustering of spatially distributed objects, even if performed using aspatial variables, will typically present interesting spatial patterns and relations. One can say that the exclusion of the spatial coordinates in the SOM training phase is generally compensated by the spatial autocorrelation of the variables used.

Finally, in the last category we consider explicit spatial clustering. In this group, as the name suggests, SOM based methods use the spatial components explicitly as input data, together with the aspatial variables. This category can still be divided in two groups: 1) Undifferentiated explicit spatial clustering methods which deal with spatial components the same way as other variables or; 2) Differentiated explicit spatial clustering methods, which use variants of SOM that take into account the special properties exhibited by spatial data, treating spatial variables differently from the other ones.

A. Examples of geometric spatial clustering

Some applications performing geometric spatial clustering are: 1) the SOM based cartographic generalization methods (Allouche and Moulin 2005; Sester 2005; Sester 2008),; 2) detection of cartographic elements (Yun and Uchimura 2007); 3) the zone design methods (Henriques and Bação 2004); 3) location/allocation methods proposed solutions (Lozano, Guerrero *et al.* 1998; Hsieh and Tien 2004; Lobo and Bação 2005; Gomes, Ribeiro *et al.* 2007) and; 4) methods to solve the travelling salesperson problem (Lobo and Bação 2005).

In section 2.3.1.1 we already presented the cartographic generalization approaches (Allouche and Moulin 2005; Sester 2005; Sester 2008) as methods to facilitate the visualisation of geospatial data. However, in these cases, cartographic generalization is obtained through clustering of spatial objects. These clusters are based only on the spatial components and produce new spatial objects from the units' weights. Thus, these applications are both examples of geovisualisation and geometric spatial clustering.

Yun and Uchimura (2007) propose a semi-automatic method for road network extraction from high-resolution satellite images. These images are first pre-processed with fuzzy logic techniques excluding all the areas that are not candidates for a road. An SOM is then applied using the spatial coordinates of these candidate areas, followed by a road tracking method.

Henriques and Bação (2004) proposed a zone design solution using the SOM. In this approach, regions are clustered forming larger aggregated regions. An example using the Portuguese Municipalities is used to create larger equally populated Portuguese

regions. Using the population variable, random points are created inside each Municipal spatial outline. The number of points created is proportional to the population variable, making the result similar to a dot map representation. The points' *x* and *y* coordinates are used as input data to the SOM. The SOM's units are therefore the cluster centroids and Municipalities are grouped according to their distance to these centroids. The result is the creation of larger demographic units, where the population is, as far as possible, equally distributed among them.

Location allocation problems can also be viewed as clustering the customers into groups supplied by the supply centres to achieve a minimum distance criterion (Hsieh and Tien 2004). Lozano *et.al.* (1998) propose a solution to locate the supply centres for 49 demand centres, corresponding to major towns in Spain and Portugal. The two spatial coordinates of the demand centres were used as input patterns, and the frequency with which they are presented is proportional to the demand: $p(Coord_i) = D_i / \sum_{k=1}^{N} D_k$ where $Coord_i$ are the coordinates of centre *i*, D_i is the demand for centre *i,* and *N* equals the number of demand centres. The final SOM's units will then be used in a refinement process to select the best supply centres locations to serve a set of customers. A similar approach was proposed in (Hsieh and Tien 2004) but in this case the training set is composed by each demand centre location multiplied by its demand value, making this a faster method than the proposed in (Lozano, Guerrero *et al.* 1998). Also, Gomes *et.al.* (2007) proposed an SOM based solution for the location of CCA (chromated copper arsenate)-treated wood waste remediation units. Using a GIS model, places where treated wood was in use and potentially would become a waste were identified. Knowing the set of discrete locations with demand for this problem, spatial clustering was applied to propose the optimal or near optimal locations for these remediation units.

Related with this issue is the travelling salesperson problem (Cook 1983). In (Lobo and Bação 2005) SOM is used to obtain optimized paths for patrol vessels in non-military missions. The coordinates of reported occurrences of fishing violations and search and rescue (SAR) operations, near the Portuguese coastline are used as input patterns in the SOM. The use of a one-dimension SOM with a large number of units provided the optimal or near optimal route that a patrol should take to visit all the occurrence locations.

B. Examples of implicit spatial clustering

Implicit spatial clustering is the most common use of SOM in geospatial data. Clustering and exploratory data analysis using different sources of data are proposed in the literature. Examples include agricultural data (Koua and Kraak 2004), health and demographic data (Openshaw, M. *et al.* 1995; Hatzichristos 2004; Koua and Kraak 2004; Spielman and Thill 2008), habitats data (Céréghino, Santoul *et al.* 2005; Gevrey, Worner *et al.* 2006), soil and geochemical data (Penn 2005; Fraser 2006; Mele and Crowley 2008), biophysical characterization data (Hosokawa and Hoshi 2001; Ferentinou and Sakellariou 2007), crime data (Chen, Atabakhsh *et al.* 2003), urban development data (Franzini 2001) and climate analysis (Hewitson 2008; Kropp and Schellnhuber 2008).

All this applications share the exploration of the SOM as a data mining tool, extracting behaviour patterns from high-dimensional geospatial and temporal datasets and presenting the results on both geographical and non-geographical spaces. Non-geographical spaces include representations such as component planes or U-matrices. In general, a great focus is given to the integration of multiple views allowing the brushing, linking, and zooming for exploratory analysis and clustering.

C. Examples of explicit spatial clustering

In this category, spatial coordinates are explicitly used in the training phase of SOM, together with aspatial variables. There are two ways to include spatial components. The first option, which we named *undifferentiated*, uses spatial components as any other variable. Typically, in these cases, the dimensionality of the aspatial space is not very high. This is because, if all variables have the same weight in the final result, a high number of aspatial variables will make spatial components (which will usually be only two) less relevant. Due to this decrease of relevance, the results when including spatial variables will be similar to those obtained without them. As an alternative, the user can define the weight for each variable in the process. However, finding the optimal weights can be troublesome and the different results created from different sets of weights could be difficult to interpret.

An example of this type of use of the SOM is presented in (Oyana, Boppidi *et al.* 2005). Their work combines spatial and health data as input data to the SOM. Variables used in the example include the patients' coordinates (*x* and *y*), a case control variable (whether the patient has asthma or gastroenteritis), a binary variable indicating whether the patient is near a highway, and a binary variable indicating if the patient is close to a pollution source. Jiang (2004) also used the SOM for filtering laser-scanning data and defining new spatial objects. As input data, he used laser scanned point clouds defined by four dimensions (*xyz* coordinates and the return intensity). With the help of the U-matrix, the detection of clusters is possible, which in this case represent the different spatial objects detected. Ehsani and Quiel (2008) used the SOM to cluster a Landsat ETM+ image to detect the yardangs (ridge), corridors (valley) and planar areas in the Iranian Lut desert. Geomorphometric parameters such as slope steepness, minimum, maximum curvature and cross-sectional curvatures were used as input to the SOM.

Another example combining geospatial and aspatial data is the Geo-enforced SOM (Bação, Lobo *et al.* 2005). The method is focused on the pre-processing phase weighting the geographic coordinates in order to make them as important as the whole set of demographic variables. In the example presented, each coordinate variable was multiplied by a factor to even the importance of the two geographical variables with the aspatial variables.

The second way of performing explicit spatial clustering, which we named *differentiated*, treats differently spatial and aspatial variables. Treating spatial variables in a different way will require changes to the standard SOM . This may be done by changing the SOM algorithm, or some of its parameterization. Some of these changes are just minor adjustments to the basic SOM algorithm, while others can be quite substantial.

To include spatial objects in the SOM, Babu (1997) proposes a new similarity measure. While the most common is to use the Euclidian distance, other measures of distance and similarity are available (Sneath and Sokal 1973; Kohonen 2001; Lourenco, Lobo *et al.* 2004). The proposed approach, called SOM for spatial data (SOMSD), uses a new distance measure between a point feature (units in the SOM) and the minimum bounding rectangle of the spatial data (when represented by polygons). While in the paper the author presents several two or three-dimensional examples, the method is suited for a higher number of dimensions.

Chandrasekaran *et.al.* propose a spatial temporal feature map (STFM) (Chandrasekaran, Palaniswami *et al.* 1993; Chandrasekaran and Palaniswami 1995). Contrary to the standard procedure, in the STFM only a subset of units can be chosen as the best matching unit. This subset of units that takes part in the SOM's voting phase is defined by spatial and a time-dependent functions. The spatial grating function defines a spatial area of influence of each unit. The time dependent function depends on the past activations and determines the output of the units. Using these two concepts, a *"selective competition is achieved by grating the n-dimensional feature space by a gating function with a time varying spatial frequency and then setting a criterion for the neurons in the feature map to compete"* (Chandrasekaran and Palaniswami 1995).

Another proposal from Bação *et.al.* (2005) makes use of the hierarchical SOM to combine spatial and aspatial components. The idea behind hierarchical SOM for spatial data is that instead of a single SOM that receives all the features of the data, the vector of the input pattern is separated into several parts and each of these parts is input to an SOM. The outputs of these SOMs are then used as input to a higher level SOM along with the geographical coordinates. This work is expanded in chapter 5, including other ways to use hierarchical SOM.

Based on the Hypermap (Kohonen 1991) and Kangas (Kangas 1992) SOM's architectures, Lobo *et.al.* (2004) proposed the spatial-Kangas SOM. In the Hypermap architecture, the input pattern vector is formed by two subsets of variables. The first subset contains the data while the second subset contains the context. Thus, the search for the best-matching unit is a twofold step. Initially the *context domain* is selected by searching for all the units that are within a given distance of the context. In the second phase, the best-matching unit is searched within the selected context. The Kangas architecture shares the same principle, but instead of getting the context from each input pattern vector, the context derives from the best-matching unit obtained in the previous iteration. Therefore, the selection of the best-matching unit considers only the units that fall in the neighbourhood of the best matching unit obtained in the previous iteration.

Spatial-Kangas map (Lobo, Bação *et al.* 2004) is the adaptation of these two methods using an aspatial pattern to be clustered in a spatial context. First, one creates a subset of units using a geographical distance criterion. In the second step, those units are compared with the input pattern to select the best matching unit. Spatial-Kangas map

forces units close in the output space to be close in the input space too, thus creating clusters of areas that will be geographically close (Lobo, Bação *et al.* 2004). Following this line of thought, Bação *et.al.* (2005) proposed the GeoSOM as an adaptation of SOM to consider the spatial nature of data. In GeoSOM, the search for the best matching unit (BMU) has two phases. The first phase settles the geographical neighbourhood where it is admissible to search for the BMU, and the second phase performs the final search using the other components. A parameter *k* controls the search neighbourhood defined in the output space. Using *k*=0 will necessarily select as BMU the unit geographically closer. The same result may be obtained by training a standard SOM with only the geographical locations, and then using each unit as a low pass filter (*i.e.* a sort of average) of the non-geographic features. As *k* (the geographic tolerance) increases, the unit locations will no longer be quasi-proportional to the locations of the training patterns, and the *equivalent filter* functions of the units will become more and more skewed, eventually ceasing to be useful as models. This GeoSOM architecture is expanded, implemented, and tested on real data in chapter 4.

2.3.1.3. Classification

Although SOM is an unsupervised learning method, it can be used as a classifier using labelled maps or combined with other methods to perform classification tasks. In fact, the Learning Vector Quantization (LVQ) classification algorithm proposed by (Kohonen 2001) is based on the SOM algorithm. A family of methods including the LVQ1, OLVQ1, LVQ2 and LVQ3 are described in section 2.2.8.

We do not feel that using a SOM in classification tasks is particularly interesting, since better classifications can usually be achieved with other supervised methods. In this thesis, we do not explore further this type of use of SOM. However, we must acknowledge that SOMs are very frequently used for this task.

Concerning geospatial data, most of the classification tasks using SOM and LVQ are applied to the Remote Sensing field. Some of the examples are the works of (Wan and Fraser 1993; Wan and Fraser 1994; Mather, Tso *et al.* 1998; Ji 2000; Niang, Gross *et al.* 2003; Richardson, Risien *et al.* 2003; Villmann, Merenyi *et al.* 2003; Lee and Lathrop 2006; Li and Eastman 2006). For a further review on SOM applications performing geospatial classification the reader is recommend to the survey made in (Lobo 2009).

Common to all these proposals is the combination of the SOM and the LVQ methods to which Kohonen called LVQ-SOM method (Kohonen 2001). Generally, in a first phase the SOM clusters the images into regions of pixels using the spectral band values and neighbourhood data. In a second phase, the LVQ algorithm uses classified input patterns to allow a better definition between the classes. Finally, for each input vector, the distances to all the units are calculated, and the class of the input is given by its BMU.

2.3.2. Comparison of methods proposed in the literature

Table 1 provides a summary on the characteristics of some existing and relevant approaches.

Table 1 - Comparison table of SOM-based analysis in the GISc context

Reference	1st level of taxonomy			SOM algorithm		Variables used in the learning phase			Application field
	Geovisualisation	Spatial Clustering	Classification	Standard	Variant	Only spatial	Only aspatial	Both spatial and aspatial	
(Jiang and Harrie 2003; Jiang and Harrie 2004)	X	X		X		X			Cartographic representation
(Jiang 2004)	X	X		X		X			
(Allouche and Moulin 2005)	X	X		X		X			
(Sester 2005; Sester 2008)	X	X		X		X			
(Koua and Kraak 2004)		X		X			X		Analysis of agricultural data
(Koua and Kraak 2004)		X		X			X		Health and census data analysis
(Hatzichristos 2004)		X			X				
(Oyana, Boppidi *et al.* 2005)		X		X				X	
(Céréghino, Santoul *et al.* 2005)		X		X			X		Patterning species habitat
(Gevrey, Worner *et al.* 2006)		X		X			X		
(Penn 2005)		X		X			X		Analysis of soil and geochemical datasets
(Fraser 2006)		X		X			X		

Reference	1st level of taxonomy			SOM algorithm		Variables used in the learning phase			Application field
	Geovisualisation	Spatial Clustering	Classification	Standard	Variant	Only spatial	Only aspatial	Both spatial and aspatial	
(Mele and Crowley 2008)		X		X			X		
(Hosokawa and Hoshi 2001)		X		X			X		Patterns in biophysical characterization data
(Ferentinou and Sakellariou 2007)		X		X			X		
(Chen, Atabakhsh *et al.* 2003)		X		X			X		Crime data analysis
(Franzini 2001)		X		X			X		Urban growth patterns
(Lozano, Guerrero *et al.* 1998)		X		X		X			Location/allocation problems
(Hsieh and Tien 2004)		X		X		X			
(Lobo and Bação 2005)		X		X		X			
(Gomes, Ribeiro *et al.* 2007)		X		X		X			
(Henriques and Bação 2004)		X		X		X			Zone design
(Skupin 2002)	X	X		X			X		Mapping concepts using cartographic approach
(Leitich and Topf 2007)	X	X		X			X		
(Thill, Jr *et al.* 2008)	X	X		X			X		
(Skupin 2003)	X			X				X	Map projections and cartograms
(Henriques 2006)	X			X				X	
(Wan and Fraser 1993; Wan and Fraser 1994)			X		X		X		Remote sensing
(Ji 2000)			X	X			X		
(Lee and Lathrop 2006)			X	X			X		
(Li and Eastman 2006)			X	X			X		
(Yun and Uchimura 2007)		X		X		X			
(Ehsani and Quiel 2008)	X	X		X			X		
(Hewitson 2008)		X		X			X		Climate analysis
(Kropp and Schellnhuber 2008)		X		X			X		

2.4. Discussion

In this chapter, an in depth review of the SOM is made. Several properties of SOM make it a good method to analyse geospatial and high-dimensional datasets. The number of papers using this method for several different applications in the GISc field proves its suitability. A survey of the most important proposals is made, along with a taxonomy for the use of SOM with geospatial data.

3. Building cartograms using the SOM

The basic idea of a cartogram (Raisz 1938) is to distort a geographical map by substituting the geographic area of a region by some other variable of interest. The objective is to rescale each region according to the value of the variable of interest while keeping the map, as much as possible, recognizable. There are several algorithms for building cartograms. None of these methods has proved to be universally better than any other, since the trade-offs made to get the correct distortion vary. This chapter begins with the definition of cartograms and reviews the most popular algorithms for building cartograms.

We then present a thorough description of a new algorithm (Carto-SOM) based on the use of Self-Organizing Maps. The proposed method is widely available, is easy to carry out, and yet has several appealing properties, such as easy parallelization, making up a good tool for geographic data presentation and analysis.

To test the Carto-SOM method we use three different datasets, and two other algorithms to build cartograms are used for benchmarking. Finally, we perform a quantitative evaluation of the cartograms produced by the Carto-SOM and the benchmark

algorithms. We conclude that it is competitive and, in some circumstances, can perform better than existing algorithms.

3.1. Introduction

Maps have always been an important part of human life, forming one of the oldest means of human communication. Today, maps continue to play an important role in our society describing and communicating many geographic (and other) facts. They may be regarded as a means for converting large amounts of data into information that can be easily understood by human beings. The arrival of personal computers and the consequent increase in geoprocessing capacities had an impact not only on map availability but also on map-making tools. Today almost everybody has access to tools, which make maps at the push of a button.

Cartograms are a type of map or a map variant. The difference between most traditional maps and a cartogram is the variable that defines the size of the regions. In most traditional maps this variable is the geographic area of the regions (or a good approximation of it), while cartograms may use any other georeferenced variable. In the context of the famous framework provided by the seven visual variables of Bertin (1983), the cartogram uses the visual variable *size* to encode a data variable.

According to Tobler (2004) there are two generic types of cartograms. The first form of cartogram distorts space according to a non-geographic metric, such as cost or time (Tobler 1961). The second form of cartogram distorts space according to a georeferenced variable. Depending on the cartogram regions' continuity cartograms are classified as non-continuous (Olson 1976) or as continuous. In this thesis, we will focus only on continuous cartograms, and for simplicity will use the word cartogram to refer to these.

"*Cartograms are purposely distorted maps that highlight a variable distribution by changing the area of objects on the map*" (Changming and Lin 1999). "*Area cartograms are deliberate exaggerations of a map according to some external geography–related variable that communicates information about regions through their spatial dimensions*" (Dougenik, Chrisman *et al.* 1985). "*These dimensions have no correspondence to the real world but are a representation of one variable other than the area*" (Kocmoud 1997).

In a population cartogram, for example, the sizes of regions are proportional to the number of inhabitants. While population is the most commonly used, we can use any social, economic, demographic or geographical variable to build a cartogram.

3.1.1. Problem definition

As Keim *et.al.* (2005) point out, cartogram generation is a map deformation problem. The inputs of the problem are:

- a polygon map composed of a set of regions, each with an initial area given by the true geographical area (each polygon matches a region);
- a target value for the final area of each one of the polygons (regions), representing the variable of interest in that region.

The goal of the map distortion is to approximate, as much as possible, the areas of the regions to the desired target. The cartogram problem may be formally described as follows:

Let $M = \{R_0, R_1, \ldots, R_n\}$ be a map M consisting on n regions R_i, each of which forms a polygon.
Let $T(M)$ be the contiguity matrix of M.
Let each region R_i have an area of a_i (area of its polygon) and a "value of interest" v_i (usually population, average income, or some other variable).
Produce a cartogram map $C = \{R'_0, R'_1, \ldots R'_n\}$ consisting of n regions R'_i, each of which forms a polygon with area $a'_i = kv_i$, with $k = {\sum a}/{\sum v}$, and $T(C) = T(M)$.

There is still another goal, loosely described as *shape similarity*. Let S be the shape of M and S' the shape of C. The objective is that $S \cong S'$. *Shape similarity* is an elusive concept that translates the ability of a reader to recognize C as an instance of M. This property is difficult to measure and difficult to define rigorously. The ability to recognize C is not only dependent on the ability to preserve the shape of each R_i but also on preserving certain landmark points. For instance recognizing a certain cartogram as a cartogram of the United States is much more dependent on preserving the shape of Florida than on preserving the shape of North Dakota.

There are many possible cartograms C that achieve the desired goal, but most mappings from M to C are the result of an iterative process, and only asymptotically get the desired result (or at least some approximation to it).

3.2. Methods for building cartograms

Several computer algorithms have been developed to build continuous area cartograms and, in this section, we will briefly review some of the most important ones. The reader interested in an extensive and thorough review of the work done in the last 35 years in cartograms is referred to a recently published paper by Tobler (2004). In this chapter we briefly review 8 algorithms that represent different approaches to the problem and are the most important and effective ones: *Rubber-map cartogram* (Tobler 1973), *Contiguous Area Cartogram* (Dougenik, Chrisman *et al.* 1985), *Pseudo-cartogram* (Tobler 1986), *Line Integral Method* (Gusein-Zade and Tikunov 1993), *Constraint-based Continuous Area Cartogram* (House and Kocmoud 1998), *Medial-axes-based cartogram* (Keim, North *et al.* 2002; Keim, North *et al.* 2004; Keim, North *et al.* 2005), *Rectangular Map Approximations* (Heilmann, Keim *et al.* 2004) and *Diffusion cartograms* (Gastner and Newman 2004).

The Rubber-map cartogram (Tobler 1973) was one of the first methods created for population cartograms. This method distorts a map as if it were a rubber surface. Tobler's method initially divides the map into a regular grid, computing a density value for each grid cell. On each cell vertex a displacement direction is computed to minimize the *density error* of its four adjacent cells. This displacement continues until no improvement can be made. Since there is a one-to-one mapping between the initial and final grids a double bivariate interpolation is used to project the map vertices to and from the final grid.

One of the most famous algorithms to build continuous cartograms is Dougenik's Contiguous Area Cartogram (Dougenik, Chrisman *et al.* 1985). This algorithm is based on Tobler's rubber map and uses a model where forces are exerted from each polygon centroid, acting on its boundary. In this method each polygon centroid applies a force *F* to its vertices. This force will move vertices away if *F* is a positive value and will bring them closer if *F* is negative. The intensity of the force exerted on each polygon is determined by the difference between the current and the desired polygon area. The

current area is the area of the polygon in each iteration, while the desired area is the distorted area in the final cartogram. In this method, some overlapping might, occasionally, occur due to complexity of the polygon shapes. However, it presents an improvement in speed compared to Tobler's method (Tobler 2004). Population cartograms of the USA, built with the several algorithms reviewed, are shown in Figure 22.

Another cartogram construction algorithm, introduced in 1986 by Tobler, is the pseudo-cartogram (Tobler 1986). This method is similar to the rubber-map method and seeks to reduce the area error by adjusting the lines of latitude and longitude on the map. Tobler's pseudo-cartogram algorithm starts with a grid superimposed on the map, after which the horizontal and vertical lines of the grid move, compressing or expanding the map to get equal density estimation. Just as the rubber-map method, this algorithm suffers from a high dependency on the coordinate system.

Zade and Tikunov proposed the Line Integral Method which applies radial transformations such that the density of the final map is uniform (Gusein-Zade and Tikunov 1993). The algorithm initially divides the map into a grid of cells. Radial transformations are applied to each cell making its density uniform, and changing the shape of the overall map.

House and Kocmoud (1998) proposed the Constraint-based Continuous Area Cartogram. According to this method cartogram creation is seen as a constrained optimization problem. The map is a connected collection of polygons and an area function decides each region's desired area, according to the total map area and variable of interest. The goal, as in any cartogram, is to rescale each object keeping map recognition. The search space is composed by all the maps that keep topology and respect the area function values of each region. Within these possible solutions, the best map, which is the most recognizable one, is selected using optimization heuristics

The medial-axes-based cartogram (Keim, North *et al.* 2002; Keim, North *et al.* 2004; Keim, North *et al.* 2005) uses the medial axes of an object to build a cartogram. The medial axis is a simplification of the shape of an object and can be obtained by connecting the centres of the maximum circles inscribed in the object. The regions are then expanded or compressed along lines perpendicular to the medial axis (cutting

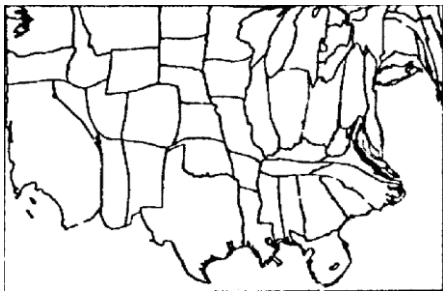
lines). This method presents as major advantage its processing time, which the authors claim to be several times lower than previous methods.

Another cartogram algorithm is the Rectangular Map Approximation (Heilmann, Keim *et al.* 2004), which represents each region as a rectangle. Each rectangle area is proportional to some variable. To best recognize the cartogram information, the method places rectangles as much as possible in the original position and as close as possible to their neighbours. This method uses Genetic Algorithms (GA) to select the best cartogram using a given cost function which is defined using the region's area, shape, topology, relative position and the spaces without any region. Although this method produces cartograms with low area errors, it presents two major drawbacks: topology error is never null, *i.e.* there will always be some topological differences between the original and the cartogram regions, and cartogram readability is hampered due to the use of rectangles instead of the original shapes.

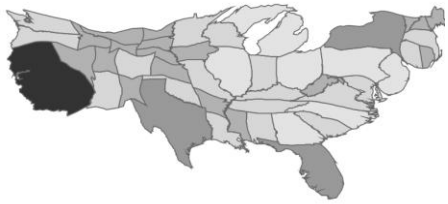
Finally, the diffusion cartograms method (Gastner and Newman 2004) produces cartograms where density variations cause diffusion from high-density areas into low-density ones. This method uses the linear process of diffusion from elementary physics to achieve a solution. It is a simple method, providing a tool to quickly calculate accurate cartograms. By changing the way that density is estimated, it also allows the user to switch between more detailed or more easily readable cartograms, thus increasing its applicability. One of its main advantages is that, using a continuous problem formulation, it can achieve accurate cartograms even for a high number of regions (more than 3000). On the other hand, it can be argued that cartograms with a large number of features (*i.e.* 3000 regions) are not as useful as smaller ones, due to the difficulty in recognizing the original regions.

From this brief review of the work done in developing algorithms for cartogram building there are three major conclusions that can be drawn: first, none of the existing methods is universally better than any other; second, most of these algorithms are computationally demanding; third, the potential user will encounter difficulties and challenges in the implementations of the algorithms. These conclusions suggest that one should examine the different options as they vary in computational efficiency, ease of implementation and performance. Thus, a new method that produces different and

competitive cartograms can be a useful addition to the set of algorithms already available.

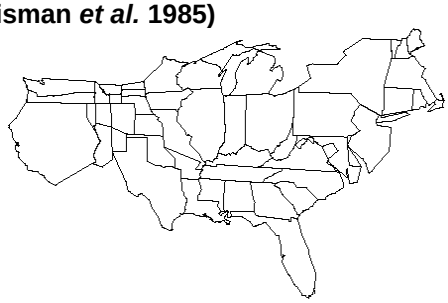


Rubber-map cartogram (Tobler 1973)



Contiguous Area Cartogram (Dougenik, Chrisman *et al.* 1985)

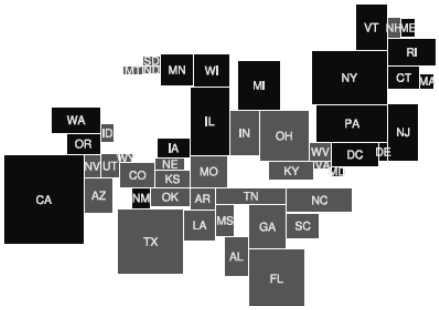
Pseudo-cartogram (Tobler 1986)



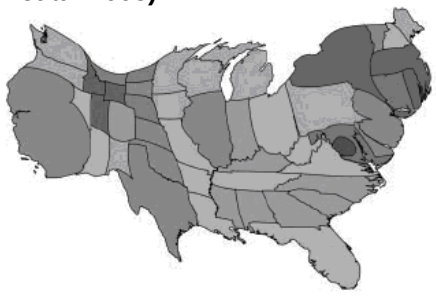
Constraint-based Continuous Area Cartogram (House and Kocmoud 1998)

Line Integral Method (Gusein-Zade and Tikunov 1993)

Medial-axes-based cartogram (Keim, North *et al.* 2002; Keim, North *et al.* 2004; Keim, North *et al.* 2005)



Rectangular Map Approximations (Heilmann, Keim *et al.* 2004)



Diffusion cartograms (Gastner and Newman 2004)

Figure 22 – Cartograms of USA population by state, using different cartogram building algorithms

3.2.1. Quantitative evaluation of cartograms

While it is subjective to compare the visual quality of different cartograms, it is easy to define a numerical value that characterizes how well the cartogram shares the available area between the different regions, according to the given variable of interest. Various such measures have been proposed. Keim *et.al.* (2004) proposes an area error function to determine the cartogram error in each region. This relative area error e_{rel}^{i} of a region R_i is given by:

$$e_{rel}^{i} = \frac{|a'_i - a_i^{current}|}{|a'_i + a_i^{current}|} \qquad \textbf{Equation VII}$$

Where a'_i is the desired area of region R_i in the cartogram (to create a perfect cartogram) and $a_i^{current}$ is the area of region R_i in the cartogram.

Other possible error measures could be used, such as the one proposed by Kocmoud (1997), or by Henriques (2006), but while having specific advantages and disadvantages, they do not produce significantly different results (Henriques 2006).

The error used in this chapter is Keim's, and may assume values in [0, 1[. For a perfect cartogram, the error is zero, since in that case each region occupies an area strictly proportional to the value of its variable of interest. The extreme but unattainable value 1 would mean that all the map space is used up by a region where the value of interest is zero, or vice versa.

3.2.2. Global cartogram error

The error measure given previously is defined for each individual region. Thus, we need to define a global error that may characterize the overall quality of the cartogram.

Based on the error $e(i, v)$ of each region R_i in regard to the variable of interest v, we may define a global measure of the *goodness* of a cartogram by its weighed mean, simple mean or quadratic mean. The quadratic mean error (mqe), is defined as:

$$mqe(v) = \frac{\sqrt{\sum_{i=1}^{N} e(i,v)^2}}{N}$$ **Equation VIII**

The weighed mean error (we), is:

$$we(v) = \sum_{i=1}^{N} |a_i \times e(i,v)|$$ **Equation IX**

The simple average error (se), is:

$$se(v) = \sum_{i=1}^{N} |e(i,v)|$$ **Equation X**

These error measures may be applied both to a cartogram and an ordinary geographical map. The global error of a geographical map is a measure of how much the variable of interest is misrepresented by the geographical area of the different regions, and thus how much distortion is necessary to get an accurate cartogram.

3.3. A New approach for building cartograms

In this section, we present a new algorithm to build cartograms based on the Self-Organizing Map (SOM). An in depth review on SOM was already presented at section 2.2.

3.3.1. Building Cartograms using SOM

The idea of using an SOM in cartography has already been explored by Skupin (2003) who uses it as a non-linear projection tool. Our idea for using SOM to build cartograms stems from the fact that it can be seen as a density estimation tool (Sarajedini and Chau 1996; Kohonen 2001; Merenyi, Jain *et al.* 2007).

The properties of the SOM as a density estimation tool have not yet been completely established, except for some very particular cases *e.g.* the one dimensional case (Ritter and Schulten 1988; Cottrell, Fort *et al.* 1998). In these cases it has been found that the

SOM has a bias towards low density areas *i.e.* $D_{units} = K \times D_{data}^{\mu}$, where D_{units} is the density of the map units, D_{data} is the density of the data patterns, K is a constant scaling factor, and μ known as magnification factor, is 2/3 for some known cases. A magnification effect of one would mean strict proportionality between the density of input patterns and the density of their assigned units. With a magnification factor of less than one low density areas will be proportionally over represented. Some work has been done to calculate the magnification factor in more general cases (Dersch and Tavan 1996; Fort 2006; Merenyi, Jain *et al.* 2007), and experimental evidence suggests that when using the original SOM algorithm the magnification factor is always less than one. Some methods have been proposed to explicitly control the magnification factor (Bauer, Der *et al.* 1996; Merenyi, Jain *et al.* 2007), but in this chapter we will use the original SOM algorithm.

In spite of the magnification effect, the more input patterns exist in a specific area of the input space the more units are assigned to represent that specific area. Additionally, units that represent that area are neighbours in the output space, consequently they can be labelled with the same label. Thus, if we populate a particular map with random points, with a density distribution proportional to the value of the variable of interest, we can train the SOM with that data. We can then label the units according to the points they are representing and use the output space as a cartogram. The process can be seen as the folding of a blank sheet (in this case according to the density of the points that represent the variable of interest) painting it according to a reference map and finally spreading out the sheet completely thus obtaining the cartogram. This approach is particularly interesting because we do not alter the standard training procedure in any way and thus any SOM implementation may be used to produce cartograms.

In our proposal, the cartogram is built based on the unit's movement during the learning process of the SOM (Henriques, Bação *et al.* 2005). In essence, during that learning process, units are moved towards higher density areas. If that movement is recorded, its inverse will transform the original map into an equal density map.

Figure 23 shows an example of this process, using a simplified instance with only two regions. The two regions are geographically identical but have distinct values for the variable of interest, represented by *p*. The variable *p* has a high value in the dark region (*i.e.*, this is a region with a high population density), and a smaller value in the lighter

region. In the following step (Figure 23b) we randomly produce points with a uniform distribution (represented with triangles) inside each region. The number of points created is a linear function of the value of population *p*. Figure 23c shows an initialised two-dimensional SOM (with 4 x 4 units). The SOM units are initialised so as to form a regular grid in the input space, contrary to the usual practice, in which the units are randomly initialised in the input space. We then continue with a standard SOM training phase where the units adapt to the training patterns. This means that units are moved (in the input space) in such a way that their density mimics the density of the data patterns (Figure 23d). The algorithm continues with the labelling process (Figure 23e). This process gives each unit a label based on the data of the region where it lies. In this case, the labels are the colours that identify each region, and thus each unit will be assigned the colour of the region where it lies after training. Figure 23f represents each unit mapped back at its initial position. Based on the units' positions and labels a final population cartogram is produced (Figure 23h). In this cartogram the darker region (that had more population) is larger than the lighter one (that had fewer population), and thus the final area of the regions is proportional to their population.

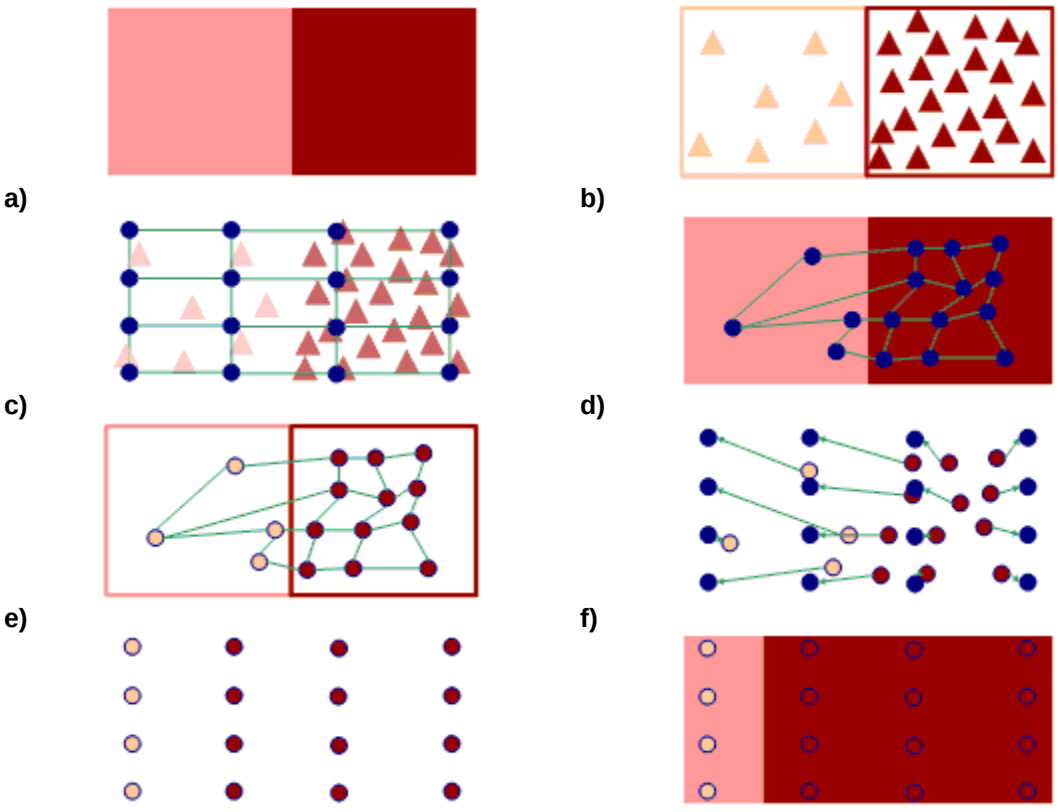
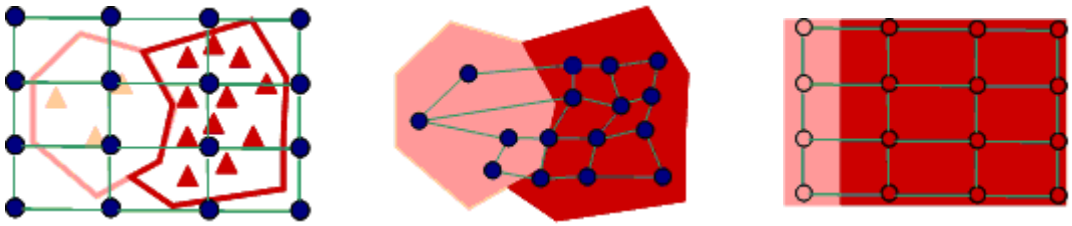


Figure 23 – Proposed method example

Most SOM implementations use rectangular shaped maps such as the one in this example. Generally, the geographic area of interest is not rectangular, and forcing the final cartogram to be rectangular, as happens with the described method, is not acceptable. The root of this problem is that when the SOM is initialised with a regular grid in a rectangle, some of its units will fall outside the geographic area of interest, as shown in Figure 24a.



a) **b)** **c)**

Figure 24 – Rectangular shaped SOM superposed on a non-regular shaped dataset; a) random point creation based on a region feature; b) SOM units after training; c) Produced cartogram

During training, those units will move to areas populated with training patterns (Figure 24b). When comparing the produced cartogram (Figure 24c) with the original regions' shape (Figure 24a) it is apparent that the cartogram regions will occupy all the (rectangular) area on the input space losing the original regions' shapes.

To address this problem we define three different areas in the input space (Figure 25). The input space is the domain area (D); the area occupied by the original regions is the region area (ra) and the remaining area is the buffer area (δ).

a) **b)**

Figure 25 – Input space nomenclature; a) SOM mapped in the input space; b) input space area definition: ra is the region area, and δ is the buffer area

The input space can now be defined as:

$$D = ra + \delta$$ **Equation XI**

As already shown, when no training patterns lie in the buffer area (δ) the cartogram produced will occupy the whole domain area (D) losing the original shape of the region area (ra). To solve this problem we created and tested several different solutions presented in detail in (Henriques 2006). Among these, the solution that produced the best cartograms consists in generating data points within the buffer area (δ) with a uniform distribution, and a density equal to the average density of data points in the whole region of interest (ra). In fact, other authors (Tobler 1973; Gastner and Newman 2004) also used a similar technique to obtain good cartograms.

3.3.2. Compensating for the magnification effect of SOM

Although the proposed algorithm may use the data available without any pre-processing, the magnification effect of the SOM, described earlier, will distort the results, producing cartograms where low density areas will be proportionally overrepresented. In some cases, this may actually be desirable, to prevent low density areas from *disappearing*. Nevertheless, we may correct, at least in part, this effect by *boosting* the original data, so that low density areas will have even lower *surrogate* densities to compensate for the magnification. We can then use this *surrogate* dataset instead of the original one. Let us now see in detail how this dataset may be generated.

For a 1-dimensional to 1-dimensional mapping, under the special conditions described by Ritter (1991), the relationship between densities of units and data points is given by:

$$D_{units} = k_1 D_{data}^{\mu} \qquad \textbf{Equation XII}$$

where D_{units} is the density of SOM units, D_{data} is the density of data points, k_1 is a proportionality constant that results from the ratio between the total number of units and the total number of data points, and μ is the magnification factor.

To build a good cartogram, we want strict proportionality between the density of units and the density of the original data. To obtain this we must have a surrogate dataset where the density D_{surr} is such that,

$$D_{units} = k_1 D_{surr}^{\mu} = k_2 D_{data} \Leftrightarrow D_{surr} = \left(\frac{k_2}{k_1}\right)^{\frac{1}{\mu}} D_{data}^{\frac{1}{\mu}} = k_3 D_{data}^{\frac{1}{\mu}} \qquad \textbf{Equation XIII}$$

where k_3 is $\left(\frac{k_2}{k_1}\right)^{1/\mu}$. For the sake of simplicity, we will not explicitly calculate each of the constants k_x involved in these calculations, since in the end we will be able to calculate the necessary variables without them. Since we will be generating, for each region, a number of points proportional to the variable of interest V and not its density, we have,

$$D_{surr} = k_3 D_{data}^{\frac{1}{\mu}} \Leftrightarrow \frac{V_{surr}}{A} = k_3 \frac{V_{data}^{\frac{1}{\mu}}}{A^{\frac{1}{\mu}}} \Leftrightarrow V_{surr} = k_3 \frac{A}{A^{\frac{1}{\mu}}} V_{data}^{\frac{1}{\mu}} = k_3 A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}} \quad \textbf{Equation XIV}$$

where V_{surr} is the surrogate variable of interest to be used instead of the variable of interest V_{data} for each region with area A. The number N of data points generated for each region is proportional to the variable of interest, *i.e.*, $N = k_4 V_{surr}$. Combining this with the previous equation we obtain, for each region,

$$N = k_5 A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}} \quad \textbf{Equation XV}$$

To compute the constant k_5 we only need to know how many data points we want overall N_{total}, since,

$$N_{total} = \sum_{allRegions} k_5 A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}} \Leftrightarrow k_5 = \frac{N}{\sum_{allRegions} A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}}} \quad \textbf{Equation XVI}$$

If the original expression is valid for 2-dimensional to 2-dimensional mappings, and we knew the magnification factor exactly, then this correction would allow a perfectly proportional representation of each region, and thus a null error in the cartogram. Since that expression is just an approximation, the error will in fact be greater than zero. As for the magnification factor μ, it is reasonable, from empirical experience, to assume it is approximately $2/3$ (Merenyi, Jain *et al.* 2007). The final number of points for each region i will thus be given by,

$$N_i = k A_i^{\frac{1}{3}} V_i^{\frac{3}{2}} \quad \textbf{Equation XVII}$$

and

$$k = \frac{N_{total}}{\sum_i A_i^{\frac{1}{3}} V_i^{\frac{3}{2}}}$$ **Equation XVIII**

Since the area outside the region of interest (δ) does not have to be faithfully represented, this correction need not be applied to that region.

3.3.3. Carto-SOM algorithm

The Carto-SOM algorithm may be defined as follows:

Given a dataset consisting of geographic map M with *n* regions R_i each with a geographical area a_i and a *value of interest* v_i, do the following:

1. Define a rectangle P encompassing all regions R_i
2. Define a new region $R_{n+1} = \delta$ consisting of all areas within P not covered by any R_i, and assign it a value of interest $v_{i+1} = \sum v_i / \sum a_i \times a_{i+1}$, where the sums are over all other regions. This corresponds to assigning a density for the *value of interest* of the area outside the original map equal to the average of the rest of the map.
3. For each region, generate $m = k \times a_i^{1/3} v_i^{3/2}$ data points with coordinates within that region, and assign to them a label i that identifies them as belonging to region R_i. The constant k is equal for all regions and it determines the total number of data points used. The choice of this number depends on the scale of the value of interest and the desired quality of the cartograms. Large values of k will produce smoother cartograms, but will require more processing power. In practice, k should be chosen so that even the smallest region gets at least 10 points, *i.e.* given the region j with smallest value of interest and smallest area, $k \geq 10 / \left(a_i^{1/3} v_i^{3/2} \right)$. The generated points may have a random uniform distribution within the region, or may be evenly spaced within that region. Optionally, this third step may be simplified by generating $m = k \times v_i$ data points within each region. This last option assumes that the magnification is 1, and thus leads to a slightly larger cartogram error, as we shall see later.
4. Generate an initial SOM with $x \times y$ units evenly spaced in the geographic rectangle P. The ratio x/y should be the same as ratio of width to height of

rectangle P, and the larger the value of x the smoother the cartogram will be. In practice, the value of x should be as high as possible, but taking into account the processing power available. Depending on the complexity of the shapes a value of 100 will usually be enough.

5. Train the SOM with the data points obtained in 3, using the standard algorithm and standard training parameters (in section 3.4.2.3 we study more closely the effects of different training parameters).
6. Label the SOM with the labels of the data points that are mapped to it. It is possible to do this with the standard SOM labelling routine available in most SOM software packages. In some rather awkward situations, this may produce a few unlabeled units. While not critical, these units may be labelled using the reverse labelling process described in (Henriques 2006), which basically consists in assigning to each SOM unit the label of the closest data pattern.
7. Print out the labels of each unit, using as coordinates each unit's position in the SOM (*i.e.*, in the output space). This is the desired cartogram.

Steps 1 to 3 are pre-processing steps that can be performed using any geographic information system, such as ArcGIS© or MapInfo, purpose built MATLAB™ routines (Henriques, Bação *et al.* 2010), or even a general-purpose program such as MS Excel®.

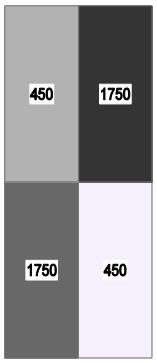
Steps 4 to 7 may be performed with any SOM software, such as SOM Toolbox for MATLAB™, SOM-PAK, or SAS® Enterprise Miner. However, to obtain the best results and avoid possible problems with unlabeled units, the routines available for MATLAB™ at (Henriques, Bação *et al.* 2010) should be used.

3.4. Results

In this section, we test the Carto-SOM method using three different datasets. An analysis of Carto-SOM's sensitivity to training parameters is performed along with a comparison with Dougenik and Gastner's cartograms.

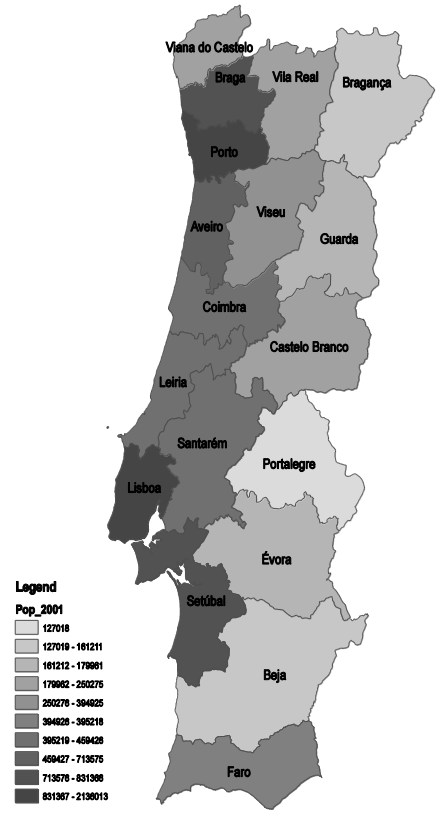
3.4.1. Experimental Settings

To test the Carto-SOM method we used an artificial dataset, Portugal's population data for 2001 and USA's population data for 2000 (Figure 26).



a) Artificial dataset

b) USA population for 2001



c) Portuguese population for 2001

Figure 26 – Datasets used to test Carto-SOM

We chose the Dougenik cartogram method (Dougenik, Chrisman *et al.* 1985) and the Diffusion method (Gastner and Newman 2004) as benchmarks for the Carto-SOM. The choice of Dougenik's cartograms was due to the fact that these are quite popular and the software (to implement them in an ArcGIS© environment) is freely available at http://arcscripts.esri.com/details.asp?dbid=14226. The choice of Diffusion cartograms was due to the fact that they seem to be the ones that manage to obtain the best quantitative errors when representing the variable of interest.

To test the Carto-SOM method we trained the SOM using the MATLAB™ SOM Toolbox (Vesanto, Himberg *et al.* 2000). Several training processes were performed, changing the initial SOM parameters. In Table 2 we present the parameters that provided the best cartograms. In the next section, we present the results of the sensitivity analysis performed based on the USA population dataset.

Table 2 - Best SOM parameters used in the Artificial, Portuguese and USA datasets

row	column	r	α	E
258	388	26	0.1	40

row and column are the number of rows and columns used in the SOM; r is the initial neighbourhood radius, α is the initial learning rate, E is the number of epochs used in the training phase.

For Dougenik´s method, we used an ArcGIS© script file to produce the cartograms (Schmid 2005). This script allows the ArcGIS© user to select an input layer and to choose the number of iterations used. In this test, we used five iterations to produce Dougenik's cartograms. For the diffusion cartogram we used an implementation produced by Michael Gastner available at his homepage (Gastner 2005).

3.4.2. Sensitivity Analysis of Carto-SOM

To analyse the sensitivity of the Carto-SOM method we performed several sets of tests changing in each set one of the parameters. In the first set, we test Carto-SOM's robustness to initialisation and random effects by using the same parameters in several tests. In the second set, we assumed different magnification factors when boosting the original data, to conclude about the effect of SOM's magnification problem in the cartograms. In the third set of tests, which we call the SOM parameters evaluation, we test changes in the neighbourhood radius, learning rate and the number of epochs. Finally, in the fourth set we evaluate influence of the number of units of the SOM and input data points.

3.4.2.1. Carto-SOM robustness test

To evaluate the influence of random effects due to the particular sample of input data points chosen, and the order by which they are selected and presented to the algorithm, we performed ten tests. In these tests, all Carto-SOM parameters were maintained except the input data points, which were generated each time (Table 3).

Table 3 - SOM parameters used to test input data points influence

row	column	r	α	E	nD
258	388	26	0.1	40	20000

row and column are the number of rows and columns used in the SOM; r is the initial neighbourhood radius, α is the initial learning rate, E is the number of epochs used in the training phase and nD is the number of input data points used which were randomly generated for each test.

In Figure 27 we present the Keim simple error for each cartogram produced. For the 10 tests performed we achieved an average error of 4.36% with a standard deviation of 0.32.

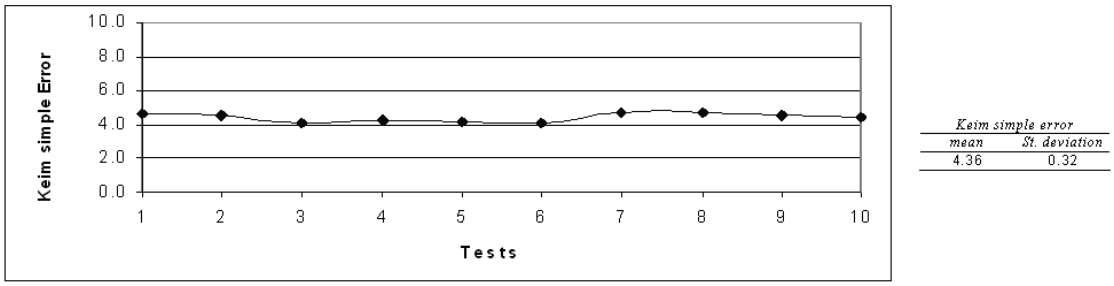


Figure 27 – Carto-SOM robustness to the choice of input data points

From these results, we may conclude that Carto-SOM is quite robust to different samples of input data points. These tests prove that generating input data points in a random way will not affect the final cartogram provided that the density is maintained. They also prove that even though Carto-SOM uses a neural network that may produce slightly different results each time it is used, the differences are negligible.

3.4.2.2. Magnification effect

In order to evaluate the magnification effect (used when boosting the original data), we performed several different tests changing the magnification factor value as shown in Table 4. Test 11, where $\mu = 1$ corresponds to a proportional input data point generation, *i.e.* there is no compensation for the magnification effect.

Table 4 - Magnification factor (μ) variation

Test	1	2	3	4	5	6	7	8	9	10	11
μ	0.5	0.55	0.6	0.65	0.67	0.7	0.75	0.8	0.85	0.9	1
Keim error	17.43	12.67	8.17	5.01	5.19	4.27	7.01	9.32	11.81	13.75	18.05

In these tests, we used 258×388 units, an initial neighbourhood radius of 26, an initial learning rate of 0.1, 40 epochs and 20000 input data points.

In Figure 28 the Keim simple error for each of the eleven cartograms produced is presented. The lowest error is obtained when the value of μ is 0.7 (similar to the predicted value of 2/3). From this result we conclude that boosting the original data using the magnification compensation of the input data points produces better results than using a proportional relation ($\mu = 1$).

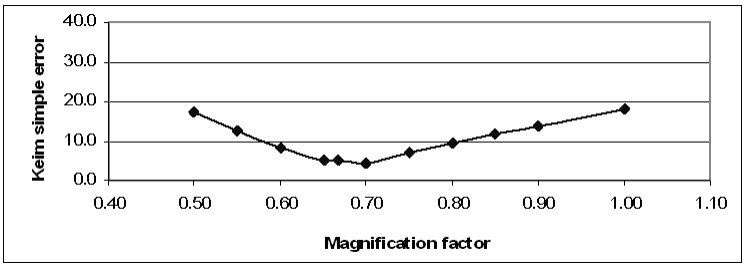


Figure 28 – Carto-SOM error as a function of the magnification factor assumed for boosting original data

3.4.2.3. SOM parameters evaluation

To test how the Carto-SOM methodology depends on the SOM parameters we performed a set of tests using different neighbourhood radius (r), learning rates (α) and number of epochs (E).

A. Neighbourhood radius

In Table 5 we present the ten tests performed using different neighbourhood radii and maintaining the remaining parameters.

Table 5 - Variation of the neighbourhood radius

Test	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
radius	1	3	5	13	26	52	77	103	129	155
Keim error	36.61	30.76	22.72	4.87	4.55	6.58	6.95	7.97	9.68	11.22

In these tests, we used 258×388 units, an initial learning rate of 0.1, 40 epochs and 20000 input data points.

As expected, if the value of the initial neighbourhood radius is very small then the SOM will have more difficulty to adjust to the input data, resulting in cartograms with higher errors (Figure 29). On the other hand, if we increase the neighbourhood radius too much then the SOM will be less stable and the final results tend to have higher errors. In the

tests performed we achieved an optimum error using a radius of 26 which corresponds to 10% of the number of units in row.

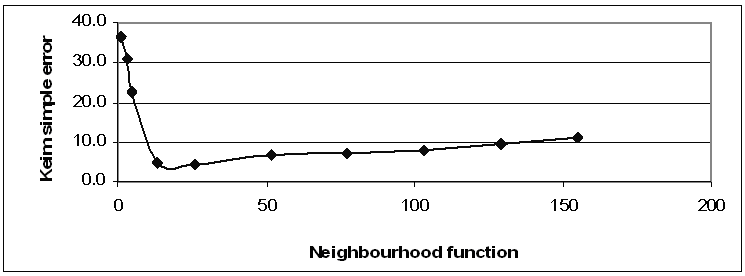


Figure 29 – Carto-SOM error as a function of neighbourhood radius

B. Learning rate

We evaluate the effect of the learning rate by using ten different values for this parameter while maintaining all the other parameters (Table 6).

Table 6 - Variation on the learning rate

Test	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Learning rate	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Keim error	*4.55*	*5.69*	*4.5*	*4.87*	*5.5*	*6.59*	*4.36*	*4.11*	*4.86*	*4.23*

In these tests, we used 258×388 units, an initial learning rate of 0.1, 40 epochs and 20000 input data points.

In Figure 30, we present the Keim simple error for each cartogram using different learning rates. Although changing this parameter produces different cartogram errors these are not relevant and we conclude the learning rate isn't a sensitive parameter.

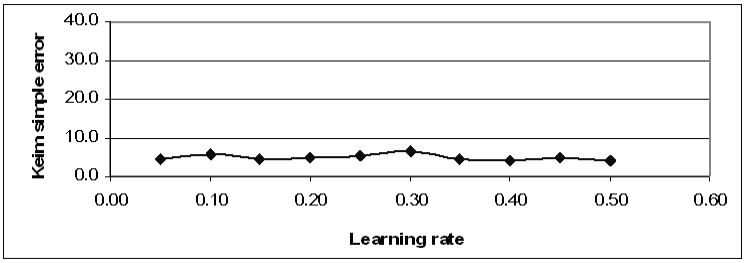


Figure 30 – Carto-SOM error as a function of learning rate

C. Number of epochs

The third parameter tested was the number of epochs used in the SOM training phase. Again, ten tests were executed to evaluate the influence of the number of epochs on the cartogram error (Table 7).

Table 7 - Variation on the number of epochs

Test	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Epochs	20	25	30	35	40	45	50	55	60	65
Keim error	*4.87*	*5.8*	*4.86*	*4.95*	*5.5*	*6.56*	*4.14*	*4.06*	*4.83*	*4.18*

In these tests, we used 258×388 units, an initial learning rate of 0.1, 40 epochs and 20000 input data points.

In the following figure (Figure 31), we present the Keim simple error for the tests performed. As in the learning rate evaluation, the number of epochs used does not have a great influence in the final cartogram error, provided that the number of epochs is greater than 5.

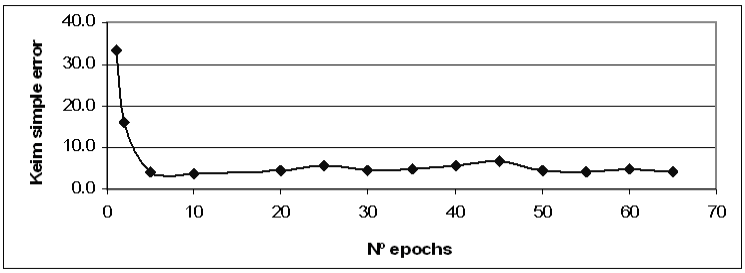


Figure 31 – Carto-SOM error as a function of the number of epochs used

3.4.2.4. SOM dimension parameters

In these experiments we want to test how the Carto-SOM methodology depends on the SOM dimension parameters, *i.e.* how does the number of units and the number of input data points influence the error obtained in the cartograms.

A. Number of units

In this analysis we want to evaluate the cartogram errors when using different numbers of units (Table 8). The ratio between the number of rows and columns is the same as the

ratio of the width and height of the original region of interest. In this case (which refers to the USA) a row/column ratio of $1/1.5$ is used.

Table 8 - Variation on the number of units used

Test	1	2	3	4	5	6	7	8	9	10
nº of rows	18	26	37	58	82	115	183	258	408	577
nº of columns	28	38	54	86	122	174	273	388	613	867
total nº of units	504	988	1998	4988	10004	20010	49959	100104	250104	500259
Keim error	*27.16*	*22.82*	*13.22*	*4.65*	*6.04*	*5.18*	*4.61*	*4.75*	*4.27*	*4.97*

In these tests, we used 258×388 units, an initial learning rate of 0.1, 40 epochs and 20000 input data points.

Figure 32 shows the errors obtained. As we can see there is a number of units (4988 units) for which the error tends to stabilize. There is reason to believe that the exact number of units for which the error stabilizes depends on the complexity of the regions being mapped.

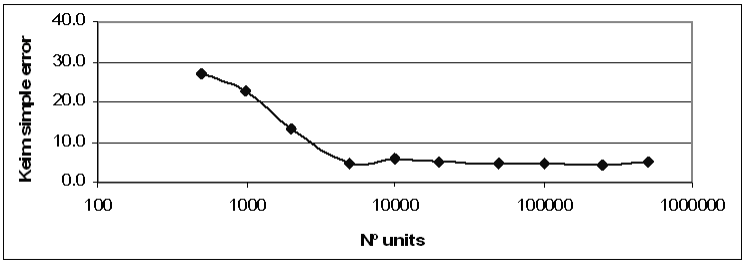


Figure 32 – Carto-SOM error as a function of the number of units

B. Number of input data points

Finally, we evaluate the effect of the number of input data points by performing ten tests using different numbers of input data points (Table 9).

Table 9 - Variation of the number of input data points

Test	1	2	3	4	5	6	7	8	9	10
nD	8100	9000	10000	15000	20000	25000	30000	35000	40000	45000
Keim error	*5.12*	*4.59*	*4.30*	*4.23*	*3.70*	*4.42*	*4.41*	*4.36*	*4.69*	*5.08*

In these tests, we used 258×388 units, an initial learning rate of 0.1, 40 epochs and 20000 input data points.

Figure 33 presents the cartogram error for each test. We did not perform any tests with less than 8100 data points because we imposed that at least 10 data points should be generated for each region. If care is not taken to guarantee this, some regions will have

a very high error because there simply are not enough data points to represent them. As long as this is taken into account, as we can see from this figure, the number of input data points has only a small influence on the cartogram error.

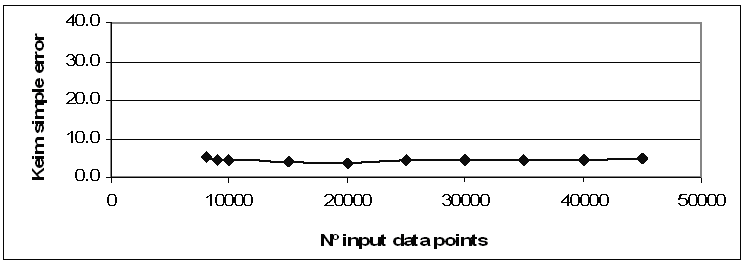


Figure 33 – SOM error as a function of the number of input data points

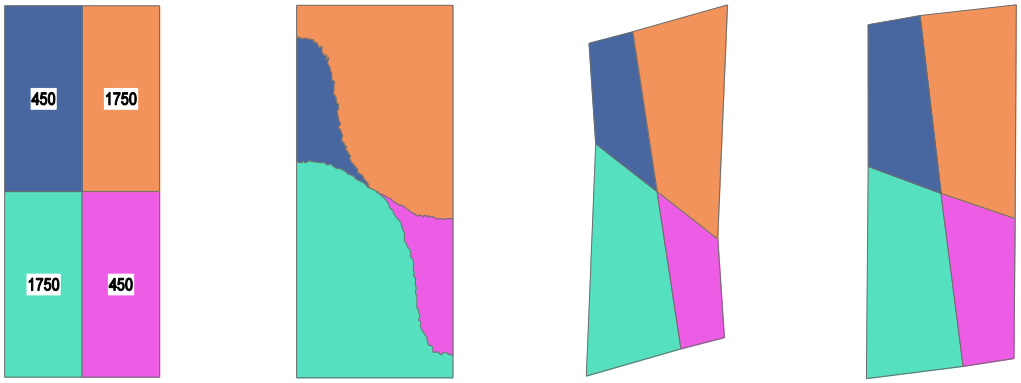
In summary, we may conclude from this sensitivity analysis that:

- Random input pattern generation does not affect the final cartogram error;
- Boosting the original data using a magnification factor near $2/3$ minimizes the cartogram errors;
- Using an initial neighbourhood radius near 10% of the number of units in row (smallest dimension) produces a minimum error;
- The initial learning rate has not a great impact in the cartogram error;
- The number of epochs used does not have a great influence in the cartogram error, provided that the number of epochs is greater than 5;
- The number of units for which the error tends to stabilize is, for this specific case, near 5000.
- Provided that the number of input data points generated is large enough to guaranty that at least 10 points exist for each region, this is not an important parameter.

3.4.3. Comparison between Carto-SOM, Dougenik and Diffusion Cartograms

In this section, we present the final cartograms for the three datasets: an artificial dataset, Portuguese 2001 population and USA 2000 population. We also compare the results of the Carto-SOM with the benchmark methods selected.

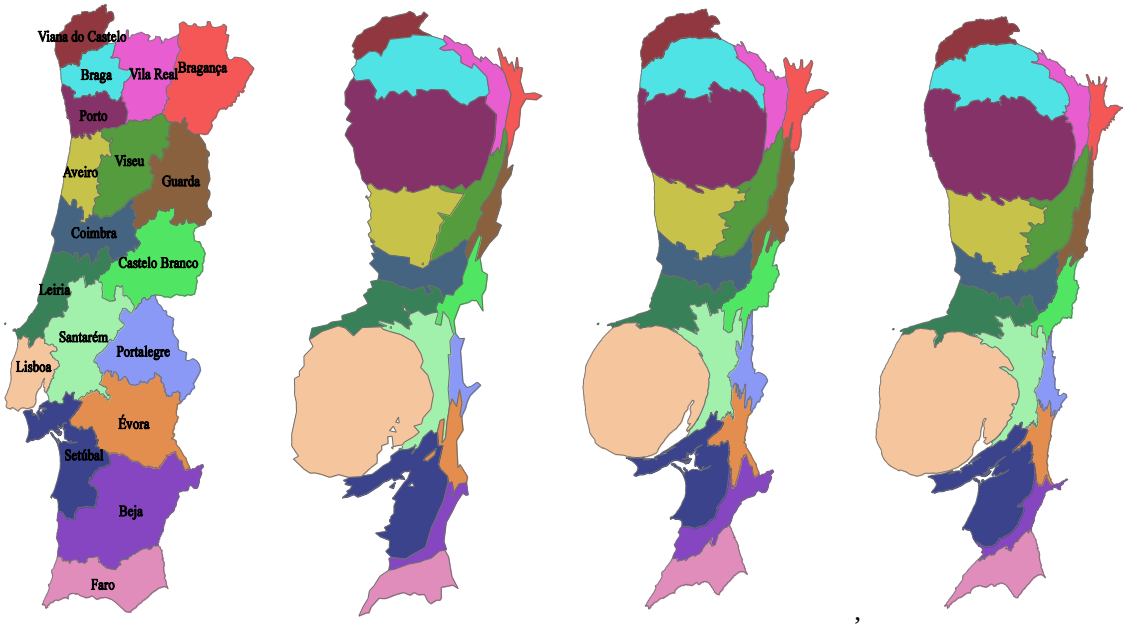
Figure 34 presents the artificial dataset and the resulting cartograms using the Carto-SOM, Dougenik and Gastner's methods. Visually these are good results since the cartograms represent the variable density by the size of each region while preserving topology, *i.e.*, all areas have the same neighbours. Furthermore, preserving important points, such as the junction of all four regions in the centre, increases the cartogram's recognition.



a) Original map **b) Carto-SOM** **c)Dougenik cartogram** **d)Diffusion cartogram**

Figure 34 – Original map, Carto-SOM, Dougenik and diffusion cartograms of the artificial dataset

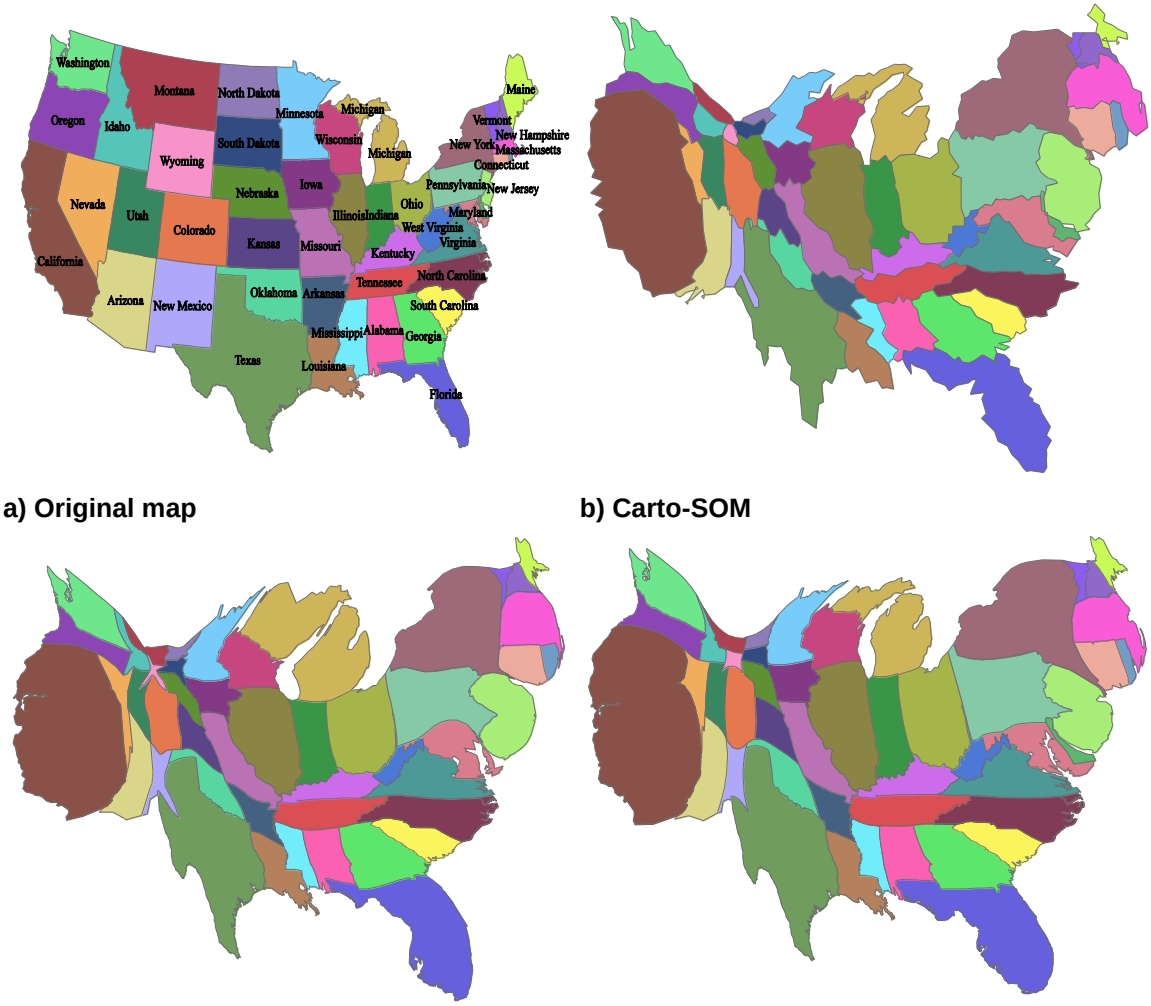
Figure 35 presents Portuguese population cartograms using Carto-SOM, Dougenik and Gastner's methods.



a) Original map **b) Carto-SOM** **c)Dougenik cartogram** **d)Diffusion cartogram**

Figure 35 – Portuguese population cartograms using the Carto-SOM, Dougenik and Diffusion methods

In Figure 36 we present the USA 2000 population cartograms using the three different methodologies: Carto-SOM, Dougenik and Gastner's.



a) Original map **b) Carto-SOM**

c) Dougenik cartogram **d) Diffusion cartogram**

Figure 36 – USA population cartograms using the Carto-SOM, Dougenik and Diffusion methods

Through visual analysis of the cartograms produced we can see that Carto-SOM manages to keep the shapes of individual regions in a way that is generally comparable to Dougenik and Diffusion Cartograms. The Dougenik cartogram has the disadvantage of performing radial transformations on the shapes of the regions, making them circular as the number of iterations increases. This is clearly observed in Portugal, where Lisbon and Oporto appear almost as circles, and to a lesser extent in the USA where California tends to an egg shape. The diffusion cartogram also suffers from this bubble effect, albeit to a lesser extent. Because of the same effect in the Dougenik cartogram, some regions are unevenly squashed, making Texas look skewed (it looks straight in the

Carto-SOM), while Setúbal and Santarém in Portugal are unevenly distorted by neighbouring Lisbon. The Carto-SOM also manages to keep the overall shape of the cartogram similar to the geographical map, even though, as we shall see in the next section, the variable of interest is better represented. Unfortunately, there are some cases in which certain regions are *broken up* into non-contiguous areas in the cartogram. To minimize this effect we may increase the number of neurons used, but no definite solution has been developed for this problem.

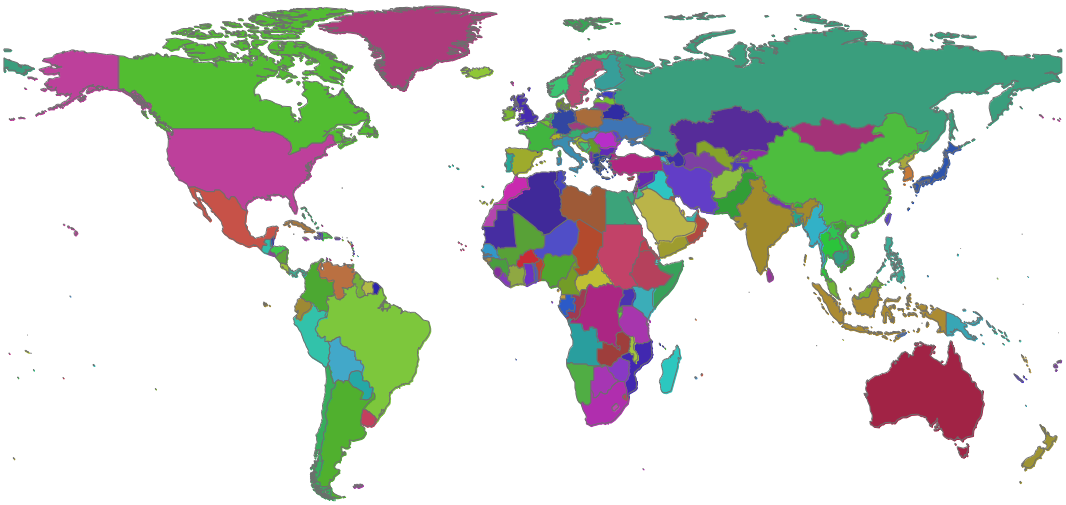
In Table 10 we present the cartogram errors calculated for each cartogram produced. The Carto-SOM performance measured by this error is significantly better except for the case of Portugal, where it is still quite competitive.

Table 10 - Keim error evaluation using different criteria on the various datasets

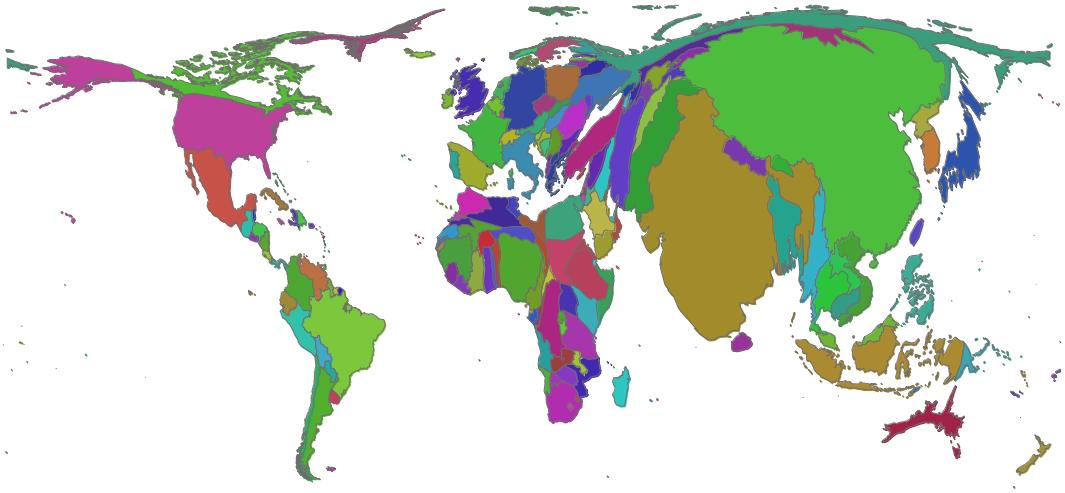
	Artificial			USA			Portugal		
	se (%)	mqe (%)	we (%)	se (%)	mqe (%)	we (%)	se (%)	mqe (%)	we (%)
Dougenik	14.64	1.83	10.3	16.02	3.08	11.09	12.29	3.80	8.65
Diffusion	22.47	2.72	16.41	5.33	1.45	4.71	1.57	0.46	1.18
Carto-SOM	3.88	2.34	2.47	3.70	0.74	0.65	3.97	1.07	1.29

3.4.4. Related issues

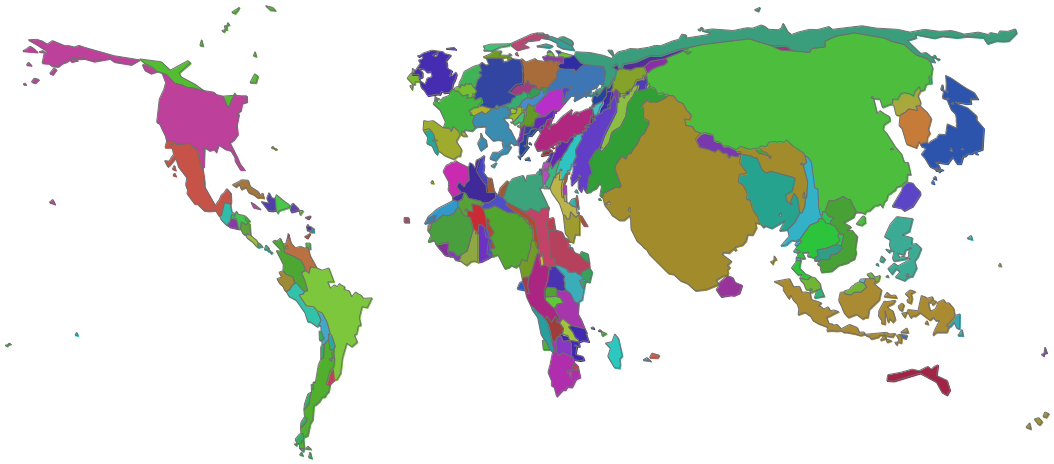
An important factor in cartogram construction is the number of areas used. Several methods are able to produce good cartograms only if a small number of areas is used. One of the most famous methods that is capable of using a high number of objects is Gastner's diffusion cartograms. Using this method, cartograms were built for the population of the countries of the world (aprox. 200 objects) and of the USA counties (aprox. 3000 objects). The proposed Carto-SOM method can produce satisfactory cartograms for such large datasets, as is shown in Figure 37 and Figure 38.



a) Original world population map

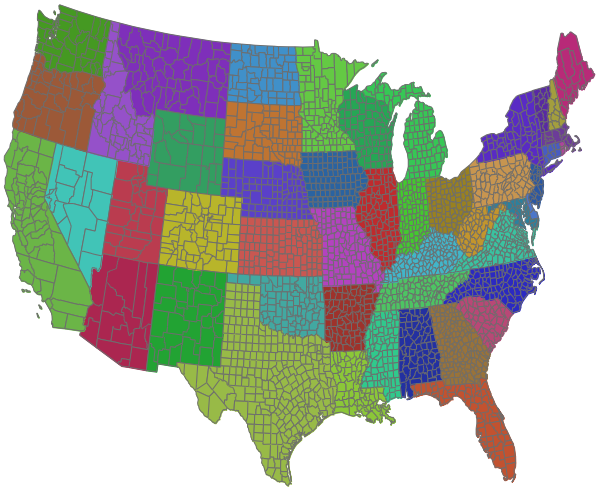


b) World population diffusion cartogram



c) World population Carto-SOM cartogram

Figure 37 – World countries population cartogram



a) Original USA counties population

b) USA counties diffusion cartogram

c) USA counties Carto-SOM cartogram

Figure 38 – USA counties population cartogram

Another issue related with the use of a high number of areas to build a cartogram is the possibility of using parallel processing. The proposed Carto-SOM method can make very good use of parallel processing. One of the problems with parallel systems is that it is sometimes difficult to parallelize the algorithms, *i.e.* split the algorithm into different tasks requiring little interaction. Fortunately, the SOM training algorithm is easily adapted to parallel systems (Ultsch and Siemon 1990; Przytula and Viktor 1993). We may even use widely available networked PCs running Windows or Linux to efficiently train very large SOM's (Bandeira, Lobo *et al.* 1998). This will not only allow us to drastically reduce the processing time but also to increase the number of SOM units that can be used, thus producing high quality cartograms in a short time.

3.5. Discussion

In this chapter, we presented a new method for building cartograms, called Carto-SOM, which is based on the Self-Organizing Map (SOM) algorithm. To a certain extent, this constitutes a completely different approach for building cartograms. In this case, instead of developing a new algorithm we make use of the widely available SOM algorithm and use it in an unexpected and creative way to build cartograms.

To evaluate the quality of the cartograms obtained with this method, tests were performed and assessed, comparing the proposed algorithm with existing ones, both using quantitative measures and visual inspection. The quantitative measure used was Keim's cartogram error. The tests suggest that the cartograms created using the Carto-SOM are good and accurate representations of the variables of interest. Visually we conclude that Carto-SOM is an efficient cartogram building algorithm, usually achieving better results than the Dougenik and Gastner's method used as benchmarks.

Finally, it must be emphasized that, using the Carto-SOM method, the only software necessary to create a cartogram is a standard implementation of the SOM algorithm. Such implementations are widely available, both in commercial data analysis programs and public domain packages.

4. GeoSOM Suite: a tool for geospatial clustering

Clustering constitutes one of the most popular and important tasks in data analysis. This is true for any type of data, and geographic data is no exception. In fact, in geographic knowledge discovery the aim is, more often than not, to explore and let spatial patterns surface rather than develop predictive models. The size and dimensionality of the existing and future databases stress the need for efficient and robust clustering algorithms. This need has been successfully addressed in the context of general-purpose knowledge discovery. Geographic knowledge discovery, nonetheless, can still benefit from better tools, especially if these tools are able to integrate geographic information and aspatial variables in order to assist the geographic analyst's objectives and needs. Typically, the objectives are related with finding spatial patterns based on the interaction between location and aspatial variables. When performing cluster-based analysis of geographic data, user interaction is essential to understand and explore the emerging patterns, and the lack of appropriate tools for this task hinders a lot of otherwise very good work.

In this chapter, we present the GeoSOM Suite as a tool designed to bridge the gap between clustering and the typical geographic information science objectives and needs. The GeoSOM Suite implements both the original SOM and the GeoSOM algorithm (Bação, Lobo *et al.* 2008), which changes the SOM algorithm to explicitly take into account geographic information. We present a case study, based on census data from Lisbon, exploring the GeoSOM Suite features and exemplifying its use in the context of exploratory data analysis.

4.1. Introduction

Advances in database technologies and in data collecting devices originated a huge growth in the amount of spatial data available. Processing these amounts of data requires powerful data mining tools, which form the core of the spatial data mining field. Spatial data mining can be defined as the discovery of interesting relationships, spatial patterns and characteristics that may exist in spatial databases.

One of the most used data mining techniques is clustering. Clustering is a well established scientific field (Fisher 1936; Kaufman and Rousseeuw 1990) allowing the partition of data into groups of similar objects. These objects are usually represented as a vector of measurements or a point in a multidimensional space (Jain, Murty *et al.* 1999). Spatial clustering (Han 2005) is the partition of spatial objects into groups so that objects within a cluster are as similar as possible. Due to spatial dependency, an intrinsic characteristic of spatial data explained by the 1st law of geography (Tobler 1970), clusters are expected to be grouped in space. Tobler's first law (TFL) states that "*everything is related to everything else, but near things are more related than distant things*". It can be argued that, as mentioned before, the First Law of Cognitive Geography (''*People think that closer things are more similar*'') (Montello, Fabrikant *et al.* 2003) can be a more accurate description of reality. Although even Tobler himself (Tobler 2004) recognizes the first part of TFL is not always true (Sui 2004) correlation is likely to be stronger at short distances.

In spite of TFL, we often see clusters produced from spatial datasets which are spatially apart. Some of the known causes are: 1) the aggregation and the scale of data (Openshaw 1984), 2) the spatial heterogeneity (Anselin 1988) and 3) the multivariate nature of the clustering.

The problems raised by the aggregation and the scale of data are known as the modifiable areal unit problem (MAUP)(Openshaw 1984). The problem is that spatial phenomena are normally continuous, but have to be aggregated to obtain a manageable discrete description. The exact outline of the area over which the description is obtained will influence critically the perception of the phenomena. Moreover, if this aggregation is obtained at different scales, that perception will be even more biased.

The spatial heterogeneity is the property that makes each place on Earth unique due to its specific attributes (Anselin 1988). This variation implies that standards and design decisions successfully adopted in one region cannot always be generalized and applied in other regions (Goodchild 2008). This uniqueness of each place makes spatial clustering an even more complex task.

The third problem with spatial clustering is that it usually involves multidimensional data. This is a problem since each of the variables may have a different level of spatial autocorrelation, and thus the global spatial autocorrelation depends a lot on the relative importance given to each of them. Even in the case when all variables share a similar global spatial autocorrelation (O'Sullivan and Unwin 2002), it is usually space dependant, and thus the local patterns of this dependency can be very different.

Nevertheless, many applications require spatially contiguous clusters that contain regions as homogeneous as possible (within each cluster), separated from each other by discrete boundaries. Same examples of these applications are image segmentation (Awad, Chehdi *et al.* 2007), creation of areas for precision farming (Fleming, Heermann *et al.* 2004), estuarine management areas (Bação, Caeiro *et al.* 2005) and zone design problems (Bação, Lobo *et al.* 2005; Cockings and Martin 2005).

Several methods are available for spatial clustering (Guha, Rastogi *et al.* 1998; Sander, Ester *et al.* 1998; Sheikholeslami, Chatterjee *et al.* 1998; Ng and Jiawei 2002; Hu and Sung 2005). For a more detailed survey on available methods, the reader is referred to (Han, Kamber *et al.* 2001).

However, many of these methods are not aware of spatial dependence and spatial heterogeneity, assuming that space coordinates are just two (or three) more variables. These methods are based on general-purpose clustering methods which have limited

capabilities in recognizing spatial patterns that include neighbours (Guo, Peuquet *et al.* 2003).

GeoSOM proposed in (Bação, Lobo *et al.* 2008) is an extension of self-organizing maps (SOM). It is specially oriented towards spatial data mining. As one of the most known unsupervised artificial neural network architecture, SOM has been successfully applied to a wide array of spatial data (Agarwal and Skupin 2008). GeoSOM, while implementing the SOM, recognizes the special inter-relation of spatial dimensions and the importance of this sub-space in the Geographer's analysis. GeoSOM takes into account Tobler's first Law, searching for clusters within certain geographic boundaries instead of global clusters produced by standard SOM.

This chapter extends and consolidates (Bação, Lobo *et al.* 2008) in three major ways:

- It presents a user-friendly and stand-alone implementation for SOM and GeoSOM (GeoSOM Suite).
- It provides several enhancements to the original GeoSOM, by adding visualisation and data analysis tools.
- It provides an example of how GeoSOM can be used, by analysing Lisbon's census data.

GeoSOM Suite enables the user to interact with data and combine multiple clustering solutions, thus gathering knowledge about data and the clusters produced. This tool implements the standard SOM and the GeoSOM algorithm with a few improvements providing a friendly and ready to use environment for spatial data exploration. Some of the improvements on the GeoSOM are: 1) a tool for cluster outline on the U-matrix; 2) auxiliary tools to help on that outline, such as hierarchical clustering in the U-matrix; 3) inclusion of parallel coordinate plots 4) visualisation of the hits combined with the selection; 5) support for hierarchical SOM (see chapter 5); 6) support for comparisons amongst several different SOM; 7) export of outputs for visualisation within GoogleEarth (via KML). In Appendix 2, we present a User Manual for GeoSOM Suite, and in Appendix 3 its source code.

The chapter is organized as follows: section 4.2 presents prior work relevant for this chapter. Section 4.3 reviews the GeoSOM method and in section 4.4, the two datasets

used to exemplify this tool are presented. Section 4.5 presents GeoSOM Suite tool, and section 4.6 demonstrates a case study using Lisbon's Metropolitan Area 2001 census dataset. Finally, section 4.7 draws some conclusions about the work developed in this chapter.

4.2. Related work

According to Guo and Gahegan (2006), when analysing geo-referenced data, there are three ways to combine spatial and non-spatial variables. These are: 1) embed the spatial information as *normal* variables (and for that they proposed encoding and ordering spatial data in a particular way); 2) create new data mining algorithms that take into account both types of characteristics, treating spatial variables in a special way; or 3) use multiple views to visually link patterns across different spaces (spatial and non spatial).

Several tools combining exploratory spatial data analysis and data mining methods have been proposed. One of the oldest tools is GeoMiner (Han, Koperski *et al.* 1997), which was based on a relational data mining system known as DbMiner (Han, Cai *et al.* 1993). GeoMiner proposed a new language (geographic mining query language) to define characteristic rules, comparison rules and association rules. Another characteristic of this system is the integration of data mining, data warehousing technologies and geographic information systems, presenting various outputs, such as maps, tables and charts.

Maceachren *et.al.* (1999) proposed the GKConstruck, allowing the integration of knowledge discovery in databases (KDD) and geographic visualisation (GVis), with spatiotemporal environmental data. The authors proposed a prototype capable of presenting three dynamically linked representation forms: the geographic map, 3D scatter plots and parallel coordinate plots. These three linked windows allow spatial data exploration through dynamic brushing, focusing and colour manipulation.

Another tool for spatial data analysis and visualisation is GeoVista Studio (Takatsuka and Gahegan 2002). In this tool, the user is able to build his own exploratory methods by visual programming. Dynamically linked visual representations such as maps, scatter plots and parallel coordinate visualisations are used for exploration and analysis.

Anselin proposed the GeoDA tool (Anselin, Syabri *et al.* 2006), including histograms, box plots, scatter plots, choropleth maps, global and local indicators of spatial association (LISA) (Anselin 1993) and spatial regression. This tool also makes use of dynamically linked windows, combining maps with statistical plotting.

In a recent paper, Mu and Wang (2008) proposed a scale-space clustering method for spatial data. This method produces several clustering sets for different scales just like in hierarchical clustering. At the top of the hierarchy there is only one cluster, and at the base the number of clusters is equal to the number of data objects. The method starts by calculating aggregation scores based on the characteristics of each object and its neighbours. These scores allow the creation of directional links, which enables the definition of local minima and maxima: local minima are objects with all directional links pointing toward other objects while local maxima are objects with all directional links pointing toward them. In the next phase, the method groups objects iteratively, from local minima to local maxima, according to the directional links. This method has, amongst others, the advantage of producing clusters that are always spatially contiguous.

Despite the variety of toolboxes and methods of geospatial clustering and data exploration systems it is not by any measure a closed research field and there is a lot of room for improvement.

In this chapter, we present the GeoSOM Suite, which provides an exploratory environment allowing the detection of spatial patterns over different spaces. One of the main contributions of this new tool is the implementation of a new method (GeoSOM) for spatial clustering. In the next section, we present a more detailed explanation of the GeoSOM algorithm.

4.3. GeoSOM outline

GeoSOM is an adaptation of SOM to consider the spatial nature of data. In GeoSOM, the search for the best matching unit (BMU) has two phases. The first phase settles the geographical neighbourhood where it is admissible to search for the BMU, and the second phase performs the final search using the other components. A parameter k controls the search neighbourhood defined in the output space. Using k=0 will necessarily select as BMU the unit geographically closer. The same result may be

obtained by training a standard SOM with only the geographical locations, and then using each unit as a low pass filter (*i.e.* a sort of average) of the non-geographic features. As *k* (the geographic tolerance) increases, the unit locations will no longer be quasi-proportional to the locations of the training patterns, and the *equivalent filter* functions of the units will become more and more skewed, eventually ceasing to be useful as models. Setting *k* equal to the size of the SOM is equivalent to treating spatial coordinates as any other variable.

Formally, the GeoSOM may be described by the following algorithm:

Let

$\boldsymbol{X}$ be the set of n training patterns $\boldsymbol{x_1, x_2, \ldots, x_n}$ each of these having a set of geospatial components $\boldsymbol{geo_i}$ and another set of non geospatial components $\boldsymbol{ngf_i}$.

$\boldsymbol{W}$ be a $\boldsymbol{p \times q}$ grid of units $\boldsymbol{w_{ij}}$ where i and j are their coordinates on that grid, and each of these units having a set of geospatial components $\boldsymbol{wgeo_{ij}}$ and another set of non geospatial components $\boldsymbol{wngf_{ij}}, \boldsymbol{w_{ij}} = \left[\boldsymbol{wgeo_{ij}}\ \boldsymbol{wngf_{ij}}\right]$.

α be the learning rate, assuming values in $]0,1[$ initialised to a given initial learning rate

r be the radius of the neighbourhood function $\boldsymbol{h(w_{ij}, w_{mn}, r)}$, initialised to a given initial radius

k be the radius of the geographical BMU that is to be searched

1 Repeat
2 For $m = 1$ *to* n
3 For all $\boldsymbol{w_{ij}} \in \boldsymbol{W}$,
4 Calculate $\boldsymbol{d_{ij}} = \left\| \boldsymbol{wgeo_{ij}} - \boldsymbol{wgeo_m} \right\|$
5 Select the unit that minimizes $\boldsymbol{d_{ij}}$ as the geowinner $\boldsymbol{w_{BMUgeo}}$
6 Select a set $\boldsymbol{W_{BMU}}$ *of* $\boldsymbol{w_{ij}}$ such that the distance in the grid between $\boldsymbol{w_{BMUgeo}}$ and $\boldsymbol{w_{ij}}$ is smaller or equal to k
7 For all $\boldsymbol{w_{ij}} \in \boldsymbol{W_{BMU}}$,
8 Calculate $\boldsymbol{d_{ij}} = \left\| \boldsymbol{x_k} - \boldsymbol{x_{ij}} \right\|$
9 Select the unit that minimizes $\boldsymbol{d_{ij}}$ as the winner $\boldsymbol{w_{BMU}}$
10 Update each unit $\boldsymbol{w_{ij}} \in \boldsymbol{W}: \boldsymbol{w_{ij}} = \boldsymbol{w_{ij}} + \boldsymbol{\alpha . h(w_{BMU}, w_{ij}, r)} \left\| \boldsymbol{x_k} - \boldsymbol{x_{ij}} \right\|$
11 Decrease the value of α and r
12 Until α reaches 0 (or some other stopping criteria is met)

4.4. Datasets used in this chapter

In this chapter, we use two different datasets to illustrate the use of the GeoSOM Suite. The first dataset (Squareville) is a fictional example (Lobo, Bação *et al.* 2009), consisting of data points with two spatial variables (the *x* and *y* coordinates) and one aspatial variable. The second dataset is taken from the Lisbon Metropolitan Area (LMA) census for 2001, and it is used for a more detailed case study.

4.4.1. Squareville dataset

Squareville is a fictional dataset used as benchmark in spatial clustering problems. Squareville is a small town with square boundaries and $10000\,m^2$ of area. Squareville has 100 houses evenly spaced with coordinates $x \in [5{,}95]$ and $y \in [5, 95]$. For each house we know the average salary s, which has a uniform distribution in $[0{,}100]$ for $35 \leq x \leq 65$ and a uniform distribution in $[900{,}1000]$ otherwise. Figure 39 shows the houses' distribution and their average salaries along Squareville.

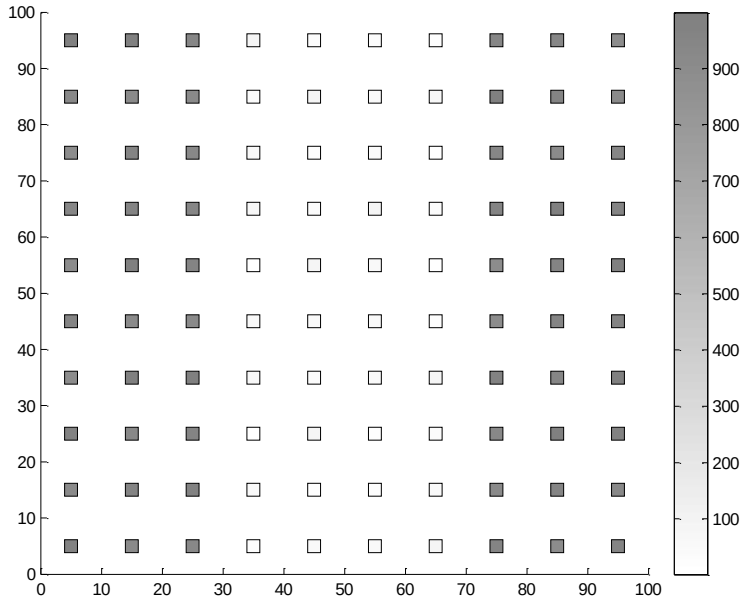


Figure 39 – Squareville (*x* and *y* represent the geographic coordinates while the colour represents the average salary by house)

Although this dataset is not normalized, (variable *s* is one order of magnitude higher than the others) we wish to use it as is. Our objective is to emulate the situation where a great

number of non-spatial variables exist, and therefore spatial variables have a relatively small weight in the final clustering process.

4.4.2. Lisbon census

This dataset refers to the 2001 census data from Lisbon's Metropolitan Area (LMA), obtained from the Portuguese Statistical Institute (*Statistics Portugal*). This data is aggregated in 3978 enumeration districts (ED) (*secções estatisticas* in Portuguese) and describes buildings, families, households, resident's age, education levels and economic activities using more than 65 variables. An ESRI™ shapefile with the ED's spatial outline and attributes is used as starting point for processing by the GeoSOM Suite. One may also use other file types, such as *.csv* or *.mat* files, but in that case, the geographical map will not be produced. Figure 40 presents a map with LMA delineation and relative location within Portugal.

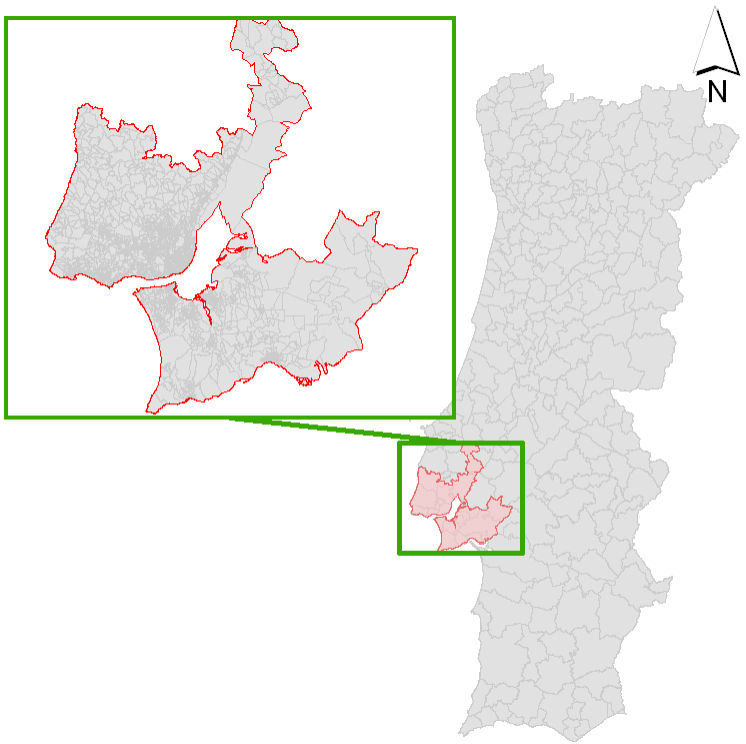


Figure 40 – Lisbon metropolitan area enumeration districts

4.5. GeoSOM Suite tool

The GeoSOM Suite is implemented in MATLAB™ and uses the public domain SOM toolbox (Vesanto, Himberg *et al.* 1999). It consists of a number of MATLAB™ routines (*m-files*) that may be used independently by researchers. A stand-alone graphical user interface (GUI) was built, allowing non-programming users to evaluate the SOM and GeoSOM algorithms. The GeoSOM Suite is freely available at (Lobo, Bação *et al.* 2009). Figure 41 shows the general GeoSOM Suite architecture that consists of: 1) routines to access spatial and non-spatial data; 2) MATLAB™ runtime components, SOM toolbox and core GeoSOM routines; 3) a graphical user interface (GUI) and; 4) routines that produce the output views. These views consist of geographic maps, U-matrices, component planes, hit-map plots, boxplots and parallel coordinate plots, which will be explained later. The GeoSOM Suite allows multiple analysis to be shown at the same time. For example, one may use several different SOM and GeoSOM on the same dataset, and visually compare the results.

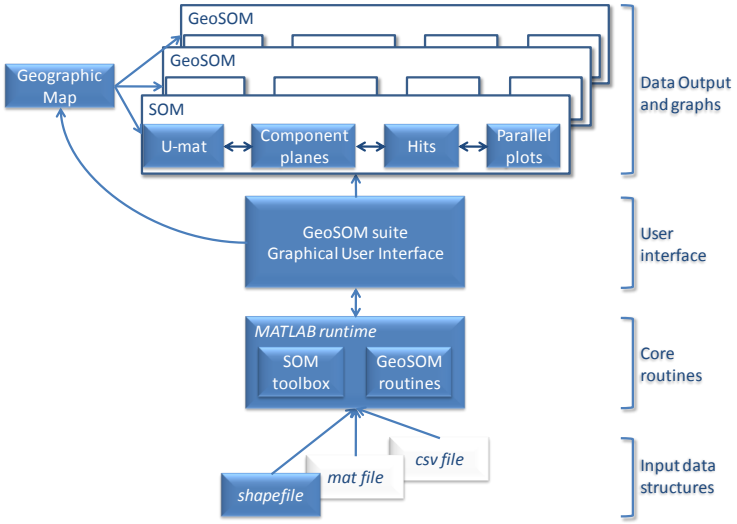


Figure 41 – GeoSOM Suite architecture

Figure 42 presents a screen-shot of the GeoSOM Suite tool. The main window contains a table of attributes and a tree view pointing to all the views created. The figure also shows three examples of views: a geographic data, a U-matrix and a box plot views.

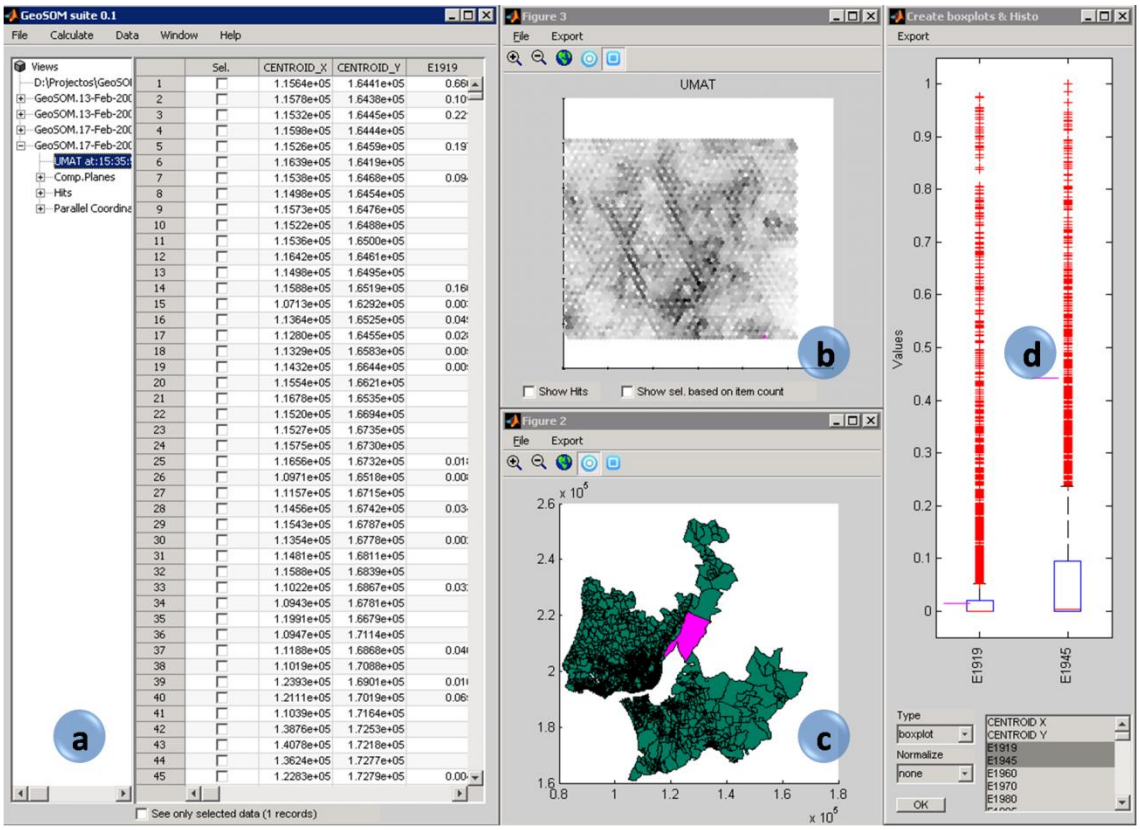


Figure 42 – GeoSOM Suite window. From the left to the right, top to bottom: GeoSOM Suite main window (a) with a tree-list of available analysis, and the full dataset with all attributes; U-matrix (b) obtained using census data; geographic map (c) of Lisbon Metropolitan Area; and a boxplot (d) showing the distribution of two variables

The GeoSOM Suite´s main functionalities are: 1) present spatial data; 2) train SOM and GeoSOM with the chosen data; 3) produce several representations (views) and (4) establish dynamic links between windows, allowing an interactive exploration of the data.

4.5.1. Views

Views are different representations of data allowing the user to analyse it from different perspectives, making interpretation easier. Presently, GeoSOM Suite includes the following views:

- Geographic maps;
- U-matrices;
- Component plane plots;
- Hit-map plots;
- Parallel coordinate plots;

- Boxplots and histograms;

U-matrices (Ultsch and Siemon 1990) have been explained in 2.2.4.4, and are calculated by finding the distances in the input space of neighboring units in the output space. The most common way to visualise them is to use a colour scheme or a grey scale to represent these distances. In this case, black represents the highest value while white represents the lowest value (Figure 43d). Low values in the U-matrix (shown as white areas) are an indication that data density is high, thus there is a cluster of data. High values in the U-matrix (shown as dark areas), are an indication that data density is low, thus there is a separation between clusters.

Component planes (Kohonen 2001), that were explained in 2.2.4.3, are another SOM representation where each unit gets a colour based on the weight of each variable used in the analysis. A component plane exists for each variable showing the units' weights for that variable (Figure 43c). By observing the component planes one can see how a given variable varies along the map. This may be useful, for example, to understand what characterizes each cluster. By comparing two or more component planes, one can visually identify correlations between variables, both globally and at a local scale.

Another possible view in the GeoSOM Suite is the hit-map plot (Kohonen 2001), explained in 2.2.4.5. This representation is usually superimposed on the U-matrix or on the component planes, and gives information about the number of data items represented by each unit, *i.e.* data items with the same BMU (Figure 43c, red hexagons). It can be used to see how a certain set of data points are mapped on the SOM, gaining more information about the clustering structure.

A parallel coordinate plot (Inselberg 1985), already discussed in 2.2.4.7, is a data analysis technique for plotting multivariate data. This technique starts by drawing a parallel axis for each variable. A line connecting each variable axis value then maps each data item (Figure 43e). This allows us to visually compare multivariate data vectors.

Other possible representations in GeoSOM Suite are the boxplots or box plots, also known as box-and-whisker diagrams and histograms (Figure 43b), discussed in 2.2.4.7. Boxplots are 2-dimensional graphics displaying several statistics for each variable: the

smallest observation of a given variable, the lower quartile, the median, the upper quartile, the largest observation and the outliers. Besides boxplots it is possible to plot histograms, which are a graphical display of tabulated frequencies, shown as bars. Figure 43 shows some views created by GeoSOM Suite using the Squareville dataset. In this example, we trained an SOM with 10 x 10 units using the houses coordinates (*x, y*) and the average salary (*s*). Also in this figure, it is possible to notice the dynamically linked property of the views allowing the brushing of data items through different views.

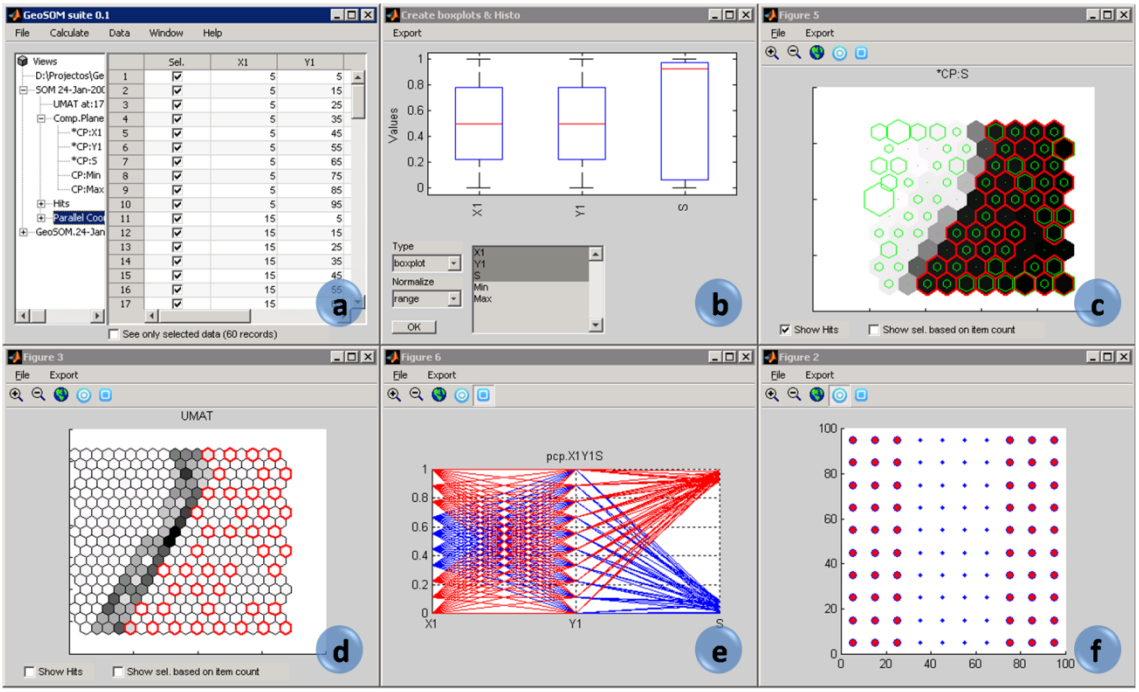


Figure 43 – Dynamically linked views created by GeoSOM Suite (selection made in the U-matrix is in red); a) GeoSOM Suite main interface, with a tabular view of the dataset; b) boxplot view of the three variables; c) the average salary component plane, with a hit-map (in green) superimposed; d) the U-matrix; e) parallel coordinate plot of all the data and f) the geographic map

The use of dynamically linked windows promotes interaction with the data, allowing users to analyse data from different perspectives. Observing the U-matrix (Figure 43d) we detect two well-defined clusters: the two white regions separated by the dark region. Selecting all units belonging to one of the clusters we are implicitly selecting a subset of the original data, and that data will be highlighted in all other views. Thus, when analyzing the salary component plane it is possible to find that the units selected on the U-matrix correspond to the highest value of average salary. This is reinforced by inspecting the parallel coordinate plot, where high average salary houses are selected. In this plot, high and low income houses are mixed in the *y* component, but clearly

separated in the *x* component. Finally, in the geographic map, it is possible to view the spatial distribution of the houses with a higher average salary.

4.5.2. Clustering in the GeoSOM Suite

Clustering with SOM can be made using *k-means SOM* (Bacao, Lobo *et al.* 2005) or *emergent SOM* (Ultsch 2005), as explained in 2.2.3.1. In the first case, each unit is a cluster centroid (similar to *k*-means clustering), while in emergent SOM each cluster is composed of a large number of units, and identified by the borders on a U-Matrix. GeoSOM Suite allows the users to use both methods for clustering. While *k-means* clustering does not require any special tool, to use *emergent SOM* clustering GeoSOM Suite allows the user to delineate the clusters on top of U-matrices. To help users outline clusters, two extra tools are available in GeoSOM Suite: a hierarchical clustering of units and, what we call a *z*-level tool. In the first case, we cluster the units based on distance and position (on the U-matrix) using a hierarchical algorithm (single-linkage), and label each unit on the U-matrix. The *z*-level tool is a simple query on the U-matrix that highlights all units that are below a certain threshold. If we consider the U-matrix in three dimensions (assuming as third dimension the distance between units) high-density areas correspond to valleys while low-density areas correspond to mountains. Thus, clusters match valley zones while mountains are cluster frontiers. After defining clusters on top U-matrices, it is possible to see them on all open views (Figure 44).

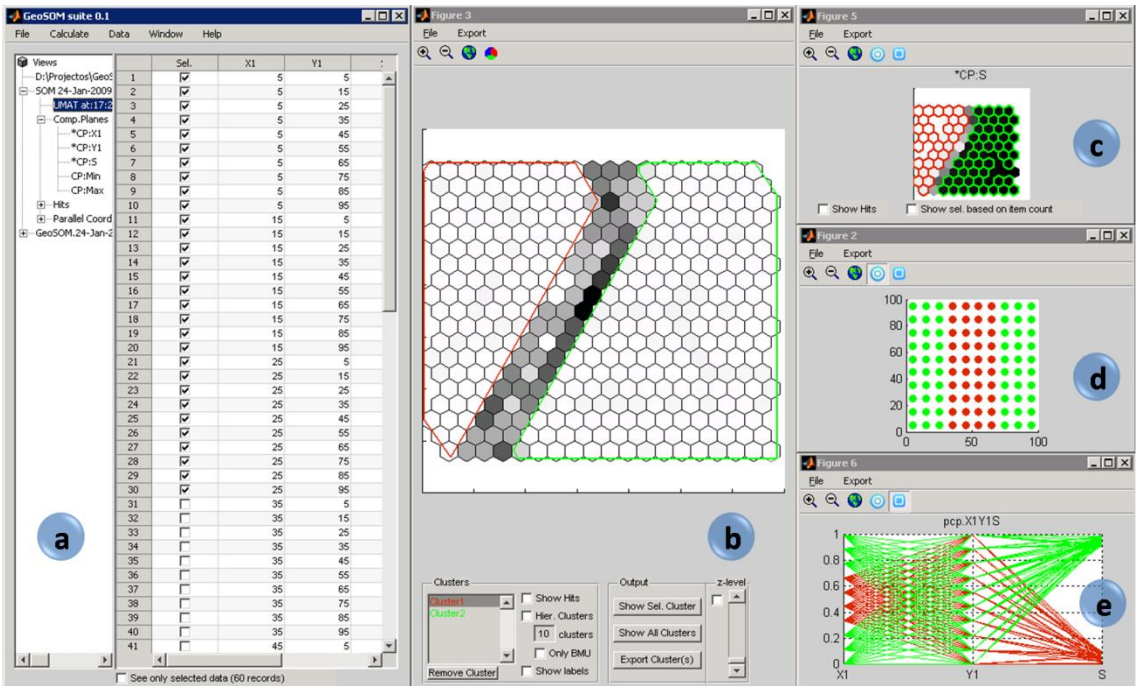


Figure 44 – Defining clusters from a standard SOM trained with Squareville data (a). Two clusters (represented by red and green) are delimited by the user on top of the U-matrix (b) produced from the SOM. The average salary plane (c), the geographic map (d) and the parallel coordinate plot (e) are also presented (right column) showing the clusters

Figure 44 shows the two clusters from the Squareville dataset. From the clusters' outline that is manually drawn on top of the U-matrix (green and red lines) it is possible to analyse the clusters on the salary component plane, on the geographic map and on the parallel coordinate plot.

4.5.3. Clustering spatial data

As shown before, the standard SOM algorithm allows us to detect two clusters in the Squareville dataset. However, a visual inspection of the geographic map will suggest us three clusters. This is because SOM merges in a single cluster the two regions with a high average salary. In fact, their main difference is only spatial and since in this dataset spatial variables have less weight than the average salary, SOM will produce only one cluster. By visualising the high-income cluster in the geographic space, we see that it is not contiguous, and therefore should be separated into two separate clusters. If more weight is given to the geographical variables (and thus less weight to the income) the cluster structure will be lost altogether, as we shall discuss later. To include spatial data and ensure the clusters' contiguity we apply the GeoSOM algorithm (Figure 45).

Figure 45 shows the result of applying GeoSOM algorithm to Squareville data. In this case, three clusters are detected (green, blue and red) matching the two spatially apart regions with high average salary and the middle one with a small average salary.

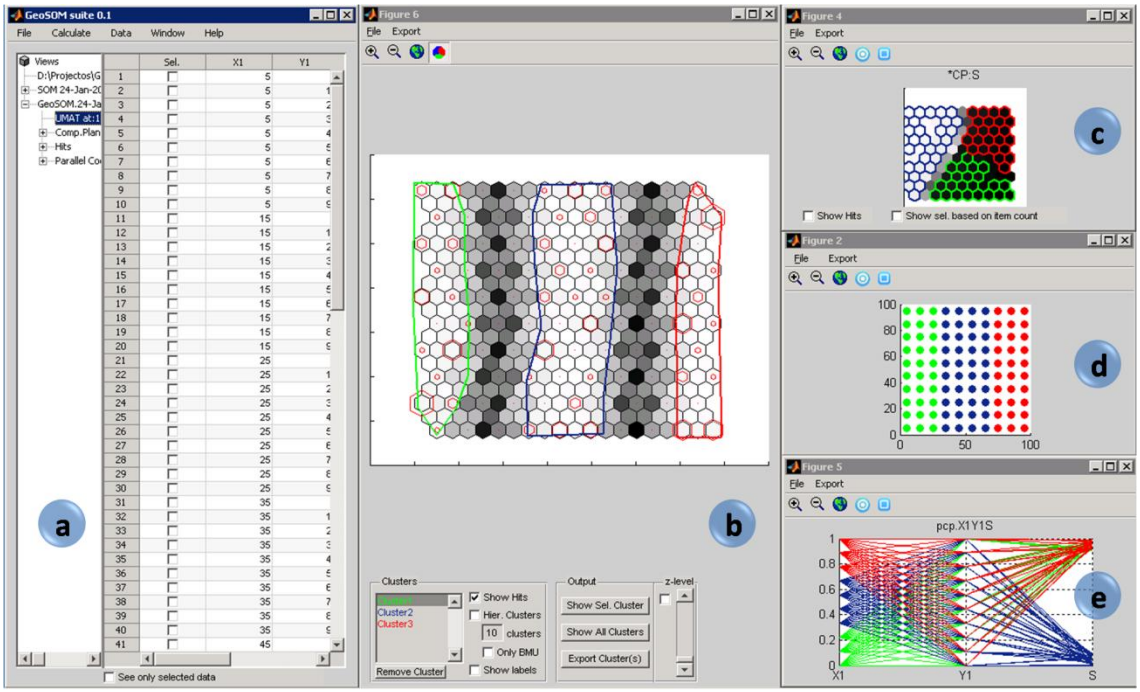


Figure 45 – Defining clusters from GeoSOM method trained with Squareville data (a). Three clusters (represented by green, blue and red) are delimited by the user on top of the U-matrix (b) produced from the SOM. The average salary plane (c), the geographic map (d) and the parallel coordinate plot (e) are also presented (right column) showing the clusters

4.5.4. Combining multiple cluster in GeoSOM Suite

Another analysis possible with GeoSOM Suite is to train several SOM or GeoSOM using the same dataset. This possibility has different applications. First, it allows the comparison between the *k-means* and *emergent SOM* clustering methods using the same dataset. Therefore, the user can compare the clusters produced using a predefined number of clusters or searching for *natural* clusters.

This feature also allows a sensitivity analysis of SOM and GeoSOM by comparing results using different input parameters. Comparisons between SOM and GeoSOM algorithms in clustering can also be performed. Using multiple clustering options at the same time will give the user a better insight on the nature of the data. Using the two examples shown before (Figure 44 and Figure 45), it is possible to compare the SOM

and GeoSOM results (Figure 46), and thus understand the geographical separation of the high income cluster.

Finally, the user might choose to make several SOM's using different subsets of variables. This can be thought of as building different thematic classifications. For the census dataset, for example, one may separately use building characteristics, family characteristics, or unemployment characteristics, which can, in the end, be evaluated together, or as we shall see in the next chapter, hierarchically.

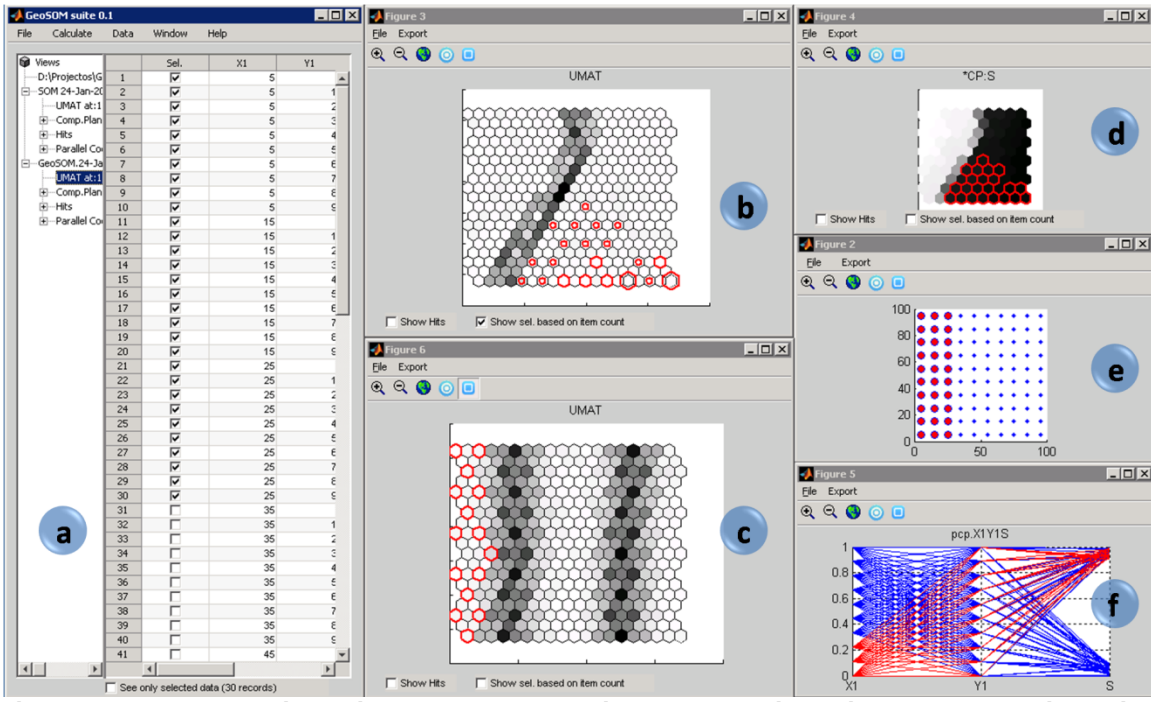


Figure 46 – Comparison between SOM and GeoSOM clustering. GeoSOM has the capability of detecting spatial contiguous clusters, while SOM produces global clusters. The selection in red shows one region with high average salary in the west part of the map. This region is not detected in the SOM due to the presence of another region with similar average salary in another region. a) Main GeoSOM window; b) U-matrix produced from a standard SOM; c) U-Matrix produced from GeoSOM; d) Average salary component plane of the standard SOM; e) Geographic map and f) parallel coordinate plot

Figure 46 shows the comparison between SOM and GeoSOM U-matrices. By selecting one cluster in the GeoSOM U-matrix (an area with a high salary), it is possible to analyse that cluster's distribution on the previously trained SOM. At this point, a reasonable doubt can arise: if SOM does not detect these two clusters due to the low weight of spatial variables, why not increase their weight instead of using a new method (GeoSOM)? The answer to this question is that by increasing the importance of the geographical variables we will have to decrease the importance of the other variables,

and thus blur the distinction between clusters. The clear distinction between clusters will fade away as the importance of the variables that define them decreases. In the limit, if only geographical variables are used, no clusters whatsoever will emerge since the geographical variables vary uniformly along space. At no point along this process will clear cut, geographically separate but otherwise similar clusters arise.

4.6. Case study: Lisbon's census

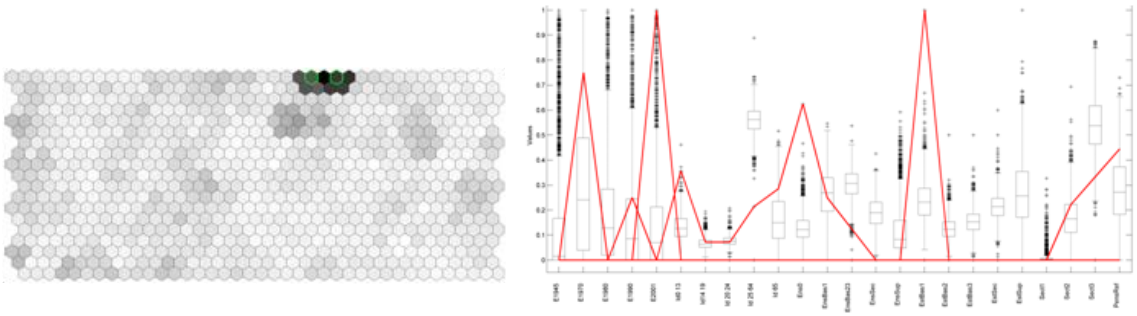
To evaluate GeoSOM Suite with real data we performed a cluster analysis using Lisbon's Metropolitan Area (LMA) 2001 census presented earlier. For this cluster analysis, we used the age of the buildings, the residents age structure, the education level and employment characterization (sector of economy). Table 11 describes the variables used in the cluster analysis.

Table 11 - Variables used in the cluster analysis of LMA census

Category	Variable name	Description
Age of the buildings	E1945	% of buildings built before 1945
	E1970	% of buildings built between 1946 and 1970
	E1980	% of buildings built between 1971 and 1980
	E1990	% of buildings built between 1981 and 1990
	E2001	% of buildings built between 1991 and 2001
Age of residents	Id0_13	% of residents with age bellow 13
	Id14_19	% of residents with age between 14 and 19
	Id_19_24	% of residents with age between 20 and 24
	Id_25_64	% of residents with age between 25 and 64
	Id_65	% of residents with more than 65 years of age
Residents' level of education	Ens0	% of resident with no formal education
	EnsBas1	% of resident with 4 years of education
	EnsBas23	% of resident with 6 to 8 years of education
	EnsSec	% of resident with 12 years of education
	EnsSup	% of resident with higher education
Residents attending school	EstBas1	% of students attending years 1-4
	EstBas2	% of students attending years 5-6
	EstBas3	% of students attending years 7-9
	EstSec	% of students attending years 10-12
	EstSup	% of students attending University
Sector of economy	Sect1	% of residents working in the primary sector
	Sect2	% of residents working in the secondary sector

Category	Variable name	Description
	Sect3	% of residents working in the tertiary sector
	PensRef	% of retired residents

We started by training a 20 x 10 SOM using all enumeration districts (EDs). Figure 47 shows the U-matrix produced and a boxplot for all the variables. Next, we identified outliers by searching for high values in the U-matrix. As can be seen, in this case, the U-matrix has a very dark area (corresponding to very low density of data) at the top. The data that is mapped to this area are clearly outliers. The values of each variable of those EDs, are superimposed on the boxplot, as red lines. As we can see in the boxplot these outliers (represented by the red lines) refer to EDs where most variables are null or deserted areas where a single house can skew the results significantly. These EDs were removed from the original dataset, so that a better understanding of the remaining EDs can be achieved. This process of removing outliers before proceeding with the analysis is quite common, since the outliers will usually *squash* the rest of the data, thus hiding the general structure of the dataset.



a) **b)**

Figure 47 – U-matrix (a) for Lisbon Metropolitan Area SOM and box plot (b) showing the outliers (red features both in U-matrix and in the boxplot)

After removing those outliers a new SOM was trained with the remaining EDs. Again, for this training set, a 20 x 10 SOM was used. Figure 48 shows the U-matrix produced and the component planes for all the variables used.

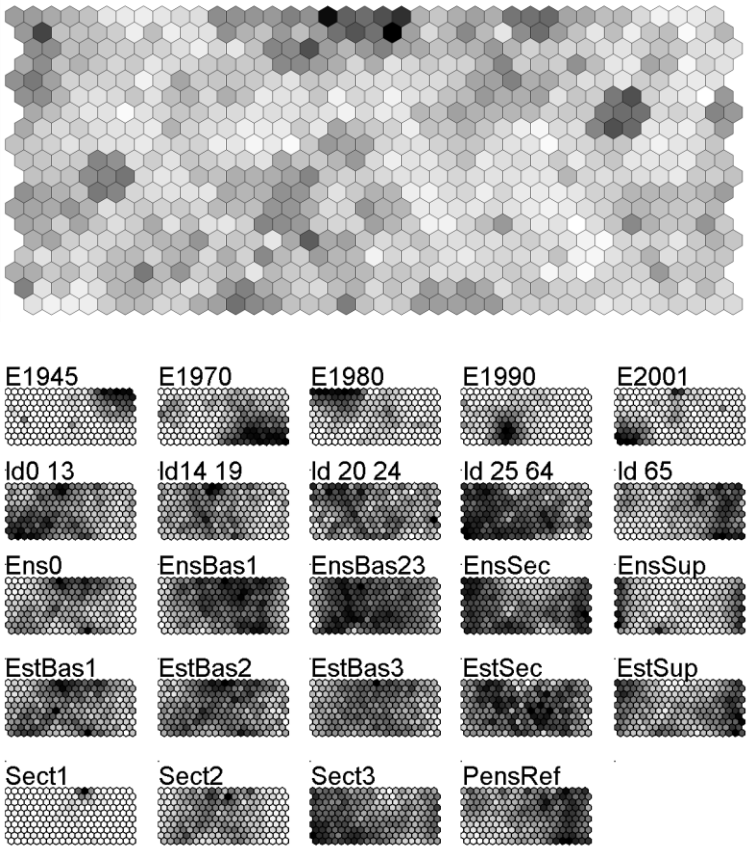


Figure 48 – U-matrix (a) and component planes (b) for Lisbon Metropolitan Area dataset after exclusion of the outliers. The top row of component planes refers to the age of the building. The next row refers to the age of the residents, the third one the student status, the forth the achieved education levels, and the last the employment sector

A detailed cross analysis between the U-matrix and the component planes reveals that the age of buildings is the factor that better defines clear clusters, and thus may be dubbed more *clusterable*. This means that these variables present natural clusters that are easily detected by SOM. Concerning the age of residents it is possible to conclude that young people (<13 years old) are more associated with *newer* areas (buildings built after the 90s) while older people (>65 years old) are found in areas with buildings built before 1970. As expected the number of students is highly related with the age of residents. Figure 49 shows the U-matrix emphasizing units with high values for residents' age, buildings age and education related variables.

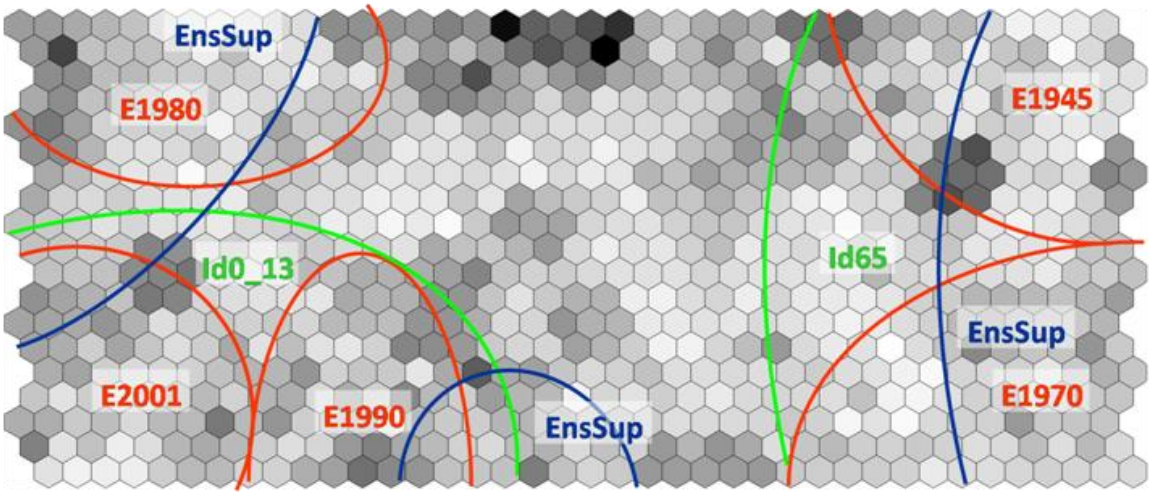


Figure 49 – U-matrix with outlines of some component plane hotspots. Areas of the component planes that have high values are shown with colours (one for each thematic group of variables) on top of the U-matrix. There are two areas where age (in green) plays a predominant role: on the right there is an area with many people over 65, and on the lower left an area with infants (under 13 years of age). There are three areas where education level (in blue) plays a predominant role: on the extreme right, upper left, and middle bottom, there are many people with tertiary education. Finally there are 5 areas (in red) where buildings have a well defined age structure: in the top right there are many old buildings (built before 1945), in the bottom right buildings built in the 60's (before 1970), in the top left, buildings of the 70's (before 1980), in the middle-left bottom the 80's (before 1990), and in the bottom left the 90's (before 2001)

Figure 50 shows the component planes for the variables *Id65* (residents with more than 65 years of age), *E1945* and *E1970* (buildings built before 1945 and between 1945 and 1970, respectively). By selecting units with higher values for variable *Id65,* it is possible, due to dynamically linked windows property of GeoSOM Suite, to verify the corresponding selection on the *E1945* and *E1970* component planes. On the two last component planes the size of the red hexagons represent the number of EDs selected. It is possible to conclude that EDs with elder people are highly related with those with buildings built before 1945 or before 1970. The spatial distribution of these EDs is also shown in Figure 50. The EDs with higher percentages of elder people are located mostly in the centre of Lisbon.

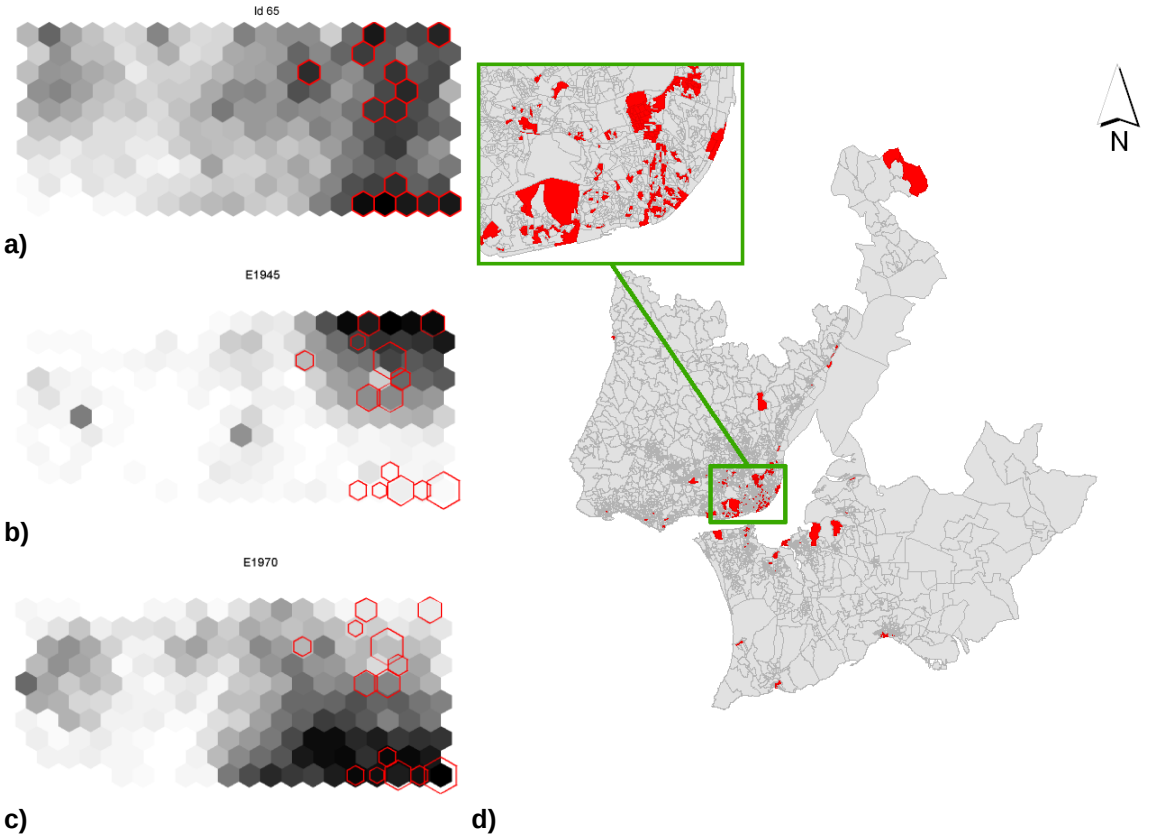


Figure 50 – Component planes for the variables *Id65* (a), *E1945* (b) and *E1970* (c) and Lisbon map (d) showing the selection of the units with higher percentage of elder people

In the following figure (Figure 51) the cluster representing the highest percentage of buildings built before 1945 is selected on the U-matrix. In the same figure, a box plot characterizing this cluster and its spatial distribution is shown.

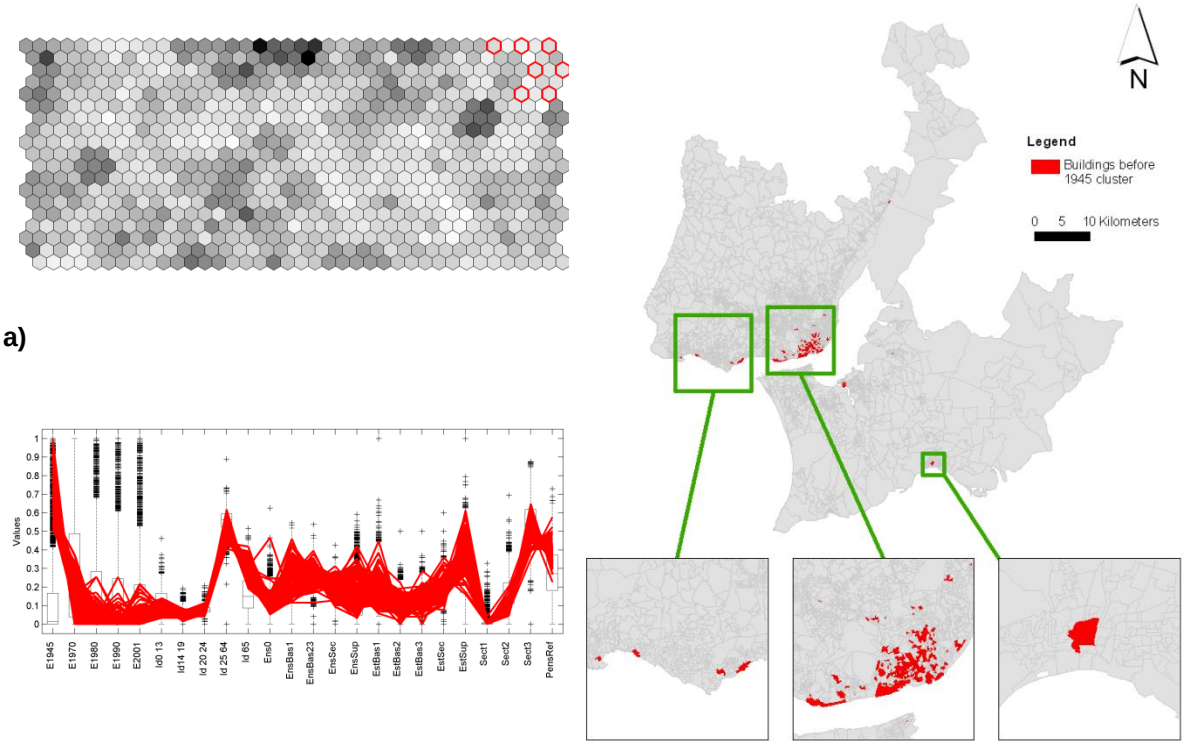


Figure 51 – U-matrix (a), parallel coordinate plot (b) and Lisbon Metropolitan Area map (c) with the highest percentage of buildings built before 1945' enumeration districts in red

From the previous figure, we conclude that the *old buildings* cluster formed by SOM is spatially distributed along Lisbon's Metropolitan Area, matching the oldest town centres (Lisbon, Cascais, Oeiras Almada and Setubal).

However, depending on the final objective, the creation of spatially homogeneous clusters can be an important goal in cluster analysis. If, for example, we wanted to decide where an historical buildings visitor centre should be located, we would like to select a cluster of old buildings that are spatially close to each other. The clusters are thus formed not only by the age dimension (age of the buildings) but also by their spatial location. To obtain these spatially homogenous clusters we applied the GeoSOM algorithm. A 20 x 10 GeoSOM was trained and the U-matrix produced is shown in Figure 52.

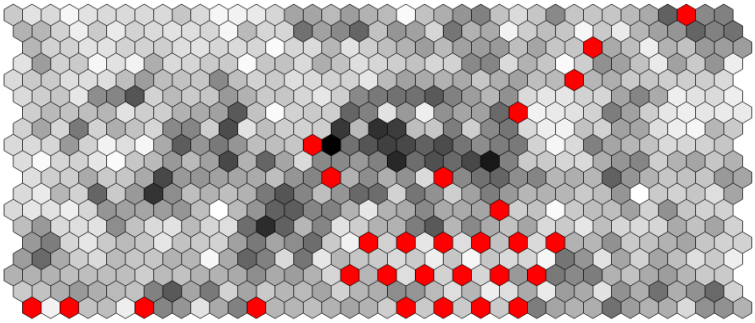
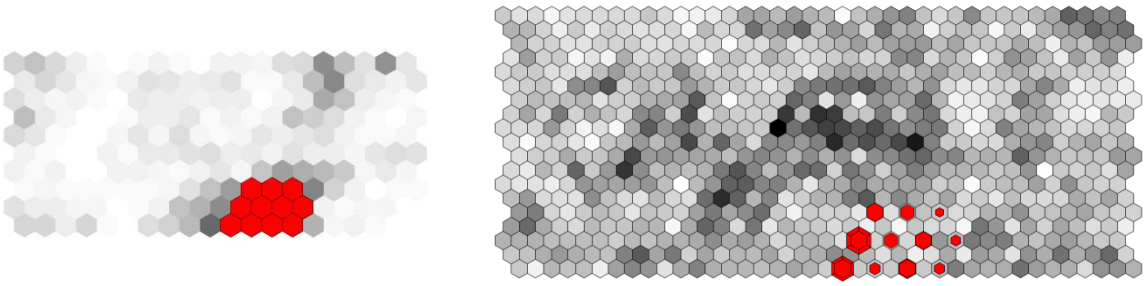


Figure 52 – U-matrix obtained with GeoSOM for Lisbon's Metropolitan Area dataset after exclusion of outliers. The original cluster of old buildings detected by the standard SOM is mapped to the red units

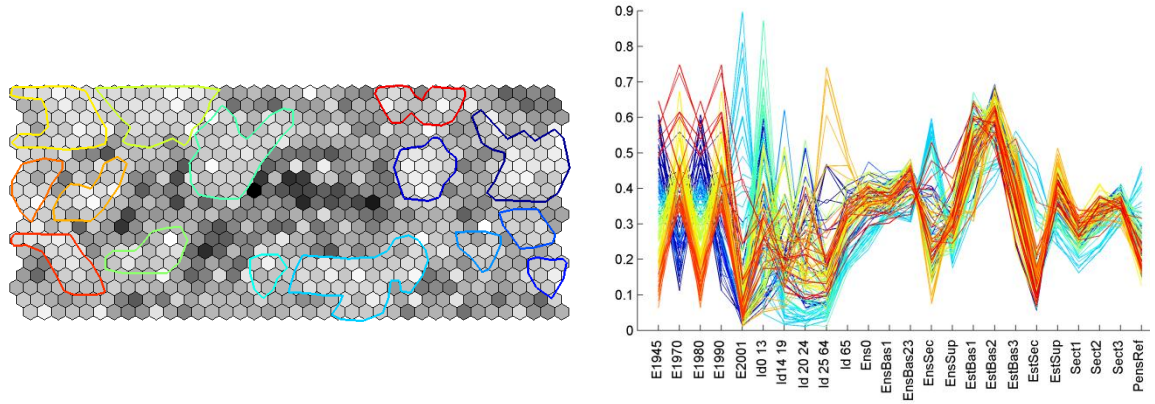
On this U-matrix we selected (red hexagons) the *old buildings* that belong to the cluster detected by the standard SOM. The highlighted GeoSOM units corroborate the fact that this cluster is not spatially contiguous. In other words, since the GeoSOM U-matrix represents the distances between its units, and this distance takes into account the attributes and geographic distances, spatial homogeneous clusters are represented by units close to each other in the U-matrix. Figure 53 shows the ED1945 component plane for the GeoSOM where high value units are selected in red. Corresponding features are also selected on the U-matrix produced from GeoSOM.



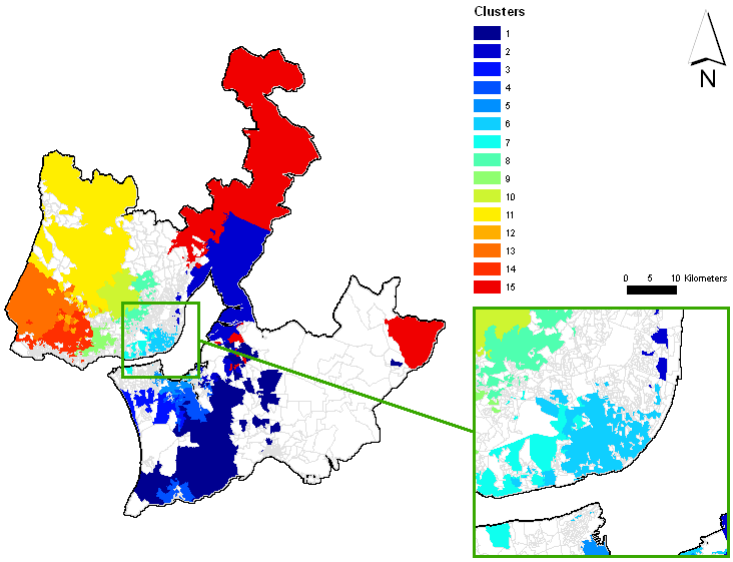
a) **b)**

Figure 53 – Oldest buildings cluster selected on the *ED1945* component plane (a) and on the U-matrix (b)

In Figure 54, clusters were delineated on top of the U-matrix produced from the GeoSOM. From this partition, EDs were coloured on the map according to their respective cluster. A parallel coordinate plot is also shown, characterizing the units belonging to each cluster.



a) **b)**



c)

Figure 54 – Clusters created for Lisbon's Metropolitan Area presented in the: a) U-matrix b) parallel coordinate plot of clustered units and c) Lisbon Metropolitan Area map

Analysing Lisbon's Metropolitan Area 2001 census data we may conclude that:

- Young people (<13 years old) are associated with *newer* areas (buildings built after the 90s), while elder people (>65 years) are found in areas with buildings built before 1970.
- EDs with elder people are located throughout all LMA, but with special focus near Lisbon's centre and centres of old villages such as *Sintra*, *Cascais*, *Oeiras*, *Almada* and *Setubal*.
- SOM and GeoSOM produce different clusters: while SOM groups ED with similar characteristics albeit quite far from each other, GeoSOM groups nearby EDs with similar characteristics.

- Using GeoSOM it is possible to create regions with similar attributes and high spatial autocorrelation (corresponding to GeoSOM clusters), but it is also possible to detect regions where the spatial autocorrelation is low (corresponding to areas outside the clearly defined clusters).

4.7. Discussion

In this chapter, we presented GeoSOM Suite as a new and efficient tool for exploratory spatial data analysis (ESDA) and clustering. This tool implements two major methods, the standard SOM (Kohonen 2001) and the GeoSOM (Bação, Lobo *et al.* 2008). The SOM is a well-known algorithm that has proved to be of interest in spatial clustering. The GeoSOM, by explicitly considering spatial autocorrelation, is able to detect spatial homogeneous and heterogeneous areas. These heterogeneous areas are regions where, although spatial attributes are related (data points are close to each other) non-spatial attributes have little correlation.

GeoSOM Suite implements several visualisation features, all dynamically linked, allowing a strong interaction between user and data, and thus an improved understanding of the data analysed. It is also possible to compare both methods through several views such as U-matrices, component planes, parallel coordinate plots, *etc*.

Spatial clusters were produced from Lisbon's Metropolitan Area 2001 census data using the SOM and GeoSOM methods available in GeoSOM Suite.

5. Hierarchical SOM for geospatial clustering

In this chapter we develop a Hierarchical Self-Organizing Map (HSOM) to perform spatial clustering and more generally, geographic knowledge discovery. We present a general review of HSOMs, and propose a taxonomy for the different variants that can be used. Geospatial data and geo-related analysis can benefit from the use of HSOM's multilayer structure, mainly due to the stratification most geospatial problems present and to the possibility of exploring the same dataset from different thematic perspectives. We present a case study, based on Lisbon's census data, exploring the HSOM and exemplifying its use in the context of exploratory data analysis.

5.1. Introduction

As we have already discussed, the amount of available geospatial data increases every day, placing additional pressure on existing analysis tools. Most of these tools were developed for a data poor environment and thus rarely address concerns of efficiency, high-dimensionality and automatic exploration (Openshaw and Openshaw 1997). Recent technological innovations have dramatically increased the availability of data on location and spatial characterization, fostering the proliferation of huge geospatial databases. To

make the most of this wealth of data we need powerful knowledge discovery tools, but we also need to consider the particular nature of geospatial data. This context has raised new research challenges and difficulties on the analysis of multidimensional geo-referenced data. The availability of methods able to perform “intelligent” data reduction on vast amounts of high dimensional data is a central issue in GISc current research agenda.

The field of knowledge discovery constitutes one of the most relevant stakes in GISc research to develop tools able to deal with “intelligent” data reduction (Miller and Han 2001; Gahegan 2003) and tame complexity. More than prediction tools, we need to develop exploratory tools, which enable an improved understanding of the available data (Openshaw 1994).

The term cluster analysis encompasses a wide group of algorithms (for a comprehensive review see (Jain, Murty *et al.* 1999)). The main goal of such algorithms is to organize data into meaningful structures. This is achieved through the arrangement of data observations into groups based on similarity. These methods have been extensively applied in different research areas including data mining (Fayyad, Piatetsky-Shapiro *et al.* 1996; Miller and Han 2001), pattern recognition (Fukunaga 1990; Duda, Hart *et al.* 2001), and statistical data analysis (Kaufman and Rousseeuw 1990). GISc has also relied heavily on clustering algorithms (Plane and Rogerson 1994; Han, Kamber *et al.* 2001). Research on geodemographics (Openshaw, M. *et al.* 1995; Openshaw and Wymer 1995; Birkin and Clarke 1998; Feng and Flowerdew 1998), identification of deprived areas (Fahmy, Gordon *et al.* 2002), and social services provision (Birkin, Clarke *et al.* 1999) are examples of the relevance that clustering algorithms have within today’s GISc research.

One of the most challenging aspects of clustering is the high dimensionality of most problems. While in general describing phenomena requires the use of many variables, the increase in dimensionality will have a significant impact on the performance of clustering algorithms and the quality of the results. First, it will increase the search space affecting the cluster algorithm efficiency, due to the effect usually known as the “curse of dimensionality” (Bellman 1961). Second, it will yield a more complex analysis of the output, as the clusters are more difficult to characterize due to the contribution of

multiple variables to the final structure. Thus, in a typical clustering problem, the user is asked to select a low number of variables that optimize the phenomena's description.

However, to produce an accurate representation of the phenomenon, it is sometimes necessary to measure it from several perspectives. A typical example is the use of census variables to study the socio-economic environment in urban context. Usually, the census covers a wide range of themes describing the characteristics of the population such as the demography, households, families, housing, economic status, among others (Rees, Martin *et al.* 2002). In these cases, some variables are strongly correlated, independently of the subject they are covering. In fact, with the increase in dimensionality, there is a higher probability of correlation between variables. In addition, due to the spatial context of census data, variables have strong spatial autocorrelation (Goodchild 1986). Spatial autocorrelation measures the degree of dependency amongst observations in a geographic space. This spatial autocorrelation corroborates Tobler's (1970) first law (TFL) which expresses the tendency of nearby objects to be similar.

To GIScientists, clusters are usually more representative and easier to understand if they present spatial contiguity. However, several reasons can cause the clusters to present spatial discontinuity. Among these, the scale or zoning scheme of the geographical units, known as the modifiable areal unit problem (MAUP) (Openshaw 1984), discussed earlier, can affect the expected spatial patterns. In addition, the combination of different variables, that presents distinct levels of spatial autocorrelation, affects the clusters' spatial patterns.

Traditional clustering methods, in which self-organizing maps (Kohonen 2001) are included, are very sensitive to divergent variables. Divergent variables are those that present significant differences to the general tendency. These variables have a great impact on the clustering process and are crucial in the final partition. For instance, when clustering using a set of variables where all, except one, present spatial autocorrelation, the divergent variable will have a higher impact. In most cases, the clusters created will not follow the spatial arrangement suggested by the majority of the variables, but will get distorted by the variables presenting odd spatial distribution.

To avoid this problem we propose the use of a hierarchical structure to explore and cluster geospatial data. Variables are grouped in themes, and each theme will be independently clustered. These partial clusters are then used to create a global partition.

One of the interesting properties of SOM is the capability of detecting small differences between objects. SOMs have proved to be a useful and efficient tool in finding multivariate data outliers (Muñoz and Muruzábal 1998; Nag, Mitra *et al.* 2005; Hadzic, Dillon *et al.* 2007).

In this chapter, we propose the use of Hierarchical SOMs to perform geospatial clustering. Several characteristics of geospatial data make it a good candidate to benefit from the HSOM specific features. The classic layer organization used in GIScience fits perfectly the layered structure of HSOM. HSOM provides an appropriate framework to perform the clustering task based on individual themes, which can then be compared with the clusters created from the combination of several themes. HSOM is less sensitive to divergent variables because these will only have a direct impact on their own theme.

There are many types of hierarchical SOM, so in this chapter we propose a taxonomy to classify existing methods according to their objectives and structure.

This chapter is organized as follows: section 5.2 reviews Hierarchical SOMs. In addition, a taxonomy is presented along with some of the most important methods proposed in the literature. Section 5.3 presents the proposed method used and its implementation in the GeoSOM Suite. Section 5.4 presents the experimental settings adopted to evaluate the proposed method and results are presented in section 5.5. Finally, section 5.6 presents the major conclusions.

5.2. Hierarchical SOM

Hierarchical SOMs (Luttrell 1989; Koikkalainen and Oja 1990; Miikkulainen 1990; Lampinen and Oja 1992; Kemke and Wichert 1993) share many characteristics with other methods such as the multi-layer SOMs (Luttrell 1988; Ichiki, Hagiwara *et al.* 1991), multi-resolution SOMs (Graham and D'Eleuterio 1991), multi-stage SOMs (Li 1989; Lee

and Ersoy 2005), fusion SOMs (Saavedra, Salas *et al.* 2007) or Tree-SOMs (Sauvage 1997).

All these methods share the idea of constructing a system using SOMs as building blocks. They vary in the way these SOMs interact with each other, and with the original data. We consider as Hierarchical SOMs, those where, at some stage, one of the SOMs receives as inputs the outputs of another SOM, as will be described later. This type of structure resembles a multi-layer perceptron (MLP) neural network in the sense that multiple layers exist connected in a feed-forward way. However, Hierarchical SOMs have completely different training algorithms and types of interaction between layers.

General multilayer SOMs may have many completely different interactions between layers. As an example, a data pattern may be mapped onto a given SOM, and then all data patterns mapped to that unit may be visualised on second SOM. Another common type of architecture presents several SOMs in linked windows (Bacao, Lobo *et al.* 2005), providing an environment where a data pattern is visualised simultaneously in several SOMs. We do not consider these as Hierarchical SOM because the outputs of one SOM are not used to actively train another SOM, nor does the second SOM in any way, use information from the first map to map the original data patterns.

We consider that, to be recognized as a Hierarchical SOM, the interaction between different SOMs must be of the train/map type. This type of interaction is one where the outputs of one SOM are used to train the other SOM, and this second one maps (represents) the original data patterns using the outputs of the first one. If these two characteristics are not present, we consider we do not have a true Hierarchical SOM, because it is the train/map relationship that establishes a strict subordination between SOMs, that in turn is necessary for a hierarchy to exist.

The train/map type of interaction encompasses different specific ways of passing information from one SOM to another. As examples, when a data pattern is presented to the first level SOM, it may pass the information onto the second level by passing the index of the best matching unit (BMU), the quantization error, the coordinates of the BMU, all activation values for all units of the first level, or any other type of data. The important issue is that whatever data is passed on, it is used to train the second level SOM. A particular case of output of one SOM layer may be the original data pattern

itself, or an empty data pattern. This is the case of a first level gating SOM that filters which data patterns are sent to each upper level SOM: it may or may not pass the pattern, depending on some characteristic.

Still, many different configurations are possible for Hierarchical SOMs. They may vary in the number of layers used, in the different ways connections are made and even in the information send through each connection.

5.2.1. Why use Hierarchical SOMs?

There are mainly two reasons for using a Hierarchical SOM (HSOM) instead of a standard SOM:

- An HSOM can require less computational effort than a standard SOM to achieve certain goals;
- An HSOM can be better suited to model a problem that has, by its own nature, some sort of hierarchical structure.

The reduction of computational effort can be achieved in two ways: by reducing the dimensionality of the inputs to each SOM, and by reducing the number of units in each SOM. Instead of having a SOM that uses all components of the input patterns, we may have several SOMs, each using a subset of those components, and in this way, we minimize the effect of the *curse of dimensionality* (Bellman 1961). The distance functions used for training the different SOMs will be simpler, and thus faster to compute. This simplicity will more than compensate for the increase in the number of different functions that have to be computed. Speed gains can also be achieved by using less units in each SOM. The finer distinction between different clusters (units) can be achieved in upper level SOMs that will only have to deal with some of the input patterns. This *divide and conquer* strategy will avoid computing distances and neighbourhoods to units that are very different from the input patterns being processed in each instant.

The second reason for using HSOMs is that, in general, they are better suited to deal with problems that present a hierarchical/thematic structure. In these cases, HSOM can map the natural structure of the problem, by using a different SOM for each hierarchical level or thematic plane. This separation of the global clustering or classification problem into different levels may not only represent the true nature of the phenomena (such as

the kingdom/class/family/species levels of a Linnaeus Taxonomy of living beings), but it may also provide an easier interpretation of the results, by allowing the user to see what clustering was performed at each level. GIS science applications, as already discussed, have a strong thematic structure that can be expressed with a different SOM for each theme, and an upper level (hierarchically superior) SOM, that fuses the information to produce globally distinct clusters.

HSOMs are often used in application fields where a structured decomposition into smaller and layered problems is convenient. Some examples include: remote sensing classification (Lee and Ersoy 2005), image compression (Barbalho, Duarte *et al.* 2001), ontology (Ichiki, Hagiwara *et al.* 1991; Chifu and Letia 2008), speech recognition (Kasabov and Peev 1994) pattern classification and extraction using health data (Hanke, Beckmann *et al.* 1996; Douzono, Tokushima *et al.* 2002; Zheng, Ahmad *et al.* 2007), species data (Vallejo, Cody *et al.* 2007), financial data (Tsao and Chou 2008), climate data (Salas, Moreno *et al.* 2007), music data (Carpinteiro 1999; Law and Phon-Amnuaisuk 2008) and electric power data (Carpinteiro and Alves da Silva 2001).

5.2.2. A taxonomy for Hierarchical SOMs

Based on the survey of the work made on the field, we propose the following taxonomy to classify the HSOM methods (Figure 55).

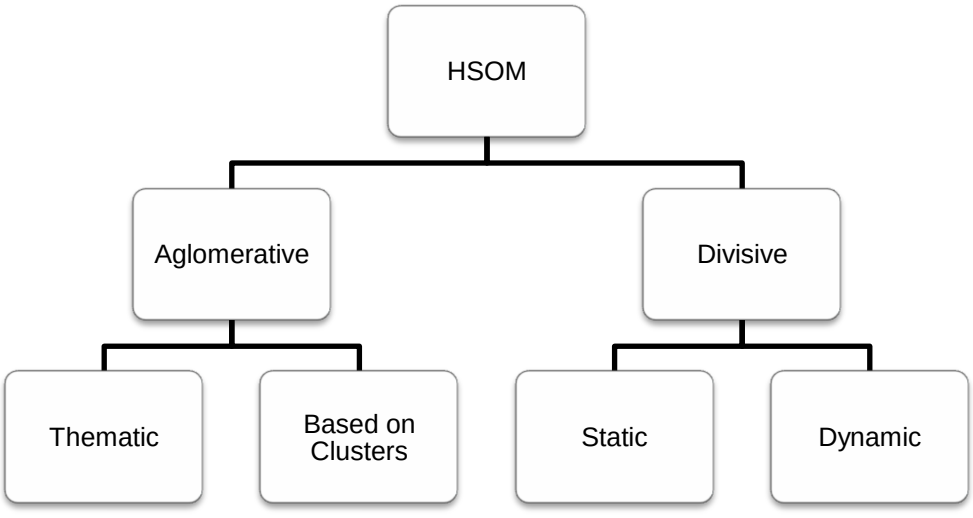


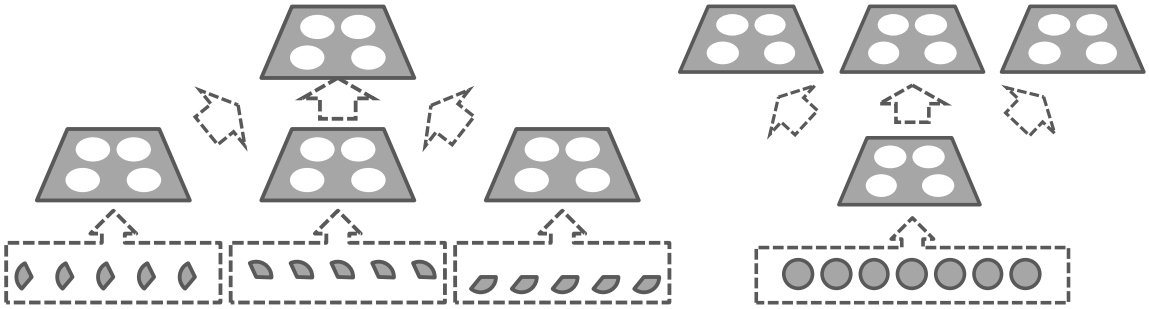
Figure 55 – HSOM taxonomy

This is a possible taxonomy for the HSOM based on their objective and on the type of structure used. Therefore, the first partition groups HSOM methods in two main types:

the agglomerative and divisive HSOMs (Figure 56). This partition results from the type of approach adopted in each HSOM method. In an agglomerative HSOM, we usually have several SOMs in the first layer (*i.e.*, the layer directly connected to the original data patterns), and then fuse the outputs in a higher level SOM, while in the divisive HSOM, we will usually have a single SOM in the first layer, and then have several SOMs in the second layer.

In the agglomerative HSOM (Figure 56a), the level of data abstraction increases as we progress up the hierarchy. Thus, usually the first level on the HSOM is the more detailed representation (or a representation of a particular aspect of the data) and, as we ascend in the structure, the main objective is to create clusters that will be more general and provide a simpler, and arguably easier, way of seeing the data.

In the divisive HSOM, the first level is usually less accurate and uses small networks. The main objective of this level is to create rough partitions, which will be more detailed and accurate as we ascend in the levels of HSOM.



a) Agglomerative HSOM **b) Divisive HSOM**

Figure 56 – Types of hierarchical SOMs: a) agglomerative and; b) divisive

In the second taxonomic level, agglomerative HSOMs can be divided into thematic and based on clusters while divisive HSOMs can be divided into static or dynamic. In the following sections, we will present a description on each category.

2.2.2.1 Thematic Agglomerative HSOM

The first class of agglomerative HSOMs is named thematic. The name results from the fact that the input space is regarded as a collection of subspaces, each one forming a theme. Figure 57 presents a diagram exemplifying how HSOM methods are generally structured in this category.

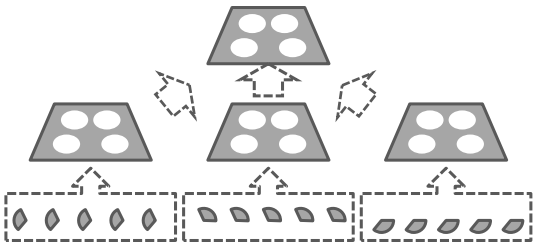


Figure 57 – Thematic HSOMs

In thematic HSOMs, the variables of the input patterns are grouped according to some criteria, forming several themes. For instance, in the case of census data, variables can be grouped into different themes such as economic, social, demographic or other. Each of these themes forms a subspace that is then presented to its own SOM, which's output will be used to train a final merging SOM. As already stated, the type of output sent from the lower level SOM to the upper level can vary in different methods.

In Figure 57, each theme is represented by a subset of the original variables. Assuming that each original data pattern (with all its variables) would get represented by a gray circle, a portion of that circle is used to represent the subset of each data pattern used in each theme.

This structure presents several advantages when performing multidimensional clustering. The first advantage is the reduction of computation caused by the partition of the input space into several themes. This partition also allows the creation of thematic clusters that, *per se*, may be interesting to the analyst. Thus, since different clustering perspectives are presented in the lower level, these can be compared to the global clustering solution allowing the user to better understand and explore the emerging patterns.

5.2.2.1. Agglomerative HSOM based on clusters

Agglomerative HSOM based on clusters are composed by two levels, each using a standard SOM (Figure 58). The first level SOM learns from the original input data, while its output is used in the second level SOM. The second level SOM is usually smaller, allowing a coarser, but probably easier to use, definition of the clusters. In this architecture, if only the coordinates of the bottom level SOM are passed as inputs to the top level, each unit of the top level SOM is BMU for several units from the first level. In this case, the top level is simply clustering together units of the bottom level, and the

final result is similar to using a small standard SOM. However, this method has the advantage of presenting two SOMs mapping the same data with different levels of detail, without having to train the top level directly with the original patterns.

Figure 58 presents the diagram of this category of HSOM.

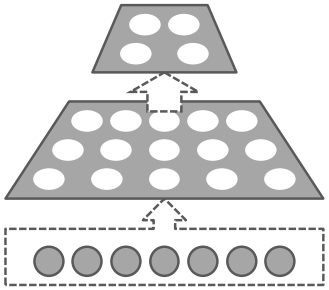


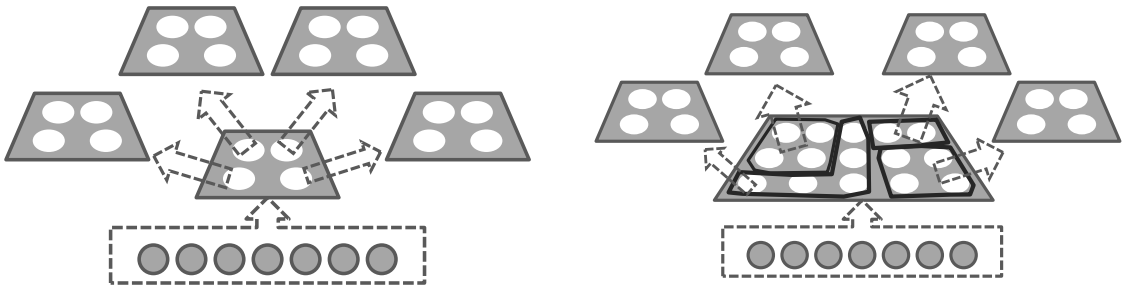
Figure 58 – HSOMs based on clusters

An HSOM based on clusters will be significantly different from a standard SOM if, instead of using only the coordinates of the BMU, more information is passed as input to the top level. As an example, one might use both the coordinates and the quantization error of the input patterns as inputs to the top level. In this case, the top level SOM will probably cluster together patterns that have high quantization error (*i.e.* patterns that are badly represented) in the first level. Thus the top level SOM could be used to detect input patterns that, by being misrepresented in the first level, require further attention.

The name proposed for this class (HSOM based on clusters) stems from the fact that the bottom level SOM uses the full patterns to obtain clusters, and the information about those clusters is the input to the top level SOM. Depending on what cluster information is passed on, the HSOM based on clusters may be similar or very different from the standard SOM.

5.2.2.2. Static divisive HSOM

In this category, HSOM has a static structure, defined by the user. Thus, the number of levels and the connections between SOMs are predefined according to the objective. Figure 59 presents two examples of HSOM structures possible in this category.



a) **b)**

Figure 59 – Static HSOMs: a) structure in which each unit will origin a new SOM and; b) structure in which a group of units will origin a new SOM

In the first case (Figure 59a) the bottom level SOM creates a rough partition of the dataset and, in a second level, an SOM is created for each unit of the first level SOM. Each of these second level SOMs receives as input only the data patterns represented by its origin unit in the bottom level, that acts as a gating device.

In the second case (Figure 59b), each top level SOM receives data from several bottom level units. This allows different levels of detail for different areas of the first level SOM.

The main advantages of Static divisive SOMs over large standard SOMs are the reduction of computational effort due to the small number of first level units (and only some of the top level units will be used in each case), and the possibility of having different detail levels for different areas of the SOM. If, for example, we want to train a 100x100 unit SOM, we may use bottom level 10x10 unit SOM, and a series of 10x10 unit SOMs to form a mosaic in the second level. While each training pattern will require the computation of 10.000 distances in the first case, it will require only 100+100=200 distances in the second.

5.2.2.3. Dynamic divisive HSOM

Finally, the category of dynamic divisive HSOMs is characterized by the structure's self-adaptation to data. These methods, also known as Growing HSOM (Dittenbach, Merkl *et al.* 2002), allow the growth of the structure during the learning phase. Two types of growth are allowed: horizontal and vertical growth. The first increases the number of units of each SOM, while the second increases the number of layers in the HSOM.

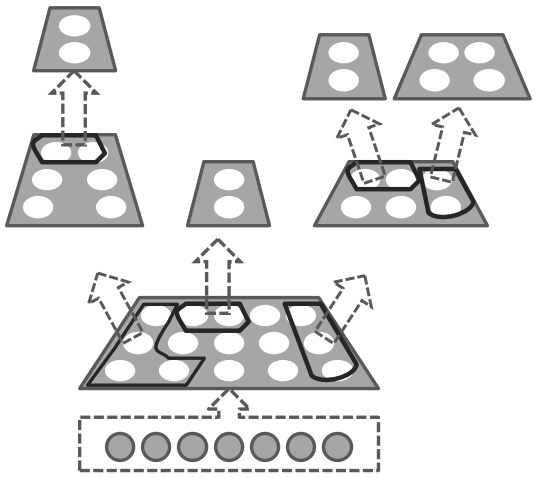


Figure 60 – Dynamic HSOMs

A diagram of this type of HSOM is shown in Figure 60. The size of each level's SOM and the number of levels is defined during the learning phase and relies on some criteria such as the quantization error.

5.2.3. Some HSOM implementations proposed in the literature

One of the first works related to HSOM was proposed in (Luttrell 1989). In this work, hierarchical vector quantization is proposed as a specific case of multistage vector quantization. This work stresses the difference in the input dimensionality between standard and hierarchical vector quantization and proves that distortion in a multistage encoder is minimised by using SOM.

Lampinen and Oja (1992) analyse the HSOM as a clustering tool. The structure proposed is based on choosing, for each input vector, the index of the best-matching unit from the first level to train the second level map. The first level produces many small mini-clusters, while the second produces a smaller number of broader and more understandable clusters.

HSOM has proved to be quite valuable for processing temporal data, often using different time scales at different hierarchical levels. An example is the work presented in (Carpinteiro 1999; Carpinteiro and Alves da Silva 2001), where the authors use HSOM to perform sequence classification and discrimination in musical and electric power load

data. Another example is where HSOM is used to process sleep apnea data (Guimarães and Urfer 2000).

Another class of HSOM is proposed in (Dittenbach, Merkl *et al.* 2002) with the Growing Hierarchical Self-Organizing Map (GHSOM). This neural network model is composed of several SOMs, each of which is allowed to grow vertically and horizontally. During the training process, until a certain criterion is met, each SOM is allowed to grow in size (horizontal growth) and the number of layers is allowed to grow (vertical growth) to form a layered architecture such that relations between input data patterns are further detailed at higher levels of the structure. One of the problems of GHSOM is the definition of the two thresholds used to control the two types of growth. Several authors proposed some variants to this method to better define these criteria. One example is the Enrich-GHSOM (Chifu and Letia 2008). Its main difference is the possibility to force the growth of the hierarchy along some predefined paths. This model is classifying data into a pre-defined taxonomic structure. Another example of a GHSOM variant is the RoFlex-HSOM extension (Salas, Moreno *et al.* 2007). This method is suited to non-stationary time-dependent environments by incorporating robustness and flexibility in the incremental learning algorithm. RoFlex-HSOM exhibits plasticity when finding the structure of the data, and gradually forgets (but not catastrophically) previous learned patterns. Also Pampalk *et.al.* (2004) proposed a Tension and Mapping Ratio extension (*TMR*) to the GHSOM. Two new indexes are introduced, the mapping ratio (*MR*) and the tension (*T*) that will control the growth of the GHSOM. *MR* measures the ratio of input patterns that get better represented by a virtual unit, placed between two existing units. *T* measures how similar are the distances between all the units.

Another example of HSOM is proposed in (Suganthan 1999) with the Hierarchical Overlapped SOM (HOSOM). The process starts by using just one SOM. After completing the unsupervised learning phase, each unit is labelled. Then, a supervised learning method is used (LVQ2) and units are merged or removed, based on the number of mapped patterns. After this, a new LVQ2 is applied and, based on the classification quality, additional layers can be created. The process is then repeated for each of these layers.

A similar structure is presented in (Endo, Ueno *et al.* 2002), which proposes a cooperative learning algorithm for the hierarchical SOM. In the first layer, some BMUs

are selected, and for each of these BMUs a SOM in the second layer is created. Input patterns used in this second level SOM are derived from the original BMU.

Ichiki *et.al* (1991) propose a hierarchical SOM do deal with semantic maps. In this proposal, each input pattern is composed by two parts: the attribute and the symbol, $\boldsymbol{X_i} = [\boldsymbol{X_{ai}} \quad \boldsymbol{X_{si}}]$**.** The attribute part $\boldsymbol{X_{ai}}$ is composed by the variables describing the input pattern, while the symbol part $\boldsymbol{X_{si}}$ is a binary vector. The first level SOM is trained using both parts of the patterns, while the second level SOM only uses the symbol set and information from the first level.

HSOM has also been used for phoneme recognition (Kasabov and Peev 1994). The authors use sound signal attributes in a first level SOM to classify the phonemes into pause, vocalised phoneme, non-vocalised phoneme, and fricative segment. After phonemes are classified, a feature frequency-scale vector is used to train the corresponding second level SOM.

A different approach called tree structured topological feature map (TSTFM) is presented in (Koikkalainen and Oja 1990). This approach uses a hierarchical structure to search for the BMU, thus reducing computation times. While the purpose of this approach is strictly to reduce computation times, its tree searching strategy is in effect a series of static divisive HSOMs.

Miikkulainen (1990) proposes a hierarchical feature map to recognize an input story (text) as an instance of a particular script by classifying it in three levels: scripts, tracks and role bindings. At the lowest level, a standard SOM is used for a gross classification of the scripts. The second level SOMs receives only the input patterns relative to its scripts, and different tracks are classified at this level. Finally, in the third level a role classification is made.

Table 12 provides a classification using the proposed taxonomy for HSOM methods presented before.

Table 12 - Comparison table of HSOM methods

Method	Classification in proposed taxonomy		Main objective
	1st level	2nd level	
(Luttrell 1989)	Agglomerative	thematic	Vector quantization
(Lampinen and Oja 1992)	Agglomerative	cluster based	Clustering
(Carpinteiro 1999; Carpinteiro and Alves da Silva 2001)	Agglomerative	cluster based	Sequential data classification and discrimination
(Dittenbach, Merkl *et al.* 2002)	Divisive	dynamic	Exploratory data mining
(Chifu and Letia 2008)	Divisive	static	Exploratory data mining
(Salas, Moreno *et al.* 2007)	Divisive	dynamic	Exploratory data mining
(Pampalk, Widmer *et al.* 2004)	Divisive	dynamic	Exploratory data mining
(Suganthan 1999)	Divisive	static	Exploratory data mining
(Endo, Ueno *et al.* 2002)	Divisive	static	Clustering
(Ichiki, Hagiwara *et al.* 1991)	Agglomerative	cluster based	Create Semantic maps
(Kasabov and Peev 1994)	Agglomerative	thematic	Phoneme recognition
(Koikkalainen and Oja 1990)	Divisive	static	Clustering
(Law and Phon-Amnuaisuk 2008)	Agglomerative	thematic	Capture the various levels of information in a musical piece
(Miikkulainen 1990)	Divisive	static	Story recognition

5.3. Proposed method

We propose the use of a hierarchical SOM in geospatial clustering. More precisely, we will use a thematic agglomerative hierarchical SOM (see taxonomy in Figure 55), and refer to it simply as HSOM. Figure 61 presents a scheme of the HSOM where several thematic SOMs are created, according to the themes used.

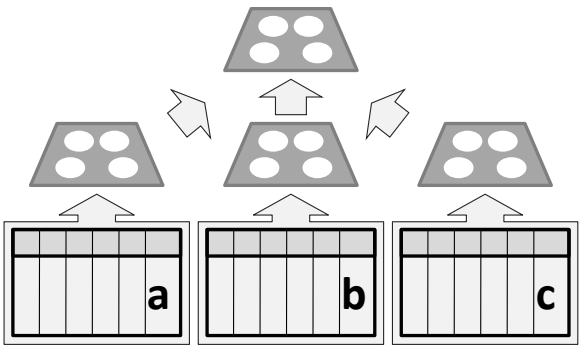


Figure 61 – Hierarchical SOM (HSOM) used. Labels *a*, *b* and *c* refer to different themes

This HSOM divides the input data space into several subspaces according to different themes. Figure 61 shows an example of HSOM using three themes: *a*, *b* and *c*. Each of these themes can be viewed as a subspace created by a subset of variables from the dataset. For instance, if theme *a* is demography, some of the possible variables to use in it are the age structure, the number of inhabitants, the number of births, *etc*..

Each of these subspaces data is used to train a SOM, which's output will be used to train a final merging SOM. When compared to the standard SOM, this approach has the advantage of setting an equal weight for each theme.

Generally, HSOM can be described as follows:

Let
$\boldsymbol{X}$ be the set of n training patterns $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_n}$.
$\boldsymbol{x_i}$ be a vector with $\boldsymbol{m}$ components $\boldsymbol{d_1}, \ldots \boldsymbol{d_m}$
t be a theme composed by k_t components of $\boldsymbol{x_i}$ from $\boldsymbol{d_1}, \ldots \boldsymbol{d_{k_t}}$
S_t be a thematic SOM map relative to the theme t, *i.e.* a SOM trained with the components of $\boldsymbol{x_i}$ belonging to theme $\boldsymbol{t}$.
$\boldsymbol{o_i}$ be the image of $\boldsymbol{x_i}$ in the maps S_t, *i.e.* the concatenation of the outputs of all the maps S_t when pattern $\boldsymbol{x_i}$ is presented.
$\boldsymbol{O}$ be the set of all $\boldsymbol{o_i}$. This set constitutes the modified training set for the top level SOM.

Do
1 For each theme t
2 Train each thematic SOM map S_t in a standard way using as input the relevant components of $\boldsymbol{X}$.
3 Create the set of modified training patterns $\boldsymbol{O}$ as a concatenation of the possible outputs of maps S_t, using for each input pattern:
  a. The coordinates of its BMU.
  b. Its quantization error.
  c. Its distance to each unit(*i.e.*, all quantization errors).
4 Train the top level SOM using as input the set of modified training patterns $\boldsymbol{O}$.

5.3.1. GeoSOM Suite's HSOM implementation

HSOM was implemented in the GeoSOM Suite as shown in Figure 62. GeoSOM Suite presents an interface where the user can choose the HSOM inputs, based on the SOMs created before, and/or the original variables. Thus, to create a structure like the one

presented in Figure 61, the user must create three first level SOMs. Each of these SOMs will use the variables relative to one theme. Then the user can create the HSOM by choosing as input data the outputs obtained from the three SOMs. Figure 62 presents a screen-shot of GeoSOM Suite in which this selection and the HSOM parameterisation is shown.

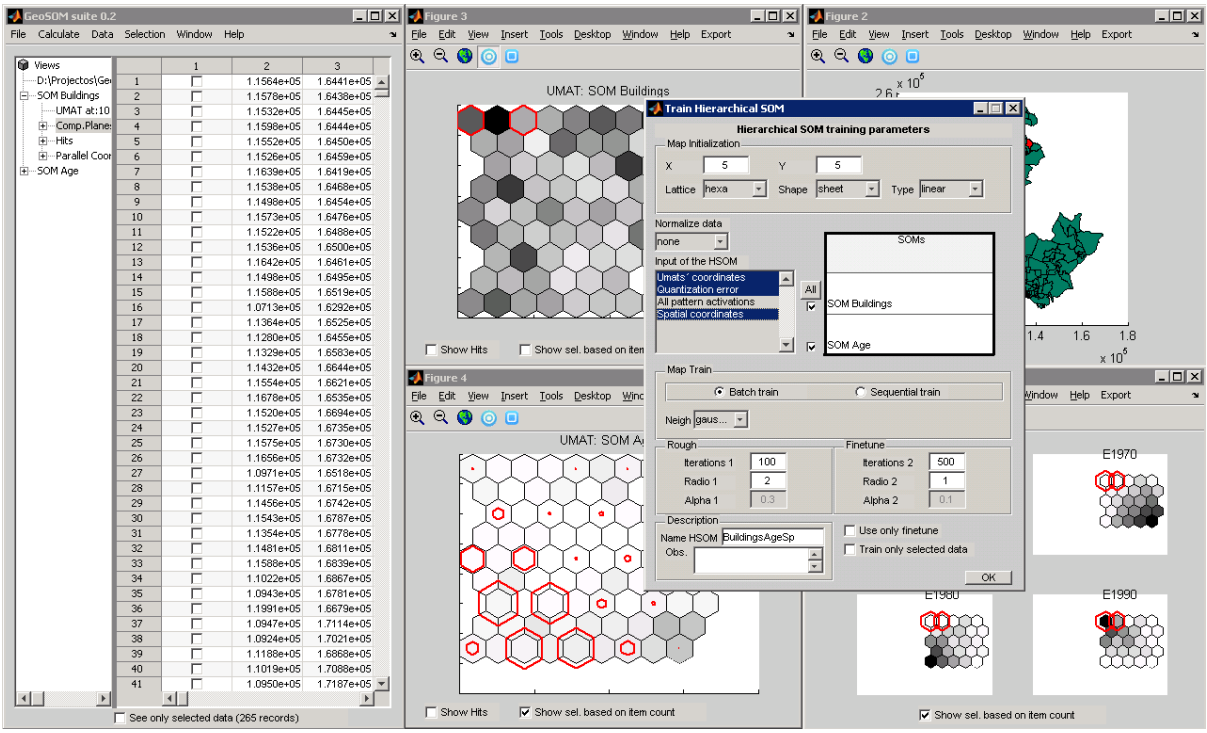


Figure 62 – HSOM implementation in GeoSOM Suite. In this example, two SOMs are trained using buildings and population age data. An HSOM is parameterised using these two SOM's outputs (BMU coordinates and quantization error) and the geographical coordinates of each ED

5.4. Experimental settings

The HSOM method was tested using the 2001 census data from Lisbon's Metropolitan Area (LMA) from the Portuguese Statistical Institute (Statistics Portugal), already used in the previous chapter. This dataset is composed by 3978 enumeration districts (ED) (*secções estatisticas* in Portuguese) describing buildings, families, households, age and education levels and economic activities through 70 variables. Figure 63 presents a map with LMA delineation and relative location within Portugal.

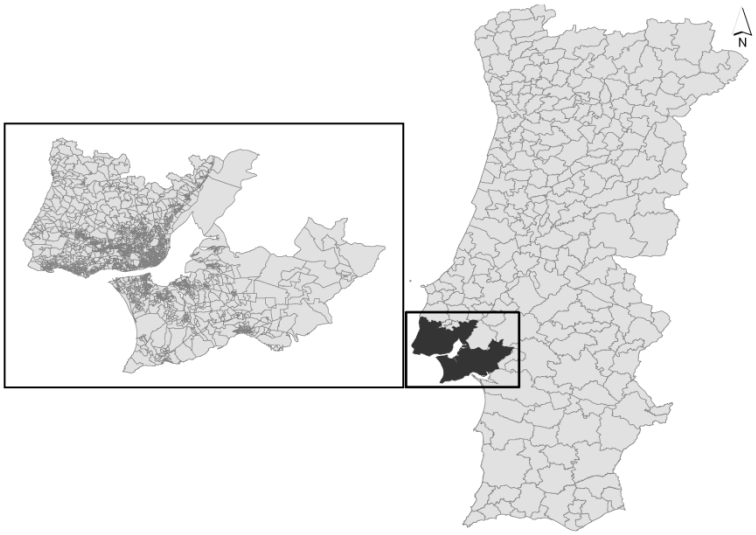


Figure 63 – Lisbon metropolitan area enumeration districts

As already defined, the proposed HSOM method uses thematic subspaces by grouping the original variables. 0 presents the variables used in each of the themes created.

To test the HSOM method, a standard SOM using the 70 original variables was used as benchmark. Table 13 presents the parameters used in both methods.

Table 13 - Parameters used in the SOM and HSOM tests.

Parameters	1st training phase	2nd training phase
Type of training	Sequential	
Size of the map	15 x 10 (in HSOM all 6 first level SOMs also have 15x10)	
Epochs	50	100
Initial neighbourhood radius	8	3
Learning rate	0.5	0.1

In the case of the HSOM, the input data used is based on the different themes created: type of lodgings; buildings (age, size and type of construction); type of families; age structure of inhabitants; education level; and employment. Thus, six standard SOMs were created and trained using the respective theme variables as input data. A top level SOM is then trained using as input the BMU's coordinates of the input patterns on each thematic SOM. Although GeoSOM Suite also has the possibility of using the BMU's quantization error or the distance from patterns to the SOM's units, at this phase we just used the BMU's coordinates.

5.5. Results

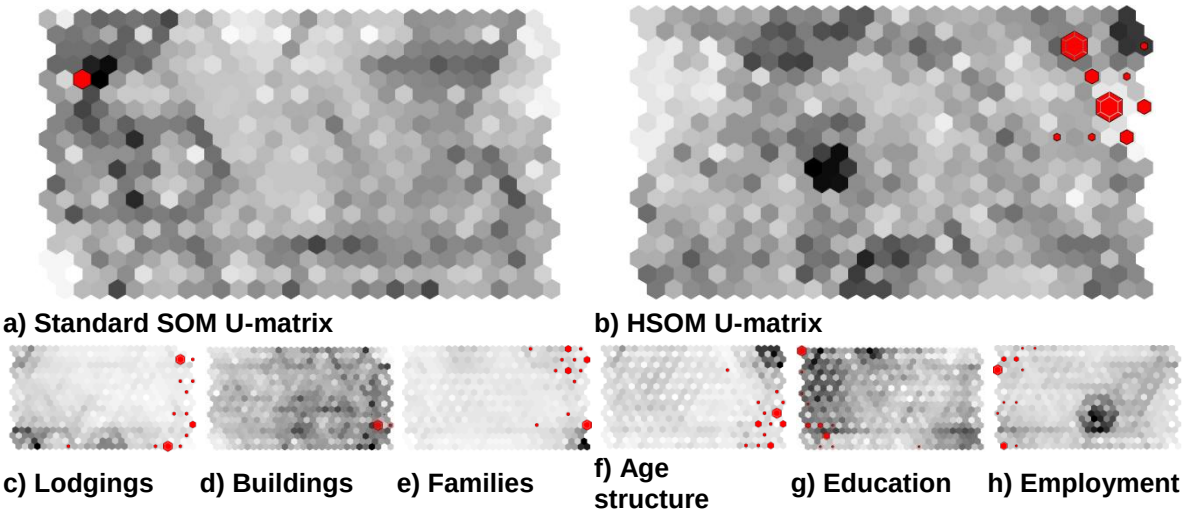
In this section, we present the results obtained with HSOM and with the standard SOM methods. The results are evaluated in two phases: in the first phase, we present a qualitative analysis of the partitions while in the second phase we quantitatively evaluate the results.

5.5.1. Qualitative evaluation

In this analysis, we use two different approaches to compare the results: 1) we start by analysing the outlier regions identified in each method and; 2) we identify one specific neighbourhood and compare its classification and neighbours in both methods.

5.5.1.1. Outliers analysis

Figure 64 presents the U-matrices created from the standard SOM and the HSOM methods. Also in Figure 64(c to h) we present the thematic U-matrices from HSOM, trained using each of the thematic subspaces. We start by analysing the most outlying unit from the standard SOM, identified as such on its U-matrix. The selection of this unit (Figure 64a) allows the identification of the set of EDs best represented by this unit, which are therefore updated on the HSOM U-matrix (Figure 64b) and consequently on the thematic SOMs.



a) Standard SOM U-matrix **b) HSOM U-matrix**

c) Lodgings **d) Buildings** **e) Families** **f) Age structure** **g) Education** **h) Employment**

Figure 64 – Visualisation of U-Matrices. Outlier selection (in red) on the a) Standard SOM, and selection update on: b) HSOM; c) Lodgings; d) Buildings; e) Families; f) Age structure; g) Education level and; h) Employment

From the analysis of Figure 64, we verify that the set of EDs identified in the standard SOM as outliers, are not outliers in the HSOM U-matrix. In fact, these EDs are mapped in a well-defined cluster region in the HSOM (Figure 64b). Analysing the thematic U-matrices from HSOM allows us to conclude that these EDs are not particularly outliers in any of the themes used.

Figure 65 presents a dataset boxplot for all the variables (in grey) used in the training phase, as well as the mean of the selected EDs (thick black line).

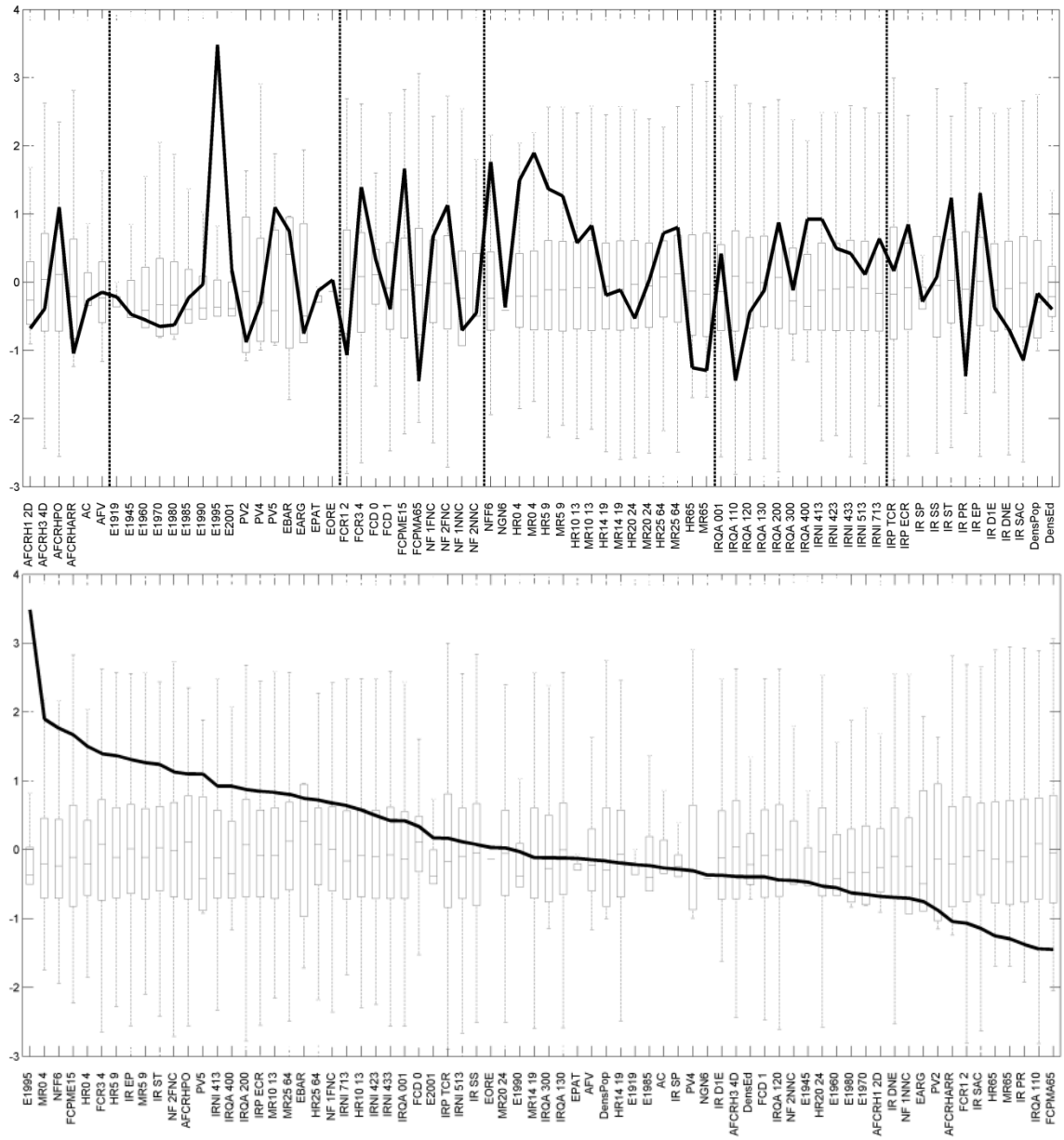
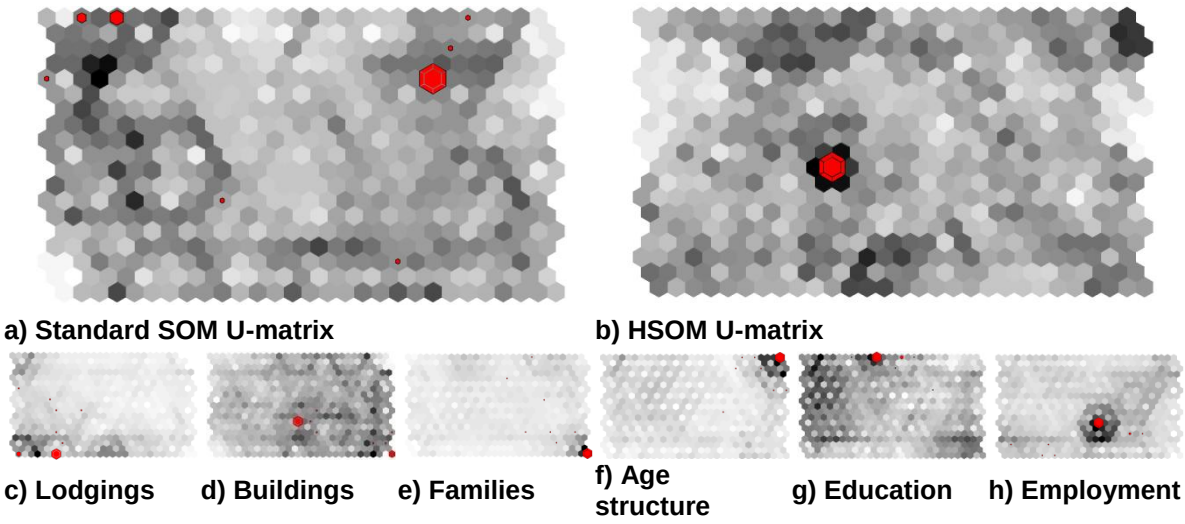


Figure 65 – Boxplot of all the variables used showing their distribution in the dataset (in grey). The black line connects the mean value of the selected EDs for each variable. The top graph has the variables grouped by themes, while in the bottom graph they are ordered by decreasing difference between the selection, and total average

Analysing the variables' mean for the selected EDs, we conclude that some variables caused SOM to consider these EDs as outliers. These variables have a great influence in the outliers' definition due to the very high and very low values presented. The variables presenting values far above the dataset mean are the percentage of buildings built in the first part of the 90s (*E1995*), the percentage of women under four years old (*MR0_4*) and the percentage of nucleus with sons under the age of six (*NFF6*). On the other hand, the variables that present values far below the mean are the percentage of

pensioners (*IR_PR*), the percentage of individuals with only the 1st degree of basic education (*IRQA_110*) and the percentage of families with more than one person with an age of over 65 (*FCPMA65*).

In an analogous way, Figure 66 presents the selection of the most outlying unit in the HSOM. Again, the set of EDs best represented by this unit was selected on all U-matrices.



a) Standard SOM U-matrix **b) HSOM U-matrix**

c) Lodgings **d) Buildings** **e) Families** **f) Age structure** **g) Education** **h) Employment**

Figure 66 – Visualisation of U-Matrices. Outlier selection (in red) on the b) HSOM, and selection update on: a) Standard SOM; c) Lodgings; d) Buildings; e) Families; f) Age structure; g) Education level and; h) Employment

The outlier EDs identified by the HSOM, present a slightly disperse distribution in the standard SOM. This means that these EDs are not all in the same cluster (Figure 66a). It is also possible to see that these EDs were identified as outliers in several themes, such as in the families, the age structure, the education and the employment themes.

This allows us to conclude that, contrary to the standard SOM, the outliers identified by the HSOM are not outliers due to extreme values in one or a few variables, but to strange values in several themes. Thus, we can conclude that while the SOM has a great capability of detecting EDs with extreme values in one or a few variables, HSOM is more concerned with finding EDs with extreme values in most variables, or more correctly in most themes.

5.5.1.2. Neighbourhood analysis

In this section, a particular neighbourhood was identified and characterized using both methods. *Bela Vista* is a neighbourhood near *Setúbal* on the south region of LMA. *Bela Vista* (which means *nice view* in Portuguese) was built in the 70s to accommodate people from shantytowns and refugees from the Portuguese ex-colonies. Today, it is one of the most problematic neighbourhoods in LMA, with several recent confrontations between the locals and the police.

Figure 67 presents the location of *Bela Vista* in the LMA.

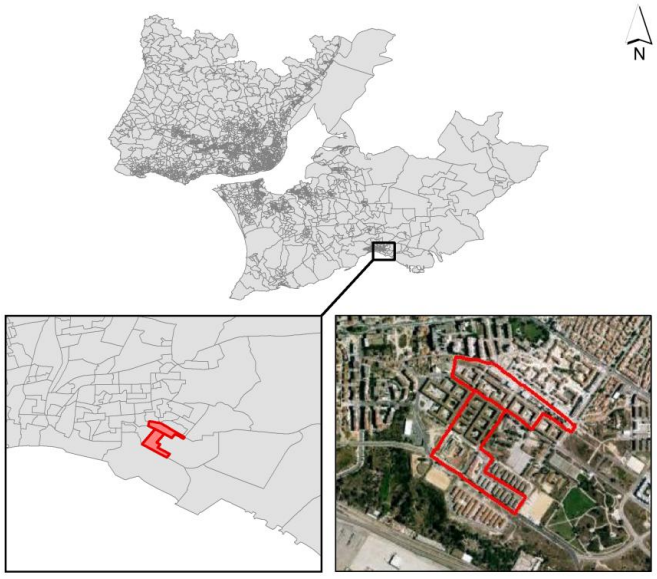
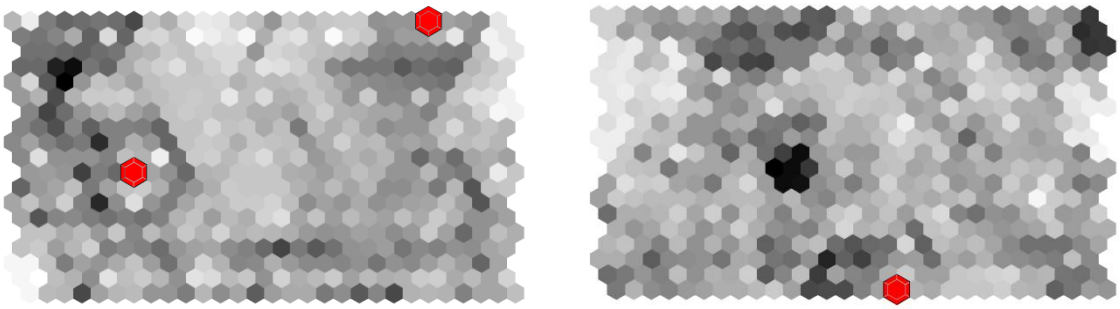


Figure 67 – *Bela Vista* neighbourhood

Let us start by selecting two EDs belonging to *Bela Vista* neighbourhood and compare their positions in the U-matrices created from HSOM and SOM (Figure 68).



a) Standard SOM U-matrix **b) HSOM U-matrix**

Figure 68 – Selection (in red) of two EDs belonging to the *Bela Vista*: a) U-matrix from the standard SOM and b) U-matrix created from the HSOM

As we can see in Figure 68, while these EDs share the same cluster in the HSOM they are represented by two distant units in the standard SOM. This means that from the SOM's perspective, these two EDs belong to very different clusters.

Analysing the variables from these two EDs (Figure 69), we can see that these EDs present significant differences in the:

- the percentage of rented houses (*AFCRHARR*)
- percentage of buildings built in concrete (*EBAR*)
- percentage of families with 1 person unemployed (*FCD_1*)
- percentage of nucleus with one (*NF1FNC*) or two (*NF2FNC*) unmarried grandchildren

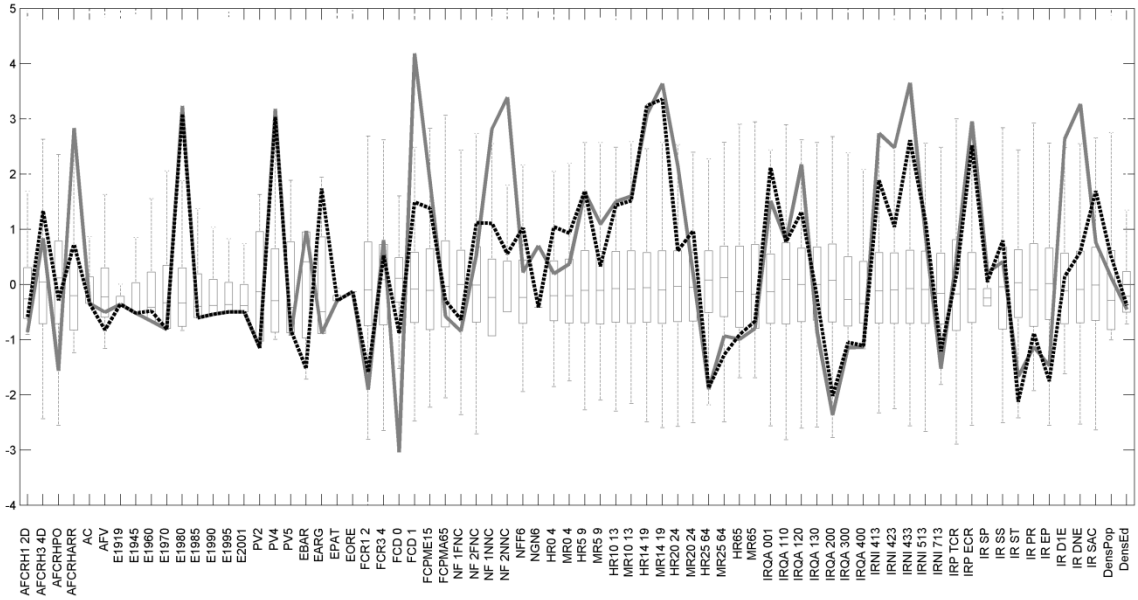
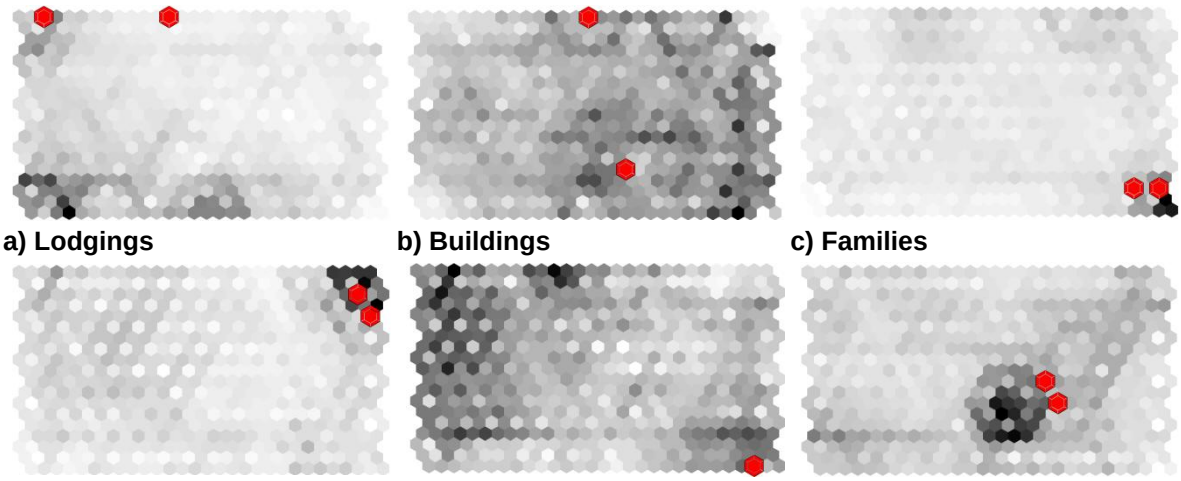


Figure 69 – Characterization of two EDs belonging to *Bela Vista* neighbourhood using a PCP

By now, a reasonable doubt can arise: why are the two EDS in the same unit in the HSOM if they present distinct characteristics? The answer to that question is illustrated in Figure 70, where we present the selected EDs mapped on the thematic U-matrices.



a) Lodgings **b) Buildings** **c) Families**

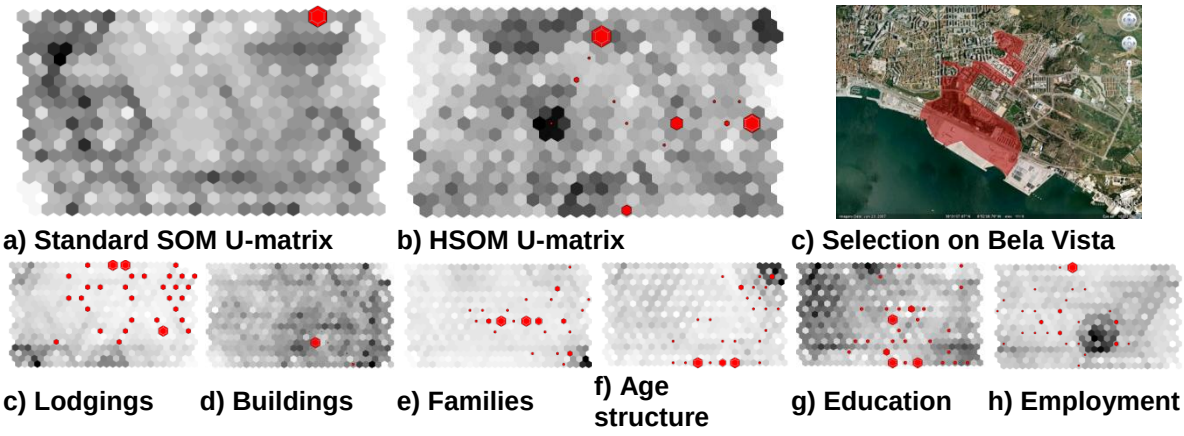
d) Age structure **e) Education** **f) Employment**

Figure 70 – Characterization of two EDs belonging to *Bela Vista* neighbourhood through the thematic U-matrices

Figure 70 shows that the two EDs from the *Bela Vista* neighbourhood are very similar in four themes (families, age structure, education and employment), presenting some differences in the lodgings and buildings themes. Since the two EDs are similar in most of the thematic SOMs, HSOM represents them in the same unit. This means that the HSOM is less sensitive to outlier variables than the SOM, since an ED has to be an outlier in several themes to become an outlier in the top level SOM.

Let us now, try a different approach in the analysis of the *Bela Vista* neighbourhood. Selecting one of the EDs belonging to the *Bela Vista* neighbourhood allows the identification of its BMU in the HSOM and in the standard SOM. We may then select each of these units, and characterize the different sets of selected EDs. In other words, the selected EDs are the most similar to *Bela Vista* from two different perspectives (SOM and HSOM perspectives).

Figure 71 presents the case in which we selected the unit that best represents *Bela Vista* from the standard SOM U-matrix (Figure 71a). The set of EDs that share this unit as BMU is then selected in the HSOM U-matrix, in the thematic SOMs and in the geographical map.



a) Standard SOM U-matrix **b) HSOM U-matrix** **c) Selection on Bela Vista**

c) Lodgings **d) Buildings** **e) Families** **f) Age structure** **g) Education** **h) Employment**

Figure 71 – Selection of similar EDs (in red) to *Bela Vista* in the standard SOM: a) SOM U-matrix; b) HSOM U-matrix; c) geographical map with EDs selection; d) Lodgings' U-matrix; e) Buildings' U-matrix; f) Families' U-matrix; g) Age structure U-matrix; h) Education level U-matrix and; i) Employment U-matrix

Comparing the position of the selected EDs on the standard SOM and on the HSOM U-matrices it is possible to conclude that this set does not form a cluster in the HSOM. An analysis on the lower level of HSOM (thematic SOMs) reinforces this fact by presenting a large dispersion of the EDs along the U-matrices. Analysing the variables distribution (Figure 72) we can conclude that this cluster is characterized by a high percentage of buildings with walls of cemented masonry (*EARG*), from the 70s (*E1980*) with three or four floors (*PV4*).

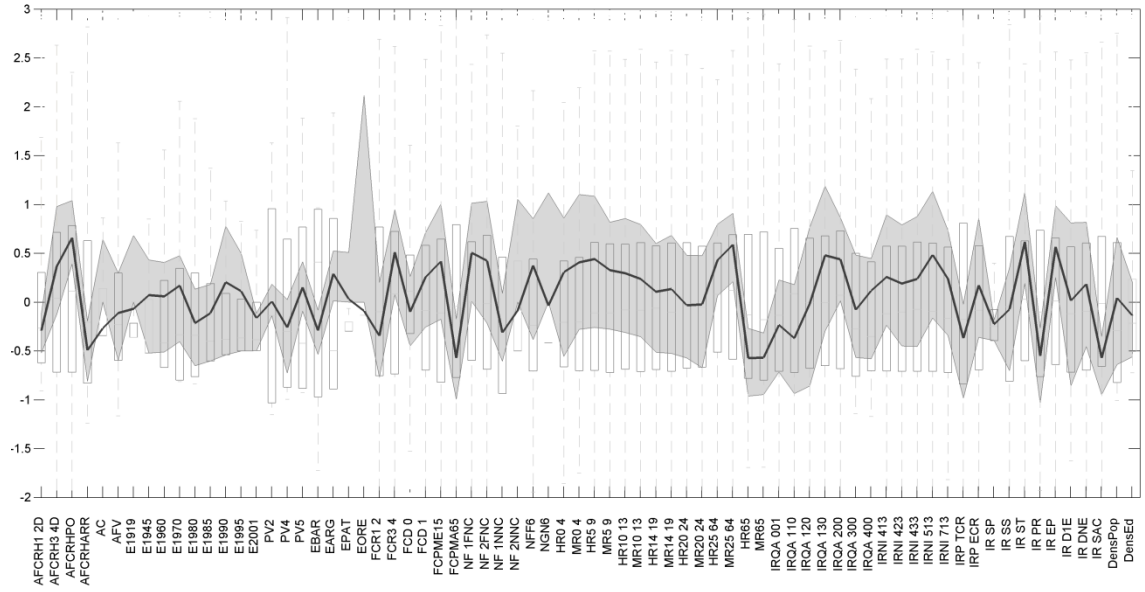
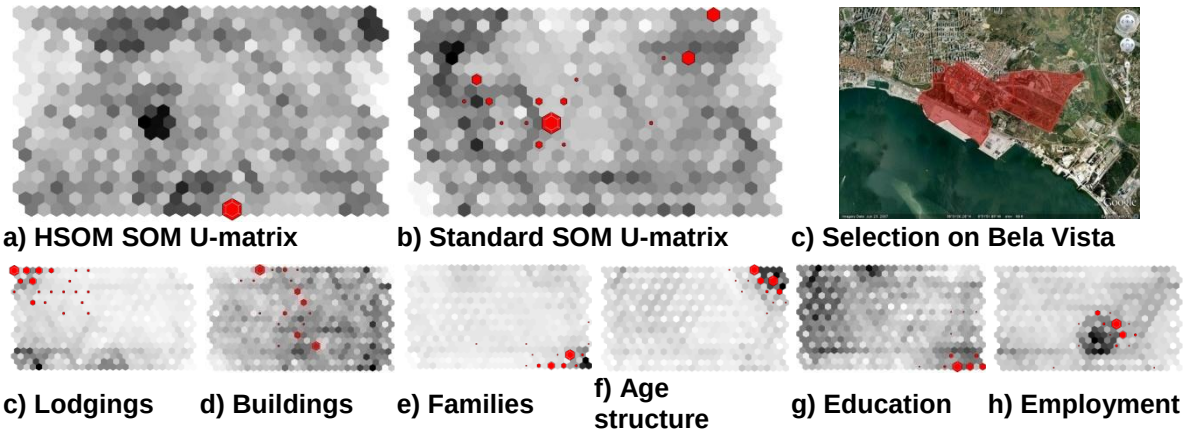


Figure 72 – Boxplot of all the variables used showing the distribution of the selected EDs. Variables were normalized using *z*-score (mean equals zero)

Figure 73 presents the same example, but this time we selected *Bela Vista's* BMU in the HSOM, and obtained the set of EDs from this unit.



a) HSOM SOM U-matrix **b) Standard SOM U-matrix** **c) Selection on Bela Vista**

c) Lodgings **d) Buildings** **e) Families** **f) Age structure** **g) Education** **h) Employment**

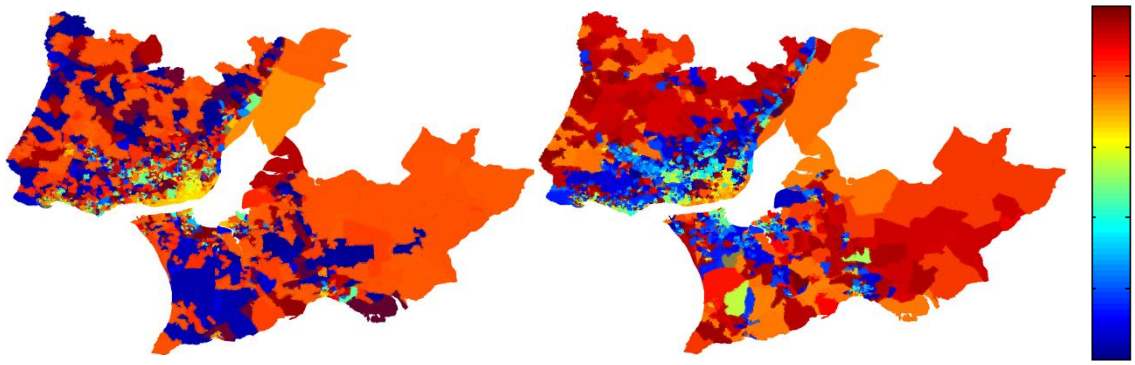
Figure 73 – Selection of similar EDs (in red) to *Bela Vista* in the HSOM: a) geographical map of the *Bela Vista* selected ED, b) SOM U-matrix; c) HSOM U-matrix; d) Lodgings' U-matrix; e) Buildings' U-matrix; f) Families' U-matrix; g) Age structure U-matrix; h) Education level U-matrix and; i) Employment U-matrix

Analysing the SOM's U-matrix (Figure 73b) we can conclude that although the selected EDs are very similar from HSOM's perspective, from a standard SOM's point of view they belong to very different partitions. Analysing the thematic SOM U-matrices we can see that in most of the themes used in HSOM the selected EDs present similar distributions. The exception is the buildings theme where EDs present different characteristics. However, because in five of the six themes these EDs present similar behaviour, HSOM groups them in the same partition.

5.5.2. Quantitative evaluation

To compare the results presented by the SOM and HSOM, we considered each unit as a separate cluster. We thus obtain 150 clusters, represented by their centroids (SOM units). This clustering approach is known as *k-means SOM* (Bacao, Lobo *et al.* 2005) due to the resemblance with *k*-means clustering.

Figure 74 presents the geographic representation of the clusters created by each method.



a) SOM clusters **b) HSOM clusters**

Figure 74 – Geographic representation of the 150 clusters created using: a) standard SOM and; b) HSOM. Each cluster is represented by a unique colour. Since the two methods create different partitions, the colours are not comparable between the two solutions. However, for each solution the colour codes guarantee that similar clusters in the SOM share similar colours

The comparison of clusters is made through the computation of the following quantitative indexes:

- The **modified quantization error** mq_e; this index is similar to the SOM's quantization error (q_e) (Kohonen 2001) and their values should be equal if the SOM is well trained and border effect does not take place. The border effect occurs when units on the borders of the map do not stretch out as much as they should (Kohonen 2001). Instead of calculating distances from each pattern to its BMU as q_e does, mq_e uses the distance between patterns and the centroids obtained from the patterns belonging to the same BMU. This index calculates the average error using only the input patterns' space. This is an important property since we are dealing with multiple SOM structures and the comparison of the input spaces of each SOM is not feasible. mq_e is defined as:

$$mq_e = \frac{\sum_{k=1}^{N_{BMU}} \| x_k - \overline{x_{BMU}} \|}{N_{BMU}} \qquad \textbf{Equation XIX}$$

  Where x_k is the training pattern, $\overline{x_{BMU}}$ is the mean of all training patterns sharing the BMU with x_k and N_{BMU} is the number of training patterns mapped to that unit.

- The **neighbourhood-cluster ratio** $ncr(k)$; this is the ratio between the number of polygons that have at least *k* neighbours of the same cluster and the total number of EDs. This ratio is an indicator of the geographical compactness of the

clusters obtained. If clusters do have geographical compactness, all ED have at least one neighbour that belongs to the same cluster, then $ncr(1) = 100\%$. On the other hand, if each cluster is geographically dispersed, and none of its ED has a neighbour belonging to the same cluster, $ncr(1) = 0\%$. As the value of *k* increases, we are searching for ever larger number of neighbours. To compute this ratio we start by computing, for each ED, the number of neighbours N_i that belong to the same cluster:

$$N_i = \sum_{j=1; j \neq i}^{N} W_{ij} \times C_{ij} \qquad \textbf{Equation XX}$$

Where W is a binary contiguity matrix, where $W_{ij} = 1$ if and only if ED_i and ED_j are neighbours. C is a binary cluster matrix, where $C_{ij} = 1$ if and only if ED_i and ED_j belong to the same cluster. To compute $ncr(k)$ we simple count the number of ED for which $N_i \geq k$, and divide that by the total number of ED.

Figure 75 presents, for each method, the modified quantization error (mq_e) obtained from Equation XIX. This error can be decomposed into geo (from geographical) and non-geo errors by using the geographical coordinates or the aspatial variables in the distances computations.

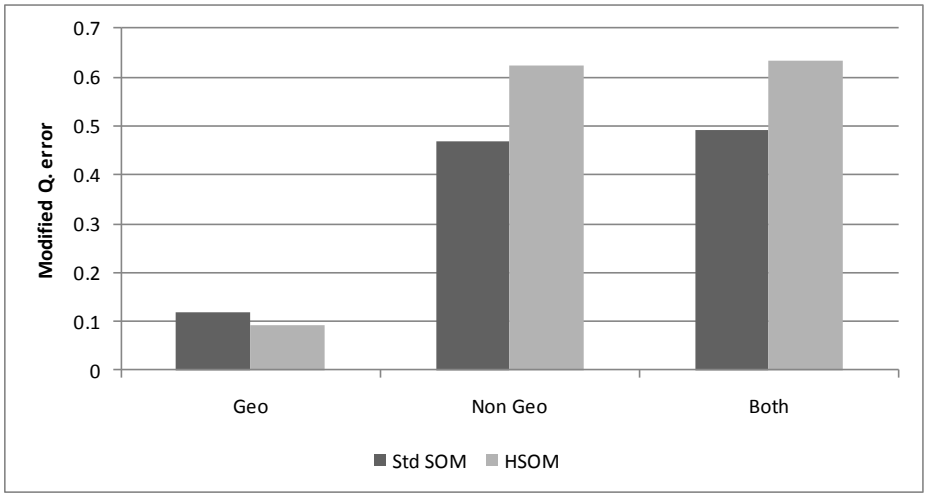


Figure 75 – Modified quantization error for each standard SOM and HSOM, using only geographical coordinates, only aspatial variables, and using both

From Figure 75 it is possible to verify that the geographical error decreases from the standard SOM to the HSOM. In an inverse way, the error using the aspatial variables is

larger in the HSOM. This is an expected result, and can be explained by the fact that HSOM due to the use of themes, is less sensitive (in the clustering process) to outliers in a particular variable, that do increase the quantization error. Finally, the combination of both the geo and non-geo components presents a larger modified quantization error in the case of the HSOM.

Figure 76 presents the ratio between the number of EDs that has from one to fourteen neighbours sharing the same cluster and the total number of EDs.

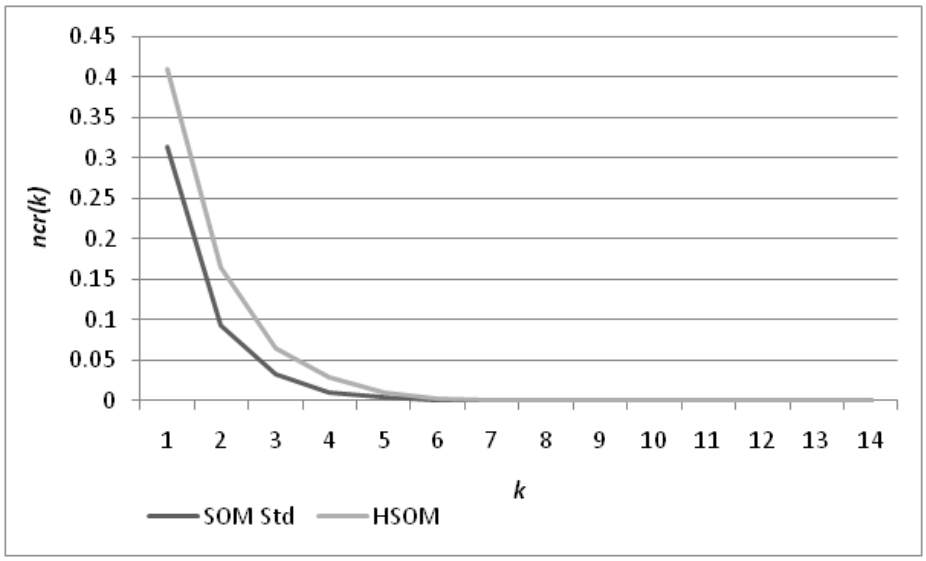


Figure 76 – Neighbourhood cluster ratio (*ncr*) calculated for *k*=1 to *k*=14. *ncr* gives the percentage of total EDs sharing *k* spatial neighbours with the same cluster

From Figure 76 it is possible to conclude that HSOM groups EDs in such a way that the number of neighbours sharing the cluster is larger than in the standard SOM. For instance, in the HSOM more than 40% of the EDs share one neighbour from the same cluster while in the standard SOM only 30% of them do.

In summary, the clusters created in the HSOM are spatially more contiguous, since the number of neighbours belonging to the same cluster is higher.

5.6. Discussion

In this chapter, we defined what a hierarchical SOM is, and the ways in which different level SOMs can interact. A taxonomy was proposed for the different hierarchical SOM

methods, and a brief review of some of the most important hierarchical SOM implementations was made.

We presented a new approach for clustering geospatial data using hierarchical SOMs. This new method was implemented in a publically available software package (GeoSOM Suite), and several tests were performed using Lisbon's Metropolitan Area (LMA) census data.

To evaluate the quality of the clustering performed by the HSOM, the results were compared with those of a standard SOM. Both a qualitative and a quantitative evaluation was made, and the results suggest that HSOM creates clusters with higher spatial autocorrelation.

HSOM proved to be a valuable complement to the standard SOM. Amongst the differences found, we would highlight the fact that HSOM has shown to be less sensitive to outlier values. This is especially important in the context of geospatial data, in which the analyst is often interested in analysing groups of variables, known as themes, instead of singular variables. The use of hierarchical structures in clustering allows the creation of thematic and multi-level clustering. The exploration of these sub products is an important step in the knowledge discovery process, allowing a multiple perspective analysis of data. This exploration is made easier with the use of GeoSOM Suite (Lobo, Bação *et al.* 2009). This freely available software package, which already provided several SOM-based tools to help in the exploratory spatial data analysis (ESDA), is now enriched with the implementation of the HSOM method. In summary the HSOM:

- is less sensitive to outliers resulting from one or a few variables;
- is more suitable for geospatial data due to this robustness to variations in a single variable, and due to the fact it shares with GIS the use of thematic layers;
- allows the use of a higher number of variables without degradation in performance. With the increase of dimensionality, a standard SOM is less efficient in detecting patterns, and takes a longer time to train;
- allows the use of larger SOMs. The reduction in computational effort (described in section 2.2) allows far larger SOMs to train in reasonable time.

6. Mobile sensor network path definition problem

In mobile sensor networks, the choice of the paths those sensors should follow is a challenging problem. As this type of networks become cheaper and more widespread, this problem becomes more important. Due to our collaboration with a research project being developed at the Portuguese Naval Academy, we were posed with the problem of defining patrol routes for a group of unmanned aerial vehicles (UAV), which are a particular type of mobile sensor network. These UAVs are used to detect ships, possibly involved in illegal activities. The goal of this chapter is to use an SOM to define the path of the UAVs so as to maximize the detection of ships in a given area. Although this line of work was not explored to its full length we chose to include it in this thesis because it is an interesting and creative way of using a SOM, and it might, in the future, mature into a productive research line.

6.1. Introduction

Technological advances in communication systems and the ability to build low power and inexpensive mobile systems make the deployment of a group of networked vehicles a feasible task. Recent development of unmanned aerial vehicles (UAV) and recent implementations have proven their benefits in several applications such as fire detection (Merino, Caballero *et al.* 2005), agricultural monitoring (Herwitz, Dunagan *et al.* 2003), ocean surveillance (Rubio, Vagners *et al.* 2004), traffic surveillance (Kaaniche, Champion *et al.* 2005) and military (Zengin and Dogan 2006; Nowak, Price *et al.* 2007).

UAV capabilities are improving, and new methods for deployment and management are required to coordinate the operation of UAV networks. Controlling a network of UAV implies a set of definitions such as goal assignment, resource allocation and trajectory optimization problems (Alighanbari, Kuwata *et al.* 2003).

Different approaches have been used in the past to solve UAV path planning problems and it is not our aim to give a survey of all of these. In (Beard, McLain *et al.* 2002), the authors use a Voronoi approach to define the UAV path. This Voronoi diagram is built based on predefined waypoints and each edge of the Voronoi diagram is assigned two costs: a threat cost and a length cost. This method deals with the problem of simultaneous arrival of multiple UAV at their targets, using threat-avoiding trajectories. In (Flint, Polycarpou *et al.* 2002) a dynamic programming method is presented to generate near-optimal trajectories for multiple UAV to cooperatively search for targets. A coordination framework in which the UAV network seeks out local maxima or minima in the environmental field is presented in (Ogren, Fiorelli *et al.* 2004). The network has adaptation capabilities in response to the sensed environment in order to optimize its gradient climb. In (Schouwenaars 2006) a decentralized trajectory planning of multiple UAV is presented, using a receding horizon strategy based on mixed integer linear programming. None of these approaches is universally better than any other, and there is plenty of room for improvement.

The objective of this chapter is to present a new method for a UAV network path definition in an autonomously way. The path definition is obtained in an iterative process, defining for each instant the path each UAV should take. The method presented allows the self-organization of a network of UAV using SOMs, allowing real-time adjustment of

the network. As study case, we use a scenario simulator for a UAV network monitoring ships.

This chapter is organized as follows. In section 6.2 we present the proposed method. In section 6.3 the scenario simulator is presented, and the experimental settings are defined in section 6.4. The tests results are presented in section 6.5. Finally, section 6.6 is devoted to the discussion of the main results obtained in this chapter.

6.2. Proposed method

The method proposed in this chapter supports the UAV network path definition in an autonomously way, taking into consideration the density of the detected events at each moment, in each place. The goal is to control the UAV network so that the number of detected events in the field of view of the sensors, at each instant, is maximized. To guarantee this optimum detection we assume each UAV should not share its field of view with others.

Since the method is based on the detection of event density, SOMs (Kohonen 2001) are applied to detect these patterns.

In our application, we use a 2-dimensional SOM. Each SOM unit represents an UAV, and the weights of the unit represent the UAV coordinates in the study area. Each ship is a data point, characterized by its two coordinates and the SOM is repeatedly being trained using the position of the detected ships as input data points. Each training phase will produce a new self-organized map, and each unit's weights will correspond to the new UAV coordinates. However, some UAV physical restrictions need to be considered in the SOM training phase. The SOM has to be aware of the UAV's maximum speed and to the fact that their fields of view must not intersect. Due to this fact the standard SOM algorithm was changed to fulfil these two restrictions.

For experimental purposes, we assume a UAV network that is deployed to perform ship detection on a specific area of interest. The UAV network is composed by several UAVs equipped with a camera and a radio transceiver. It is not our purpose to study the detection process or the communication issues since extensive work has been made in this field (Todorovic and Nechyba 2004; Kaaniche, Champion *et al.* 2005).

6.2.1. The UAV path definition algorithm

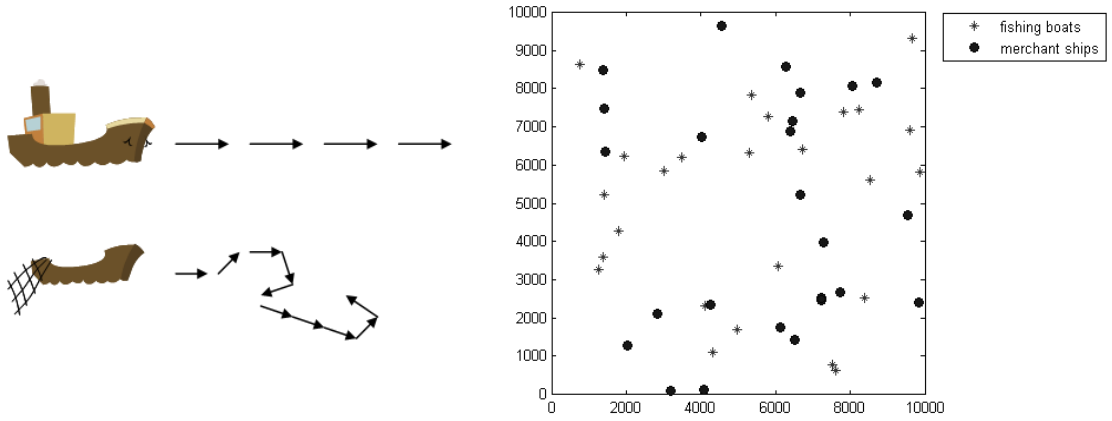
The UAV management algorithm based on SOM may be described as follows:

Let
tf be the duration of the simulation
x_0, y_0, x_1, y_1 define the area of interest
$uavVel$ be the maximum UAV velocity
$uavSen$ be the maximum UAV sensing range
$UAVpos$ be a collection of coordinates x_i, y_i defining the UAV initial positions
$ShipsPos$ be a collection of coordinates x_i, y_i defining the ships initial positions

1 Randomly define each ship position ($ShipsPos$)
2 Repeat
3 Simulate the ships behaviour (movement) in the area of interest for the instant t (update $ShipsPos$)
3 Define the detected ships based on UAVpos, ShipsPos and uavSen
4 Train the SOM according to the ships and UAV positions, taking in consideration $uavVel$ and $uavSen$
5 Move the UAV according to the SOM units' weights (update UAVpos)
6 Increase t
7 Until t reaches tf

6.3. The scenario simulator

To simulate the ships and its paths a ship simulator was implemented in MATLAB™, using a set of predefined characteristics such as type of ship (fishing or merchant), the number of ships and the minimum and maximum velocities for each type of ship. Each ship is initially randomly created, and the total number of ships present in the area of interest is constant. The type of ship defines its behaviour, *i.e.* a fishing boat will stay in the same area for a while, moving afterwards to a different place, while a merchant ship has a constant direction and velocity (Figure 77).



a) **b)**

Figure 77 – The ship simulator: a) fishing boats versus merchant ships behaviour b) initial distribution of the ships

6.4. Experimental evaluation

To measure the effectiveness of the proposed method, we must define a metric by which it can be assessed, and some benchmark methods with which we may compare it. Since the objective is to monitor as best as possible the points of interest, an obvious metric is the number of ships observed, by the UAV, for the instant *t*, divided by the total number of ships present in the area of interest at the same instant. We call this ratio the instant coverage level (*icl*).

$$icl_t = \frac{\sum_{i=1}^{s} nshipsDetected_{it}}{nships_t}$$ **Equation XXI**

Where s is the number of UAV in the network, $nships_t$ is the number of ships present in the study area at instant t, and $nshipsDetected_{it}$ is the number of ships detected by the UAV sensor i at instant t.

Another metric we use is the global coverage level, defined as the total percentage of ships detected until the present time (*gcl*).

$$gcl_t = \sum_{t=0}^{T} \frac{\sum_{i=1}^{s} nnewshipsDetected_{it}}{nnewships_t}$$ **Equation XXII**

Where T is the current instant, $nnewshipsDetected_{it}$ is the number of ships detected by the UAV sensor i at instant t that has not been detected before and $nnewships_t$ is the number of new ships crossing the study area.

Finally we calculate the average time monitoring ships (*amt*). This metric will evaluate if a UAV can follow a moving ship.

$$amt = \frac{\sum_{i=1}^{N} monitoringTime_b}{nships}$$ **Equation XXIII**

Where the $monitoringTime_i$ is the amount of time ship b is being observed and $nships$ is the total number of detected ships.

6.4.1. The benchmark UAV algorithms

As a comparison benchmark, we chose two different methods for defining patrol itineraries. In the first method we assume fixed sensors in a regular matrix along the area of interest. In the second method we define for each UAV a region to cover in the area of interest, and each one will follow a zigzag trajectory covering its corresponding region. Figure 78 exemplifies the benchmark methods' behaviour in the area of interest.

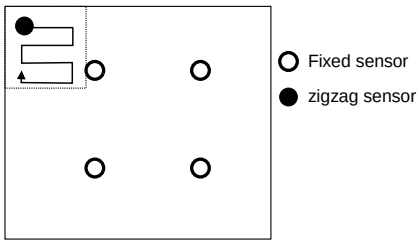


Figure 78 – Benchmark methods: fixed locations and zigzag trajectories for the sensors

Figure 79 presents a 3x3 UAV network for detection of ships of two different types, in four different time instants.

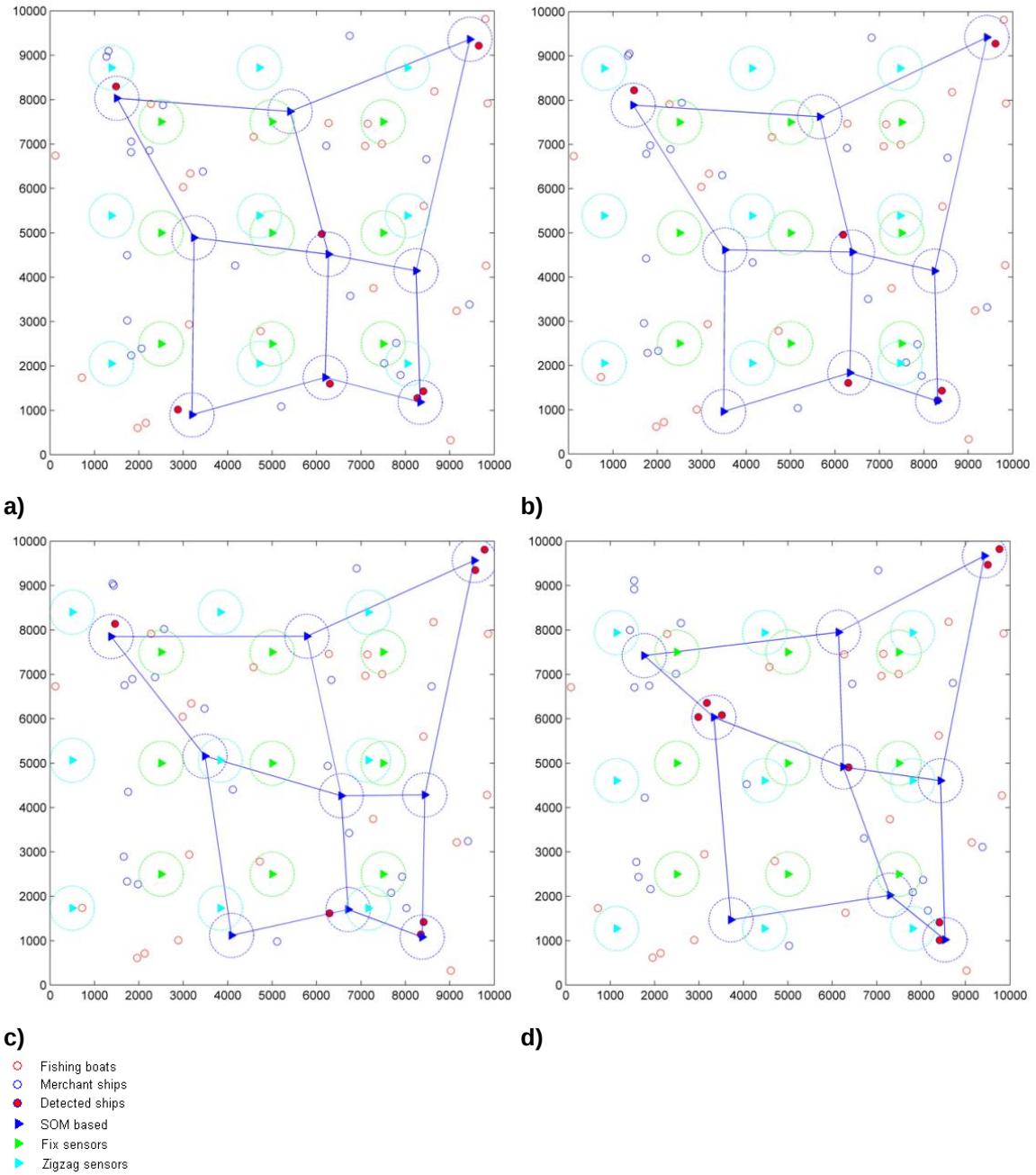


Figure 79 – Ship detection using an SOM based, fixed and zigzag UAV methods in a area of 10000x10000 meters, at for different instants: a) instant *t;* b) instant *t*+1; c) instant *t*+2 and; d) instant *t*+3

6.5. Results

Figure 80 presents the instant coverage level, the global coverage and the average monitoring time for the SOM based method and the two benchmark methods. In this

case, and due to the random ships' position, we calculate an average over 10 simulations.

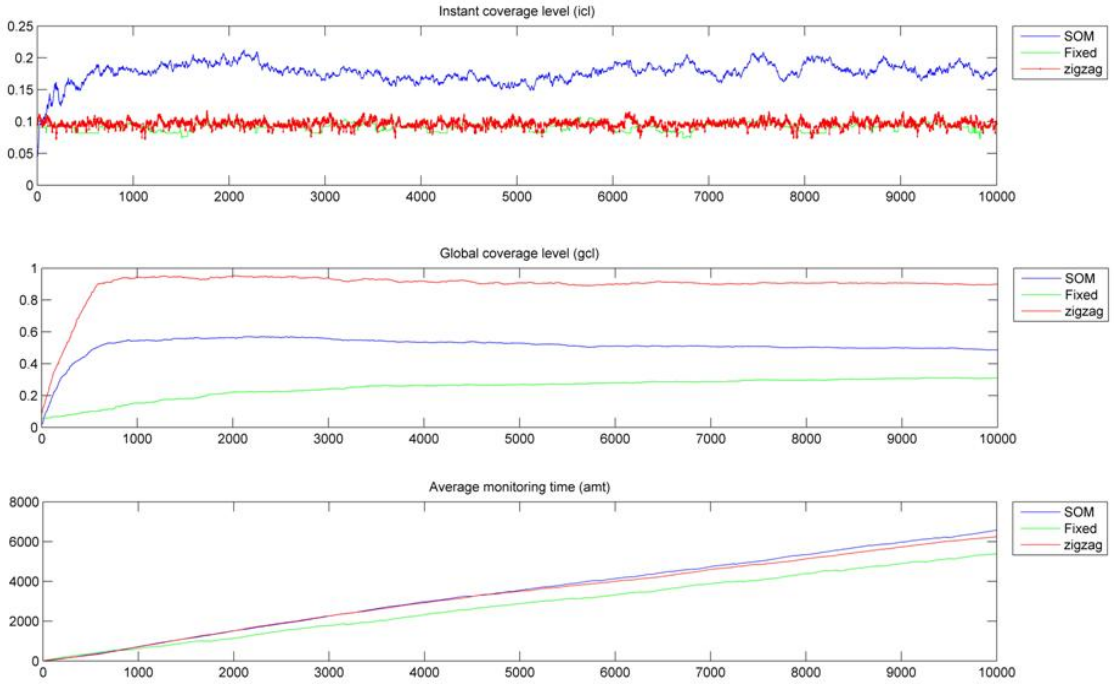


Figure 80 – Statistics for SOM and benchmark methods: fixed and zigzag trajectories sensors

From Figure 80 we can conclude that our method presents the best instant coverage, while the fixed and zigzag trajectories methods present similar instant coverage. However, the zigzag method presents a higher global coverage, detecting almost all ships that cross the area of interest. Our method only captures approximately 50% of the crossing ships. Inversely, SOM method presents a better average monitoring time, representing a higher capacity for tracking the ships once they are detected.

To better understand the SOM method we evaluate the same metrics changing the number of sensors and the number of ships.

6.5.1. Changing the number of UAV in the network

In this test, eight different UAV networks were used (with 4, 9, 16, 25, 36, 49, 64 and 81 UAV) and their instant coverage level is calculated (Figure 81). From the analysis of

Figure 81 we conclude, as expected, that an increase in the number of UAV implies a higher instant coverage.

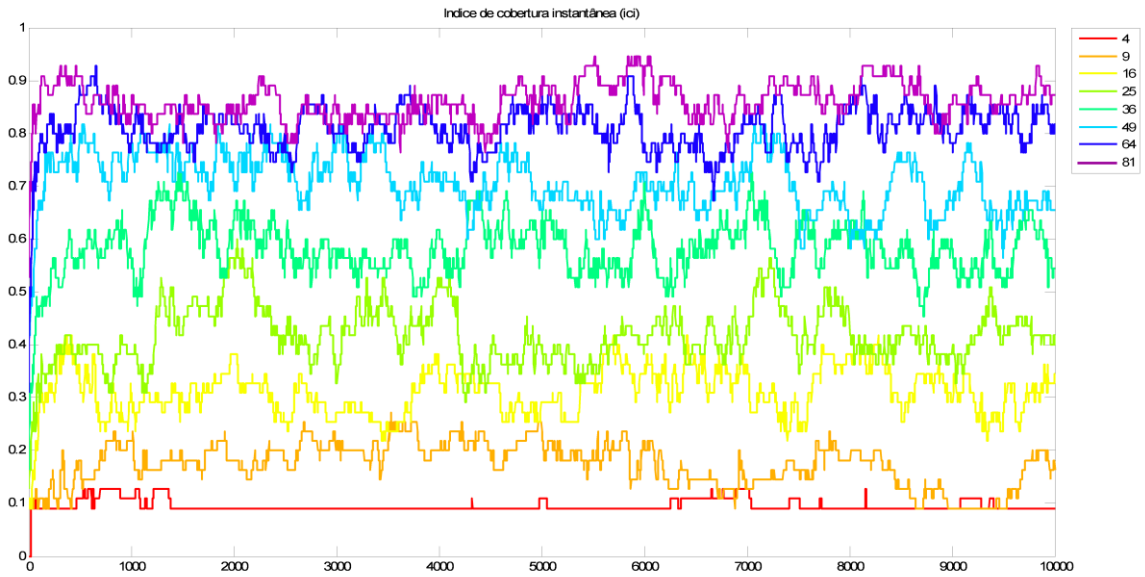


Figure 81 – Instant coverage level calculated for 8 sets of UAV

6.5.2. Changing the number of ships

For this test, we used a set of 9 UAV and we increased the number of ships present in the area of interest (Figure 82). From the analysis of this figure, we conclude, as expected, that an increase in the number of ships to detect by the UAV network results in a smaller coverage of the area.

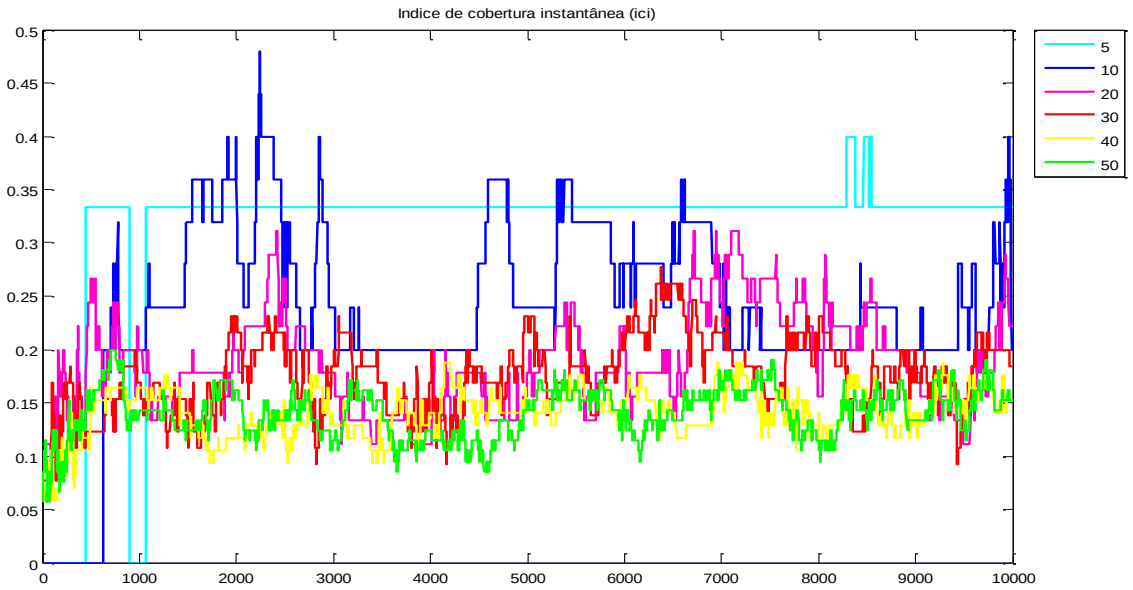


Figure 82 – Instant coverage level calculated using 9 UAV and increasing the number of ships in the area of interest

6.6. Discussion

In this chapter, we presented a first approach to the use of Self-Organizing Maps as a UAV network path definition model. The working example used in this study was based on ship detection in the ocean. The tests made proved that the proposed method improves the number of ships instantaneously detected using the SOM based method, when compared to the benchmark methods.

7. Conclusions

The transition from poor to rich environments in computation and data available poses new challenges to GISc researchers, as well as most other scientific fields. Artificial intelligence techniques can be used to help facing some GIScience challenges. Given the relevance of data in recent advances in science, it is realistic to assume that the answer to many problems may lie in the huge databases we now have available. For this to become true we need to take the most sophisticated tools and creatively explore the data we have available. In the case of GIScience, this may involve adjustments in the general data mining methods since GIS-minded analysis usually require more than the results provided by these general tools.

In this thesis, focusing on this assumption, we proved that SOM-based data mining tools can be valuable in the geospatial context, and constitutes a less dependent assumption method to extract relations and patterns from geospatial databases. We presented several well succeed examples of applications from the literature and proposed three SOM-based solutions for GISc problems. To include geospatial reasoning in these solutions, adaptations were proposed and implemented and some examples of how they

should be used are presented. The particular characteristics of geospatial data allied with GIS-minded analysis require, in some cases, adjustments to the general data mining methods while in other cases, the original methods are well suited. These adjustments are dependent on the problem we are dealing with and on the expected results and analysis.

We started by present a review on SOMs, describing the algorithm and the parameterisation associated with it. Special attention was given to the SOM visualisation tools, which were comprehensively listed and described. A taxonomic classification to group these tools was proposed, and each of the visual exploration methods was classified accordingly.

A second review focused on the use of SOM to deal with geospatial data. Different approaches using SOM in the geographic context were analysed and, to provide a structured view of all these methods, a taxonomy of those methods was also proposed. This review presents an extensive analysis on the possibilities of adapting the SOM algorithm to the GIS-minded analysis.

Four different applications were developed in this thesis, belonging to three different GISc's topics: 1) cartography and visualisation; 2) spatial data mining and 3) location/allocation optimisation. The proposed applications of SOM were:

1. In the cartography and visualisation context
    - A new method to create area cartograms;
2. In the spatial clustering context
    - An improved method to perform spatial clustering from high-dimensional datasets;
    - A new method to perform thematic spatial clustering from geospatial databases;
3. In the location/allocation optimisation context
    - A new method to manage unmanned vehicles defining an optimised path to best cover events in a specific region.

In all of these problems, density estimation plays a central role, whether we refer to spatial density (the density in the geographic space) or to the density in a hyperspace of characteristics. That is one of the reasons SOM proved to be well suited to solve those problems. Other reasons were explained in the in-depth review of SOM, focusing on some characteristics familiar to GIS scientists, such as the topology and visualisation techniques.

7.1. Contributions

From this thesis, we would like to emphasize the following major contributions:

1. We proposed a taxonomy for SOM visualisation methods, organizing most of the techniques used to visualise the SOM results. The visualisation of these results shares many of the challenges encountered in the geovisualisation. Because GIScientists are proficient in using most of these visualisation tools, they are in a good position to take advantage of SOM's visualisation tools.

2. We reviewed the most important applications of SOM in the geospatial context and proposed a taxonomy for SOM based methods used in the geospatial context. Many different approaches were analysed and categorised according to their objectives and characteristics.

3. We proposed a new method called Carto-SOM, which uses a standard SOM to create area cartograms based on a georeferenced variable. Besides a useful tool for creating cartograms, Carto-SOM has also become an important instrument to evaluate the magnification factor of SOM. In the case study presented (USA population), this distortion was also measured (giving a value of aprox. 2/3) and we proposed an inverse distortion in the input data creation to overcome this undesired effect.

4. We improved, extended and implement the GeoSOM spatial clustering method. The GeoSOM method constitutes an adaptation of SOM to incorporate the first law of Geography, assuming location as a particular dimension. This method was implemented and improved by adding new analysis and visualisation tools,

providing an efficient framework for exploratory spatial analysis and knowledge discovery. To prove this capability we used Lisbon's Metropolitan census data as an example.

5. We reviewed most Hierarchical SOMs proposals and proposed a taxonomy, making clear the differences between general multi-layer architectures and what we assumed as hierarchical SOMs.

6. We proposed the application of hierarchical SOMs in the exploration and thematic clustering of geospatial data, more specifically using census data. Hierachical SOMs proved to very useful in the context of census data mainly due to their hierarchical structure and to the possibility of analysing thematic clusters.

7. Finally, we proposed a new SOM-based method for definition, in real time, of trajectories to be follow by a network of mobile sensors to optimise the amount of covered events in a study region.

7.2. Future work

The applications proposed in this thesis are not closed research and future developments can be expected.

Carto-SOM is still a prototype and it was not developed with computational efficiency issues in mind. As future developments, we foresee its implementation in a more computationally efficient programming language. Another open option to increase the efficiency and processing time is the use of a parallel processing paradigm. In addition, it would be interesting to include methods to compute smoother region boundaries, since at present boundaries are defined by the grid of discrete units. Finally, better cartograms can be produced by significantly increasing the number of units used in the SOM.

Concerning GeoSOM Suite, future research will focus on evaluating the performance of GeoSOM with factors such as scale, zoning and time. In addition, some research will be done using GeoSOM to detect clusters in different thematic areas and to produce final clusters based on these clusters.

In the Hierarchical SOMs and thematic clustering, future work will explore the use of different outputs sent from lower level SOM top level SOM. While the BMU's coordinates were used, it would be interesting to include spatial indexes such as the local spatial correlation. Yet another topic for future research is the inclusion of geographical data on the hierarchical structure to increase the contiguity of final solution.

Finally, concerning the UAV path definition using SOM, we would like to include aerodynamics restrictions of each UAV, such as the curving angles, maximum and minimum acceleration and breaking. These restrictions will require further adaptations of the SOM algorithm.

Concerning the use of general-purpose artificial intelligence tools in the GISc context, many proposals are expected in the near future, capable of exploration and semi-automatic analysis on very large size repositories.

Specifically, in the case of SOMs, we envision its widespread use for exploratory spatial data analysis (ESDA), geovisualisation and spatial clustering. Several properties of SOM and its capability to easily learn and mimic the data structure make it a well known and widely used method in geographical information science. We proved in this thesis that SOM can be an important tool in GISc, providing that the special nature of geospatial data is considered.

References

Agarwal, P. and A. Skupin (2008). Self-Organising Maps: Applications in Geographic Information Science, Wiley.

Alighanbari, M., Y. Kuwata and J. P. How (2003). Coordination and control of multiple UAVs with timing constraints and loitering. Proceedings of the 2003 American Control Conference, 2003.

Allouche, M. K. and B. Moulin (2005). "Amalgamation in cartographic generalization using Kohonen's feature nets." International Journal of Geographical Information Science **19**(8): 899 - 914.

Anselin, L. (1988). Spatial Econometrics: Methods and Models (Studies in Operational Regional Science), Springer.

Anselin, L. (1989). What is special about spatial data. Technical Report 89–4. S. B. University of California, National Center for Geographic Information and Analysis. Santa Barbara, National Center for Geographic Information and Analysis.

Anselin, L. (1993). Local Indicators of Spatial Association - LISA. GISDATA (Geographic Information Systems Data) Specialist Meeting on GIS (Geographic Information Systems) and Spatial Analysis, Amsterdam, Netherlands, Ohio State Univ Press.

Anselin, L. (1998). Exploratory spatial data analysis in a geocomputational environment. Geocomputation: A Primer. L. P. A., B. S. M., M. Rachael and M. Bill. New York, John Wiley & Sons**:** 77-94.

Anselin, L., I. Syabri and Y. Kho (2006). "GeoDa: An Introduction to Spatial Data Analysis." Geographical Analysis **38**(1): 5-22.

Atkinson, P. and D. Martin (2000). Introduction. Innovations in GIS 7: GeoComputation. P. Atkinson and D. Martin. London, Taylor and Francis.

Awad, M., K. Chehdi and A. Nasri (2007). "Multicomponent Image Segmentation Using a Genetic Algorithm and Artificial Neural Network." Geoscience and Remote Sensing Letters, IEEE **4**(4): 571-575.

Babu, G. P. (1997). "Self-organizing neural networks for spatial data." Pattern Recognition Letters **18**(2): 133-142.

Bação, F., S. Caeiro, M. Painho, P. Goovaerts and M. Costa (2005). Delineation of Estuarine Management Units: Evaluation of an Automatic Procedure. Geostatistics for Environmental Applications. P. Renard, H. Demougeot-Renard and R. Froidevaux. Netherlands, Springer-Verlag**:** 429-441.

Bacao, F., V. Lobo and M. Painho (2005). Geo-SOM and its integration with geographic information systems. WSOM 05, 5th Workshop On Self-Organizing Maps. University Paris 1, Panthéon-Sorbonne**:** 5-8.

Bacao, F., V. Lobo and M. Painho (2005). On the Particular Characteristics of Spatial Data and its Similarities to Secondary Data Used in Data Mining. GIS PLANET 2005, II International Conference and Exhibition On Geographic Information, Estoril, Portugal.

Bacao, F., V. Lobo and M. Painho (2005). Self-organizing Maps as Substitutes for K-Means Clustering. Lecture Notes in Computer Science,. V. S. Sunderam, G. v. Albada, P. Sloot and J. J. Dongarra. Berlin Heidelberg, Springer-Verlag. **3516:** 476-483.

Bação, F., V. Lobo and M. Painho (2005). Applying genetic algorithms to zone design. Soft Computing. **9:** 341-348.

Bação, F., V. Lobo and M. Painho (2005). The self-organizing map, the Geo-SOM and relevant variants for geosciences. Computers and Geosciences, Elsevier. **31:** 155-163.

Bação, F., V. Lobo and M. Painho (2008). Applications of Different Self-Organizing Map Variants to Geographical Information Science Problems. Self-Organising Maps: Applications in Geographic Information Science. P. Agarwal and A. Skupin**:** 21-44.

Bandeira, N., V. Lobo and F. Moura-Pires (1998). Training a Self-organizing Map distributed on a PVM network. International Joint Conference on Neural Networks IEEE World Conference on Computational Intelligence, Anchorage.

Barbalho, J. M., A. Duarte, D. Neto, J. A. F. Costa and M. L. A. Netto (2001). Hierarchical SOM applied to image compression. International Joint Conference on Neural Networks, 2001. IJCNN '01, Washington, DC.

Bauer, H.-U., R. Der and M. Herrmann (1996). Controling the magnification factor of self-organizing feature maps. Neural Computation. **8:** 757-765.

Bauer, H.-U. and K. R. Pawelzik (1992). "Quantifying the neighbourhood preservation of self-organizing feature maps." IEEE Transactions on Neural Networks **3**(4): 570-579.

Beard, R. W., T. W. McLain, M. A. Goodrich and E. P. Anderson (2002). "Coordinated target assignment and intercept for unmanned air vehicles." IEEE Transactions on Robotics and Automation **18**(6): 911-922.

Bellman, R. (1961). Adaptive Control Processes: A Guided Tour. Princeton, New Jersey, Princeton University Press.

Bertin, J. (1983). Semiology of Graphics: Diagrams, Networks, Maps, University of Wisconsin Press.

Birkin, M. and G. Clarke (1998). "GIS, geodemographics and spatial modeling in the UK financial service industry." Journal of Housing Research **9**: 87-111.

Birkin, M., G. Clarke and M. Clarke (1999). GIS for Business and Service Planning. Geographical Information Systems. M. Goodchild, P. Longley, D. Maguire and D. Rhind. Cambridge, Geoinformation.

Bishop, C. M., M. Svensen and C. K. I. Williams (1997). Magnification factors for the SOM and GTM algorithms. Procedings Workshop on Self-Organizing Maps, Helsinki, Helsinki University of Technology.

Brassel , K. E. and D. Reif (1979). "A procedure to generate Thiessen polygons." Geographical Analysis **11**: 289-303.

Brown, K. Q. (1979). "Voronoi diagrams from convex hulls." Information Processing Letters **9**(5): 223-228.

Buttenfield, B. and R. McMaster (1991). Map Generalization: Making Rules for Knowledge Representation. London, Longman.

Carpinteiro, O. A. s. (1999). "A Hierarchical Self-Organizing Map Model for Sequence Recognition." Neural Processing Letters **9**(3): 209-220.

Carpinteiro, O. A. S. and A. P. Alves da Silva (2001). "A Hierarchical Self-Organizing Map Model in Short-Term Load Forecasting." Journal of Intelligent and Robotic Systems **31**(1): 105-113.

Céréghino, R., F. Santoul, A. Compin and S. Mastrorillo (2005). "Using self-organizing maps to investigate spatial patterns of non-native species." Biological Conservation **125** (4): 459-465.

Chandrasekaran, V. and M. Palaniswami (1995). "Spatio-temporal Feature Maps using Gated Neuronal Architecture." IEEE Transactions on Neural Networks **6**(5): 1119-1131.

Chandrasekaran, V., M. Palaniswami and T. M. Caelli (1993). An extended self-organizing map with gated neurons. IEEE International Conference on Neural Networks, Piscataway, NJ.

Changming, D. and L. Lin (1999). Constructing contiguous area cartogram using arcview avenue. The proceedings of Geoinformatics'99 Conference, Ann Arbor, The Association of Chinese Professionals in GIS.

Chen, H., H. Atabakhsh, T. Petersen, J. Schroeder, T. Buetow, L. Chaboya, C. O'Toole, M. Chau, T. Cushna, D. Casey and Z. Huang (2003). COPLINK: visualization for

crime analysis. Proceedings of the 2003 annual national conference on Digital government research. Boston, MA, Digital Government Society of North America.

Chifu, E. S. and I. A. Letia (2008). Text-Based Ontology Enrichment Using Hierarchical Self-organizing Maps. Nature inspired Reasoning for the Semantic Web (NatuReS2008), Karlsruhe, Germany.

Clark Labs (2008). IDRISI Taiga. Worcester, USA.

Claussen, J. C. (2003). "Winner-relaxing and winner-enhancing Kohonen maps: Maximal mutual information from enhancing the winner." Complexity (Wiley) **8**(4): 15 - 22.

Cockings, S. and D. Martin (2005). "Zone design for environment and health studies using pre-aggregated data." Social Science & Medicine **60**(12): 2729-2742.

Cook, S. A. (1983). "An overview of computational compexity." Communications of the ACM **26**(6): 400-408.

Cottrell, M., J. C. Fort and G. Pages (1998). "Theoretical Aspects of the SOM algorithm." Neurocomputing, Elsevier **21**: 119-138.

Couclelis, H. I. E. (1998). Geocomputation in context. GeoComputation: a primer. P. Longley, S. Brooks, R. McDonnell and B. Macmillan. New York, Wiley**:** 17-29.

Demšar, J., B. Zupan, G. Leban and T. Curk (2004). Orange: From Experimental Machine Learning to Interactive Data Mining. Knowledge Discovery in Databases: PKDD 2004**:** 537-539.

Dersch, D. R. and P. Tavan (1996). "Asymptotic level density in topological feature maps." IEEE Transactions on Neural Networks **6**(1): 230-236.

Dittenbach, M., D. Merkl and A. Rauber (2002). Organizing And Exploring High-Dimensional Data With The Growing Hierarchical Self-Organizing Map. Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2002), Orchid Country Club, Singapore.

Dougenik, J., N. Chrisman and D. Niemeyer (1985). An algorithm to construct continuous area cartograms. Professional Geographer. **37:** 75-81.

Douzono, H., H. Tokushima, S. Hara and Y. Noguchi (2002). A design method of DNA chips using hierarchical self-organizing maps. Proceedings of the 9th International Conference on Neural Information Processing. ICONIP '02, Orchid Country Club, Singapore.

Duda, R. O., P. E. Hart and D. G. Stork (2001). Pattern Classification, Wiley-Interscience.

Durbin, R. and G. Mitchison (1990). "A dimension reduction framework for understanding cortical maps." Nature **343**: 644-647.

Ehsani, A. H. and F. Quiel (2008). "Application of Self Organizing Map and SRTM data to characterize yardangs in the Lut desert, Iran." Remote Sensing of Environment **112**(7): 3284-3294.

Endo, M., M. Ueno and T. Tanabe (2002). "A Clustering Method Using Hierarchical Self-Organizing Maps." The Journal of VLSI Signal Processing **32**(1): 105-118.

Fahmy, E., D. Gordon and S. Cemlyn (2002). Poverty and Neighbourhood Renewal in West Cornwall. Social Policy Association Annual Conference, Nottingham, UK.

Fayyad, U., G. Piatetsky-Shapiro and P. Smyth (1996). "From data mining to knowledge discovery in databases." AI Magazine **17**: 37-54.

Fayyad, U. M., G. Piatetsky-Shapiro and P. Smyth (1996). From Data Mining to Knowledge Discovery: An Overview. Advances in Knowledge Discovery and Data Mining. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, AAAI Press/ The MIT Press**:** 1-43.

Feng, Z. and R. Flowerdew (1998). Fuzzy geodemographics: a contribution from fuzzy clustering methods. Innovations in GIS 5. S. Carver. London, Taylor & Francis**:** 119-127.

Ferentinou, M. D. and M. G. Sakellariou (2007). "Computational intelligence tools for the prediction of slope performance." Computers and Geotechnics **34**(5): 362-384.

Fincke, T., V. Lobo and F. Bação (2008). Visualizing self-organizing maps with GIS. GI Days 2008, Munster, Germany, Institute for Geoinformatics at the University of Münster (Germany).

Fisher, R. A. (1936). "The use of Multiple Measurements in Taxonomic Problems." Annals of Eugenics **VII**(II): 179-188.

Fleming, K. L., D. F. Heermann and D. G. Westfall (2004). "Evaluating Soil Color with Farmer Input and Apparent Soil Electrical Conductivity for Management Zone Delineation." Agron J **96**(6): 1581-1587.

Flint, M., M. Polycarpou and E. Fernandez-Gaucherand (2002). Cooperative control for multiple autonomous UAV's searching for targets. Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, Nevada, USA.

Fort, J. C. (2006). "SOM's mathematics." NEURAL NETWORKS **19**(6-7): 812-816.

Foster, D. (2009). "GPX: the GOS exchange format." Retrieved 19-08-2009, from http://www.topografix.com/gpx.asp.

Franzini, L. B., P.; Diappi, L. (2001). Self Organizing Maps,: A Clustering neural method for urban analysis. Rencontres de Théo Quant.

Fraser, S. a. D., BL (2006). "Data Mining Geoscientific Data Sets Using Self Organizing Maps." Mastering the Data Explosion in the Earth and Environmental Sciences, Extended Abstracts: 5-7.

Fukunaga, K. (1990). Introduction to statistical pattern recognition, Academic Press Inc.

Gahegan, M. (2003). "Is inductive machine learning just another wild goose (or might it lay the golden egg)?" International Journal of Geographical Information Science **17**(1): 69-92.

Gastner, M. (2005). "Diffusion cartogram - cartogram code." from http://www.santafe.edu/~mgastner/.

Gastner, M. and M. E. J. Newman (2004). Diffusion-based method for producing density-equalizing maps. Proceedings of the National Academy of Sciences of the United States of America.

Gevrey, M., S. Worner, N. Kasabov, J. Pitt and J.-L. Giraudel (2006). "Estimating risk of events using SOM models: A case study on invasive species establishment." Ecological Modelling **197**(3-4): 361-372.

Gomes, H., A. B. Ribeiro and V. Lobo (2007). "Location model for CCA-treated wood waste remediation units using GIS and clustering methods." Environmental Modelling & Software **22**(12): 1788-1795.

Goodchild, M. (1986). Spatial Autocorrelation. Norwich, Geo Books.

Goodchild, M. (2007). "Citizens as sensors: the world of volunteered geography." GeoJournal **69**(4): 211-221.

Goodchild, M. F. (1992). "Geographical information science." International Journal of Geographical Information Science **6**(1): 31 - 45.

Goodchild, M. F. (2008). Geographic information science: the grand challenges. The Handbook of Geographic Information Science. J. P. Wilson and A. S. Fotheringham. Malden, MA, Blackwell**:** 596-608.

Goodhill, G. J. and T. J. Sejnowski (1997). "A unifying objective function for topographic mappings." Neural Computation **9**(6): 1291-1303.

Google. (2005). "Google Maps." Retrieved 10-07-2009, from http://maps.google.com/.

Google. (2009). "KML Documentation Introduction." Retrieved 19-08-2008, from http://code.google.com/apis/kml/documentation/.

Gorricha, J. and V. Lobo (2009). Visualização de Clusters em Dados Georreferenciados: uma abordagem com recurso ao Self-Organizing Map 3D. XVI Jornadas de Classificação e Análise de Dados - JOCLAD, Faro, Universidade do Algarve.

Graham, D. P. W. and G. M. T. D'Eleuterio (1991). A hierarchy of self-organized multiresolution artificial neural networks for robotic control. International Joint Conference on Neural Networks, IJCNN-91, Seattle.

Guha, S., R. Rastogi and K. Shim (1998). CURE: an efficient clustering algorithm for large databases. Proceedings of the 1998 ACM SIGMOD international conference on Management of data. Seattle, Washington, United States, ACM.

Guimaraes, G. (2000). Temporal Knowledge Discovery for Multivariate Time Series with Enhanced Self-Organizing Maps. IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Como, Italy, ijcnn.

Guimarães, G. and W. Urfer (2000). Self-Organizing Maps and its Applications in Sleep Apnea Research and Molecular Genetics, University of Dortmund - Statistics Department.

Guo, D. and M. Gahegan (2006). "Spatial ordering and encoding for geographic data mining and visualization." J. Intell. Inf. Syst. **27**(3): 243-266.

Guo, D., D. J. Peuquet and M. Gahegan (2003). "ICEAGE: Interactive Clustering and Exploration of Large and High-Dimensional Geodata." GeoInformatica **7**(3): 229-253.

Gurney, K. (1997). An Introduction to Neural Networks. Bristol, PA, USA, Taylor & Francis, Inc.

Gusein-Zade, S. and V. Tikunov (1993). A new technique for constructing continuous cartograms. Cartography and Geographic Information Systems. **20 (3):** 167-173.

Hadzic, F., T. S. Dillon and H. Tan (2007). Outlier detection strategy using the self-organizing map. Knowledge Discovery and Data Mining: Challenges and Realities. X. Z. I. Davidson. Hershey, PA, USA, Information Science Reference**:** 224-243.

Haese, K. and G. J. Goodhill (2001). Auto-SOM: recursive parameter estimation for guidance of self-organizing feature maps. Neural Computation. **13:** 595-619.

Haining, R., S. Wise and J. Ma (1998). "Exploratory Spatial Data Analysis in a Geographic Information System Environment." Journal of the Royal Statistical Society. Series D (The Statistician) **47**(3): 457-469.

Han, J. (2005). Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers Inc.

Han, J., Y. Cai and N. Cercone (1993). "Data-Driven Discovery of Quantitative Rules in Relational Databases." IEEE Trans. on Knowl. and Data Eng. **5**(1): 29-40.

Han, J., M. Kamber and A. K. H. Tung (2001). Spatial clustering methods in data mining: A survey. Geographic Data Mining and Knowledge Discovery. H. J. Miller and J. Han. London, Taylor and Francis**:** 188-217.

Han, J., K. Koperski and N. Stefanovic (1997). "GeoMiner: a system prototype for spatial data mining." SIGMOD Rec. **26**(2): 553-556.

Hand, D. J., P. Smyth and H. Mannila (2001). Principles of data mining, MIT Press.

Hanke, J., G. Beckmann, P. Bork and J. G. Reich (1996). "Self-organizing hierarchic networks for pattern recognition in protein sequence." Protein Science **5**(1): 72-82.

Harvey, J. M. and J. Han (2001). Geographic data mining and knowledge discovery: An overview. Geographic Data Mining and Knowledge Discovery. H. J. Miller and J. Han, CRC Press**:** 3 - 32.

Hatzichristos, T. (2004). "Delineation of demographic regions with GIS and computational intelligence." Environment and Planning B: Planning and Design **31**(1): 39-49.

Heilmann, R., D. A. Keim, C. Panse and M. Sips (2004). RecMap: rectangular map approximations. IEEE Symposium on Information Visualization, Austin, Texas, IEEE.

Henriques, R. (2006). Carto-Som – Cartogram creation using self-organizing maps. Lisbon, Institute of Statistics and Information Management in the New University of Lisbon (ISEGI / UNL). **Master of Science**.

Henriques, R. and F. Bação (2004). soMGis: uma ferramenta para construir regiões V Congresso da Geografia Portuguesa, Campus de Azurém, Guimarães, Associação Portuguesa de Geógrafos.

Henriques, R., F. Bação and V. Lobo (2005). Carto-SOM: cartogram creation using self-organizing maps. 14th European Colloquium on Theoretical and Quantitative Geography [CD-ROM], Tomar, e-Geo-Geography and Regional Planning Research Centre.

Henriques, R., F. Bação and V. Lobo. (2010). "CartoSOM code." from http://www.isegi.unl.pt/labnt/roberto/.

Herwitz, S. R., S. Dunagan, D. Sullivan, R. Higgins, L. Johnson, Z. Jian, R. Slye, J. Brass, J. Leung, B. Gallmeyer and M. Aoyagi (2003). Solar-powered UAV mission for agricultural decision support. Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International.

Hewitson, B. C. (2008). Climate Analysis, Modelling, and Regional Downscaling Using Self-Organizing Maps. Self-Organising Maps: Applications in Geographic Information Science. P. Agarwal and A. Skupin**:** 137-154.

Hosokawa, M. and T. Hoshi (2001). Landform classification method using self-organizing map and its application to earthquake damage evaluation. Geoscience and Remote Sensing Symposium, 2001. IGARSS '01. IEEE 2001 International.

House, D. and C. Kocmoud (1998). Continuous cartogram construction. Proceedings of IEEE Visualization, Research Triangle Park, NC, IEEE.

Hsieh, K.-H. and F.-C. Tien (2004). "Self-organizing feature maps for solving location-allocation problems with rectilinear distances." Computers and Operations Research **31**(7): 1017-1031.

Hu, T. and S. Sung (2005). "Clustering spatial data with a hybrid EM approach." Pattern Analysis & Applications **8**(1): 139-148.

Ichiki, H., M. Hagiwara and M. Nakagawa (1991). Self-organizing multilayer semantic maps. International Joint Conference on Neural Networks, IJCNN-91, Seattle.

Inselberg, A. (1985). "The plane with parallel coordinates." The Visual Computer **1**(2): 69-91.

Jain, A. K., M. N. Murty and P. J. Flynn (1999). "Data clustering: a review." ACM Comput. Surv. **31**(3): 264-323.

Ji, C. Y. (2000). "Land-use classification of remotely sensed data using Kohonen Self-Organizing Feature Map neural networks." Photogrammetric engineering and remote sensing **66**(12): 1451-1460.

Jiang, B. (2004). Extraction of Spatial Objects from Laser-scanning Data Using a Clustering Technique. ISPRS 2004 Photogrammetry and Remote Sensing, Istanbul, Turkey, ISPRS Congress.

Jiang, B. and L. Harrie (2003). Cartographic selection using self-organizing maps. Fifth workshop on progress in automated map generalisation, Paris, France, ICA, commission on map generalisation.

Jiang, B. and L. Harrie (2004). "Selection of Streets from a Network Using Self-Organizing Maps." Transactions in GIS **8**(3): 335-350.

Jiang, B. and L. Harrie (2004). Spatial clustering for mining knowledge in support of generalization processes in GIS. Workshop on generalisation and multiple representation, Leicester, United Kingdom, ICA, commission on map generalisation.

Jiang, F., H. Berry and M. Schoenauer (2009). The impact of network topology on self-organizing maps. Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation. Shanghai, China, ACM.

Kaaniche, K., B. Champion, C. Pegard and P. Vasseur (2005). A Vision Algorithm for Dynamic Detection of Moving Vehicles with a UAV. Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005 Barcelona, Spain.

Kangas, J. (1992). Temporal Knowledge in Locations of Activations in a Self-Organizing Map. ICANN'92 - International Conference on Artificial Neural Networks, Brighton, England, Elsevier Science Publisher.

Kasabov, N. and E. Peev (1994). Phoneme Recognition with Hierarchical Self Organised Neural Networks and Fuzzy Systems - A Case Study. Proc. ICANN'94, Int. Conf. on Artificial Neural Networks, Springer.

Kaski, S. and T. Kohonen (1996). Exploratory data analysis by the self-organizing map: structures of welfare and poverty in the world. Neural Networks in Financial Engineering. N. Apostolos-Paul, Yaser Refenes, Yaser Abu-Mostafa, John Moody and A. Weigend. Singapore, World Scientific**:** 498-507.

Kaski, S., J. Nikkilä and T. Kohonen (1998). Methods for interpreting a self-organized map in data analysis. Proceedings of ESANN'98 , 6th European Symposium on Artificial Neural Networks, Bruges, Belgium, D-Facto.

Kaski, S., J. Nikkila, M. Oja, J. Venna, P. Toronen and E. Castren (2003). "Trustworthiness and metrics in visualizing similarity of gene expression." BMC Bioinformatics **4**(1): 48.

Kaufman, L. and P. J. Rousseeuw (1990). Finding groups in data : an introduction to cluster analysis. New York, John Wiley & Sons.

Keim, D. A., S. C. North and C. Panse (2004). "CartoDraw: A fast algorithm for generating contiguous cartograms." IEEE Transactions on Visualization and Computer Graphics **10**(1): 95-110.

Keim, D. A., S. C. North and C. Panse (2005). Medial-axes based cartograms. IEEE Computer Graphics and Applications. **25(3):** 60-68.

Keim, D. A., S. C. North, C. Panse and J. o. Schneidewind (2002). Efficient cartogram generation: a comparison. Proceedings of the IEEE Symposium on Information Visualization 2002**:** 33.

Kemke, C. and A. Wichert (1993). Hierarchical {S}elf-{O}rganizing {F}eature {M}aps for Speech Recognition. Proc. WCNN'93, World Congress on Neural Networks, Lawrence Erlbaum.

Kim, M. Y. and H. Cho (2004). "Three-dimensional map building for mobile robot navigation environments using a self-organizing neural network." Journal of Robotic Systems **21**: 323-343.

Kiviluoto, K. (1996). Topology preservation in self-organizing maps. IEEE International Conference on Neural Networks, Washington, DC, USA.

Kocmoud, C. (1997). Constructing continuous cartograms: a constraint-based approach. Office of Graduate Studies. Texas, Texas A&M University. **Master of Science**.

Kohonen, T. (1982). "Self-organizing formation of topologically correct feature maps." RecMap: rectangular map approximations **43 (1)**: 59-69.

Kohonen, T. (1991). The Hypermap Architecture. Artificial Neural Networks. T. Kohonen, K. Mäkisara, O. Simula and J. Kangas. Amsterdam, Netherlands, Elsevier Science Publishers. **2:** 1357-1360.

Kohonen, T. (1995). Self-Organizing Maps. Berlin, Springer.

Kohonen, T. (2001). Self-Organizing Maps. Berlin, Springer.

Kohonen, T., J. Hynninen, J. Kangas and J. Laaksonen (1996). SOM PAK: The Self-Organizing Map program package.

Koikkalainen, P. and E. Oja (1990). Self-organizing hierarchical feature maps. International Joint Conference on Neural Networks, IJCNN Washington, DC, USA.

Koua, E. and M.-J. Kraak (2004). "Geovisualization to support the exploration of large health and demographic survey data." International Journal of Health Geographics **3**(1): 12.

Koua, E. L. and M. J. Kraak (2004). "Alternative visualization of large geospatial datasets." Cartographic Journal **41**(3): 217-228.

Kropp, J. P. and H. J. Schellnhuber (2008). Prototyping Broad-Scale Climate and Ecosystem Classes by Means of Self-Organising Maps. Self-Organising Maps: Applications in Geographic Information Science. P. Agarwal and A. Skupin**:** 155-176.

Kruskal, J. (1964). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis." Psychometrika **29**(1): 1-27.

Lacayo, M. and A. Skupin (2007). A GIS-based Module for Training and Visualization of Self-Organizing Maps. Workshop: "From geovisualization toward geovisual analytics" Helsinki.

Lampinen, J. and E. Oja (1992). "Clustering properties of hierarchical self-organizing maps." Journal of Mathematical Imaging and Vision **2**(2): 261-272.

Law, E. and S. Phon-Amnuaisuk (2008). Towards Music Fitness Evaluation with the Hierarchical SOM. Applications of Evolutionary Computing**:** 443-452.

Lee, J. and O. K. Ersoy (2005). Classification of remote sensing data by multistage self-organizing maps with rejection schemes. Proceedings of 2nd International Conference on Recent Advances in Space Technologies, RAST 2005, Istanbul, Turkey.

Lee, S. and R. G. Lathrop (2006). "Subpixel analysis of landsat ETM+ using self-organizing map (SOM) neural networks for urban land cover characterization." IEEE transactions on geoscience and remote sensing **44**(6): 1642-1654.

Leitich, S. and M. Topf (2007). Globe of Music - Music library visualization using GeoSOM. 8th International Conference on Music Information Retrieval (ISMIR 2007), Vienna, Austria, Österreichische Computer Gesellschaft (OCG).

Li, J. M., Constantine N. (1989). Multistage vector quantization based on the self-organization feature maps. Visual Communications and Image Processing IV, SPIE.

Li, X., J. Gasteiger and J. Zupan (1993). "On the topology distortion in self-organizing feature maps." Biological Cybernetics **70**(2): 189-198.

Li, Z. and J. R. Eastman (2006). Commitment and Typicality Measurements for the Self-Organizing Map. UCGIS 2006 Summer Assembly, Vancouver, Washington.

Lobo, V. (2002). Ship noise classification: a contribution to prototype based classifier design. College of Science and Technology. Lisbon, University of Lisbon, . **Doctor of Philosophy**.

Lobo, V. and F. Bação (2005). One dimensional Self-Organizing Maps to optimize marine patrol activities. Proceedings from the Oceans'05 IEEE Conference and Exhibition, Brest, France.

Lobo, V., F. Bação and R. Henriques. (2009). "GeoSOM Repository: SquareVille dataset." Retrieved 17-07-2009, 2009, from http://www.isegi.unl.pt/labnt/geosom/georepository/.

Lobo, V., F. Bação and R. Henriques. (2009). "GeoSOM suite." Retrieved 15-11-2009, from www.isegi.unl.pt/labnt/geosom.

Lobo, V., F. Bação and M. Painho (2004). The Self-Organizing Map and it's Variants as Tools for Geodemographical Data Analysis: the Case of Lisbon's Metropolitan Area. 7th AGILE Conference on GIScience, Heraklion, Crete/Greece.

Lobo, V. J. A. S. (2009). Application of Self-Organizing Maps to the Maritime Environment. Information Fusion and Geographic Information Systems**:** 19-36.

Longley, P. A., M. F. Goodchild, D. J. Maguire and D. W. Rhind (2005). Geographic Information Systems and Science. Chichester, England, John Wiley & Sons.

Lourenco, F., V. Lobo and F. Bacao (2004). Binary-based similarity measures for categorical data and their application in self-organizing maps. JOCLAD 2004 - XI Jornadas de Classificacao e Análise de Dados, Lisbon, CLAD.

Lozano, S., F. Guerrero, L. Onieva and J. Larrañeta (1998). "Kohonen maps for solving a class of location-allocation problems." European Journal of Operational Research **108**(1): 106-117.

Luttrell, S. P. (1988). Self-organising multilayer topographic mappings. IEEE International Conference on Neural Networks, San Diego, California.

Luttrell, S. P. (1989). "Hierarchical vector quantisation." Communications, Speech and Vision, IEE Proceedings I **136**(6): 405-413.

Maceachren, A. M., M. Wachowicz, R. Edsall, D. Haug and R. Masters (1999). "Constructing knowledge from multivariate spatiotemporal data: integrating geographical visualization with knowledge discovery in database methods." International Journal of Geographical Information Science **13**(4): 311 - 334.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observation. 5th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press.

Mark, D. M. (2003). Geographic Information Science: Defining the Field. Foundations of Geographic Information Science. M. Duckham, M. F. Goodchild and M. Worboys, CRC Press**:** 3 - 18.

Mather, P. M., B. Tso and M. Koch (1998). "An evaluation of Landsat TM spectral data and SAR-derived textural information for lithological discrimination in the Red Sea Hills, Sudan." International Journal of Remote Sensing **19**(4): 587-604.

McMaster, R. and S. Shea (1992). Generalization in digital cartography. Washington, DC, Association of American Geographers.

Mele, P. M. and D. E. Crowley (2008). "Application of self-organizing maps for assessing soil biological quality." Agriculture, Ecosystems & Environment **126**(3-4): 139-152.

Merenyi, E., A. Jain and T. Villmann (2007). "Explicit Magnification Control of Self-Organizing Maps for “Forbidden” Data." IEEE Transactions on Neural Networks **8**(3): 786-797.

Merino, L., F. Caballero, J. R. Martinez-de Dios and A. Ollero (2005). Cooperative Fire Detection using Unmanned Aerial Vehicles. Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, Barcelona, Spain.

Miikkulainen, R. (1990). "Script Recognition with Hierarchical Feature Maps." Connection Science **2**(1): 83 - 101.

Miller, H. and J. Han (2001). Geographic Data Mining and Knowledge Discovery. London, UK, Taylor and Francis.

Miller, H. and J. Han (2001). Geographic data mining and knowledge discovery, an overview. Geographic Data Mining and Knowledge Discovery. H. Miller and J. Han. London, UK, Taylor and Francis**:** 3-32.

Montello, D., S. Fabrikant, M. Ruocco and R. Middleton (2003). Testing the First Law of Cognitive Geography on Point-Display Spatializations. Spatial Information Theory**:** 316-331.

Mu, L. and F. Wang (2008). "A Scale-Space Clustering Method: Mitigating the Effect of Scale in the Analysis of Zone-Based Data." Annals of the Association of American Geographers **98**(1): 85 - 101.

Muñoz, A. and J. Muruzábal (1998). "Self-organizing maps for outlier detection." Neurocomputing **18**(1-3): 33-60.

Nag, A., A. Mitra and S. Mitra (2005). "Multiple outlier detection in multivariate data using self-organizing maps title." Computational Statistics **20**(2): 245-264.

NCGIA. (1990). "NCGIA Core Curriculum in GIS." from http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/toc.html.

Ng, R. T. and H. Jiawei (2002). "CLARANS: a method for clustering objects for spatial data mining." Knowledge and Data Engineering, IEEE Transactions on **14**(5): 1003-1016.

Niang, A., L. Gross, S. Thiria, F. Badran and C. Moulin (2003). "Automatic neural classification of ocean colour reflectance spectra at the top of the atmosphere with introduction of expert knowledge." Remote Sensing of Environment **86**(2): 257-271.

Nowak, D. J., I. Price and G. B. Lamont (2007). Self organized UAV swarm planning optimization for search and destroy using SWARMFARE simulation. Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come. Washington D.C., IEEE Press.

O'Reilly, T. (2005). "What Is Web 2.0." Retrieved 15-07-2009, 2009, from http://oreilly.com/web2/archive/what-is-web-20.html.

O'Sullivan, D. and D. J. Unwin (2002). Geographic information analysis, John Wiley and Sons.

Ogren, P., E. Fiorelli and N. E. Leonard (2004). "Cooperative control of mobile sensor networks:Adaptive gradient climbing in a distributed environment." IEEE Transactions on Automatic Control **49**(8): 1292-1302.

Olson, J. (1976). "Noncontiguous area cartograms." The Professional Geographer **28, 4**: 371-380.

Openshaw, S. (1984). The modifiable areal unit problem. Norwich, England, GeoBooks - CATMOG 38.

Openshaw, S. (1994). Two exploratory space-time-attribute pattern analysers relevant to GIS. Spatial Analysis and GIS. S. Fotheringham and P. Rogerson. London, Taylor and Francis**:** 83-104.

Openshaw, S. (1994). 'What is GISable spatial analysis? New Tools for Spatial Analysis. Luxembourg Eurostat**:** 36-44.

Openshaw, S. (1999). Geographical data mining: key design issues. Proceedings of the 4th International Conference on GeoComputation, Mary Washington College Fredericksburg, Virginia, USA.

Openshaw, S. (2000). GeoComputation. GeoComputation. S. Openshaw and R. J. Abrahart. London, Taylor and Francis**:** 1-31.

Openshaw, S., M. Charlton, C. Wymer and A. Craft (1987). "A Mark 1 Geographical Analysis Machine for the automated analysis of point data sets." International Journal of Geographical Information Science **1**(4): 335 - 358.

Openshaw, S., B. M. and W. C. (1995). "Using Neurocomputing Methods to Classify Britain's Residential Areas." from http://www.geog.leeds.ac.uk/papers/95-1/.

Openshaw, S. and C. Openshaw (1997). Artificial Intelligence in Geography, John Wiley & Sons, Inc.

Openshaw, S. and C. Wymer (1995). Classifying and regionalizing census data. Census users handbook. S. Openshaw. Cambrige, UK, GeoInformation International**:** 239-268.

OpenStreetMap. (2004). "OpenStreetMap." Retrieved 10-07-2009, from http://www.openstreetmap.org/.

Oyana, T. J., D. Boppidi, J. Yan and J. S. Lwebuga-Mukasa (2005). Exploration of geographic information systems (GIS)-based medical databases with self-organizing maps (SOM): A case study of adult asthma. Proceedings of the 8th International Conference on GeoComputation, Ann Arbor, University of Michigan.

Pampalk, E., G. Widmer and A. Chan (2004). "A new approach to hierarchical clustering and structuring of data with Self-Organizing Maps." Intell. Data Anal. **8**(2): 131-149.

Pearson, K. (1901). "LIII On lines and planes of closest fit to systems of points in space." Philosophical Magazine Series 6 **2**(11): 559 - 572.

Penn, B. S. (2005). "Using self-organizing maps to visualize high-dimensional data." Computers and Geosciences **31**(5): 531-544

Plane, D. A. and P. A. Rogerson (1994). The Geographical Analysis of Population: With Applications to Planning and Business. New York, John Wiley & Sons.

Pölzlbauer, G. (2004). Survey and comparison of quality measures for self-organizing maps. Proceedings of the 5th workshop on Data Analysis, Elfa Academic Press.

Przytula, K. W. and K. P. Viktor (1993). Parallel digital implementations of neural networks, Prentice-Hall, Inc.

Raisz, E. (1938). General cartography. New York, London, McGraw-Hill Book Company.

Rees, P., D. Martin and P. Williamson (2002). Census data resources in the United Kingdom. The Census Data System. P. Rees, D. Martin and P. Williamson. Chichester, Wiley**:** 1-24.

Richardson, A. J., C. Risien and F. A. Shillington (2003). "Using self-organizing maps to identify patterns in satellite imagery." Progress in Oceanography **59**(2-3): 223-239.

Ritter, H. (1991). "Asymptotic level density for a class of vector quantization processes." IEEE Transactions on Neural Networks **2**(1): 173-175.

Ritter, H. (1999). Self-organizing maps in non-euclidean spaces. Kohonen maps. E. Oja and S. Kaski. Berlin Springer**:** 97-108.

Ritter, H. and K. Schulten (1986). "On the stationary state of Kohonen's self-organizing sensory mapping." Biological Cybernetics **54**: 99-106.

Ritter, H. and K. Schulten (1988). Extending Kohonens Self-organizing Mapping Algorithm to Learn Ballistic Movements. NATO ASI Series, Vol F41 - Neural Computers, Springer-Verlag. **F41**.

Robinson, A. H., J. L. Morrison, P. C. Muehrcke, A. J. Kimerling and S. C. Guptill (1984). Elements of Cartography. New York, John Wiley & Sons.

Robinson, W. S. (1950). "Ecological Correlations and the Behavior of Individuals." American Sociological Review **15**(3): 351-357.

Rubio, J. C., J. Vagners and R. Rysdyk (2004). Adaptive Path Planning for Autonomous UAV Oceanic Search Missions. Proceedings of the AIAA 1st Intelligent Systems Technical Conference, American Institute of Aeronautics and Astronautics Press, Reston, VA, AIAA.

Saavedra, C., R. Salas, S. Moreno and H. Allende (2007). Fusion of Self Organizing Maps. Computational and Ambient Intelligence**:** 227-234.

Salas, R., S. Moreno, H. Allende and C. Moraga (2007). "A robust and flexible model of hierarchical self-organizing maps for non-stationary environments." Neurocomput. **70**(16-18): 2744-2757.

Sammon, J. W. (1969). "A Nonlinear Mapping for Data Structure Analysis." IEEE Transactions on Computers **C18**(5): 401-409.

Sander, J., M. Ester, H.-P. Kriegel and X. Xu (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications." Data Mining and Knowledge Discovery **2**(2): 169-194.

Sarajedini, B. A. and P. M. Chau (1996). Cumulative Distribution Estimation With Neural Networks. WCNN'96, California, Laurence Erlbaum.

SAS Institute Inc. (2009). SAS Enterprise Miner. Cary, USA

Sauvage, V. (1997). The T-SOM (Tree-SOM). Advanced Topics in Artificial Intelligence**:** 389-397.

Schmid, S. (2005). "CartoCreator: cartogram creator." Retrieved 10-08-2005, from http://arcscripts.esri.com/details.asp?dbid=14226.

Schouwenaars, T. (2006). Safe Trajectory Planning of Autonomous Vehicles. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology. **Doctor of Philosophy**.

Seiffert, U. and B. Michaelis (1995). Three-dimensional self-organizing maps for classification of image properties. 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems(ANNES'95), Dunedin, New Zealand, IEEE Computer Society.

Sester, M. (2005). "Optimization approaches for generalization and data abstraction." International Journal of Geographical Information Science **19**(8): 871 - 897.

Sester, M. (2008). Self-Organizing Maps for Density-Preserving Reduction of Objects in Cartographic Generalization. Self-Organising Maps: Applications in Geographic Information Science. P. Agarwal and A. Skupin**:** 107-120.

Sheikholeslami, G., S. Chatterjee and A. Zhang (1998). WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. Proceedings of the 24rd

International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.

Shuman, B. A. (1992). Foundations and Issues in Library and Information Science. Englewood, Colorado, Libraries United Inc.

Silipo, R. (1999). Neural Networks. Intelligent Data Analysis: An Introduction. M. Berthold and D. J. Hand, Springer-Verlag New York, Inc.**:** 432.

Silva, N. C. and N. C. Rosa (2002). Estimative of SOM learning parameters using genetic algorithms. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando , FL USA SCI 2002.

Skupin, A. (2002). "A cartographic approach to visualizing conference abstracts." Computer Graphics and Applications, IEEE **22**(1): 50-58.

Skupin, A. (2003). A novel map projection using an artificial neural network. Proceedings of 21st International Cartographic Conference, Durban, South Africa, ICC.

Skupin, A. and R. Hagelman (2005). "Visualizing Demographic Trajectories with Self-Organizing Maps." GeoInformatica **9**: 159-179.

Sneath, P. H. A. and R. R. Sokal (1973). Numerical taxonomy: The principles and practice of numerical classification, W.H. Freeman.

Sokal, R. R. and P. H. A. Sneath (1963). Principals of Numerical Taxonomy, W.H. Freeman.

Spielman, S. E. and J.-C. Thill (2008). "Social area analysis, data mining, and GIS." Computers, Environment and Urban Systems **32**(2): 110-122.

SPSS Inc. (2009). SPSS (Statistical Package for the Social Sciences). Chicago, Illinois, USA.

Suganthan, P. N. (1999). "Hierarchical overlapped SOM's for pattern classification." Neural Networks, IEEE Transactions on **10**(1): 193-196.

Sui, D. Z. (2004). "Tobler's First Law of Geography: A Big Idea for a Small World?" Annals of the Association of American Geographers **94**(2): 269-277.

Takatsuka, M. and M. Gahegan (2002). "GeoVISTA Studio: a codeless visual programming environment for geoscientific data analysis and visualization." Computers & Geosciences **28**(10): 1131-1144.

Thill, J.-C., W. A. K. Jr, I. Casas and X. Yao (2008). Detecting Geographic Associations in English Dialect Features in North America within a Visual Data Mining Environment Integrating Self-Organizing Maps. Self-Organising Maps:

Applications in Geographic Information Science. P. Agarwal and A. Skupin**:** 87-106.

Tobler, W. (1961). Map transformation of geographic space, University of Washington.

Tobler, W. (1973). "A continuous transformation useful for districting." Annals, New York Academy of Sciences **219**: 215-220.

Tobler, W. (1986). "Pseudo-cartograms." The American Cartographer **13,1**: 43-50.

Tobler, W. (2004). "On the First Law of Geography: A Reply." Annals of the Association of American Geographers **94**(2): 304 - 310.

Tobler, W. (2004). Thirty-five years of computer cartograms. Annals, Assoc. American Geographers. **94(1):** 58-73.

Tobler, W. R. (1970). "A Computer Movie Simulating Urban Growth in the Detroit Region." Economic Geography **46**: 234-240.

Todorovic, S. and M. C. Nechyba (2004). Intelligent missions for MAVs: visual contexts for control, tracking and recognition. Proceedings of the IEEE International Conference on Robotics and Automation, ICRA '04, New Orleans, LA, USA, IEEE.

Tomlinson, R. (1998). The Canada geographic information system. The History of Geographic Information Systems: Perspectives from the Pioneers. T. W. Foresman. Upper Saddle River, NJ, Prentice Hall**:** 21-32.

Tomlinson, R. F. (1984). Keynote address: geographical information systems-a new frontier. International Symposium on Spatial Data Handling,. Zurich. **1:** 2-3.

Torgerson, W. (1952). "Multidimensional scaling: I. Theory and method." Psychometrika **17**(4): 401-419.

Tsao, C.-Y. and C.-H. Chou (2008). Discovering Intraday Price Patterns by Using Hierarchical Self-Organizing Maps. JCIS-2008 Proceedings, Advances in Intelligent Systems Research, Shenzhen, China, Atlantis Press.

Turner, A. (2006). Introduction to neogeography, O'Reilly.

UCGIS. (2001). "Bylaws of the University Consortium for Geographic Information Science." Retrieved 20-07-2009, from http://www.ucgis.org/fByLaws.html.

Ultsch, A. (2005). Clustering with SOM: U*C. WSOM 2005, Paris.

Ultsch, A., L. Hermann and p. 1-6). (2005). Architecture of emergent self-organizing maps to reduce projection errors. 13th European Symposium on Artificial Neural Networks ESANN 2005, Bruges, Belgium.

Ultsch, A. and F. Moerchen (2005). ESOM-Maps: tools for clustering,visualization, and classification with Emergent SOM, Dept. of Mathematics and Computer Science, University of Marburg, Germany.

Ultsch, A. and H. P. Siemon (1990). Kohonen´s self-organizing neural networks for exploratory data analysis. Proceedings of the International Neural Network Conference, Paris, Kluwer.

Vallejo, E., M. Cody and C. Taylor (2007). Unsupervised Acoustic Classification of Bird Species Using Hierarchical Self-organizing Maps. Progress in Artificial Life**:** 212-221.

Vesanto, J. (1999). "SOM-based data visualization methods." Intelligent Data Analysis **3**: 111-126.

Vesanto, J. (2000). Using SOM in data mining. Department of Computer Science and Engineering. Espoo, Finland, Helsinki University of Tecnology. **Licenciate:** 50.

Vesanto, J., J. Himberg, E. Alhoniemi and J. Parhankangas (1999). Self-organizing map in Matlab: the SOM Toolbox. Proceedings of the Matlab DSP Conference, Espoo, Finland, Comsol Oy.

Vesanto, J., J. Himberg, E. Alhoniemi and J. Parhankangas (2000). SOM Toolbox for MATLAB 5. Technical Report A57. Helsinki Helsinki University of Technology**:** 60.

Vesanto, J., M. Sulkava and J. Hollmen (2003). On the decomposition of the self-organizing map distortion measure. Proceedings of the Workshop on Self-Organizing Maps (WSOM'03), Hibikino, Kitakyushu, Japan.

Villmann, T., R. Der, M. Herrmann and T. M. Martinetz (1997). "Topology preservation in self-organizing feature maps: Exact definition and measurement." IEEE Transactions on Neural Networks **8**(2): 256-266.

Villmann, T., E. Merenyi and B. Hammer (2003). "Neural maps in remote sensing image analysis." Neural Networks **16**(3-4): 389-403.

Viscovery Software GmbH (2009). Viscovery SOMine 5.0. Vienna, Austria.

Visipoint Ltd. (2009). VisiSOM. Kuopio, Finland.

Wan, W. and D. Fraser (1993). M2dSOMAP: clustering and classification of remotely sensed imagery by combining multiple Kohonen self-organizing maps and associative memory. Proceedings of 1993 International Joint Conference on Neural Networks, IJCNN '93 Nagoya, Japan, Supermolecular Science Division Electrotechni.

Wan, W. and D. Fraser (1994). A self-organizing map model for spatial and temporal contextual classification. IGARSS'94, Surface and Atmospheric Remote Sensing : Technologies, Data, Analysis and Interpretation, Pasadena, California, USA, IEEE.

Wikimapia. (2006). "Wikimapia." Retrieved 10-07-2009, from http://en.wikipedia.org/wiki/Wikimapia.

Wikipedia. (2009). "Application programming interface." Retrieved 19-08-2009, from http://en.wikipedia.org/wiki/Application_programming_interface.

Wu, Y. and M. Takatsuka (2005). The Geodesic Self-Organizing Map and its error analysis. Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38. Newcastle, Australia, Australian Computer Society, Inc.

Wu, Y. and M. Takatsuka (2006). "Spherical self-organizing map using efficient indexed geodesic data structure." Neural Networks **19**(6-7): 900-910.

Yun, L. and K. Uchimura (2007). "Using Self-organizing Map for Road Network Extraction from Ikonos Imagery." International Journal of Innovative Computing, Information and Control (IJICIC) **3**(3): 641-656.

Zengin, U. and A. Dogan (2006). Real-Time Target Tracking for Autonomous UAVs in Adversarial Environments: A Gradient Search Algorithm. 45th IEEE Conference on Decision and Control, San Diego, CA, USA.

Zheng, C., K. Ahmad, A. Long, Y. Volkov, A. Davies and D. Kelleher (2007). Hierarchical SOMs: Segmentation of Cell-Migration Images. Advances in Neural Networks – ISNN 2007**:** 938-946.

Zrehen, S. and F. Blayo (1992). A geometric organization measure for Kohonen's map. Neuro-Nîmes 92 : neural networks & their applications, Nîmes, France.

Appendixes

Appendix 1. Carto-SOM code

Functions index


```
%% batch file
% Parameters
%testes=[Lin Col Epoc1 Epoc2 R1 R2 NumTotalDados m MagnifiedOcean DoubleTrain]
testes =[400 588 -99    40  -99 8   100000     0.67   0               0 ];

% paths
path(path,'D:\cartoSOM\Somtoolbox');

% Load pop data
str=strcat(cd,'\dados\State_Area_Pop.mat');
% load area info: State_Area_Pop
load (str);

% create error matrix
Erros=zeros(length(testes),9); % ceM,ceQ,ceW,kmM,kmQ,kmW,kdM,kdQ,kdW

% load points data
str=strcat(cd,'\dados\Matriz_Dados.mat');
load (str);

% testing
iter=size(testes);
iter=iter(1,1);
for i = 3:3 % each set of tests
    str=strcat('Db','_',num2str(i));
    disp(str);

    xNeurons=testes(i,1);
    yNeurons=testes(i,2);
    nEpochs_1=testes(i,3);
    nEpochs_2=testes(i,4);
    radius_ini_1=testes(i,5);
    radius_ini_2=testes(i,6);

    alpha_ini_1=0.4;
    alpha_ini_2=0.1;

    % load the correct proportion of points for each state
    proporcaoDes=cartoSom_CalculateNumInputPatterns(State_Area_Pop, ...
        testes(i,8), testes(i,7),testes(i,9));

     if(length(find(proporcaoDes==0))>0)
         %WARNING
         warning('There exist some states with 0 points. This calculation is aborted!');
         % create a new folder
         mkdir (strcat('testes\',str));
         save (strcat('testes\',str,'\ProblemaNosDados.mat'), 'proporcaoDes','-mat');
```

```
        continue
     end

     %get population sample
    [Dados]=cartoSom_GetPopulationSample_v2(proporcaoDes, Matriz_Dados, 1);
    clear Matriz_Dados

    % create cartogram using SOMPAK
    NumberUnits_State= cartoSOM_SomPak_setembro07(str,...
                        xNeurons,yNeurons,nEpochs_1,nEpochs_2,radius_ini_1,...
                        radius_ini_2,alpha_ini_1,alpha_ini_2,Dados,...
                        testes(i,8),testes(i,7),testes(i,10));

    % global cartogram errors
    Erros(i,:)=cartoSOM_CalculateErrors(State_Area_Pop,NumberUnits_State);
    save ('testes\Erros.mat', 'Erros','-mat');

    % cartogram errors by region
    parcialErrors=cartoSOM_CalculateParcialErrors(State_Area_Pop,NumberUnits_State);
    str=strcat('testes\',str,'\parcialErrors.mat');
    save (str, 'parcialErrors','-mat');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[nInputPatterns]=cartoSom_CalculateNumInputPatterns(StateMatrix, m,
TotalNumPoints, MagnifiedOcean)
% [nInputPatterns]=cartoSom_CalculateNumInputPatterns(desired, baseSample, rnd)
%
% NAME: cartoSom_CalculateNumInputPatterns
%
% OBJECTIVE: Obtains the number of input patterns for each state
%
% INPUT PARAMETERS:
%       StateMatrix - nx3 matrix. 1st columns is ID, 2nd is Area,
%                 and third is Population
%       m           - magnification factor
%       MagnifiedOcean - 0/1 variable defining if the number of input
%                        in ocean is magnified (1) or linearly
%                        calculated(0)
%
% OUTPUT PARAMETERS:
%       nInputPatterns - nx1 matrix with the Numb of Input Patterns
%                  for each state
%
% COMMENTS:
% V.0.1 2007/08/30 By R.Henriques, V.Lobo & F.Bacao

% exp1=1-(1/m)
% exp2=1/m
%a=area.^(exp1);
%p=pop.^(exp2);
%k= TotalNumPoints./ sum(a.*p);
%nInputPatterns=round(k.*a.*p);

exp1=1-(1/m);
exp2=1/m;

if (MagnifiedOcean)
    a=StateMatrix(:,2).^(exp1);
    p=StateMatrix(:,3).^(exp2);
    k= TotalNumPoints./ sum(a.*p);
    nInputPatterns=round(k.*a.*p);
else
    numOceanPnts=round(StateMatrix(1,3)./sum(StateMatrix(:,3)).*TotalNumPoints);

    a=StateMatrix(2:length(StateMatrix),2).^(exp1);
    p=StateMatrix(2:length(StateMatrix),3).^(exp2);
    k= (TotalNumPoints-numOceanPnts)./ sum(a.*p);
```

```
    nInputPatterns=round(k.*a.*p);
    nInputPatterns=[numOceanPnts ;nInputPatterns];
end
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [sample]=cartoSom_GetPopulationSample_v2(desired, baseSample, rnd)
% [sample]=cartoSom_GetPopulationSample(desired, baseSample, rnd)
%
% NAME: cartoSom_GetPopulationSample
%
% OBJECTIVE: Obtain a samples population set
%
% INPUT PARAMETERS:
%       desired - nx1 matrix with the desired population for each state
%       baseSample - nx3 matrix base sample. 1st columns is X, 2nd is Y,
%                  and third is state ID
%       rnd     - flag indicating if the sample should be randomized (=1)
%
% OUTPUT PARAMETERS:
%       sample - nx3 matrix with the desired sample. 1st columns is X, 2nd
%                is Y,and third is state ID
%
% COMMENTS:
%       state ID starts with 0, which is the ID for the ocean (or
%       non-populated areas
%

% V.0.1 2007/08/27 By V.Lobo, R.Henriques, & F.Bacao

nstates=length(desired);           % find how many states there are
ndesired=sum(desired);
sample=zeros(ndesired,3);              % pre-dimension the sample

ii=1;fi=1;                             % initial and final i for each state
for state=0:nstates-1
    candidates=find(baseSample(:,3)==state);%find the desired state
    ncandidates=length(candidates);      %see how many points there are
                                            %for this state in the base
    if (ncandidates>0)
        ds=desired(state+1);            %n of desired samples for state
        tmpindex=ceil(ncandidates*rand(ds,1)+eps); %index into avaiable samples
        indexes=candidates(tmpindex);           %find the true indexes
        fi=ii+desired(state+1)-1;
        sample(ii:fi,:)=baseSample(indexes,:);   %select the samples;
        ii=fi+1;
    end
end;

sample2=sample(find(sample(:,1)~=0&sample(:,2)~=0),:);

if rnd==1
    indexes=randperm(length(sample2));
    sample=sample2(indexes,:,:);
end;

return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function NumberUnits_State=cartoSOM_SomPak(testName,...
xNeurons,yNeurons,nEpochs_1,nEpochs_2,radius_ini_1,radius_ini_2,...
alpha_ini_1,alpha_ini_2,Dados,magnificationFactor, NumTotalDados, TreinoDuplo);

% CartoSOM
%
% CartoSOM algorithm using all units and using the SOMPak software
%
% cartoSOM_var2_SomPak(xNeurons,yNeurons,niterations_1, niterations_2,
```

```
%radius_ini_1,radius_ini_2,alpha_ini_1,alpha_ini_2);
%
%  xNeurons      = number of units in x
%  yNeurons      = number of units in y
%  niterations_1 = number of epochs in the first train
%  niterations_2 = number of epochs in the second train
%  radius_ini_1  = initial radius in the first train
%  radius_ini_2  = initial radius in the second train
%  alpha_ini_1   = alpha initial value in the first train
%  alpha_ini_2   = alpha initial value in the second train
%
%  TreinoDuplo  = 1 para treinar 2 vezes e 0 para treinar apenas 1 vez

% Roberto Henriques, Victor Lobo & Fernando Bacao Setembro 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
tic;

cd testes
%% create a new folder
mkdir (testName);
cd ..

%%
%load DADOS txt file
%str=strcat('pontos\',pointFile);
%usa=load (str, '-ascii');
[num_dados num_features]=size(Dados);
num_features=num_features-1;    % os ficheiros incluem labels

%create Som Data
sD=som_data_struct(Dados(:,1:2));
%clear usa;

%initialize som
sM1 = som_randinit_roberto( sD, 'msize',[xNeurons yNeurons],'rect','sheet');
sM1.neigh = 'gaussian';

if (TreinoDuplo==1)
    cd Sompak
    %% train SOM
    disp(1st phase');

    sM1=sompak_train(sM1,[],[],[],sD,[],nEpochs_1*num_dados,alpha_ini_1,radius_ini_1);

    cd ..
end

cd Sompak

disp('2nd phase');

sM1=sompak_train(sM1,[],[],[],sD,[],nEpochs_2*num_dados,alpha_ini_2,radius_ini_2);

cd ..

%% progress
disp('autolabeling');

%% load point file
% str=strcat('pontos\',pointFile);
% usa=load (str, '-ascii');

disp('autolabeling 1');
usa_labels=Dados(:,num_features+1);
[n1,n2]=size(usa_labels);

disp('autolabeling 2');
```

```
label_cell=mat2cell(usa_labels,ones(1,n1),[n2]); % criar lista de labels
sMlixo=som_randinit_roberto( sD, 'msize',[xNeurons yNeurons],'rect','sheet');

disp('autolabeling 3');
sM1.labels=sMlixo.labels;
clear sMlixo;
clear usa_labels;
clear usa;

disp('autolabeling 4');
sD.labels=label_cell;
%sM1=som_autolabel(sM1,sD,'vote');
clear label_cell;

disp('autolabeling 5');
M= labels2mat(sM1.labels);

%Alteracao do colormap
jet1=jet;
jet1(1,:)=[1 1 1];

disp('BMUS');

% Figure with the BMU (invertido) --> CARTOGRAM
figure(1);
Bmus=carto_nearestDado (sM1.codebook, sD);
outMat=reshape(Bmus,sM1.topol.msize(1),sM1.topol.msize(2));
outMat=[outMat ; zeros(1,sM1.topol.msize(2))];
outMat=[outMat zeros(sM1.topol.msize(1)+1,1)];
h = pcolor(flipud(outMat));
title(strcat('Cartograma-', testName));
COLORMAP(jet1);

%save figure
str=strcat('Testes\',testName,'\cartograma');
saveas(h,str)

%save bmus --> from this file we can build a GIS environment cartogram
str=strcat('testes\',testName,'\BMUs.txt');
save (str, 'Bmus' ,'-ascii');

%% Calculate Number of units for each state
Estados=labels2mat(sD.labels);
EstadoMin=min(Estados);
EstadoMax=max(Estados);

EstadosDiferentes=unique(Estados);
nUnitsState=zeros(length(EstadosDiferentes)-1,2); % sem o mar

i=0;
%for each  state=EstadoMin:EstadoMax
for estado=0:EstadoMax % exlude the ocean (state=-1)
    i=i+1;
    nUnitsState(i,1)= estado;
    nUnitsState(i,2)=length(find(Bmus == estado));
end;

str=strcat('testes\',testName,'\contagem.txt');
save (str, 'nUnitsState' ,'-ascii');

% return the area matrix for error calculation
NumberUnits_State=nUnitsState;

%% Save parameters used in the test to an ascii file
str=strcat('testes\',testName,'\parametros.txt');
fid = fopen(str,'w');
fprintf(fid,'%s \t %3g\n','xNeurons' , xNeurons);
fprintf(fid,'%s \t %3g\n','yNeurons' , yNeurons);
fprintf(fid,'%s \t %5g\n','niterations_1' , nEpochs_1);
```

```
fprintf(fid,'%s \t %5g\n','niterations_2' , nEpochs_2);
fprintf(fid,'%s \t %5g\n','radius_ini_1' , radius_ini_1);
fprintf(fid,'%s \t %5g\n','radius_ini_2' , radius_ini_2);
fprintf(fid,'%s \t %5g\n','alpha_ini_1' , alpha_ini_1);
fprintf(fid,'%s \t %5g\n','alpha_ini_2' , alpha_ini_2);
fprintf(fid,'%s \t %5g\n','magnificationFactor' , magnificationFactor);
fprintf(fid,'%s \t %5g\n','NumTotalDados' , NumTotalDados);
fprintf(fid,'%s \t %5g\n','ProcessTime(sec)' , toc);

fclose(fid);

str=strcat('testes\',testName,'\mapa.mat');
save (str, 'sM1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Error=cartoSOM_CalculateErrors(State_Area_Pop,NumberUnits_State);
% Error=cartoSOM_CalculateErrors(PopAreaState,NumberUnits_State);
%
% cartoSOM_CalculateError
%
% cartoSOM_var2_SomPak(xNeurons,yNeurons,niterations_1, niterations_2,...
% radius_ini_1,radius_ini_2,alpha_ini_1,alpha_ini_2);
%
% State_Area_Pop     = matrix with the population and area for each state
% NumberUnits_State= matrix with the id and number of units for each state

% Roberto Henriques September 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Errors
% Cartogram Error   (ce) = Ai(%)- Vi(%)
% Keim Error        (km) = (Ai(%)- Vi(%) / Ai(%)+ Vi(%)
% Kocmoud Error     (kd) = (Ai(%)- Vi(%) / Vi(%)

% grouping method
% Mean error        (mean of abs values)        (ceM,kmM,kdM)
% Quadratic error   (mean square root quadratic sum) (ceQ,kmQ;kdQ)
% Weighted          (weighted with Vi(%))       (ceW;kmW;kdW)

NumberStates=length(NumberUnits_State);
SumUnits=sum(NumberUnits_State);
ai=NumberUnits_State(:,2)./SumUnits(:,2);
%vi=PopAreaState(:,4);
vi=State_Area_Pop(:,3)./(sum(State_Area_Pop(:,3)));

%% Cartogram Error (ce)
ce= ai-vi;

ceM=mean(abs(ce));
ceQ=sqrt(sumsqr(ce))/NumberStates;
ceW=sum(abs(ce.*vi));

%% Keim Error
km=(ai-vi)./(ai+vi);

kmM=mean(abs(km));
kmQ=sqrt(sumsqr(km))/NumberStates;
kmW=sum(abs(km.*vi));

%% Kocmoud Error
kd=(ai-vi)./vi;

kdM=mean(abs(kd));
kdQ=sqrt(sumsqr(kd))/NumberStates;
kdW=sum(abs(kd.*vi));

%% Return Errors
Error=[ceM,ceQ,ceW,kmM,kmQ,kmW,kdM,kdQ,kdW];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function parcialError=cartoSOM_CalculateParcialErrors(State_Area_Pop,NumberUnits_State);
% parcialError=cartoSOM_CalculateParcialErrors(State_Area_Pop,NumberUnits_S
% tate);
%
% NAME: cartoSOM_CalculateParcialErrors
%
% OBJECTIVE: Calculate cartogram Error per state
%
% INPUT PARAMETERS:
%       State_Area_Pop=     matrix with the population and area for each
%                           state
%       NumberUnits_State=  matrix with the id and number of units for each
%                           state
%
% OUTPUT PARAMETERS:
%       parcialError - nx1 matrix with cartogram Error per state
%
% COMMENTS:
%       state ID starts with 0, which is the ID for the ocean (or
%       non-populated areas
%

% V.0.1 2007/08/31 By  R.Henriques, V.Lobo, & F.Bacao

NumberStates=length(NumberUnits_State);
SumUnits=sum(NumberUnits_State);
ai=NumberUnits_State(:,2)./SumUnits(:,2);
vi=State_Area_Pop(:,3)./(sum(State_Area_Pop(:,3)));

%% Cartogram Error (ce)
ce= ai-vi;

%% Keim Error
km=(ai-vi)./(ai+vi);

%% Kocmoud Error
kd=(ai-vi)./vi;

%% Return parcial Errors
parcialError=[ce,km,kd];

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix 2. GeoSOM Suite manual

Manual index

A2.1 Introduction

GeoSOM Suite is exploratory analysis and clustering tool providing several methods to accomplish that task.

Beside some general methods from descriptive statistics and basic visual exploration tools such as the boxplots or histograms, the GeoSOM Suite presents, at present time, three different methods to perform clustering, with special focus to geospatial data. Those are:

1. Self-Organizing Maps (SOM);
2. Geo Self-Organizing Maps (GeoSOM) and;
3. Hierarchical Self-Organizing Maps (HSOM)

A2.1.1. Self Organizing Maps

Teuvo Kohonen proposed the Self-organizing maps (SOM) in the beginning of the 1980s (Kohonen 1982). The SOM is usually used for mapping high-dimensional data into one, two, or three-dimensional feature maps. The basic idea of an SOM is to map the data patterns onto an *n*-dimensional grid of units or neurons. That grid forms what is known as the output space, as opposed to the input space that is the original space of the data patterns. This mapping tries to

preserve topological relations, *i.e.* patterns that are close in the input space will be mapped to units that are close in the output space, and *vice-versa*. The output space is usually two-dimensional, and most of the implementations of SOM use a rectangular grid of units. To provide even distances between the units in the output space, hexagonal grids are sometimes used (Kohonen 2001). Each unit, being an input layer unit, has as many weights as the input patterns, and can thus be regarded as a vector in the same space of the patterns.

When training an SOM with a given input pattern, the distance between that pattern and every unit in the network is calculated. Then the algorithm selects the unit that is closest as the winning unit (also known as best matching unit- BMU), and that pattern is mapped on to that unit. If the SOM has been trained successfully, then patterns that are close in the input space will be mapped to units that are close (or the same) in the output space. Thus, SOM is 'topology preserving' in the sense that (as far as possible) neighbourhoods are preserved through the mapping process.

A2.1.2. GeoSOM

GeoSOM is an adaptation of SOM to consider the spatial nature of data. In GeoSOM, the search for the best matching unit (BMU) has two phases. The first phase settles the geographical neighbourhood where it is admissible to search for the BMU, and the second phase performs the final search using the other components. A parameter k controls the search neighbourhood defined in the output space. Using k=0 will necessarily select as BMU the unit geographically closer. The same result may be obtained by training a standard SOM with only the geographical locations, and then using each unit as a low pass filter (*i.e.* a sort of average) of the non-geographic features. As k (the geographic tolerance) increases, the unit locations will no longer be quasi-proportional to the locations of the training patterns, and the "equivalent filter" functions of the units will become more and more skewed, eventually ceasing to be useful as models.

Setting k equal to the size of the SOM is equivalent to treat spatial coordinates as any other variable.

A2.1.3. Hierarchical SOM

Hierarchical SOMs (Luttrell 1989; Koikkalainen and Oja 1990; Miikkulainen 1990; Lampinen and Oja 1992; Kemke and Wichert 1993) share many characteristics with other methods such as the multi-layer SOMs (Luttrell 1988; Ichiki, Hagiwara *et al.* 1991), multi-resolution SOMs (Graham and D'Eleuterio 1991), multi-stage SOMs (Li 1989; Lee and Ersoy 2005), fusion SOMs (Saavedra, Salas *et al.* 2007) or Tree-SOMs (Sauvage 1997).

All these methods share the idea of constructing a system using SOMs as building blocks. They vary in the way these SOMs interact with each other, and with the original data. We consider as

Hierarchical SOMs, those where, at some stage, one of the SOMs receives as inputs some form of output of another SOM. This type of structure resembles a multi-layer perceptron (MLP) neural network in the sense that multiple layers exist connected in a feed-forward way. However, Hierarchical SOMs have completely different training algorithms and types of interaction between layers.

General multilayer SOMs may have many completely different interactions between layers. As an example, a data pattern may be mapped onto a given SOM, and then all data patterns mapped to that unit may be visualized on second SOM. Another common type of architecture presents several SOMs in linked windows (Bacao, Lobo *et al.* 2005), providing an environment where a data pattern is visualised simultaneously in several SOMs. We do not consider these as Hierarchical SOM because the outputs of one SOM are not used to actively train another SOM, nor does the second SOM in any way, use information from the first map to map the original data patterns.

A2.1.4. GeoSOM Suite main functionalities

GeoSOM suite has a set of functionalities, allowing the user to interact with data and perform SOM and GeoSOM analyses. Some of the main functionalities are:

- Use of dynamic windows, allowing the user to visualize and/or select in different representations of data.
- Show geographic maps when data has spatial attributes
- Build U-matrices (U-Mat)
- Build component planes (CP)
- Build Hit maps (Hits)
- Build parallel coordinates plots (PCP)
- Create clusters on the U-Mat based on:
    - Visualization
    - Hierarchical clustering

Some of the GeoSOM suite inputs are:

- *ESRI Shapefile*
- *.mat file
- *.csv file
- Database connection (to appear in future versions)

Some of the GeoSOM suite outputs are:

- Save GeosomStruct

- Export all the graphics
- Export input data with each BMU and distance
- Export the different views in shapefile format
- Export the clusters

A2.1.5. GeoSOM technology

GeoSOM suite was implemented using Matlab® and the SOM toolbox. A guide user interface (GUI) was built allowing non-programmers users to evaluate the SOM and GeoSOM algorithms.

A2.2 GeoSOM suite tool

A2.2.1. Interfaces

Main interface

The GeoSOM suite has the following main interface (Fig. 1) constituted by a tree view which will show all the GeoSOM results and a main data grid used to present the data imported by the user.

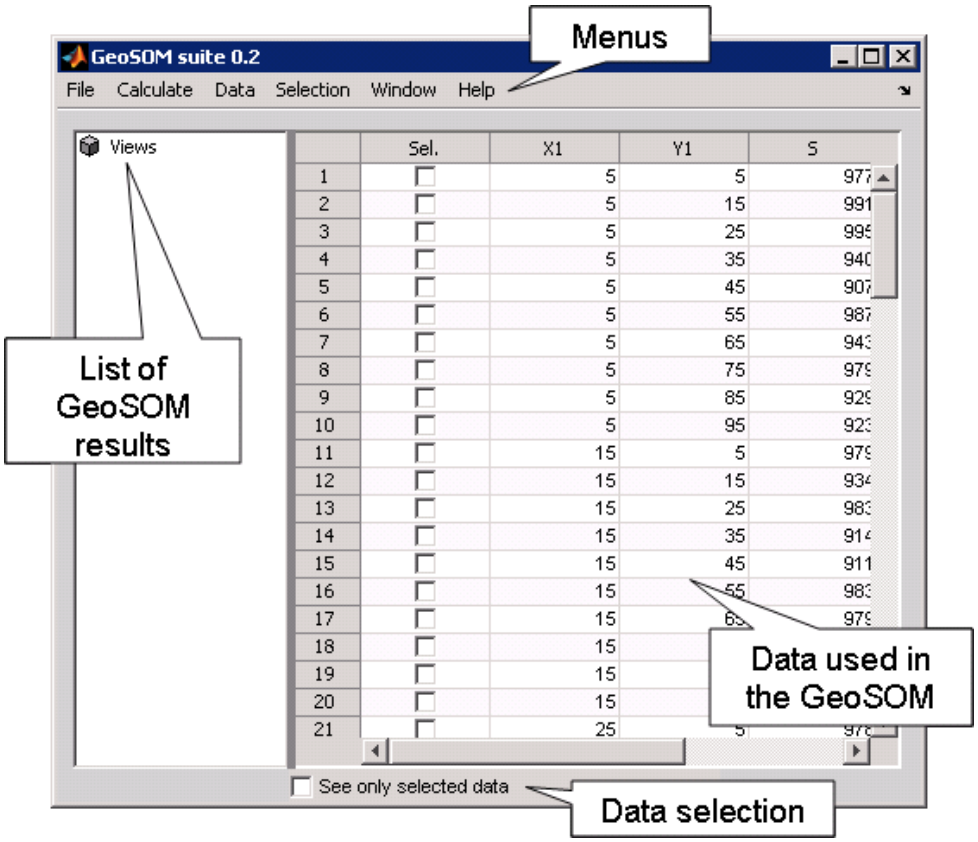
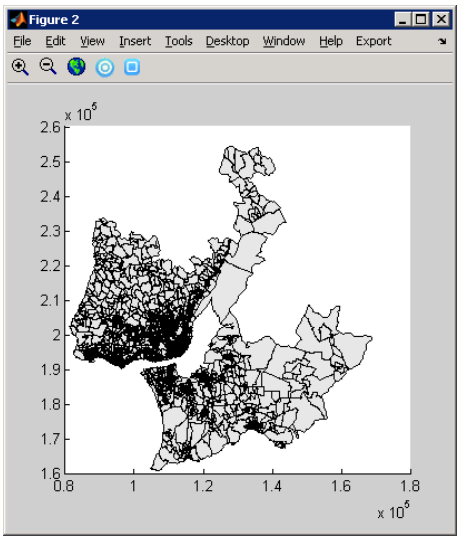


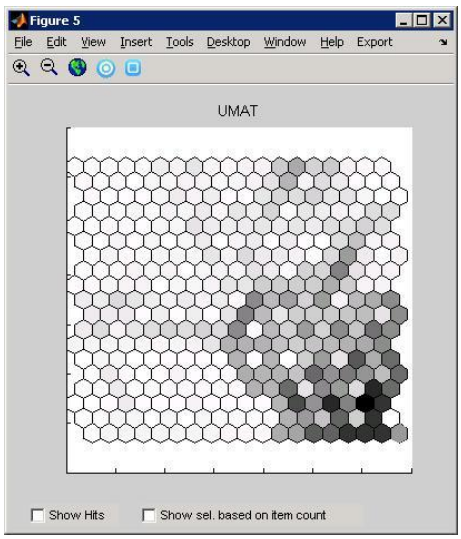
Fig. 1 – GeoSOM main interface

Views

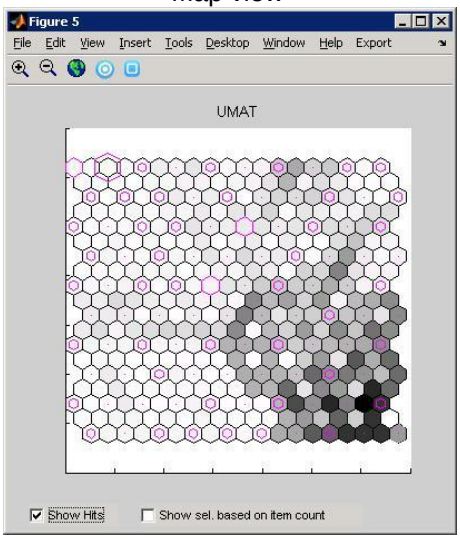
From the main interface, it is possible to create different views. Some of the views possible are the maps, the U-matrices, the component planes, the hits plot and the parallel coordinates plot.



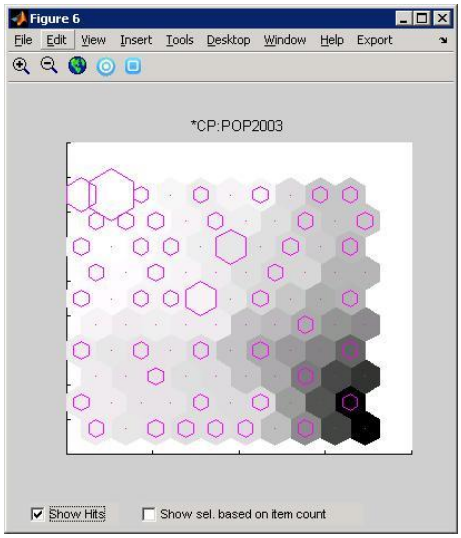
Map view



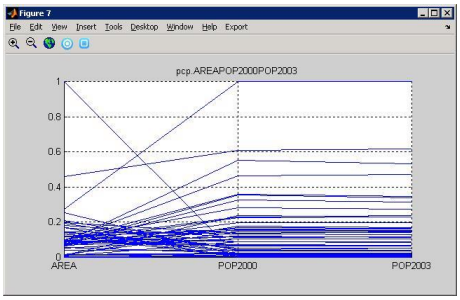
U-mat view



U-mat view with hits



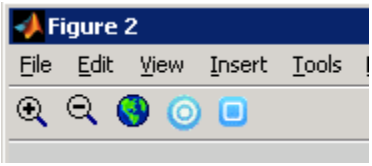
Component plane view with hits



Parallel coordinates plot

Fig. 2 – GeoSOM views

In each view a set of tools are available at the button toolbar:



Button	Description
	Zoom in
	Zoom out
	Full extent
	Select based on one click (pressing shift and left mouse buttons will add the selected features to the previous selection; pressing control and left mouse buttons will remove the selected features to the previous selection)
	Selection based on area (shift and control buttons have the same properties as before)

Also, in each view, a new menu is added to the default MATLAB menu (Export) (Fig. 3). This menu has two sub menus: export to shapefile and to a new figure. The first allows the user to export each view as a shapefile, while the second exports to a new figure only the axes.

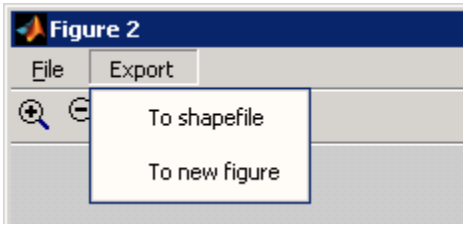


Fig. 3 – New menu in View interface

In the case of a UMAT or CP view, some extra-features are added (Fig. 4).

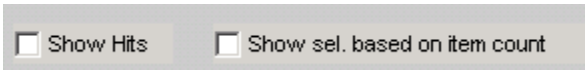


Fig. 4 – Extra features in the case of u-mat and component planes

Checking the “Show Hits” will superpose the hits to the current view. The “Show sel. based on item count” will select units based on its number of data items. Thus, for example in Fig. 5 the units are selected according to the number of data items that share the BMU. Note that the size of the selection feature (a hexagon in the example) is not absolute, *i.e.* for the group of selected

units the one with more data items will be represented with the bigger hexagon, while if a unit has zero data items no hexagon is shown.

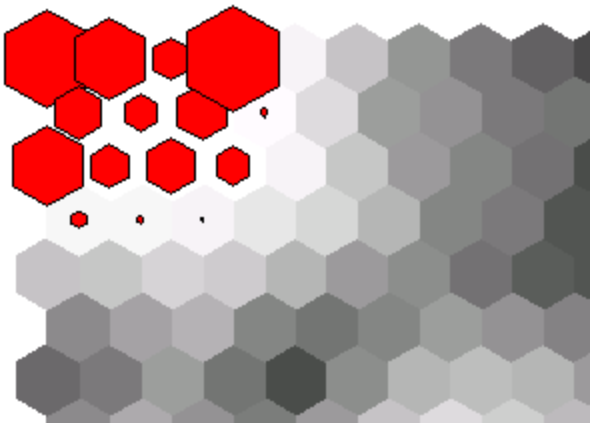


Fig. 5 – "Show sel. based on item count" tool in a cp

A2.2.2. Menus

GeoSOM has a set of menus allowing the user to access the functions described above:

- **File**
  - **Load shapefile** – allows the user to load a shapefile ("*popular geospatial vector data format for geographic information systems software. It is developed and regulated by ESRI as a (mostly) open specification for data interoperability among ESRI and other software products.[1] A "shapefile" commonly refers to a collection of files with ".shp", ".shx", ".dbf", and other extensions on a common prefix name (e.g., "lakes.*"). The actual shapefile relates specifically to files with the ".shp" extension, however this file alone is incomplete for distribution, as the other supporting files are required"*). from wikipedia.
  - **Load csv/mat file –** allows the user to load a csv or a mat file. While csv file has the ability the store numeric and non numeric data, mat files only store numeric data. Thus, when using mat files, in order to load data and data headers, the user must create a *mat* file called *header_theNameOfTheOriginalMatFile.* For instance, if we are loading a file called iris.mat, then the user must create a cell array with the headers for this data and save it as header_iris.mat in the same location where iris.mat is saved. In case this file is not supplied then GeoSOM will use var1 to varN as headers.
  - **Connect to database –** allows the user to import data from a database (not supported in this version)
  - **Load GeoSOM struct –** allows to load a pre-saved GeoSOM struct
  - **Save GeoSOM struct -** allows to save the current GeoSOM struct

- **Close GeoSOM –** closes GeoSOM

- **Calculate**
  - **Train SOM –** opens a new form allowing to define all SOM parameters to be used in the SOM construction
  - **Train GeoSOM -** opens a new form allowing to define all GeoSOM parameters to be used in the GeoSOM construction

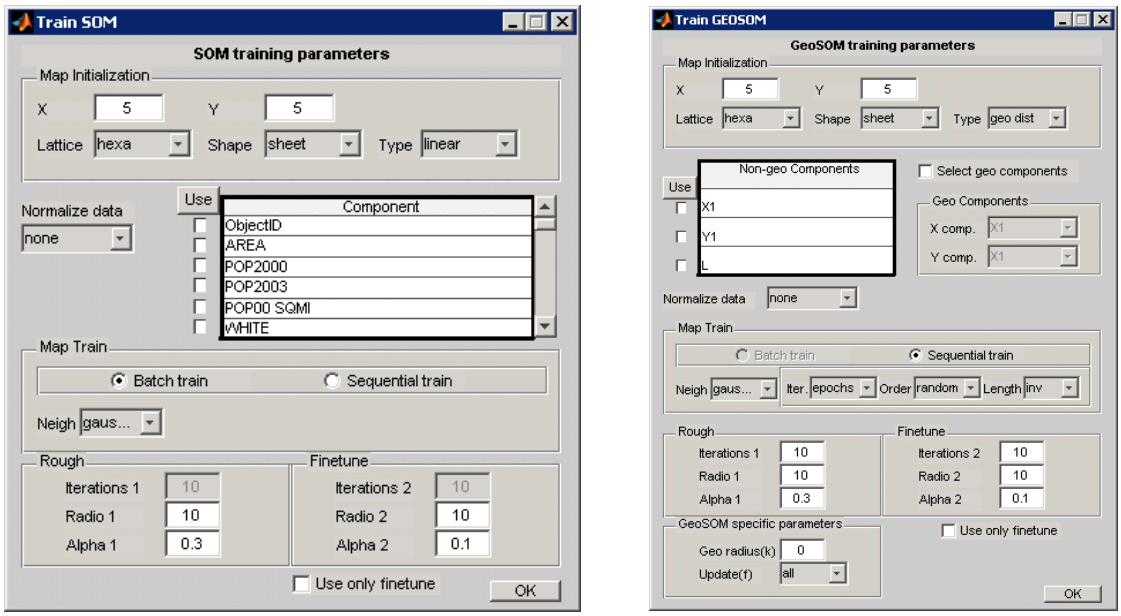


Fig. 6 – GeoSOM calculate SOM and GeoSOM interface

These two interfaces are similar, allowing the user to define all SOM parameters. The differences in the case of GeoSOM are:

- Definition of a new parameter, the geo-radius (*k*)
- Definition of a new parameter , the update type (*f*)
- Definition of the geo components. The user has the ability to select the geo components from the list of attributes (in case data has an *x* and *y* fields). Leaving this field unselected will mean that the spatial information will be collected from the spatial data (the centroid is calculated using a mean on the feature bounding box)

- **Data**
  - **Query –** allows the user to query data. The user has the ability of querying data based on a simple query (Select all data items where variable <=> than a value) or based on a Matlab expression (find(data(:,1)>0 & data (:,2)<1 → this would select all items with variable1 bigger than 0 and variable 2 smaller than 1). The Matlab expression feature will only work with numeric variables.

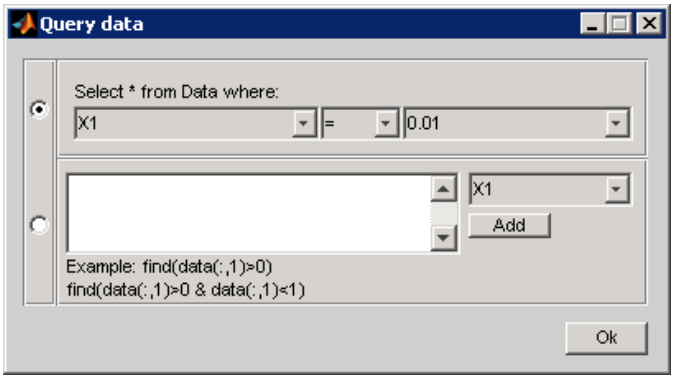


Fig. 7 – Query tool

- **Export –** allows the user to export all/selected data
- **Create box histo –** allows the user to explore data using histograms and box plots. Notice that selected features will also be shown.

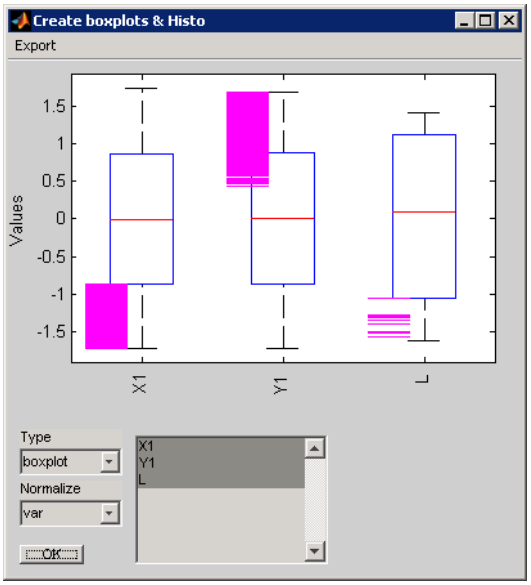


Fig. 8 – Box plot and histogram tool

- **Window**
  - **Dock windows –** docking windows function (not supported yet)
  - **Close all views –** deletes all open views
  - **Preferences** - allows the user to define GeoSOM suite preferences

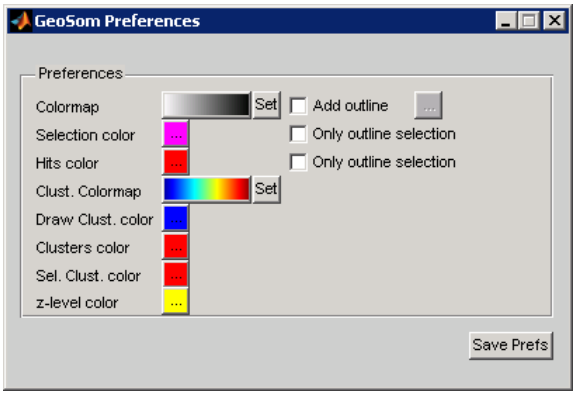
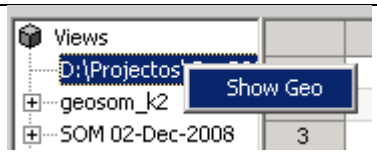
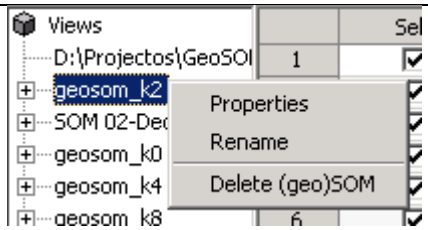
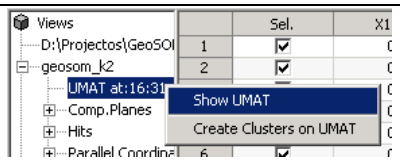
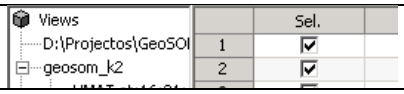


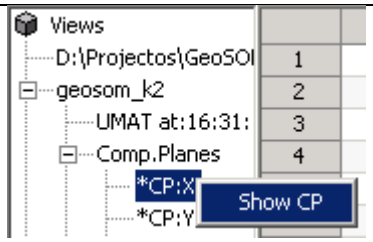
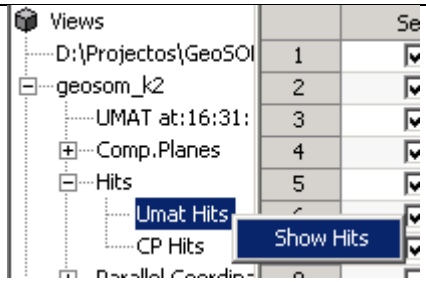
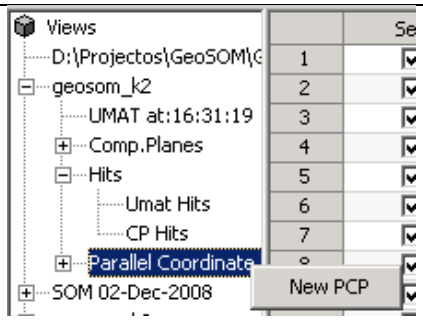
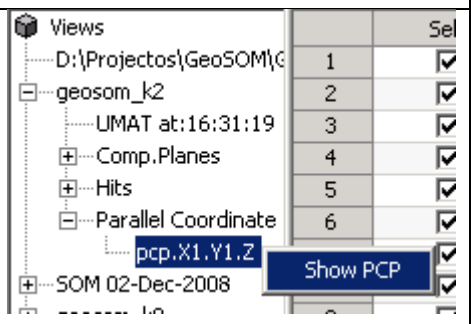
Fig. 9 – GeoSOM suite preferences

- **Help**
  - **Help –** opens the GeoSOM help file
  - **About –** GeoSOM credits

A2.2.3. Popup menus

To access functions related to the views presented in the tree view from the main interface, a set of popup menus are available. These functions can be used by selecting the item in the tree view and a right click on the mouse. Depending on the type of element selected the available functions change.

Element	Functions	Description	Image
Map	Show Geo	Opens a view with a map	
SOM or GeoSom element	Properties	Show the parameters used in the training phase	
	Rename	Change the name of the (geo)SOM	
	Delete (geo)SOM	Delete the (geo)SOM	
U-mat	Show U-mat	Shows the u-mat	
	Create clusters on UMAT	Allow the user to create clusters based on the u-mat	
Component planes	Show CPs used in SOM	Opens a new figure with the CPs for all the	

Element	Functions	Description	Image
		variables used in the training	
	Show all CP	Opens a new figure with the CPs for all the variables	
Component plane	Show CP	Opens a specific component plane	
Hits Umat hits CP hits	Show hits	Opens the hits plot adapted to the u-mat or cps	
Parallel coordinates Plots (group)	New PCP	Creates a new pcp based on the variables selected by the user	
Parallel coordinates plot	Show PCP	Opens a specific parallel coordinates plot	

A2.3 Quick start with GeoSOM Suite

Lets us present an example of using GeoSOM suite tool for data analysis. In this example we are using the USA.shp with census data obtained from the ESRI Data & Maps CD.

1. Open GeoSOM
2. Select *File>Load shapefile* and select *USA.shp*
3. The data grid in the main form will be filled with *USA.shp* attributes and a new window with the spatial information appears. Also, in the Views tree a new item is added. This item has the same name has the path of *USA.shp* file.

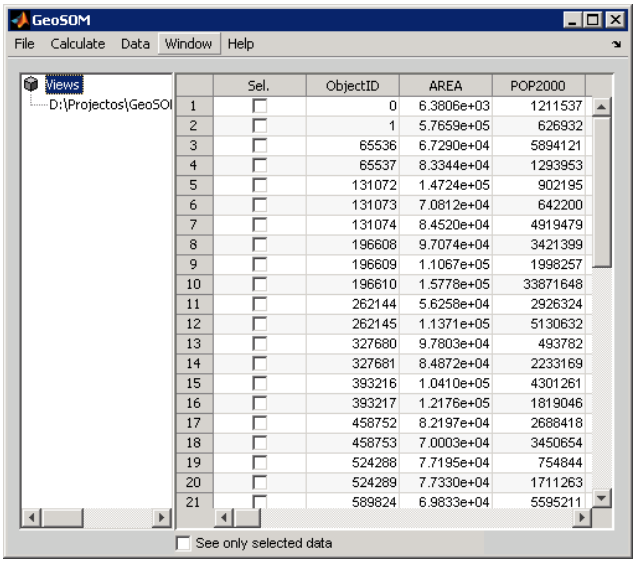


Fig. 10 – GeoSOM Suite main window

By default the geographic window is opened, but in case the user closes this window, by selecting the respective item in the tree view and clicking with the right mouse button a menu *Show Geo* appears allowing the user to reopen this view.

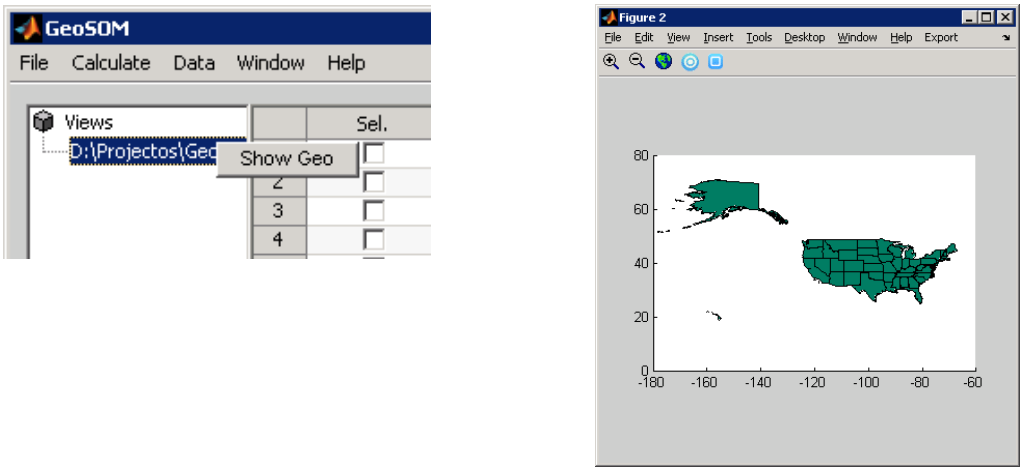


Fig. 11 – Show geographic information in GeoSOM Suite

4. The next step is to select one record by clicking in the respective checkbox present in the first column (Sel.). If the geographic window is open, then the selection is also showed in the map, by coloring the correspondent state.

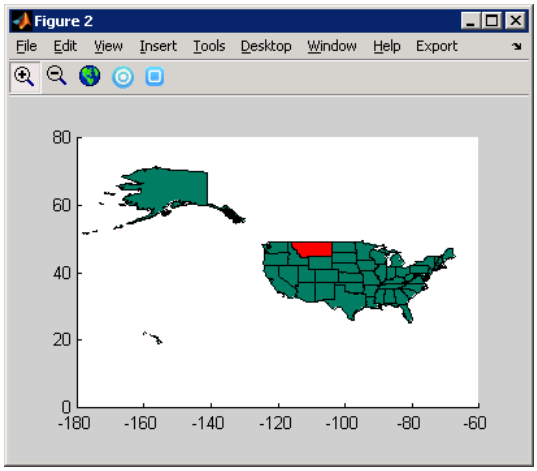


Fig. 12 – Geographic selection in GeoSOM Suite

5. Lets us now create a SOM. Select Calculate>Train SOM. The following form will appear:

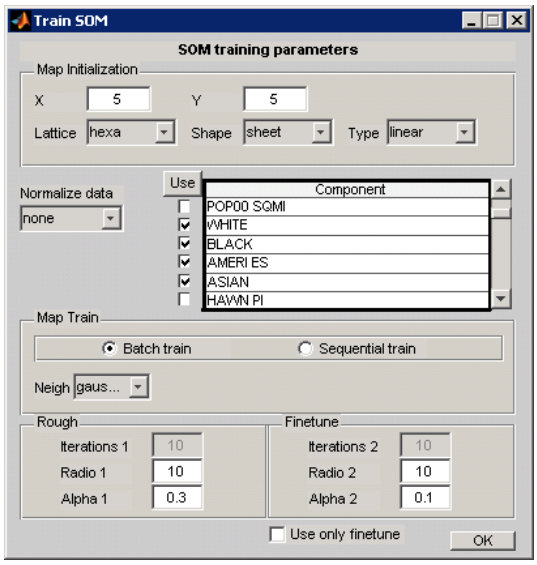


Fig. 13 – Training parameters of SOM

6. Click OK. A new SOM is calculated using the parameters defined in the previous figure.
7. Also the Views tree is updated with this new SOM.

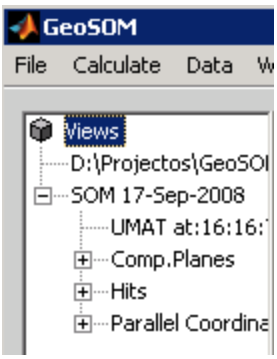


Fig. 14 – Views tree

8. Inside the SOM item several representations are built. Please select UMAT, right mouse button and Show UMAT. Do the same for Component planes.

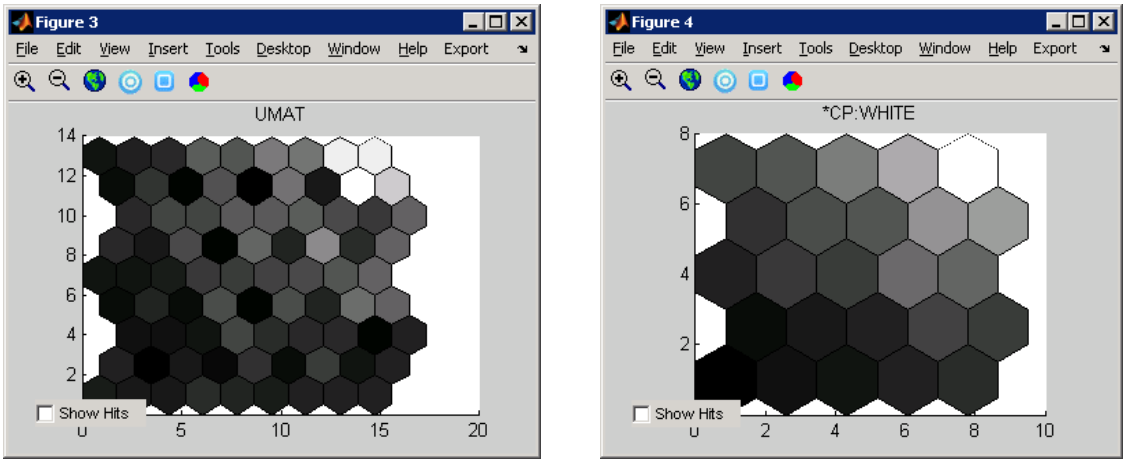
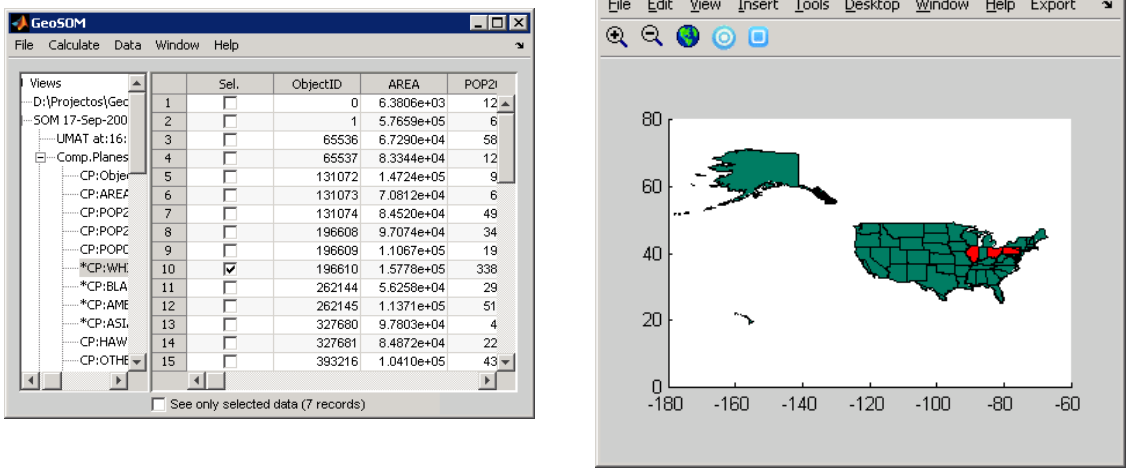


Fig. 15 – U-matrix and component plane

9. Using the Select Area button, select some feature in any of the views and check the selection in all the other views.



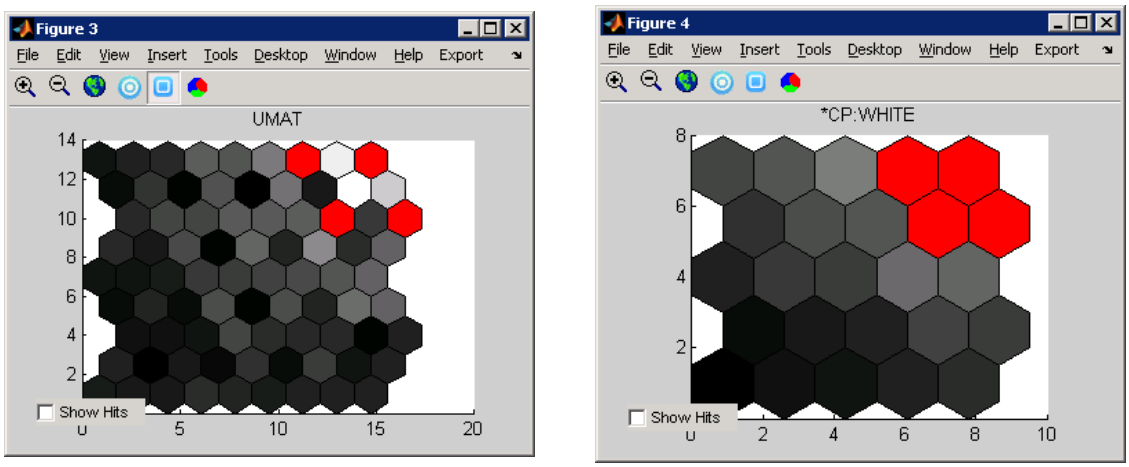


Fig. 16 – Selection in several views of GeoSOM suite

A2.4 GeoSOM Suite Installation

GeoSOM Suite is a Free academic software available at www.isegi.unl.pt/labnt/geosom. On the downloads section, two installers are available:

Depending if you already have the Matlab Compiler Runtime (version 7.8) installer in your machine, you can choose the version with/without this component.

After downloading, run the *exe* file you just download. Because this is a compressed file and you cannot set the extraction folder, you should copy the downloaded *exe* file to the desired GeoSOM installation folder (e.g., c:\programs\geosom or other)

If you are installing the version with MCR, a popup will appear for setting some MCR installation parameters.

After installation is done click on GeoSOM_v01.exe.

Appendix 3. GeoSOM Suite code

Functions index

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function GeoSOM()
    %GeoSOM
    %   Description: GEOSOM Main Form
    %   Inputs:
    %   Outputs:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: Dec 22, 2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    %%% Current Version
    Vs='0.2';
    Created='January 2010';

    fh=figure('Position',[200 200 500 400]);
    % clear default menu and toolbars and add title
```

```
set(fh,'MenuBar','none','toolbar','none','NumberTitle','off','name',...
    ['GeoSOM suite ' Vs ]);
handles.geosom=fh;

% Menus
%   file
mh=uimenu('Label','File');
h=uimenu(mh,'Label','Load shapefile');
handles.Menu_Load_shapefile=h;
h=uimenu(mh,'Label','Load csv/mat file');
handles.Menu_Load_csvmat=h;
h=uimenu(mh,'Label','Connect to database','enable','off');
handles.Menu_ConnectDB=h;
h=uimenu(mh,'Label','Load GeoSom struct','separator','on');
handles.Menu_LoadGeosom=h;
h=uimenu(mh,'Label','Save GeoSom struct');
handles.Menu_SaveGeosom=h;
h=uimenu(mh,'Label','Close GeoSOM','separator','on');
handles.Menu_Close=h;

%   calculate
mh=uimenu('Label','Calculate');
h=uimenu(mh,'Label','Train SOM','enable','off');
handles.Menu_Train_SOM=h;
h=uimenu(mh,'Label','Train GeoSOM','separator','on','enable','off');
handles.Menu_Train_GeoSOM=h;
h=uimenu(mh,'Label','Train Hierarch.SOM','separator','on','enable','off');
handles.Menu_Train_HSOM=h;

%   data
mh=uimenu('Label','Data');
h=uimenu(mh,'Label','Query','enable','off');
handles.Menu_Query=h;
h=uimenu(mh,'Label','Export','enable','off');
handles.Menu_ExportData=h;
h=uimenu(mh,'Label','Create box histo','enable','off');
handles.Menu_CreateBoxPlotsExportData=h;

%   selection
mh=uimenu('Label','Selection');
h=uimenu(mh,'Label','Clear','enable','off');
handles.Menu_SelectionClear=h;
h=uimenu(mh,'Label','Invert','enable','off');
handles.Menu_SelectionInvert=h;

%   window
mh=uimenu('Label','Window');
h=uimenu(mh,'Label','Dock windows','enable','off');
handles.Menu_Dock=h;
h=uimenu(mh,'Label','Close all views');
handles.Menu_CloseViews=h;
h=uimenu(mh,'Label','Preferences');
handles.Menu_Preferences=h;

%   help
mh=uimenu('Label','Help');
h=uimenu(mh,'Label','Help');
handles.Menu_Help=h;
h=uimenu(mh,'Label','About');
handles.Menu_About=h;

%   tables
h = uitable('tag', 'uitable1',...
       'Parent',fh,...
       'Position', [10 380 50 15]);
handles.uitable1=h;

%   treeview
root = uitreenode('Views', 'Views', [], false);
```

```
    tree = uitree( fh,'Root', root,'ExpandFcn', @uitree1_ExpandFc);
    handles.uitree1=tree;

    %   checkbox
    h = uicontrol('Style', 'checkbox',...
           'String', 'See only selected data',...
           'tag', 'chkSel',...
           'Position', [10 380 50 15]);
    handles.chkSel=h;

    %   draggable label
    h = uicontrol('Style', 'text',...
           'String', '',...
           'tag', 'textDragg',...
           'Position', [10 10 10 10],...
           'Enable','off',...
           'BackgroundColor',[.5 .5 .5]);
    handles.textDragg=h;


    %%% CALLBACKS with handles %%%
    %menu file
    set(handles.Menu_Load_shapefile,'Callback',{@menuLoadShapefile_Callback handles});
    set(handles.Menu_Load_csvmat,'Callback',{@menu_Load_csvmat_Callback handles});
    set(handles.Menu_LoadGeosom,'Callback',{@Menu_Load_Geosom_Callback handles});
    set(handles.Menu_SaveGeosom,'Callback',{@Menu_Save_Geosom_Callback handles});
    set(handles.Menu_Close,'Callback',{@Menu_Close_Callback handles});
    %menu train
    set(handles.Menu_Train_SOM,'Callback',{@menuTrainSOM_Callback handles});
    set(handles.Menu_Train_GeoSOM,'Callback',{@menuTrainGeoSOM_Callback handles});
    set(handles.Menu_Train_HSOM,'Callback',{@menuTrainHSOM_Callback handles});
    %menu data
    set(handles.Menu_Query,'Callback',{@menu_Query_Callback handles});
    set(handles.Menu_ExportData,'Callback',{@menu_Export_Callback handles});
    set(handles.Menu_CreateBoxPlotsExportData,'Callback',{@Menu_CreateBoxPlots_Callback
handles});
    %menu selection
    set(handles.Menu_SelectionClear,'Callback',{@Menu_SelectionClear_Callback handles});
    set(handles.Menu_SelectionInvert,'Callback',{@Menu_SelectionInvert_Callback
handles});
    %menu window
    set(handles.Menu_CloseViews,'Callback',{@menuCloseViews_Callback handles});
    set(handles.Menu_Preferences,'Callback',{@menuPreferences_Callback handles});
    %menu Help
    set(handles.Menu_Help,'Callback',{@menuHelp_Callback handles});
    set(handles.Menu_About,'Callback',{@menuAbout_Callback Vs Created});

    %other callbacks
    set(handles.geosom,'CloseRequestFcn',{@geosom_Close});
    set(handles.geosom,'ResizeFcn',{@geosom_ResizeFcn handles});
    set(handles.Menu_Dock,'Callback',{@menuDock_Callback handles});
    set(handles.chkSel,'Callback',{@chkSel_Callback handles});
    set(handles.uitable1,'CellEditCallback',{@uitable1_CellEditCallback handles});
    set(handles.uitree1, 'NodeSelectedCallback', @uitree1_nodeSelected);
    set(handles.uitree1.Tree, 'MousePressedCallback', {@uitree1_MousePress handles});
    set(handles.textDragg,'ButtonDownFcn',{@dragg handles});

    % create a global varible with the main figure handles
    global mainHandles
    mainHandles=handles;

    %%% load prefs (prefs.mat file located in the geosom root)
    geosom_loadPrefs();
return

function geosom_ResizeFcn(h,evd,handles)
    % resize function

    if isempty(handles)==1
```

```
        return
    end

    % new figure position -> x y width height
    Position = getpixelposition(handles.geosom);

    iborder=10;
    % textDragg
    setpixelposition(handles.textDragg,[Position(3)*1/4+iborder-1 iborder*2 5
Position(4)-30]);
    % uitree
    setpixelposition(handles.uitree1,[iborder iborder*2 Position(3)*1/4 Position(4)-30]);
    % uitable
    setpixelposition(handles.uitable1,[Position(3)*1/4+iborder iborder*2 Position(3)*3/4-
2*iborder Position(4)-30]);
    % chksel
    setpixelposition(handles.chkSel,[Position(3)*1/4+iborder 0 250 20]);
return

function menuLoadShapefile_Callback(h,evd,handles)
    % load shapefile function
    global gss
    global mainHandles

    %get the file
    [filename, pathname, filterindex] = uigetfile('*.shp', 'Pick a
shapefile',mainHandles.prefs.DataFolder);
    if filename == 0
        return
    end

    try
        set(handles.geosom,'Pointer','watch');
        gss = geosom_ReadShapeFile(strcat(pathname, filename));
        set(handles.geosom,'Pointer','arrow');
        if pathname~=0
            mainHandles.prefs.DataFolder=pathname;
        end
    catch
        set(handles.geosom,'Pointer','arrow');
        msgbox('Cannot read this file! Make sure you selected an shp file.'...
            ,'Problem rading file','warn');
        return
    end

    %%visualization of the data map is generated once the data is fully loaded
    %%automatically and the possibility of create another one is only available
    %%once some training is done
    geosom_ShowView(1,gray);

    %enable the train functions
    enableTrainMenu (handles)

    % refresh the treeview
    RefreshUitree (handles)

    % load uitable --> this has to be the last thing happening
    LoadUitable(handles);
return;

function LoadUitable(handles)
    % loads uitable with shapefile atribute table
    global gss

    %numrows=size(gss.NumData,1);
    cols_num = size(gss.NumData,2);
    cols_cell = size(gss.CellData,2);

    %clear
```

```
    data=[];
    set(handles.uitable1,'Data',data);

    %add data
    %if no cell data exist then use matrix instead of cells
    set(handles.uitable1,'Data',[num2cell(gss.SelectedIndex) num2cell(gss.NumData)
gss.CellData]);
    set(handles.uitable1,'ColumnName',['Sel.' gss.NumDataLabel gss.CellDataLabel]);
    columneditable =[true(1,1) false(1,cols_num+cols_cell)];
    set(handles.uitable1,'columneditable',columneditable);

    % write in the check selected data label the number of selected features
    if sum(gss.SelectedIndex)>0
        set (handles.chkSel,'string',strcat('See only selected data (' ...
            ,num2str(sum(gss.SelectedIndex)), ' records)'));
    else
        set (handles.chkSel,'string','See only selected data')
    end
return

function enableTrainMenu(handles)
    % this function enalbes the menus with the SOM and GeoSOM training
    % functions
    set (handles.Menu_Train_SOM,'enable','on');
    set (handles.Menu_Train_GeoSOM,'enable','on');
    set (handles.Menu_Train_HSOM,'enable','on');
    set (handles.Menu_Query,'enable','on');
    set (handles.Menu_ExportData,'enable','on');
    set (handles.Menu_CreateBoxPlotsExportData,'enable','on');
    set (handles.Menu_SelectionClear,'enable','on');
    set (handles.Menu_SelectionInvert,'enable','on');
return

function chkSel_Callback(hObject, eventdata, handles)
    % hObject    handle to chkSel (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hint: get(hObject,'Value') returns toggle state of chkSel
    global gss

    numrows=size(gss.NumData,1);
    cols_num = size(gss.NumData,2);
    cols_cell = size(gss.CellData,2);

    data=[num2cell(gss.SelectedIndex) num2cell(gss.NumData) gss.CellData];
    if get(hObject,'Value')==1 % selected
        set(handles.uitable1,'Data',data(find(gss.SelectedIndex==1),:));
        columneditable =false(1,cols_num+cols_cell+1);
    else % not selected
        set(handles.uitable1,'Data',data);
        columneditable =[true(1,1) false(1,cols_num+cols_cell)];
    end
    set(handles.uitable1,'columneditable',columneditable);
return

function uitable1_CellEditCallback(hObject, eventdata, handles)
    % hObject    handle to uitable1 (see GCBO)
    % eventdata  structure with the following fields (see UITABLE)
    %   Indices: row and column indices of the cell(s) edited
    %   PreviousData: previous data for the cell(s) edited
    %   EditData: string(s) entered by the user
    %   NewData: EditData or its converted form set on the Data property. Empty if Data
was not changed
    %   Error: error string when failed to convert EditData to appropriate value for Data
    % handles    structure with handles and user data (see GUIDATA)

    global gss
```

```
    %make selection
    gss.SelectedIndex(eventdata.Indices(:,1))=eventdata.EditData;

    % present the selected features in all the views
    geosom_MakeSelectionAllOpenViews();

    % load uitable --> this has to be the last thing happening
    LoadUitable(handles);
return

function menuTrainHSOM_Callback(hObject, eventdata, handles)
    % hObject    handle to menuTrainSOM (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global gss
    if isfield(gss,'Som')
        if length(gss.Som)<2
            msgbox('To use Hierarchical SOM you need 2 or more trainned SOMs'...
                ,'Cannot use Hierarchical SOM','warn')
            return
        end
    else
        msgbox('To use Hierarchical SOM you need 2 or more trainned SOMs'...
            ,'Cannot use Hierarchical SOM','warn')
        return
    end
    geosom_HSomTrainingParameters();

    % refresh the treeview
    RefreshUitree(handles)
return

function menuTrainGeoSOM_Callback(hObject, eventdata, handles)
    % hObject    handle to menuTrainSOM (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    geosom_GeoSomTrainingParameters();
    % refresh the treeview
    RefreshUitree(handles)
return

function menuTrainSOM_Callback(hObject, eventdata, handles)
    % hObject    handle to menuTrainSOM (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    geosom_SomTrainingParameters();
    % refresh the treeview
    RefreshUitree(handles)
return

function menu_Query_Callback(hObject, eventdata, handles)

    %function to present a new window for querying data
    geosom_querydata();

    %present the selection in the views
    geosom_MakeSelectionAllOpenViews;

    %present the selection in the table
    geosom_TableDraw();
return

function menu_Export_Callback(hObject, eventdata, handles)

    %function to present a new window for exporting data
    geosom_ExportData();
```

```
    %present the selection in the views
    geosom_MakeSelectionAllOpenViews;

    %present the selection in the table
    geosom_TableDraw();
return

function Menu_CreateBoxPlots_Callback (hObject, eventdata, handles)
    % show geosom box plots
    geosom_CreateBoxPlotsHisto();
return

function popupViews_Callback(hObject, eventdata, handles)
    global gss

    popupStr = get(handles.popupViews, 'String');
    popupVal = get(handles.popupViews, 'Value');
    gss = geosom_ShowView (gss,popupVal,gray);
return

function menuAbout_Callback(hObject, eventdata, Vs,Created)
    % about geosom
    geosom_About (Vs,Created)
return

function menuHelp_Callback(hObject, eventdata, handles)
    % opens geosom help
    sPath=[cd '\help\help.html'];
    open (sPath)
return

function dragg(hObject, eventdata, handles)
    % dragg
    point1 = get(gcf,'CurrentPoint'); % button down detected
    rect = [point1(1,1) point1(1,2) 5 5];
    [r2] = dragrect(rect);

    %resize form
    Position = getpixelposition(handles.geosom);
    iborder=10;

    % textDragg
    setpixelposition(handles.textDragg,[r2(1)+iborder-1 iborder*2 5 Position(4)-30]);
    % uitree
    setpixelposition(handles.uitree1,[iborder iborder*2 r2(1) Position(4)-30]);
    % uitable
    setpixelposition(handles.uitable1,[r2(1)+iborder+1 iborder*2 Position(3)-2*iborder-
r2(1) Position(4)-30]);
    % chksel
    setpixelposition(handles.chkSel,[Position(3)*1/4+iborder 0 250 20]);
return

function menuPreferences_Callback(hObject, eventdata, handles)
    % open geosom prefs
    geosom_Preferences();
return

function menuCloseViews_Callback(hObject, eventdata, handles)
    % close all open views
    global gss

    button = questdlg('Are you sure you want to close all open views?'...
        ,'Close All Views','YES','NO','NO');
    if strcmp(button,'YES')
        for i=1:length(gss.ViewLabel)
            % check if this view is displayed.
            % if yes close it
            if isfield(gss.ViewLabel(i), 'WindowHandle')
                if ishandle(gss.ViewLabel(i).WindowHandle)
```

```
                    close(gss.ViewLabel(i).WindowHandle);
                end
            end
        end
    end
return

function menuDock_Callback(hObject, eventdata, handles)
    global gss

    numViews=[];
    % get the visible views
    for v=1:length(gss.ViewLabel)
        if isfield(gss.ViewLabel(v), 'WindowHandle')
            if ishandle(gss.ViewLabel(v).WindowHandle)
                numViews=[numViews;v];
            end
        end
    end
    nCols=ceil(sqrt(length(numViews)+1));
    nRows=ceil((length(numViews)+1)/nCols);

    group = setfigdocked('GroupName','GeoSOM','GridSize',[nRows nCols]);

    % add geosom main fig
    group = setfigdocked('GroupName','GeoSOM','Figure',gcf,'Figindex',1);

    % add the views
    for i=1:length(numViews)
        group =
setfigdocked('GroupName','GeoSOM','Figure',gss.ViewLabel(numViews(i)).WindowHandle,'Figin
dex',i+1);
    end
return

function Menu_Load_Geosom_Callback(hObject, eventdata, handles)
    % loads a new geosom struct

    global gss
    global mainHandles

    if isempty(gss)~=1
        button = questdlg('A GeoSomStruct is already loaded. Do you want to replace it by
this new one?'...
            ,'Load GeoSomStruct','YES','NO','NO');
        if strcmp(button,'YES')
            try
                set(handles.geosom,'Pointer','watch');
                %uiopen
                [filename, pathname, filterindex] = uigetfile('*.mat', ...
                'Pick a GeoSOM struct file',mainHandles.prefs.ProjFolder);
                load(strcat(pathname,'\',filename));


                enableTrainMenu(handles);
                LoadUitable(handles)
                set(handles.geosom,'Pointer','arrow');
                RefreshUitree(handles);
                if pathname~=0
                    mainHandles.prefs.ProjFolder=pathname;
                end
            catch
                set(handles.geosom,'Pointer','arrow');
            end
        else
            return
        end
    else
        try
```

```
            set(handles.geosom,'Pointer','watch');
            uiopen
            enableTrainMenu(handles);
            LoadUitable(handles)
            set(handles.geosom,'Pointer','arrow');
            RefreshUitree(handles);
        catch
            set(handles.geosom,'Pointer','arrow');
        end
    end
return

function geosom_Close (hObject, eventdata)
    global mainHandles
    %save Prefs with data location and proj location
    prefs=mainHandles.prefs;
    save 'prefs.mat' prefs

    %close
    delete (gcf);
return

function Menu_Close_Callback (hObject, eventdata, handles)
    % close geosom
    button = questdlg('Are you sure you want to exit?'...
        ,'Exit GeoSom','YES','NO','NO');
    if strcmp(button,'YES')
        exit
    end
return

function Menu_Save_Geosom_Callback (hObject, eventdata, handles)
    % save the geosomtruct
    global mainHandles
    try
        set(handles.geosom,'Pointer','watch');
        %uisave gss
        [filename,pathname,filterindex] = uiputfile('*.mat','Save GeoSOM
struct',strcat(mainHandles.prefs.ProjFolder,'\gss.mat'));
        if pathname~=0
            mainHandles.prefs.DataFolder=pathname;
        end
        set(handles.geosom,'Pointer','arrow');
    catch
        set(handles.geosom,'Pointer','arrow');
    end
return

function menu_Load_csvmat_Callback (hObject, eventdata, handles)
    global gss
    global mainHandles
    %get the file
    [filename, pathname, filterindex] = uigetfile({'*.csv';'*.mat'}, ...
        'Pick a csv or mat file with the input data',mainHandles.prefs.DataFolder);
    if filename == 0
        return
    end

    [gss,ok] = geosom_ReadCsvMatFile(pathname, filename);

    if ok % ok is true if everythink is ok in geosom_ReadCsvMatFile
        % % uitable
        numrows=size(gss.NumData,1);
        cols_num = size(gss.NumData,2);
        cols_cell = size(gss.CellData,2);

        set(handles.uitable1,'Data',[num2cell(gss.SelectedIndex) num2cell(gss.NumData)
gss.CellData]);
```

```
        set(handles.uitable1,'ColumnName',['Selected' gss.NumDataLabel
gss.CellDataLabel]);
        columneditable =[true(1,1) false(1,cols_num+cols_cell)];
        set(handles.uitable1,'columneditable',columneditable);
        if pathname~=0
            mainHandles.prefs.DataFolder=pathname;
        end
    end

    %enable the train functions
    enableTrainMenu (handles)
return

function Menu_SelectionClear_Callback(hObject, eventdata, handles)
    global gss

    gss.SelectedIndex(:)=0;

    %refresh table
    LoadUitable(handles);

    % present the selected features in all the views
    geosom_MakeSelectionAllOpenViews();
return

function Menu_SelectionInvert_Callback(hObject, eventdata, handles)
    global gss

    Sel=gss.SelectedIndex==1;
    NonSel=gss.SelectedIndex==0;

    gss.SelectedIndex(Sel)=0;
    gss.SelectedIndex(NonSel)=1;

    %refresh table
    LoadUitable(handles);

    % present the selected features in all the views
    geosom_MakeSelectionAllOpenViews();
return

function nodes = uitree1_ExpandFc(tree,value)
    global gss
    if strcmpi(value,'Views'); % root childs (geo,som)
        ExistsSpatialData = 0;
        % get possible geographic map
        if isfield(gss,'ViewLabel')
            for i=1:length(gss.ViewLabel)
                if strcmpi(gss.ViewLabel(i).Type,'geographicmap')
                    nodes(i) = uitreenode({'view',i,'geo'}, ...
                      gss.ViewLabel(i).Name,'',1);
                    ExistsSpatialData=1;
                end
            end

        end
        if isfield(gss,'Som')
            for i=1:length(gss.Som)
                if ExistsSpatialData % view starts in 2 but Somindex starts in 1
                    j=i+1;
                else
                    j=i;
                end
                nodes(j) = uitreenode({'som',i},...
                    gss.Som(i).name,'',0);
            end
        end
    elseif strcmpi(value(1),'som') % som or geosom childs
        %get the som_id
```

```
    somInd=value(2);
    k=0;
    cp=0;
    hits=0;
    if isfield(gss,'ViewLabel')
        for i=1:length(gss.ViewLabel)
            if gss.ViewLabel(i).SomOrigin==somInd
                if strcmpi(gss.ViewLabel(i).Type,'umat')
                    k = k+1;
                    nodes(k) = uitreenode({'view',i,'umat'}, ...
                      gss.ViewLabel(i).Name,'',1);
                elseif strcmpi(gss.ViewLabel(i).Type,'cp')
                    cp=1;
                elseif strcmpi(gss.ViewLabel(i).Type,'hits')
                    hits=1;
                end
            end
        end
        if cp % add a cp header node
            k = k+1;
            nodes(k) = uitreenode({'cp',somInd}, ...
            'Comp.Planes','',0);
        end
        if hits % add a hits header node
            k = k+1;
            nodes(k) = uitreenode({'hits',somInd}, ...
            'Hits','',0);
        end
        % add a pcp header node
        k = k+1;
        nodes(k) = uitreenode({'pcps',somInd}, ...
        'Parallel Coordinate Plots','',0);

    end
elseif strcmpi(value(1),'cp') % cp childs
    somInd=value(2);
    k=0;
    if isfield(gss,'ViewLabel')
        for i=1:length(gss.ViewLabel)
            if gss.ViewLabel(i).SomOrigin==somInd
                if strcmpi(gss.ViewLabel(i).Type,'cp')
                    k = k+1;
                    nodes(k) = uitreenode({'view',i,'cp'}, ...
                      gss.ViewLabel(i).Name,'',1);
                end
            end
        end
    end
elseif strcmpi(value(1),'hits') % hits childs
    somInd=value(2);
    k=0;
    if isfield(gss,'ViewLabel')
        for i=1:length(gss.ViewLabel)
            if gss.ViewLabel(i).SomOrigin==somInd
                if strcmpi(gss.ViewLabel(i).Type,'hits')
                    k = k+1;
                    nodes(k) = uitreenode({'view',i,'hits'}, ...
                      gss.ViewLabel(i).Name,'',1);
                end
            end
        end
    end
elseif strcmpi(value(1),'pcps') % pcp childs
    somInd=value(2);
    k=0;
    if isfield(gss,'ViewLabel')
        for i=1:length(gss.ViewLabel)
            if gss.ViewLabel(i).SomOrigin==somInd
                if strcmpi(gss.ViewLabel(i).Type,'pcp')
```

```
                        k = k+1;
                        nodes(k) = uitreenode({'view',i,'pcp'}, ...
                          gss.ViewLabel(i).Name,'',1);
                    end
                end
            end
        end

    end

    if ~exist('nodes','var')
        nodes=[];
    end
return

function uitree1_nodeSelected(tree,ev)
    cNode = ev.getCurrentNode;
    tmp = tree.FigureComponent;
    set(tmp, 'UserData', cNode);
return

function uitree1_MousePress(tree,value,handles)
    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tmp = handles.uitree1.FigureComponent;
    S = get(tmp, 'UserData');
    if ~isempty(S)
        if strcmpi(S.getName,'views')
            return
        end
        if strcmpi(S.getName,'Hits')
            return
        end
    end

    if value.getButton==3
        import javax.swing.*;
        pUpMenu = JPopupMenu;

        tag=S.getValue;
        if length(tag)==3 % end node (geo, umat, cp, pcp)
            if strcmpi(tag(3),'geo')
                cMenu1 = JMenuItem('Show Geo');
                set(cMenu1, 'ActionPerformedCallback', {@cntxtmenuTree_bt1_callback S});
                pUpMenu.add(cMenu1);
            elseif strcmpi(tag(3),'umat')
                cMenu2 = JMenuItem('Show UMAT');
                set(cMenu2, 'ActionPerformedCallback', {@cntxtmenuTree_bt1_callback S});
                pUpMenu.add(cMenu2);

                %Add separator
                pUpMenu.add(JSeparator());

                cMenu2a = JMenuItem('Create Clusters on UMAT');
                set(cMenu2a, 'ActionPerformedCallback', {@cntxtmenuTree_bt6_callback S});
                pUpMenu.add(cMenu2a);
            elseif strcmpi(tag(3),'cp')
                cMenu3 = JMenuItem('Show CP');
                set(cMenu3, 'ActionPerformedCallback', {@cntxtmenuTree_bt1_callback S});
                pUpMenu.add(cMenu3);
            elseif strcmpi(tag(3),'hits')
                cMenu4 = JMenuItem('Show Hits');
                set(cMenu4, 'ActionPerformedCallback', {@cntxtmenuTree_bt1_callback S});
                pUpMenu.add(cMenu4);
            elseif strcmpi(tag(3),'pcp')
                cMenu5 = JMenuItem('Show PCP');
```

```
                set(cMenu5, 'ActionPerformedCallback', {@cntxtmenuTree_bt1_callback S});
                pUpMenu.add(cMenu5);
            end
        else % parent node (SOM, Comp.Planes)
            if strcmpi(tag(1),'som')
                cMenu6 = JMenuItem('Properties');
                set(cMenu6, 'ActionPerformedCallback', {@cntxtmenuTree_bt2_callback S});
                pUpMenu.add(cMenu6);
                cMenu6a = JMenuItem('Select data used');
                set(cMenu6a, 'ActionPerformedCallback', {@cntxtmenuTree_bt8_callback S
handles});
                pUpMenu.add(cMenu6a);
                cMenu7 = JMenuItem('Rename');
                set(cMenu7, 'ActionPerformedCallback', {@cntxtmenuTree_bt5_callback S
handles});
                pUpMenu.add(cMenu7);

                %Add separator
                pUpMenu.add(JSeparator());

                cMenu7a = JMenuItem('Delete (geo)SOM');
                set(cMenu7a, 'ActionPerformedCallback', {@cntxtmenuTree_bt7_callback S
handles});
                pUpMenu.add(cMenu7a);

            elseif strcmpi(tag(1),'cp')
                cMenu8 = JMenuItem('Show CPs used in SOM');
                set(cMenu8, 'ActionPerformedCallback', {@cntxtmenuTree_bt3_callback S
1});
                pUpMenu.add(cMenu8);

                cMenu9 = JMenuItem('Show all CP');
                set(cMenu9, 'ActionPerformedCallback', {@cntxtmenuTree_bt3_callback S
0});
                pUpMenu.add(cMenu9);

            elseif strcmpi(tag(1),'pcps')
                cMenu10 = JMenuItem('New PCP');
                set(cMenu10, 'ActionPerformedCallback', {@cntxtmenuTree_bt4_callback S
handles});
                pUpMenu.add(cMenu10);
            end
        end

        if pUpMenu.getComponentCount>0
            pUpMenu.show(handles.uitree1.Tree, value.getX, value.getY);
            pUpMenu.repaint;
        end
    end
return

function cntxtmenuTree_bt1_callback(h,evd,selNode)
    % this button will show the views (geo, umat, cp)
    % clear context

    global mainHandles

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tag=selNode.getValue;
    if strcmpi(tag(1),'view')
        viewId=tag(2);
        % show view
        geosom_ShowView(viewId,mainHandles.prefs.cmap)
    end
return
```

```
function cntxtmenuTree_bt2_callback(h,evd,selNode)
    % this button will show the som properties
    % clear context

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tag=selNode.getValue;
    if strcmpi(tag(1),'som')
        somId=tag(2);
        geosom_showProperties(somId);
    end
return

function cntxtmenuTree_bt3_callback(h,evd,selNode,UsedOnSOM)
    % this button will open all the Cp in one figure
    % clear context
    global mainHandles
    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tag=selNode.getValue;
    geosom_ShowViewsOneFigure (tag(1),tag(2),mainHandles.prefs.cmap,UsedOnSOM)
return

function cntxtmenuTree_bt4_callback(h,evd,selNode,handles)
    % this button creates a new pcp
    % clear context
    global gss

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tag=selNode.getValue;
    SelNumData=geosom_NewParallelCoordinatePlot';
    if isempty(SelNumData)~=1 % if any variable was selected
        % check if this pcp already exists for this SOM
        for i=1:length(gss.ViewLabel)
            if strcmpi(gss.ViewLabel(i).SomOrigin,tag(2))
                if isfield(gss.ViewLabel(i),'SelVariables')
                    if isequal(gss.ViewLabel(i).SelVariables,SelNumData)
                        msgbox ('The PCP for the selected variables already exists!'...
                            ,'New PCP','help');
                        return;
                    end
                end
            end
        end
        geosom_BuildCoordinatePlots(SelNumData,tag(2))
        RefreshUitree(handles);
    end
return

function cntxtmenuTree_bt5_callback(h,evd,selNode,handles)
    % this button renames the SOM
    global gss

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end
```

```
    newName = inputdlg('Please insert the new name','Rename');
    if isempty(newName)
        return
    end
    if isvarname(newName{1})==0
        msgbox('The new name is not valid! Please select a new valid name.');
        return
    end

    tag=selNode.getValue;
    if strcmpi(tag(1),'som')
        somId=tag(2);

        if strcmpi(gss.Som(somId).method,'geosom')
            gss.Som(somId).name=char(strcat('geosom_',newName));
        elseif strcmpi(gss.Som(somId).method,'som')
            gss.Som(somId).name=char(strcat('som_',newName));
        end
        RefreshUitree(handles);
    end
return

function cntxtmenuTree_bt6_callback(h,evd,selNode)
    % Create clusters on top of umat

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tag=selNode.getValue;
    if strcmpi(tag(1),'view')
        viewId=tag(2);
        % show view
        geosom_drawCluster(viewId)
    end
return

function cntxtmenuTree_bt7_callback(h,evd,selNode,handles)
    % this button deletes the SOM

    global gss

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    button = questdlg({'Are you shure you want to delete this (geo)SOM.',
        'This cannot be undone!'},'Delete','Ok','Cancel','Cancel');
    if strcmpi(button,'Ok')==0
        return
    end

    %delete SOM includes deleting from:
    %gss.View
    %gss.ViewLabel
    %gss.ViewIndexData
    %gss.Som

    tag=selNode.getValue;
    if strcmpi(tag(1),'som')
        somId=tag(2);
        Ids=[];
        for i=1:length(gss.ViewLabel)
            if gss.ViewLabel(i).SomOrigin==somId
                Ids=[Ids; i];
            end
        end
```

```
        %deleting...
        gss.Som(somId)=[];
        gss.View(Ids)=[];
        gss.ViewLabel(Ids)=[];
        gss.ViewIndexData(:,Ids)=[];
        %refreshing...
        RefreshUitree(handles);
    end
return

function cntxtmenuTree_bt8_callback(h,evd,selNode,handles)
    % this button selects the data used in the SOM or GeoSOM
    global gss

    h = findobj(gcf,'Tag','ContextMenuTree');
    if ~isempty(h);
        delete (h);
    end

    tag=selNode.getValue;
    if strcmpi(tag(1),'som')
        somId=tag(2);
        gss.SelectedIndex(:)=0;
        gss.SelectedIndex(gss.Som(somId).topol.DataUsed)=1;
        gss.SelectedIndex=logical(gss.SelectedIndex);
    end

    %refresh table
    LoadUitable(handles);
    % present the selected features in all the views
    geosom_MakeSelectionAllOpenViews();
return

function tree=RefreshUitree(handles)
    root = uitreenode('Views', 'Views', [], false);
    handles.uitree1.setRoot(root)
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_About(Vs,Created)
    %GEOSOM_ABOUT ...
    % Syntax: geosom_About(Vs,Created)
    % Vs - Version
    % Created - Creation date
    %
    % Subfunctions: openWebsite
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 22-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    scrsz = get(0,'ScreenSize');
    fh=figure('Position',[scrsz(3)/2-100 scrsz(4)/2-50 200 100]);

    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
        'Resize','off','name','About GeoSom suite');
    handles.About=fh;

    % % add controls
    % text - title
    h = uicontrol('Style', 'text',...
           'String', ['GeoSOM vs ', Vs],...
           'tag', 'textViews',...
           'Position', [10 70 100 15],...
           'HorizontalAlignment','Left',...
           'fontweight','bold');
```

```
    h = uicontrol('Style', 'push',...
            'String','www.isegi.unl.pt/labnt/geosom',...
            'tag', 'textViews',...
            'Position', [10 55 160 15],...
            'HorizontalAlignment','Left',...
            'ForegroundColor',[0 0 1],...
            'callback',{@openWebsite});

    h = uicontrol('Style', 'text',...
           'String', 'Ga2 Team',...
           'tag', 'textViews',...
           'Position', [10 30 100 15],...
           'HorizontalAlignment','Left');


    h = uicontrol('Style', 'text',...
           'String', Created,...
           'tag', 'textViews',...
           'Position', [10 10 100 15],...
           'HorizontalAlignment','Left');

    h = uicontrol('Style', 'pushbutton',...
           'String', 'Ok',...
           'tag', 'pushBt',...
           'Position', [150 20 40 20],...
           'callback','close');
return

function openWebsite(hObject, eventdata)
    % opens geosom webpage
    web ('http://www.isegi.unl.pt/labnt/geosom/');
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildComponentPlanes(somIndex)
    %geosom_BuildComponentPlanes
    % function for building the CPs
    %
    % Syntax: geosom_BuildComponentPlanes()
    % input - somIndex
    % output - none
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 09-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    M=gss.Som(somIndex);

    %geosom_struct.NumData
    mapSize=(M.topol.msize);
    maxx = mapSize(2);
    maxy =mapSize(1);
    tmp=cell(maxx*maxy,1);
    DataUsed = gss.Som(somIndex).topol.DataUsed;

    %check data normalization
    if strcmpi(M.topol.Normalization,'none')
        bmus=som_bmus(gss.Som(somIndex),gss.NumData(DataUsed,:));
    else
        sD=som_data_struct(gss.NumData(DataUsed,:));
        sD=som_normalize(sD,M.topol.Normalization);
```

```
    bmus=som_bmus(gss.Som(somIndex),sD);
end

% check for existance of other Views
if ~isfield(gss,'View') % if does not exist create it
    gss.View =[];
    numView=0;
else
    numView = length(gss.View);
end

% create polygon structure
for i=1:maxx*maxy;tmp{i}='Polygon';end;

if strcmpi( M.topol.lattice,'rect') % rect lattice
    for v=1:length(M.comp_names)  % for each variable

        gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                'Value',0);

        for j=1:maxx
            for i=1:maxy                    % i and j are indexes of the UMAT
                xCur=j;
                yCur=i*-1;

                k=(j-1)*maxy+i;          % linear index into GS2
                gs2(k).X = [ xCur-1 xCur   xCur xCur-1 NaN ];
                gs2(k).Y = [ yCur-1 yCur-1 yCur yCur NaN ];
                gs2(k).Value = M.codebook(k,v);
                gs2(k).BoundingBox = [xCur-1 yCur-1; xCur yCur];
                %dataIndex=find(bmus==k);
                neuron = som_sub2ind(M.topol.msize,[i j]);
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
        gss.View{numView+v} = gs2;
        gss.ViewIndexData(:,numView+v)=tmpViewIndexData;

        if M.mask(v) % if it was used in the train
            gss.ViewLabel(numView+v).Name=strcat('*','CP: ',M.comp_names(v));
        else
            gss.ViewLabel(numView+v).Name=strcat('CP: ',M.comp_names(v));
        end
        gss.ViewLabel(numView+v).Type='CP';
        gss.ViewLabel(numView+v).SomOrigin=somIndex;
    end
elseif  strcmpi( M.topol.lattice,'hexa') % hexa lattice
    column=[1 2 1 2];
    column=repmat(column,1,ceil(maxy/4));
    C=1;
    A=C*.5;
    B=sind(60)*C;

    for v=1:length(M.comp_names)  % for each variable

        gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                'Value',0);

        for j=1:maxx
            for i=1:maxy
                if column(i)==2  % 2nd
                    xCur=(j-1)*(2*B)+(B);
                elseif column(i)==1  % 1st
                    xCur=(j-1)*(2*B);
                end
                yCur=-(i-1)*(A+C);

                k=(j-1)*maxy+i;         % linear index into GS2
```

```
                    gs2(k).X = [xCur xCur xCur+B xCur+(2*B) xCur+(2*B) xCur+B NaN];
                    gs2(k).Y = [yCur+A yCur+A+C yCur+A+C+A yCur+A+C yCur+A yCur NaN];

                    gs2(k).Value = M.codebook(k,v);
                    gs2(k).BoundingBox = [j-1 i-1; j i];
                    neuron = som_sub2ind(M.topol.msize,[i j]);
                    dataIndex=find(bmus==neuron);
                    tmpViewIndexData(dataIndex)=k;
                end;
            end;

            gss.View{numView+v} = gs2;
            gss.ViewIndexData(DataUsed,numView+v)=tmpViewIndexData;

            if M.mask(v) % if it was used in the train
                gss.ViewLabel(numView+v).Name=strcat('*','CP: ',M.comp_names(v));
            else
                gss.ViewLabel(numView+v).Name=strcat('CP: ',M.comp_names(v));
            end
            gss.ViewLabel(numView+v).Type='CP';
            gss.ViewLabel(numView+v).SomOrigin=somIndex;
        end
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildComponentPlanes_Geo(geosom_struct,somIndex)
    %geosom_BuildComponentPlanes
    % function for build the CPs in the geosom algorithm
    %
    % Syntax: geosom_BuildComponentPlanes_Geo()
    % input - geosom_struct
    % output - none
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 09-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    M=geosom_struct.Som(somIndex);
    DataUsed=gss.Som(somIndex).topol.DataUsed;

    %geosom_struct.NumData
    mapSize=(M.topol.msize);
    maxx = mapSize(2);
    maxy =mapSize(1);
    tmp=cell(maxx*maxy,1);

    if strcmpi(M.topol.Normalization,'none')
        bmus=som_bmus_geo(M,geosom_struct.NumData,M.topol.k);
    else
        sD=som_data_struct(geosom_struct.NumData);
        sD=som_normalize(sD,M.topol.Normalization);
        bmus=som_bmus_geo(M,sD,M.topol.k);
    end

    % check for existance of other Views
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    % create polygon structure
    for i=1:maxx*maxy;tmp{i}='Polygon';end;
```

```
if strcmpi( M.topol.lattice,'rect') % rect lattice
    for v=1:length(M.comp_names)  % for each variable

        gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                'Value',0);
        for j=1:maxx
            for i=1:maxy                    % i and j are indexes of the UMAT
                xCur=j;
                yCur=i*-1;

                k=(j-1)*maxy+i;
                %k=(i-1)*maxx+j;         % linear index into GS2
                gs2(k).X = [ j-1 j   j j-1 NaN ];
                gs2(k).Y = [ i-1 i-1 i i NaN ];
                gs2(k).Value = M.codebook(k,v);
                gs2(k).BoundingBox = [j-1 i-1; j i];
                %dataIndex=find(bmus==k);
                neuron = som_sub2ind(M.topol.msize,[i j]);
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
        gss.View{numView+v} = gs2;
        gss.ViewIndexData(:,numView+v)=tmpViewIndexData;

        if M.mask(v) % if it was used in the train
            gss.ViewLabel(numView+v).Name=strcat('*','CP: ',M.comp_names(v));
        else
            gss.ViewLabel(numView+v).Name=strcat('CP: ',M.comp_names(v));
        end
        gss.ViewLabel(numView+v).Type='CP';
        gss.ViewLabel(numView+v).SomOrigin=somIndex;
    end
elseif  strcmpi( M.topol.lattice,'hexa') % hexa lattice
    column=[1 2 1 2];
    column=repmat(column,1,ceil(maxy/4));
    C=1;
    A=C*.5;
    B=sind(60)*C;

    for v=1:length(M.comp_names)  % for each variable

        gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                'Value',0);
        for j=1:maxx
            for i=1:maxy
                if column(i)==2  % 2nd
                    xCur=(j-1)*(2*B)+(B);
                elseif column(i)==1  % 1st
                    xCur=(j-1)*(2*B);
                end
                yCur=-(i-1)*(A+C);

                k=(j-1)*maxy+i;
                %k=(i-1)*maxx+j;         % linear index into GS2
                gs2(k).X = [xCur xCur xCur+B xCur+(2*B) xCur+(2*B) xCur+B NaN];
                gs2(k).Y = [yCur+A yCur+A+C yCur+A+C+A yCur+A+C yCur+A yCur NaN];

                gs2(k).Value = M.codebook(k,v);
                gs2(k).BoundingBox = [j-1 i-1; j i];
                neuron = som_sub2ind(M.topol.msize,[i j]);
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
        gss.View{numView+v} = gs2;
        gss.ViewIndexData(DataUsed,numView+v)=tmpViewIndexData;
```

```
            if M.mask(v) % if it was used in the train
                gss.ViewLabel(numView+v).Name=strcat('*','CP: ',M.comp_names(v));
            else
                gss.ViewLabel(numView+v).Name=strcat('CP: ',M.comp_names(v));
            end
            gss.ViewLabel(numView+v).Type='CP';
            gss.ViewLabel(numView+v).SomOrigin=somIndex;
        end
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildComponentPlanesHSOM(sD,somIndex)
    %geosom_BuildComponentPlanesHSOM
    % function for build the CPs in the HSOM
    %
    % Syntax: geosom_BuildComponentPlanesHSOM()
    % input - somIndex
    % output - none
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: October 08-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    M=gss.Som(somIndex);

    %geosom_struct.NumData
    mapSize=(M.topol.msize);
    maxx = mapSize(2);
    maxy =mapSize(1);
    tmp=cell(maxx*maxy,1);
    DataUsed = gss.Som(somIndex).topol.DataUsed;

    bmus=som_bmus(gss.Som(somIndex),sD);

    % check for existance of other Views
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    % create polygon structure
    for i=1:maxx*maxy;tmp{i}='Polygon';end;

    if strcmpi( M.topol.lattice,'rect') % rect lattice
        for v=1:length(M.comp_names)  % for each variable

            gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                    'Value',0);

            for j=1:maxx
                for i=1:maxy                    % i and j are indexes of the UMAT
                    xCur=j;
                    yCur=i*-1;

                    k=(j-1)*maxy+i;          % linear index into GS2
                    gs2(k).X = [ xCur-1 xCur   xCur xCur-1 NaN ];
                    gs2(k).Y = [ yCur-1 yCur-1 yCur yCur NaN ];
                    gs2(k).Value = M.codebook(k,v);
                    gs2(k).BoundingBox = [xCur-1 yCur-1; xCur yCur];
                    %dataIndex=find(bmus==k);
                    neuron = som_sub2ind(M.topol.msize,[i j]);
                    dataIndex=find(bmus==neuron);
```

```
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
        gss.View{numView+v} = gs2;
        gss.ViewIndexData(:,numView+v)=tmpViewIndexData;

        if M.mask(v) % if it was used in the train
            gss.ViewLabel(numView+v).Name=strcat('*','CP: ',M.comp_names(v));
        else
            gss.ViewLabel(numView+v).Name=strcat('CP: ',M.comp_names(v));
        end
        gss.ViewLabel(numView+v).Type='CP';
        gss.ViewLabel(numView+v).SomOrigin=somIndex;
    end
elseif  strcmpi( M.topol.lattice,'hexa') % hexa lattice
    column=[1 2 1 2];
    column=repmat(column,1,ceil(maxy/4));
    C=1;
    A=C*.5;
    B=sind(60)*C;

    for v=1:length(M.comp_names)  % for each variable

        gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                'Value',0);

        for j=1:maxx
            for i=1:maxy
                if column(i)==2  % 2nd
                    xCur=(j-1)*(2*B)+(B);
                elseif column(i)==1  % 1st
                    xCur=(j-1)*(2*B);
                end
                yCur=-(i-1)*(A+C);

                k=(j-1)*maxy+i;         % linear index into GS2
                gs2(k).X = [xCur xCur xCur+B xCur+(2*B) xCur+(2*B) xCur+B NaN];
                gs2(k).Y = [yCur+A yCur+A+C yCur+A+C+A yCur+A+C yCur+A yCur NaN];

                gs2(k).Value = M.codebook(k,v);
                gs2(k).BoundingBox = [j-1 i-1; j i];
                neuron = som_sub2ind(M.topol.msize,[i j]);
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;

        gss.View{numView+v} = gs2;
        gss.ViewIndexData(DataUsed,numView+v)=tmpViewIndexData;

        if M.mask(v) % if it was used in the train
            gss.ViewLabel(numView+v).Name=strcat('*','CP: ',M.comp_names(v));
        else
            gss.ViewLabel(numView+v).Name=strcat('CP: ',M.comp_names(v));
        end
        gss.ViewLabel(numView+v).Type='CP';
        gss.ViewLabel(numView+v).SomOrigin=somIndex;
    end
end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildCoordinatePlots(SelNumData,somIndex)
    %GEOSOM_BUILDCOORDINATEPLOTS
    % function to build the Coordinate Plots
    % Syntax: geosom_BuildCoordinatePlots(SelNumData,somIndex)
    % SelNumData - selected rows of the NumData
    % somIndex - index of the correspondent SOM
```

```
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 01-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    global gss

    % check for existance of other Views
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    [numData numField]=size(gss.NumData);
    tmpViewIndexData=zeros(numData,1);

    % create line structure
    for i=1:numData;tmp{i}='Line';end;

    % MinMax
    Minim=repmat(min(gss.NumData),length(gss.NumData),1);
    Maxim=repmat(max(gss.NumData),length(gss.NumData),1);
    NumData=(gss.NumData-Minim)./(Maxim-Minim);

    for v=SelNumData  % for each variable
        gs2 = struct('Geometry',tmp','BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                 'Value',0);
        for i=1:numData  % for each individual
            gs2(i).X = [(1:length(SelNumData)),NaN];
            gs2(i).Y = [NumData(i,SelNumData),NaN];
            gs2(i).Value = i;
            gs2(i).BoundingBox = [1 min(gs2(i).Y);length(SelNumData) max(gs2(i).Y)];

            tmpViewIndexData(i)=i;
        end;

        gss.View{numView+1} = gs2;
        gss.ViewIndexData(:,numView+1)=tmpViewIndexData;

        strName='pcp';
        for i=SelNumData
            strName=strcat(strName,'.',cell2mat(gss.NumDataLabel(i)));
        end
        gss.ViewLabel(numView+1).Name=strName;

        gss.ViewLabel(numView+1).Type='PCP';
        gss.ViewLabel(numView+1).SomOrigin=somIndex;
        gss.ViewLabel(numView+1).SelVariables=SelNumData;
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [somIndex,geosom_struct]=geosom_BuildGeoSom(parameters);
    % geosom_BuildGeoSom - Train a GeoSOM
    % Syntax: [somIndex,geosom_struct]=geosom_BuildGeoSom(parameters)
    % input
    %           parameters - struct with all the geosom parameters
    % output
    %           somIndex - index od the som in the gss struct
    %           geosom_struct - gss struct
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 04-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008
```

```
    global gss

    %get data selection
    if parameters.TrainSelection
        DataInd=find(gss.SelectedIndex==1);
    else
        DataInd=(1:size(gss.SelectedIndex,1));
    end

    % get the centroids from spatial data or from fields list
    if parameters.chkSelGeoCom % from the list
        Data=[gss.NumData(DataInd,parameters.SelectedComponentsGeo)
gss.NumData(DataInd,:)];
    else % from spatial data
        %check the type of spatial feature (point or other)
        if strcmpi(gss.ViewLabel(1).Type,'GeographicMap') % we have a map
            if strcmpi(gss.View{1}(1).Geometry,'Point') % point shp
                % get x and y
                Centr=zeros(length(DataInd),2);
                counter=0;
                for i=DataInd' % selected points
                    counter=counter+1;
                    Centr(counter,1)=gss.View{1}(i).X;
                    Centr(counter,2)=gss.View{1}(i).Y;
                end
            else % line or polygon shp
                % get x and y mean
                Centr=zeros(length(DataInd),2);
                counter=0;
                for i=DataInd' % selected points
                    counter=counter+1;
                    Centr(counter,1)=mean(gss.View{1}(i).BoundingBox(:,1));
                    Centr(counter,2)=mean(gss.View{1}(i).BoundingBox(:,2));
                end
            end
        end
        Data=[Centr gss.NumData(DataInd,:)];
    end

    % add the component names
    parameters.ComponentsNames=['X';'Y';parameters.ComponentsNames];

    [numData numFeatures]=size(Data);
    mask=zeros(numFeatures,1);
    % add x and y to the mask
    mask(1:2)=1;
    % add non geo component to the mask
    mask(parameters.SelectedComponents+2)=1;

    sM=som_map_struct(numFeatures,...
        'msize',[parameters.ydim parameters.xdim],...
        'lattice',parameters.Lattice,...
        'shape',parameters.Shape,...
        'neigh',parameters.Neigh,...
        'comp_names',parameters.ComponentsNames,...
        'mask',mask);
    sM.name=strcat('GeoSOM.',date);

    % map type
    if strcmp(parameters.MapType,'geo dist')==1
        sM = som_randinit_geo(Data,sM);
    elseif strcmp(parameters.MapType,'linear')==1
        sM = som_lininit(Data,sM);
    else
        sM = som_randinit(Data,sM);
    end
    sD=som_data_struct(Data);
```

```
switch parameters.Normalization
    case 'none'
    otherwise
        sD=som_normalize(sD,parameters.Normalization);
end

%save normalization in the sM
sM.topol.Normalization=parameters.Normalization;
%save k in the sM
sM.topol.k=parameters.k;
%save f in the sM
sM.topol.f=parameters.f;
%save selection in the sM
sM.topol.DataUsed=DataInd;

if parameters.chkFinetune==0
    % rough training

    % batch or sequential
    if parameters.batchTrain
        %not implemented
    else
        sM = som_seqtrain_geo(sM,sD,...
        'geo_match',parameters.k,...
        'geo_update',parameters.f,...
        'radius_ini',parameters.radius_ini_1,...
        'radius_fin',0,...
        'alpha_ini',parameters.alpha_ini_1,...
        'trainlen',parameters.niterations_1, ...
        'trainlen_type',parameters.IterType,...
        'alpha_type',parameters.Length_funct,...
        'sample_order', parameters.Order,...
        parameters.IterType);
    end
end

% finetune training
if parameters.batchTrain
       %not implemented
else
    sM = som_seqtrain_geo(sM,sD,...
    'geo_match',parameters.k,...
    'geo_update',parameters.f,...
    'radius_ini',parameters.radius_ini_2,...
    'radius_fin',0,...
    'alpha_ini',parameters.alpha_ini_2,...
    'trainlen',parameters.niterations_2, ...
    'trainlen_type',parameters.IterType,...
    'alpha_type',parameters.Length_funct,...
    'sample_order', parameters.Order,...
     parameters.IterType);
end

% add a field to the map to recognize its variant
sM.method='geosom';

% Finalize, by instering the View into the main structure
if isfield(gss,'Som')
    numSom = length(gss.Som);
    somIndex = numSom+1;
    gss.Som(somIndex) = sM;
else
    gss.Som(1)=sM;
    somIndex=1;
end;

%update geosom_struct with two extra columns
geosom_struct=gss;
geosom_struct.NumData=Data;
```

```
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sD=geosom_BuildHierarqSom(parameters)
    % geosom_BuildHierarqSom()
    % NAME: GEOSOM_BUILDHIERARQSOM
    %
    % OBJECTIVE: Train a SOM with parameters from strcut parameters
    % NOTE     : Instead of the original input data, uses for each input
    %            the x and y from the umats on the selected SOMs
    %
    % INPUT PARAMETERS:
    %       parameters - struct with parameters
    %
    % OUTPUT PARAMETERS:
    %
    % COMMENTS:
    %
    % V.0.1 2009/07/07 By R.Henriques & V.Lobo

    global gss

    % get input data
    % for each input pattern, we will use the coordinates
    % (on the umat) of its BMU unit

    % use all data or selection?
    if parameters.TrainSelection
        DataInd=find(gss.SelectedIndex==1);
    else
        DataInd=(1:size(gss.SelectedIndex,1))';
    end

    % Get the Selected Input to the HSOM
    % 1 'Umats´ coordinates'
    % 2 'Quantization error '
    % 3 'All pattern activations'
    % 4 'Spatial coordinates'

    % set variables names
    VarNames={};
    % Get the Umats indexes from the selected SOMs
    indices=[];
    data=[];
    for i=parameters.SelectedComponents %for each SOM
        for v=1:length(gss.ViewLabel)
            if strcmpi(gss.ViewLabel(v).Type,'UMAT')
                if gss.ViewLabel(v).SomOrigin==i
                    indices=[indices v];
                    break
                end
            end
        end
    end

    if find(parameters.SelectedInputs==1)
        % NOTE : instead of using the SOM in the som_ind2sub we are using a matrix
        % with the Umat size (considering the interspaces features)
        datatmp=zeros(size(DataInd,1),size(parameters.SelectedComponents,2)*2);
        for i=1:length(parameters.SelectedComponents)' %for each SOM
            sizeUmat=gss.Som(parameters.SelectedComponents(i)).topol.msize;
            sizeUmat=sizeUmat.*2-1;
            sub = som_ind2sub(sizeUmat,...
                gss.ViewIndexData(DataInd,indices(i)));
            datatmp(:,i*2-1:i*2)=sub;

            VarNames(size(VarNames,1)+1,1)=cellstr(strcat('UcoordX
SOM.',gss.Som(i).name));
```

```
            VarNames(size(VarNames,1)+1,1)=cellstr(strcat('UcoordY
SOM.',gss.Som(i).name));
        end
        data=[data datatmp];
        clear datatmp;
    end

    if find(parameters.SelectedInputs==2)
        % add the qe for each individual, i.e for each unit the qe
        % indBMU=indices+1;

        datatmp=zeros(size(DataInd,1),size(indices,2));
        counter=0;
        for i=(parameters.SelectedComponents) %for each SOM
            sD=som_data_struct(gss.NumData(DataInd,:));
            if strcmpi(gss.Som(i).topol.Normalization,'none')==0
                sD=som_normalize(sD,gss.Som(i).topol.Normalization);
            end
            [Bmus, Qerrors] = som_bmus(gss.Som(i), sD);
            counter=counter+1;
            datatmp(:,counter)=Qerrors; %(DataInd);

            VarNames(size(VarNames,1)+1,1)=cellstr(strcat('Qe SOM.',gss.Som(i).name));
        end
        data=[data datatmp];
        clear datatmp;
    end

    if find(parameters.SelectedInputs==3)
        % all patterns distances

        for i=(parameters.SelectedComponents) %for each SOM
            datatmp=zeros(size(DataInd,2),prod(gss.Som(i).topol.msize));

            sD=som_data_struct(gss.NumData(DataInd,:));
            %sD=som_normalize(sD,gss.Som(i).topol.Normalization);
            if strcmpi(gss.Som(i).topol.Normalization,'none')==0
                sD=som_normalize(sD,gss.Som(i).topol.Normalization);
            end
            %[Bmus, Qerrors] = som_bmus(gss.Som(i), sD);
            %datatmp(:,1:prod(gss.Som(i).topol.msize))=som_eucdist2(sD,gss.Som(i));
            IndMask=find(gss.Som(i).mask==1);
            for j=1:size(sD.data,1);
                for k=1:size(gss.Som(i).codebook,1);
                    datatmp(j,k)=norm(sD.data(j,IndMask)-gss.Som(i).codebook(k,IndMask));
                end
            end

            for j=1:prod(gss.Som(i).topol.msize)

%VarNames(size(VarNames,1)+1,1)=cellstr(strcat('Dist.Unit.',num2str(j),'.SOM.',num2str(i)
));

VarNames(size(VarNames,1)+1,1)=cellstr(strcat('Dist.Unit',num2str(j),'.',gss.Som(i).name)
);
            end

            data=[data datatmp];
            clear datatmp;
        end
    end

    if find(parameters.SelectedInputs==4)
        % add the spatial components
        % get the centroids from spatial data or from fields list
        %check the type of spatial feature (point or other)
        if strcmpi(gss.ViewLabel(1).Type,'GeographicMap') % we have a map
            if strcmpi(gss.View{1}(1).Geometry,'Point') % point shp
                % get x and y
```

```
            Centr=zeros(length(DataInd),2);
            counter=0;
            for i=DataInd % selected points
                counter=counter+1;
                Centr(counter,1)=gss.View{1}(i).X;
                Centr(counter,2)=gss.View{1}(i).Y;
            end
        else % line or polygon shp
            % get x and y mean
            Centr=zeros(length(DataInd),2);
            counter=0;
            for i=DataInd' % selected points
                counter=counter+1;
                Centr(counter,1)=mean(gss.View{1}(i).BoundingBox(:,1));
                Centr(counter,2)=mean(gss.View{1}(i).BoundingBox(:,2));
            end
        end
    end

    VarNames(size(VarNames,1)+1,1)=cellstr('Coord X');
    VarNames(size(VarNames,1)+1,1)=cellstr('Coord Y');

    data=[data Centr];
    clear datatmp;
end

[numData numFeatures]=size(data);

sM=som_map_struct(numFeatures,...
    'msize',[parameters.ydim parameters.xdim],...
    'lattice',parameters.Lattice,...
    'shape',parameters.Shape,...
    'neigh',parameters.Neigh,...
    'comp_names',VarNames);

% map type
if strcmp(parameters.MapType,'linear')==1
    sM = som_lininit(data,sM);
else
    sM = som_randinit(data,sM);
end

sD=som_data_struct(data);

switch parameters.Normalization
    case 'none'
    otherwise
        sD=som_normalize(sD,parameters.Normalization);
end

%save normalization in the sM
sM.topol.Normalization=parameters.Normalization;

%save selection in the sM
sM.topol.DataUsed=DataInd;

if parameters.chkFinetune==0
    % rough

    % batch or sequential
    if parameters.batchTrain
        sM = som_batchtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_1,...
        'radius_fin',0,...
        'trainlen',parameters.niterations_1);
    else
        sM = som_seqtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_1,...
        'radius_fin',0,...
```

```
            'alpha_ini',parameters.alpha_ini_1,...
            'trainlen',parameters.niterations_1, ...
            'trainlen_type',parameters.IterType,...
            'alpha_type',parameters.Length_funct,...
            'sample_order', parameters.Order,...
            parameters.IterType);
        end
    end

    % finetune
    if parameters.batchTrain
        sM = som_batchtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_2,...
        'radius_fin',0,...
        'trainlen',parameters.niterations_2);
    else
        sM = som_seqtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_2,...
        'radius_fin',0,...
        'alpha_ini',parameters.alpha_ini_2,...
        'trainlen',parameters.niterations_2, ...
        'trainlen_type',parameters.IterType,...
        'alpha_type',parameters.Length_funct,...
        'sample_order', parameters.Order,...
        parameters.IterType);
    end

    %calculate Errors
    [sM.topol.qe,sM.topol.te] = som_quality(sM, sD);

    %hierarchical SOM
    sM.method='hsom';
    sM.name=parameters.SOMname;
    sM.topol.Obs=parameters.Obs;

    % Finalize, by instering the View into the main structure
    if isfield(gss,'Som')
        numSom = length(gss.Som);
        somIndex = numSom+1;
        gss.Som(somIndex) = sM;
    else
        gss.Som(1)=sM;
        somIndex=1;
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildHits(somIndex)
    %geosom_BuildHits
    % function for build the Hits views
    % creates 2 hits views
    % 1. Hits for visualization in UMAT
    % 2. Hits for visualization in CPs
    %
    % Syntax: geosom_BuildHits()
    % input - somIndex
    % output - none
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 09-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    % check for existance of other Views
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
```

```
    numView=0;
else
    numView = length(gss.View);
end

% % create hits
sM = gss.Som(gss.ViewLabel(numView).SomOrigin);
DataUsed=gss.Som(somIndex).topol.DataUsed;

%check data normalization
if strcmpi(sM.topol.Normalization,'none')
    %bmus=som_bmus(gss.Som(somIndex),gss.NumData(DataUsed,:));
    hits = som_hits(sM,gss.NumData(DataUsed,:));
else
    sD=som_data_struct(gss.NumData(DataUsed,:));
    sD=som_normalize(sD,sM.topol.Normalization);
    hits = som_hits(sM,sD);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% create hits for UMAT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sU=som_umat(sM);
[ maxy maxx ] = size(sU);

% create polygon structure
for i=1:(ceil(maxx/2)*ceil(maxy/2));tmp{i}='Polygon'; end;

gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
        'Value',0);

if strcmpi(sM.topol.lattice,'rect') % rect lattice
    hitsNorm = hits./max(hits);
    k=0;
    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            %k=(i-1)*maxx+j;          % linear index into GS2
            if mod(i,2)==1 && mod(j,2)==1  % corresponds to a neuron in the UMAT
                xCentroid=j-0.5;
                yCentroid=i*(-1)-0.5;

                %k=k+1;
                si=ceil(i/2);sj=ceil(j/2);
                k = som_sub2ind(sM.topol.msize,[si sj]);

                gs2(k).X = [xCentroid-1*hitsNorm(k)...
                            xCentroid+1*hitsNorm(k)...
                            xCentroid+1*hitsNorm(k)...
                            xCentroid-1*hitsNorm(k) NaN];
                gs2(k).Y = [yCentroid-1*hitsNorm(k)...
                            yCentroid-1*hitsNorm(k)...
                            yCentroid+1*hitsNorm(k)...
                            yCentroid+1*hitsNorm(k) NaN];
                gs2(k).Value = k;
                gs2(k).BoundingBox = [xCentroid-1*hitsNorm(k)...
                                      yCentroid-1*hitsNorm(k);...
                                      xCentroid+1*hitsNorm(k)...
                                      yCentroid+1*hitsNorm(k)];
            end
        end;
    end;
    gss.View{numView+1} = gs2;
    %gss.ViewIndexData(:,numView+1)=tmpViewIndexData;

    gss.ViewLabel(numView+1).Name=strcat('Umat Hits');
    gss.ViewLabel(numView+1).Type='Hits';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;
```

```
    elseif  strcmpi( sM.topol.lattice,'hexa') % hexa lattice
        hitsNorm = hits.*1.5./max(hits);

        column=[1 2 3 2];
        column=repmat(column,1,ceil(maxy/4));
        k=0;
        for j=1:maxx
            for i=1:maxy                    % i and j are indexes of the UMAT
                if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                    C=1;
                    A=C*.5;
                    B=sind(60)*C;

                    %k=k+1;
                    si=ceil(i/2);sj=ceil(j/2);
                    k = som_sub2ind(sM.topol.msize,[si sj]);

                    % check if it is the 1st 2nd or third row
                    if column(i)==3      % 3rd
                        xCent=(j-1)*(2*B)+(3*B);
                    elseif column(i)==2  % 2nd
                        xCent=(j-1)*(2*B)+(2*B);
                    elseif column(i)==1  % 1st
                        xCent=(j-1)*(2*B)+B;
                    end
                    yCent=-1*(i-1)*(A+C)+(1/C);

                    % redefine C A and B acording to hits
                    C=hitsNorm(k);
                    A=C*.5;
                    B=sind(60)*C;

                    %k=(i-1)*maxx+j;         % linear index into GS2
                    gs2(k).X = [xCent-B xCent-B xCent xCent+B xCent+B xCent NaN];
                    gs2(k).Y = [yCent-1/2*C yCent+1/2*C yCent+A+1/2*C yCent+1/2*C yCent-
1/2*C yCent-A-1/2*C NaN];
                    gs2(k).Value = sU(i,j);
                    gs2(k).BoundingBox = [xCent-B yCent-A-1/2*C; xCent+B yCent+A+1/2*C];
                end;
            end;
        end;
        gss.View{numView+1} = gs2;
        gss.ViewLabel(numView+1).Name=strcat('Umat Hits');
        gss.ViewLabel(numView+1).Type='Hits';
        gss.ViewLabel(numView+1).SomOrigin=somIndex;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% create hits for CPs
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    maxy =sM.topol.msize(1);
    maxx =sM.topol.msize(2);

    % create polygon structure
    for i=1:(ceil(maxx/2)*ceil(maxy/2));tmp{i}='Polygon'; end;

    gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
            'Value',0);

    if strcmpi(sM.topol.lattice,'rect') % rect lattice
        hitsNorm = hits./max(hits);
        k=0;
        for j=1:maxx
                for i=1:maxy                    % i and j are indexes of the UMAT
                %k=(i-1)*maxx+j;         % linear index into GS2
                %if mod(i,2)==1 && mod(j,2)==1  % corresponds to a neuron in the UMAT
                    xCentroid=j-0.5;
                    yCentroid=-1*i-0.5;
```

```
                    %k=k+1;
                    k = som_sub2ind(sM.topol.msize,[i j]);

                    gs2(k).X = [xCentroid-1*hitsNorm(k)...
                                xCentroid+1*hitsNorm(k)...
                                xCentroid+1*hitsNorm(k)...
                                xCentroid-1*hitsNorm(k) NaN];
                    gs2(k).Y = [yCentroid-1*hitsNorm(k)...
                                yCentroid-1*hitsNorm(k)...
                                yCentroid+1*hitsNorm(k)...
                                yCentroid+1*hitsNorm(k) NaN];
                    gs2(k).Value = k;

                    gs2(k).BoundingBox = [xCentroid-1*hitsNorm(k)...
                                          yCentroid-1*hitsNorm(k);...
                                          xCentroid+1*hitsNorm(k)...
                                          yCentroid+1*hitsNorm(k)];
                %end
            end;
        end;
        gss.View{numView+2} = gs2;
        %gss.ViewIndexData(:,numView+1)=tmpViewIndexData;

        gss.ViewLabel(numView+2).Name=strcat('CP Hits');
        gss.ViewLabel(numView+2).Type='Hits';
        gss.ViewLabel(numView+2).SomOrigin=somIndex;

    elseif  strcmpi( sM.topol.lattice,'hexa') % hexa lattice
        hitsNorm = hits.*1.5./max(hits);

        column=[1 2 1 2];
        column=repmat(column,1,ceil(maxy/4));
        k=0;
        for j=1:maxx
            for i=1:maxy                    % i and j are indexes of the UMAT

                %if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                    C=1;
                    A=C*.5;
                    B=sind(60)*C;

                    %k=k+1;
                    k = som_sub2ind(sM.topol.msize,[i j]);

                    % check if it is the 1st 2nd or third row
                    if column(i)==3      % 3rd
                        xCent=(j-1)*(2*B)+(3*B);
                    elseif column(i)==2  % 2nd
                        xCent=(j-1)*(2*B)+(2*B);
                    elseif column(i)==1  % 1st
                        xCent=(j-1)*(2*B)+B;
                    end
                    yCent=-1*(i-1)*(A+C)+(1/C);

                    % redefine C A and B acording to hits
                    C=hitsNorm(k);
                    A=C*.5;
                    B=sind(60)*C;

                    %k=(i-1)*maxx+j;         % linear index into GS2
                    gs2(k).X = [xCent-B xCent-B xCent xCent+B xCent+B xCent NaN];
                    gs2(k).Y = [yCent-1/2*C yCent+1/2*C yCent+A+1/2*C yCent+1/2*C yCent-
1/2*C yCent-A-1/2*C NaN];
                    gs2(k).Value = sU(i,j);
                    gs2(k).BoundingBox = [xCent-B yCent-A-1/2*C; xCent+B yCent+A+1/2*C];
                %end;
            end;
        end;
```

```
        gss.View{numView+2} = gs2;
        gss.ViewLabel(numView+2).Name=strcat('CP Hits');
        gss.ViewLabel(numView+2).Type='Hits';
        gss.ViewLabel(numView+2).SomOrigin=somIndex;
    end
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildHits_Geo(geosom_struct,somIndex)
    %geosom_BuildHits_Geo
    % function for build the Hits views in the GeoSOM
    % creates 2 hits views
    % 1. Hits for visualization in UMAT
    % 2. Hits for visualization in CPs
    %
    % Syntax: geosom_BuildHits()
    % input - somIndex
    % output - none
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 09-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    % check for existance of other Views
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    % % create hits
    %sM = geosom_struct.Som(geosom_struct.ViewLabel(numView).SomOrigin);
    M=geosom_struct.Som(somIndex);

    if strcmpi(M.topol.Normalization,'none')
        hits = som_hits_geo(M,geosom_struct.NumData,M.topol.k);
        %bmus=som_bmus_geo(M.Som(somIndex),M.NumData,M.topol.k);
    else
        sD=som_data_struct(geosom_struct.NumData);
        sD=som_normalize(sD,M.topol.Normalization);
        hits = som_hits_geo(M,sD,M.topol.k);
        %bmus=som_bmus_geo(M.Som(somIndex),sD,M.topol.k);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % create hits for UMAT
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%% Nota alterar o kradius para um valor dinamico em vez de 0
    sU=som_umat_geo(M);
    [ maxy maxx ] = size(sU);

    % create polygon structure
    for i=1:(ceil(maxx/2)*ceil(maxy/2));tmp{i}='Polygon'; end;

    gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
            'Value',0);

    if strcmpi(M.topol.lattice,'rect') % rect lattice
        hitsNorm = hits./max(hits);
        k=0;
        for j=1:maxx
            for i=1:maxy                    % i and j are indexes of the UMAT
                %k=(i-1)*maxx+j;         % linear index into GS2
```

```
            if mod(i,2)==1 && mod(j,2)==1  % corresponds to a neuron in the UMAT
                xCentroid=j-0.5;
                yCentroid=i*(-1)-0.5;

                %k=k+1;
                si=ceil(i/2);sj=ceil(j/2);
                k = som_sub2ind(M.topol.msize,[si sj]);

                gs2(k).X = [xCentroid-1*hitsNorm(k)...
                            xCentroid+1*hitsNorm(k)...
                            xCentroid+1*hitsNorm(k)...
                            xCentroid-1*hitsNorm(k) NaN];
                gs2(k).Y = [yCentroid-1*hitsNorm(k)...
                            yCentroid-1*hitsNorm(k)...
                            yCentroid+1*hitsNorm(k)...
                            yCentroid+1*hitsNorm(k) NaN];
                gs2(k).Value = k;
                gs2(k).BoundingBox = [xCentroid-1*hitsNorm(k)...
                                      yCentroid-1*hitsNorm(k);...
                                      xCentroid+1*hitsNorm(k)...
                                      yCentroid+1*hitsNorm(k)];
            end
        end;
    end;
    gss.View{numView+1} = gs2;
    %gss.ViewIndexData(:,numView+1)=tmpViewIndexData;

    gss.ViewLabel(numView+1).Name=strcat('Umat Hits');
    gss.ViewLabel(numView+1).Type='Hits';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;

elseif  strcmpi(M.topol.lattice,'hexa') % hexa lattice
    hitsNorm = hits.*1.5./max(hits);

    column=[1 2 3 2];
    column=repmat(column,1,ceil(maxy/4));
    k=0;
    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                C=1;
                A=C*.5;
                B=sind(60)*C;

                %k=k+1;
                si=ceil(i/2);sj=ceil(j/2);
                k = som_sub2ind(M.topol.msize,[si sj]);

                % check if it is the 1st 2nd or third row
                if column(i)==3      % 3rd
                    xCent=(j-1)*(2*B)+(3*B);
                elseif column(i)==2  % 2nd
                    xCent=(j-1)*(2*B)+(2*B);
                elseif column(i)==1  % 1st
                    xCent=(j-1)*(2*B)+B;
                end
                yCent=-1*(i-1)*(A+C)+(1/C);

                % redefine C A and B acording to hits
                C=hitsNorm(k);
                A=C*.5;
                B=sind(60)*C;

                %k=(i-1)*maxx+j;         % linear index into GS2
                gs2(k).X = [xCent-B xCent-B xCent xCent+B xCent+B xCent NaN];
                gs2(k).Y = [yCent-1/2*C yCent+1/2*C yCent+A+1/2*C yCent+1/2*C yCent-
1/2*C yCent-A-1/2*C NaN];
                gs2(k).Value = sU(i,j);
                gs2(k).BoundingBox = [xCent-B yCent-A-1/2*C; xCent+B yCent+A+1/2*C];
```

```
            end;
        end;
    end;
    gss.View{numView+1} = gs2;
    gss.ViewLabel(numView+1).Name=strcat('Umat Hits');
    gss.ViewLabel(numView+1).Type='Hits';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create hits for CPs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
maxy =M.topol.msize(1);
maxx =M.topol.msize(2);

% create polygon structure
for i=1:(ceil(maxx/2)*ceil(maxy/2));tmp{i}='Polygon'; end;

gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
        'Value',0);

if strcmpi(M.topol.lattice,'rect') % rect lattice
    hitsNorm = hits./max(hits);
    k=0;
    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            %k=(i-1)*maxx+j;          % linear index into GS2
            xCentroid=j-0.5;
            yCentroid=i*(-1)-0.5;

            %k=k+1;
            k = som_sub2ind(M.topol.msize,[i j]);

            gs2(k).X = [xCentroid-1*hitsNorm(k)...
                        xCentroid+1*hitsNorm(k)...
                        xCentroid+1*hitsNorm(k)...
                        xCentroid-1*hitsNorm(k) NaN];
            gs2(k).Y = [yCentroid-1*hitsNorm(k)...
                        yCentroid-1*hitsNorm(k)...
                        yCentroid+1*hitsNorm(k)...
                        yCentroid+1*hitsNorm(k) NaN];
            gs2(k).Value = k;

            gs2(k).BoundingBox = [xCentroid-1*hitsNorm(k)...
                                  yCentroid-1*hitsNorm(k);...
                                  xCentroid+1*hitsNorm(k)...
                                  yCentroid+1*hitsNorm(k)];
        end;
    end;
    gss.View{numView+2} = gs2;
    %gss.ViewIndexData(:,numView+1)=tmpViewIndexData;

    gss.ViewLabel(numView+2).Name=strcat('CP Hits');
    gss.ViewLabel(numView+2).Type='Hits';
    gss.ViewLabel(numView+2).SomOrigin=somIndex;

elseif  strcmpi(M.topol.lattice,'hexa') % hexa lattice
    hitsNorm = hits.*1.5./max(hits);

    column=[1 2 1 2];
    column=repmat(column,1,ceil(maxy/4));
    k=0;
    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            C=1;
            A=C*.5;
            B=sind(60)*C;

            %k=k+1;
```

```
                k = som_sub2ind(M.topol.msize,[i j]);

                % check if it is the 1st 2nd or third row
                if column(i)==3      % 3rd
                    xCent=(j-1)*(2*B)+(3*B);
                elseif column(i)==2  % 2nd
                    xCent=(j-1)*(2*B)+(2*B);
                elseif column(i)==1  % 1st
                    xCent=(j-1)*(2*B)+B;
                end
                yCent=-1*(i-1)*(A+C)+(1/C);

                % redefine C A and B acording to hits
                C=hitsNorm(k);
                A=C*.5;
                B=sind(60)*C;

                %k=(i-1)*maxx+j;         % linear index into GS2
                gs2(k).X = [xCent-B xCent-B xCent xCent+B xCent+B xCent NaN];
                gs2(k).Y = [yCent-1/2*C yCent+1/2*C yCent+A+1/2*C yCent+1/2*C yCent-1/2*C
yCent-A-1/2*C NaN];
                gs2(k).Value = sU(i,j);
                gs2(k).BoundingBox = [xCent-B yCent-A-1/2*C; xCent+B yCent+A+1/2*C];
            end;
        end;
        gss.View{numView+2} = gs2;
        gss.ViewLabel(numView+2).Name=strcat('CP Hits');
        gss.ViewLabel(numView+2).Type='Hits';
        gss.ViewLabel(numView+2).SomOrigin=somIndex;
    end
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildHitsHSOM(sD,somIndex)
    %geosom_BuildHitsHSOM
    % function for build the Hits views for the HSOM
    % creates 2 hits views
    % 1. Hits for visualization in UMAT
    % 2. Hits for visualization in CPs
    %
    % Syntax: geosom_BuildHitsHSOM()
    % input - sD,somIndex
    % output - none
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 09-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    % check for existance of other Views
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    % % create hits
    sM = gss.Som(gss.ViewLabel(numView).SomOrigin);
    DataUsed=gss.Som(somIndex).topol.DataUsed;

    hits = som_hits(sM,sD);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % create hits for UMAT
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
sU=som_umat(sM);
[ maxy maxx ] = size(sU);

% create polygon structure
for i=1:(ceil(maxx/2)*ceil(maxy/2));tmp{i}='Polygon'; end;

gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
        'Value',0);

if strcmpi(sM.topol.lattice,'rect') % rect lattice
    hitsNorm = hits./max(hits);
    k=0;
    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            %k=(i-1)*maxx+j;          % linear index into GS2
            if mod(i,2)==1 && mod(j,2)==1  % corresponds to a neuron in the UMAT
                xCentroid=j-0.5;
                yCentroid=i*(-1)-0.5;

                %k=k+1;
                si=ceil(i/2);sj=ceil(j/2);
                k = som_sub2ind(sM.topol.msize,[si sj]);

                gs2(k).X = [xCentroid-1*hitsNorm(k)...
                            xCentroid+1*hitsNorm(k)...
                            xCentroid+1*hitsNorm(k)...
                            xCentroid-1*hitsNorm(k) NaN];
                gs2(k).Y = [yCentroid-1*hitsNorm(k)...
                            yCentroid-1*hitsNorm(k)...
                            yCentroid+1*hitsNorm(k)...
                            yCentroid+1*hitsNorm(k) NaN];
                gs2(k).Value = k;
                gs2(k).BoundingBox = [xCentroid-1*hitsNorm(k)...
                                      yCentroid-1*hitsNorm(k);...
                                      xCentroid+1*hitsNorm(k)...
                                      yCentroid+1*hitsNorm(k)];
            end
        end;
    end;
    gss.View{numView+1} = gs2;
    %gss.ViewIndexData(:,numView+1)=tmpViewIndexData;

    gss.ViewLabel(numView+1).Name=strcat('Umat Hits');
    gss.ViewLabel(numView+1).Type='Hits';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;

elseif  strcmpi( sM.topol.lattice,'hexa') % hexa lattice
    hitsNorm = hits.*1.5./max(hits);

    column=[1 2 3 2];
    column=repmat(column,1,ceil(maxy/4));
    k=0;
    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                C=1;
                A=C*.5;
                B=sind(60)*C;

                %k=k+1;
                si=ceil(i/2);sj=ceil(j/2);
                k = som_sub2ind(sM.topol.msize,[si sj]);

                % check if it is the 1st 2nd or third row
                if column(i)==3      % 3rd
                    xCent=(j-1)*(2*B)+(3*B);
                elseif column(i)==2  % 2nd
                    xCent=(j-1)*(2*B)+(2*B);
```

```
                    elseif column(i)==1  % 1st
                        xCent=(j-1)*(2*B)+B;
                    end
                    yCent=-1*(i-1)*(A+C)+(1/C);

                    % redefine C A and B acording to hits
                    C=hitsNorm(k);
                    A=C*.5;
                    B=sind(60)*C;

                    %k=(i-1)*maxx+j;         % linear index into GS2
                    gs2(k).X = [xCent-B xCent-B xCent xCent+B xCent+B xCent NaN];
                    gs2(k).Y = [yCent-1/2*C yCent+1/2*C yCent+A+1/2*C yCent+1/2*C yCent-
1/2*C yCent-A-1/2*C NaN];
                    gs2(k).Value = sU(i,j);
                    gs2(k).BoundingBox = [xCent-B yCent-A-1/2*C; xCent+B yCent+A+1/2*C];
                end;
            end;
        end;
        gss.View{numView+1} = gs2;
        gss.ViewLabel(numView+1).Name=strcat('Umat Hits');
        gss.ViewLabel(numView+1).Type='Hits';
        gss.ViewLabel(numView+1).SomOrigin=somIndex;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % create hits for CPs
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    maxy =sM.topol.msize(1);
    maxx =sM.topol.msize(2);

    % create polygon structure
    for i=1:(ceil(maxx/2)*ceil(maxy/2));tmp{i}='Polygon'; end;

    gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
            'Value',0);

    if strcmpi(sM.topol.lattice,'rect') % rect lattice
        hitsNorm = hits./max(hits);
        k=0;
        for j=1:maxx
                for i=1:maxy                    % i and j are indexes of the UMAT
                %k=(i-1)*maxx+j;         % linear index into GS2
                %if mod(i,2)==1 && mod(j,2)==1  % corresponds to a neuron in the UMAT
                    xCentroid=j-0.5;
                    yCentroid=-1*i-0.5;

                    %k=k+1;
                    k = som_sub2ind(sM.topol.msize,[i j]);

                    gs2(k).X = [xCentroid-1*hitsNorm(k)...
                                xCentroid+1*hitsNorm(k)...
                                xCentroid+1*hitsNorm(k)...
                                xCentroid-1*hitsNorm(k) NaN];
                    gs2(k).Y = [yCentroid-1*hitsNorm(k)...
                                yCentroid-1*hitsNorm(k)...
                                yCentroid+1*hitsNorm(k)...
                                yCentroid+1*hitsNorm(k) NaN];
                    gs2(k).Value = k;

                    gs2(k).BoundingBox = [xCentroid-1*hitsNorm(k)...
                                          yCentroid-1*hitsNorm(k);...
                                          xCentroid+1*hitsNorm(k)...
                                          yCentroid+1*hitsNorm(k)];
                %end
            end;
        end;
        gss.View{numView+2} = gs2;
        %gss.ViewIndexData(:,numView+1)=tmpViewIndexData;
```

```
        gss.ViewLabel(numView+2).Name=strcat('CP Hits');
        gss.ViewLabel(numView+2).Type='Hits';
        gss.ViewLabel(numView+2).SomOrigin=somIndex;

    elseif  strcmpi( sM.topol.lattice,'hexa') % hexa lattice
        hitsNorm = hits.*1.5./max(hits);

        column=[1 2 1 2];
        column=repmat(column,1,ceil(maxy/4));
        k=0;
        for j=1:maxx
            for i=1:maxy                    % i and j are indexes of the UMAT

                %if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                    C=1;
                    A=C*.5;
                    B=sind(60)*C;

                    %k=k+1;
                    k = som_sub2ind(sM.topol.msize,[i j]);

                    % check if it is the 1st 2nd or third row
                    if column(i)==3      % 3rd
                        xCent=(j-1)*(2*B)+(3*B);
                    elseif column(i)==2  % 2nd
                        xCent=(j-1)*(2*B)+(2*B);
                    elseif column(i)==1  % 1st
                        xCent=(j-1)*(2*B)+B;
                    end
                    yCent=-1*(i-1)*(A+C)+(1/C);

                    % redefine C A and B acording to hits
                    C=hitsNorm(k);
                    A=C*.5;
                    B=sind(60)*C;

                    %k=(i-1)*maxx+j;         % linear index into GS2
                    gs2(k).X = [xCent-B xCent-B xCent xCent+B xCent+B xCent NaN];
                    gs2(k).Y = [yCent-1/2*C yCent+1/2*C yCent+A+1/2*C yCent+1/2*C yCent-
1/2*C yCent-A-1/2*C NaN];
                    gs2(k).Value = sU(i,j);
                    gs2(k).BoundingBox = [xCent-B yCent-A-1/2*C; xCent+B yCent+A+1/2*C];
                %end;
            end;
        end;
        gss.View{numView+2} = gs2;
        gss.ViewLabel(numView+2).Name=strcat('CP Hits');
        gss.ViewLabel(numView+2).Type='Hits';
        gss.ViewLabel(numView+2).SomOrigin=somIndex;
    end
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildSom(parameters);
    % geosom_BuildSom()
    % NAME: GEOSOM_BUILDSOM
    % OBJECTIVE: Train a SOM with parameters from strcut parameters
    % INPUT PARAMETERS:
    %       parameters - struct with parameters
    % OUTPUT PARAMETERS:
    %
    % COMMENTS:
    %       This is the basic fuction used to import data into the GeoSOM
    %       programs.
    %
    % V.0.1 2007/01/23 By V.Lobo & M.Loureiro
    % V.0.2 2007/01/24 By V.Lobo & M.Loureiro (bug review)
```

```
% V.0.3 2007/01/28 By V.Lobo & M.Loureiro - Major overhall. ONly creates
%                   the SOM itself.
% V.0.4 2008/06/30 By R.Henriques & V.Lobo (more SOM parameters included)

global gss

%get data selection
if parameters.TrainSelection
    DataInd=find(gss.SelectedIndex==1);
else
    DataInd=(1:size(gss.SelectedIndex,1));
end

[numData numFeatures]=size(gss.NumData(DataInd,:));
mask=zeros(numFeatures,1);
mask(parameters.SelectedComponents)=1;

sM=som_map_struct(numFeatures,...
    'msize',[parameters.ydim parameters.xdim],...
    'lattice',parameters.Lattice,...
    'shape',parameters.Shape,...
    'neigh',parameters.Neigh,...
    'comp_names',parameters.ComponentsNames,...
    'mask',mask);

% map type
if strcmp(parameters.MapType,'linear')==1
    sM = som_lininit(gss.NumData(DataInd,:),sM);
else
    sM = som_randinit(gss.NumData(DataInd,:),sM);
end

sD=som_data_struct(gss.NumData(DataInd,:));

switch parameters.Normalization
    case 'none'
    otherwise
        sD=som_normalize(sD,parameters.Normalization);
end

%save normalization in the sM
sM.topol.Normalization=parameters.Normalization;

%save selection in the sM
sM.topol.DataUsed=DataInd;

if parameters.chkFinetune==0
    % rough

    % batch or sequential
    if parameters.batchTrain
        sM = som_batchtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_1,...
        'radius_fin',0,...
        'trainlen',parameters.niterations_1);
    else
        sM = som_seqtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_1,...
        'radius_fin',1,...
        'alpha_ini',parameters.alpha_ini_1,...
        'trainlen',parameters.niterations_1, ...
        'trainlen_type',parameters.IterType,...
        'alpha_type',parameters.Length_funct,...
        'sample_order', parameters.Order,...
        parameters.IterType);
    end
end

% finetune
```

```
    if parameters.batchTrain
        sM = som_batchtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_2,...
        'radius_fin',0,...
        'trainlen',parameters.niterations_2);
    else
        sM = som_seqtrain(sM,sD,...
        'radius_ini',parameters.radius_ini_2,...
        'radius_fin',1,...
        'alpha_ini',parameters.alpha_ini_2,...
        'trainlen',parameters.niterations_2, ...
        'trainlen_type',parameters.IterType,...
        'alpha_type',parameters.Length_funct,...
        'sample_order', parameters.Order,...
        parameters.IterType);
    end

    %calculate Errors
    [sM.topol.qe,sM.topol.te] = som_quality(sM, sD);

    % stand SOM
    sM.method='som';
    sM.name=parameters.SOMname;
    sM.topol.Obs=parameters.Obs;

    % Finalize, by instering the View into the main structure
    if isfield(gss,'Som')
        numSom = length(gss.Som);
        somIndex = numSom+1;
        gss.Som(somIndex) = sM;
    else
        gss.Som(1)=sM;
        somIndex=1;
    end;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildViewUmat(somIndex)
    % geosom_BuildViewUmat
    % NAME: geosom_BuildViewUmat
    % OBJECTIVE: Build a View containing the UMAT relative to a certain SOM
    %
    % INPUT PARAMETERS:somIndex
    % OUTPUT PARAMETERS:
    %
    % COMMENTS:
    %       This is the basic fuction used to import data into the GeoSOM
    %       programs.
    %
    % V.0.1 2009/07/07 By R.Henriques

    global gss

    M=gss.Som(somIndex);
    sU=som_umat(M);
    DataUsed=gss.Som(somIndex).topol.DataUsed;

    %check data normalization
    if strcmpi(M.topol.Normalization,'none')
        % version 01 bmus=som_bmus(gss.Som(somIndex),gss.NumData);
        bmus=som_bmus(gss.Som(somIndex),gss.NumData(DataUsed,:));
        gss.Som(somIndex)
    else
        sD=som_data_struct(gss.NumData(DataUsed,:));
        sD=som_normalize(sD,M.topol.Normalization);
        bmus=som_bmus(gss.Som(somIndex),sD);
    end
```

```
% Generate a geostruct2 with the info of a UMAT
[ maxy maxx ] = size(sU);
tmp=cell(maxx*maxy,1);
for i=1:maxx*maxy;tmp{i}='Polygon';end;
gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
            'Value',0);
[numData numField]=size(gss.NumData(DataUsed,:));
tmpViewIndexData=zeros(numData,1);

% drawing sequence
% 1   4   7
% 2   5   8
% 3   6   9

if strcmp(M.topol.lattice,'rect')==1

    for j=1:maxx % i and j are indexes of the UMAT
        for i=1:maxy
            k=(j-1)*maxy+i;
            %k=(i-1)*maxx+j;         % linear index into GS2
            xCur=j;
            yCur=i*-1;
            gs2(k).X = [ xCur-1 xCur xCur xCur-1 NaN ];
            gs2(k).Y = [ yCur-1 yCur-1 yCur yCur NaN ];
            gs2(k).Value = sU(i,j);
            gs2(k).BoundingBox = [xCur-1 yCur-1; xCur yCur];
            if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                si=ceil(i/2);sj=ceil(j/2);
                neuron = som_sub2ind(M.topol.msize,[si sj]);
                gs2(k).SomIndex=neuron;
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
    end;

elseif strcmp(M.topol.lattice,'hexa')==1
    %          .
    %      .   |   . C
    %   .    A |      .
    %  --------|--------
    %                B
    %_____________________________________
    % Point X Coordinate  Y Coordinate
    % 1           0               A + C
    % 2           0               A
    % 3           B               0
    % 4           2 * B           A
    % 5           2 * B           A + C
    % 6           B               2 * C

    % A=1/2*C
    % B=sin(60)*C
    % C=C

    C=1;
    A=C*.5;
    B=sind(60)*C;

    column=[1 2 3 2];
    column=repmat(column,1,ceil(maxy/4));

    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            % chech if it is the 1st 2nd or third row
            if column(i)==3      % 3rd
                xCur=(j-1)*(2*B)+(2*B);
            elseif column(i)==2  % 2nd
                xCur=(j-1)*(2*B)+(B);
```

```
                elseif column(i)==1  % 1st
                    xCur=(j-1)*(2*B);
                end
                yCur=-(i-1)*(A+C);

                k=(j-1)*maxy+i;
                %k=(i-1)*maxx+j;         % linear index into GS2
                gs2(k).X = [xCur xCur xCur+B xCur+(2*B) xCur+(2*B) xCur+B NaN];
                gs2(k).Y = [yCur+A yCur+A+C yCur+A+C+A yCur+A+C yCur+A yCur NaN];
                gs2(k).Value = sU(i,j);
                gs2(k).BoundingBox = [j-1 i-1; j i];
                if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                    si=ceil(i/2);sj=ceil(j/2);
                    neuron = som_sub2ind(M.topol.msize,[si sj]);
                    gs2(k).SomIndex=neuron;
                    dataIndex=find(bmus==neuron);
                    tmpViewIndexData(dataIndex)=k;
                end;
            end;
        end;
    end

    % Finalize, by instering the View into the main structure
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    gss.View{numView+1} = gs2;
    gss.ViewIndexData(DataUsed,numView+1)=tmpViewIndexData;
    time = fix(clock);
    gss.ViewLabel(numView+1).Name=strcat('UMAT
at:',int2str(time(4)),':',int2str(time(5)),':',int2str(time(6)));
    gss.ViewLabel(numView+1).Type='UMAT';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildViewUmat_Geo(geosom_struct,somIndex)
    % geosom_BuildViewUmat_Geo
    % NAME: geosom_BuildViewUmat_Geo
    % OBJECTIVE: Build a View containing the UMAT relative to a certain GeoSOM
    %
    % INPUT PARAMETERS:
    %   geosom_struct
    %   somIndex
    % OUTPUT PARAMETERS:
    %
    % COMMENTS:
    %       This is the basic fuction used to import data into the GeoSOM
    %       programs.
    % V.0.2 2009/03/26 By R.Henriques

    global gss

    M=geosom_struct.Som(somIndex);
    sU=som_umat_geo(M);
    DataUsed=gss.Som(somIndex).topol.DataUsed;

    %check data normalization
    if strcmpi(M.topol.Normalization,'none')
        bmus=som_bmus_geo(geosom_struct.Som(somIndex),geosom_struct.NumData,M.topol.k);
    else
        sD=som_data_struct(geosom_struct.NumData);
        sD=som_normalize(sD,M.topol.Normalization);
        bmus=som_bmus_geo(geosom_struct.Som(somIndex),sD,M.topol.k);
```

```
end

% Generate a geostruct2 with the info of a UMAT
[ maxy maxx ] = size(sU);
tmp=cell(maxx*maxy,1);
for i=1:maxx*maxy;tmp{i}='Polygon';end;
gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
            'Value',0);
[numData numField]=size(geosom_struct.NumData);
tmpViewIndexData=zeros(numData,1);

if strcmp(M.topol.lattice,'rect')==1

    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            k=(j-1)*maxy+i;
            %k=(i-1)*maxx+j;         % linear index into GS2
            xCur=j;
            yCur=i*-1;
            gs2(k).X = [ xCur-1 xCur   xCur xCur-1 NaN ];
            gs2(k).Y = [ yCur-1 yCur-1 yCur yCur NaN ];
            gs2(k).Value = sU(i,j);
            gs2(k).BoundingBox = [xCur-1 yCur-1; xCur yCur];
            if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                si=ceil(i/2);sj=ceil(j/2);
                neuron = som_sub2ind(M.topol.msize,[si sj]);
                gs2(k).SomIndex=neuron;
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
    end;

elseif strcmp(M.topol.lattice,'hexa')==1
    %          .
    %      .   |   . C
    %   .    A |      .
    %  --------|--------
    %                B
    %_____________________________________
    % Point X Coordinate  Y Coordinate
    % 1           0               A + C
    % 2           0               A
    % 3           B               0
    % 4           2 * B           A
    % 5           2 * B           A + C
    % 6           B               2 * C

    % A=1/2*C
    % B=sin(60)*C
    % C=C

    C=1;
    A=C*.5;
    B=sind(60)*C;

    column=[1 2 3 2];
    column=repmat(column,1,ceil(maxy/4));

    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            % chech if it is the 1st 2nd or third row
            if column(i)==3      % 3rd
                xCur=(j-1)*(2*B)+(2*B);
            elseif column(i)==2  % 2nd
                xCur=(j-1)*(2*B)+(B);
            elseif column(i)==1  % 1st
                xCur=(j-1)*(2*B);
            end
```

```
                yCur=-(i-1)*(A+C);

                k=(j-1)*maxy+i;
                %k=(i-1)*maxx+j;         % linear index into GS2
                gs2(k).X = [xCur xCur xCur+B xCur+(2*B) xCur+(2*B) xCur+B NaN];
                gs2(k).Y = [yCur+A yCur+A+C yCur+A+C+A yCur+A+C yCur+A yCur NaN];
                gs2(k).Value = sU(i,j);
                gs2(k).BoundingBox = [j-1 i-1; j i];
                if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                    si=ceil(i/2);sj=ceil(j/2);
                    neuron = som_sub2ind(M.topol.msize,[si sj]);
                    gs2(k).SomIndex=neuron;
                    dataIndex=find(bmus==neuron);
                    tmpViewIndexData(dataIndex)=k;
                end;
            end;
        end;
    end

        % Finalize, by instering the View into
        % the main structure
    if ~isfield(geosom_struct,'View') % if does not exist create it
        geosom_struct.View =[];
        numView=0;
    else
        numView = length(geosom_struct.View);
    end

    % add view to global gss
    gss.View{numView+1} = gs2;
    gss.ViewIndexData(DataUsed,numView+1)=tmpViewIndexData;
    time = fix(clock);
    gss.ViewLabel(numView+1).Name=strcat('UMAT
at:',int2str(time(4)),':',int2str(time(5)),':',int2str(time(6)));
    gss.ViewLabel(numView+1).Type='UMAT';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_BuildViewUmatHSOM(somIndex,sD)
    % geosom_BuildViewUmatHSOM
    % NAME: geosom_BuildViewUmatHSOM
    % OBJECTIVE: Build a View containing the UMAT relative to a certain SOM
    %            that uses the umat coordinates from other SOMs as input data
    %
    % INPUT PARAMETERS:
    %
    % OUTPUT PARAMETERS:
    %
    % COMMENTS:
    %       This is the basic fuction used to import data into the GeoSOM
    %       programs.
    % V.0.2 2009/07/01 By R.Henriques

    global gss

    M=gss.Som(somIndex);
    sU=som_umat(M);

    bmus=som_bmus(gss.Som(somIndex),sD);

    % Generate a geostruct2 with the info of a UMAT
    [ maxy maxx ] = size(sU);
    tmp=cell(maxx*maxy,1);
    for i=1:maxx*maxy;tmp{i}='Polygon';end;
    gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
                'Value',0);
    [numData numField]=size(sD);
```

```
tmpViewIndexData=zeros(numData,1);

% drawing sequence
% 1   4   7
% 2   5   8
% 3   6   9

if strcmp(M.topol.lattice,'rect')==1

    for j=1:maxx % i and j are indexes of the UMAT
        for i=1:maxy
            k=(j-1)*maxy+i;
            %k=(i-1)*maxx+j;         % linear index into GS2
            xCur=j;
            yCur=i*-1;
            gs2(k).X = [ xCur-1 xCur xCur xCur-1 NaN ];
            gs2(k).Y = [ yCur-1 yCur-1 yCur yCur NaN ];
            gs2(k).Value = sU(i,j);
            gs2(k).BoundingBox = [xCur-1 yCur-1; xCur yCur];
            if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                si=ceil(i/2);sj=ceil(j/2);
                neuron = som_sub2ind(M.topol.msize,[si sj]);
                gs2(k).SomIndex=neuron;
                dataIndex=find(bmus==neuron);
                tmpViewIndexData(dataIndex)=k;
            end;
        end;
    end;

elseif strcmp(M.topol.lattice,'hexa')==1
    %          .
    %      .   |   . C
    %   .    A |      .
    %  --------|--------
    %                B
    %_____________________________________
    % Point X Coordinate  Y Coordinate
    % 1           0               A + C
    % 2           0               A
    % 3           B               0
    % 4           2 * B           A
    % 5           2 * B           A + C
    % 6           B               2 * C

    % A=1/2*C
    % B=sin(60)*C
    % C=C

    C=1;
    A=C*.5;
    B=sind(60)*C;

    column=[1 2 3 2];
    column=repmat(column,1,ceil(maxy/4));

    for j=1:maxx
        for i=1:maxy                    % i and j are indexes of the UMAT
            % chech if it is the 1st 2nd or third row
            if column(i)==3      % 3rd
                xCur=(j-1)*(2*B)+(2*B);
            elseif column(i)==2  % 2nd
                xCur=(j-1)*(2*B)+(B);
            elseif column(i)==1  % 1st
                xCur=(j-1)*(2*B);
            end
            yCur=-(i-1)*(A+C);

            k=(j-1)*maxy+i;
            %k=(i-1)*maxx+j;         % linear index into GS2
```

```
                gs2(k).X = [xCur xCur xCur+B xCur+(2*B) xCur+(2*B) xCur+B NaN];
                gs2(k).Y = [yCur+A yCur+A+C yCur+A+C+A yCur+A+C yCur+A yCur NaN];
                gs2(k).Value = sU(i,j);
                gs2(k).BoundingBox = [j-1 i-1; j i];
                if mod(i,2)==1 && mod(j,2)==1   % if this is a neuron
                    si=ceil(i/2);sj=ceil(j/2);
                    neuron = som_sub2ind(M.topol.msize,[si sj]);
                    gs2(k).SomIndex=neuron;
                    dataIndex=find(bmus==neuron);
                    tmpViewIndexData(dataIndex)=k;
                end;
            end;
        end;
    end

        % Finalize, by instering the View into
        % the main structure
    if ~isfield(gss,'View') % if does not exist create it
        gss.View =[];
        numView=0;
    else
        numView = length(gss.View);
    end

    gss.View{numView+1} = gs2;
    gss.ViewIndexData(gss.Som(somIndex).topol.DataUsed,numView+1)=tmpViewIndexData;
    time = fix(clock);
    gss.ViewLabel(numView+1).Name=strcat('UMAT
at:',int2str(time(4)),':',int2str(time(5)),':',int2str(time(6)));
    gss.ViewLabel(numView+1).Type='UMAT';
    gss.ViewLabel(numView+1).SomOrigin=somIndex;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_struct_itemindex=geosom_ClosestItem(click, geosom_struct, viewindex);
    % geosom_struct=geosom_ReadShapeFile(filename)
    %
    % NAME: geosom_ClosestItem
    %
    % OBJECTIVE: Identify which is the closest item (point, line, polygon) to
    % a mouse click or a rectangle
    %
    % INPUT PARAMETERS:
    %       click - the [x,y] coordinates of the mouse click or the 4
    %       coordinates of the rectangle
    %       geosom_struct - where to search for the items
    %       viewindex - the index of the view in the geosom_struct
    %
    % OUTPUT PARAMETERS:
    %       geosom_struct_itemindex - The index in the geosom_struct of the
    %       closest item.
    %
    % COMMENTS:
    %
    % V.0.1 2007/01/22 By V.Lobo & M.Loureiro
    % V.0.2 2007/01/24 By V.Lobo & M.Loureiro & F.Pires
    % V.0.3 2008/07/14 By R.Henriques & V.Lobo

    tmp=length(geosom_struct.View{viewindex});
    tmp_close=zeros(1,tmp); %will be 1 for the closest, or if click inside polygons
    for i=1:tmp
        tmp_type = geosom_struct.View{viewindex}(1).Geometry;
        if strcmp(tmp_type,'Polygon')
            if size(click,1)==1 %get selection from a point
                if inpolygon(click(:,1), click(:,2),...
                       geosom_struct.View{viewindex}(i).X, ...
                       geosom_struct.View{viewindex}(i).Y);
                   tmp_close(i)=1;
```

```
                end
            else %get selection from a rectangle
                result=inpolygon(geosom_struct.View{viewindex}(i).X,...
                    geosom_struct.View{viewindex}(i).Y,...
                    click(:,1), click(:,2));
                if sum(result)>0
                    tmp_close(i)=1;
                end
            end
        elseif strcmp(tmp_type,'Line')
            if size(click,1)==1 %get selection from a point
                if inpolygon(click(:,1), click(:,2),...
                       geosom_struct.View{viewindex}(i).X, ...
                       geosom_struct.View{viewindex}(i).Y);
                   tmp_close(i)=1;
                end
            else %get selection from a rectangle
                result=inpolygon(geosom_struct.View{viewindex}(i).X,...
                    geosom_struct.View{viewindex}(i).Y,...
                    click(:,1), click(:,2));
                if sum(result)>0
                    tmp_close(i)=1;
                end
            end

        elseif strcmp(tmp_type,'Point')
            if size(click,1)==1 %get selection from a point
                if inpolygon(click(:,1), click(:,2),...
                       geosom_struct.View{viewindex}(i).X, ...
                       geosom_struct.View{viewindex}(i).Y);
                   tmp_close(i)=1;
                end
            else %get selection from a rectangle
                result=inpolygon(geosom_struct.View{viewindex}(i).X,...
                    geosom_struct.View{viewindex}(i).Y,...
                    click(:,1), click(:,2));
                if sum(result)>0
                    tmp_close(i)=1;
                end
            end

        else %not supposed to happen

        end
    end
    geosom_struct_itemindex=find(tmp_close==1);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_CreateBoxPlotsHisto()
    %GEOSOM_CREATEBOXPLOTSHISTO ...
    % Syntax: geosom_CreateBoxPlotsHisto()
    %function to show the form with box plots, histograms and pcp
    %
    %
    % Subfunctions: pushOK_Callback, geosom_ResizeFcn, ExportAxes2NewFig, PercentileCalc
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 22-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    global gss

    scrsz = get(0,'ScreenSize');
    fh=figure('Position',[scrsz(3)/2-400 scrsz(3)/2-400 400 400]);

    % clear default menu and toolbars and add title
```

```
set(fh,'MenuBar','none','toolbar','none','NumberTitle','off','name',...
    'Create boxplots & Histo','Resize','on');
handles.CreateBoxPlots=fh;

% add export capabilities
mn = uimenu('Label','Export');
uimenu(mn,'Label','To new figure','Callback',...
    {@ExportAxes2NewFig});

% add controls
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% replace the character _ by space
Comp = gss.NumDataLabel';
for i=1:length(Comp)
    Comp{i}=strrep(Comp{i}, '_', ' ');
end

% components
h = uicontrol('Style', 'listbox',...
       'String', Comp,...
       'tag', 'Listbox',...
       'Position', [100 10 150 100],...
       'HorizontalAlignment','left','Max',3,'Min',1);
handles.Listbox=h;

% type of representation
h = uicontrol('Style', 'text',...
       'String', 'Type',...
       'tag', 'textNormalize',...
       'Position', [10 100 80 15],...
       'HorizontalAlignment','left');
handles.textRepresentation=h;
h = uicontrol('Style', 'popup',...
       'String', {'boxplot';'histo'},...
       'tag', 'popupType',...
       'Position', [10 80 80 20]);
handles.popupRepresentation=h;

% normalize box
h = uicontrol('Style', 'text',...
       'String', 'Normalize',...
       'tag', 'textNormalize',...
       'Position', [10 60 80 15],...
       'HorizontalAlignment','left');
handles.textNormalize=h;
h = uicontrol('Style', 'popup',...
       'String', {'none';'var';'range';'log';'logistic';'histD';'histC'},...
       'tag', 'popupNormalize',...
       'Position', [10 40 80 20]);
handles.popupNormalize=h;

h = axes('tag', 'Boxplot','AmbientLightColor','white',...
       'units','pixels','Position', [20 150 360 230]);
handles.boxplot=h;

%checkbox : show slection values
h = uicontrol('Style', 'checkbox',...
       'String', 'Show selection',...
       'tag', 'chkSel',...
       'Position', [260 105 140 15]);
handles.chkShowSel=h;

%checkbox : link selected data items with line
h = uicontrol('Style', 'checkbox',...
       'String', 'Link data items',...
       'tag', 'chkSel',...
       'Position', [270 90 130 15]);
handles.chkLink=h;
```

```
    %checkbox : mean values
    h = uicontrol('Style', 'checkbox',...
           'String', 'Show mean values',...
           'tag', 'chkSel',...
           'Position', [270 75 130 15]);
    handles.chkMean=h;

    %checkbox : Sel Histo
    h = uicontrol('Style', 'checkbox',...
           'String', 'Show Sel. Histo',...
           'tag', 'chkSel',...
           'Position', [270 60 130 15]);
    handles.chkSelHisto=h;

    %checkbox : Sort variables
    h = uicontrol('Style', 'checkbox',...
           'String', 'Sort Dif2Mean',...
           'tag', 'chkSort',...
           'Position', [270 45 130 15]);
    handles.chkSort=h;

    %checkbox : show multiple sel.
    h = uicontrol('Style', 'checkbox',...
           'String', 'Show Multiple Sel.',...
           'tag', 'chkSel',...
           'Position', [260 25 140 15]);
    handles.chkMultiple=h;

    %checkbox : show multiple sel.Q
    h = uicontrol('Style', 'checkbox',...
           'String', 'Show 1Q. and 3Q.',...
           'tag', 'chkSel',...
           'Position', [270 10 130 15]);
    handles.chkMultipleQuart=h;

    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'OK',...
           'tag', 'pushOK',...
           'Position', [10 10 50 15]);
    handles.pushOK=h;

    %callbacks
    set(handles.pushOK,'Callback',{@pushOK_Callback handles});
    set(handles.CreateBoxPlots,'ResizeFcn',{@geosom_ResizeFcn handles});
return

function pushOK_Callback(h,evd,handles)
 %show in handles.boxplot the boxplot for the sel variables
 global gss
 global mainHandles

    % get components
    % comp = get(handles.TableComponents, 'userdata');
    % SelectedComponents=find(comp.isChecked==1);
    list_entries = get(handles.Listbox,'String');
    SelectedComponents = get(handles.Listbox,'Value');
    if isempty(SelectedComponents)
        errordlg('You must select at least one variable',...
                 'Incorrect Selection','modal')
        return
    end

    Norm=get(handles.popupNormalize,'string');
    Norm=cell2mat(Norm(get(handles.popupNormalize,'value'),:));

    if strcmpi(Norm,'none')
        data=gss.NumData(:,SelectedComponents);
```

```
    else
        data=som_normalize(gss.NumData(:,SelectedComponents),Norm);
    end
    labels={(list_entries(SelectedComponents)')};

    if get(handles.chkSort,'value')  % sort variables according to differences
        % if there is any selection
        % checked
        if sum(gss.SelectedIndex)>0
            indexes=find(gss.SelectedIndex==1);
    %        Difmean=[abs((mean(data)-mean(data(indexes,:))))'
(1:length(SelectedComponents))'];
            Difmean=[(mean(data(indexes,:))-mean(data))'
(1:length(SelectedComponents))'];
            Difmean=flipud(sortrows(Difmean,1));

            data=data(:,Difmean(:,2)');
            labels={(list_entries(SelectedComponents(Difmean(:,2)'))')};
        end
    end

    %boxplot or histo?
    type=get(handles.popupRepresentation,'string');
    type=cell2mat(type(get(handles.popupRepresentation,'value'),:));

    if strcmpi(type,'boxplot')
        %boxplot(handles.boxplot,data,'labels',labels{1})
        boxplot(handles.boxplot,data,'labels',labels{1},'colors',[.8 .8
.8],'symbol','w.');

        % if there is any selection plot in the box the selection and option is
        % checked
        if sum(gss.SelectedIndex)>0 && get(handles.chkShowSel,'value')
            indexes=find(gss.SelectedIndex==1);

            if get(handles.chkMean,'value')  % show only Mean values?
                meanSelData=mean(data(indexes,:));

                if get(handles.chkLink,'value')&& size(data,2)>1  % show items with link
                    x=(1:size(data,2));
                    y=meanSelData;
                    line(x,y,'color',mainHandles.prefs.selcolor,'LineWidth',2);
                else % show each data item with a bar in each variable
                    for i=1:length(SelectedComponents)
                        line([i-.4 i-.1],[meanSelData(i) meanSelData(i)],'Color',...
                            mainHandles.prefs.selcolor);
                    end
                end
            else % show all data items
                if get(handles.chkLink,'value')&& size(data,2)>1  % show items with link
                    for s=1:length(indexes)
                        x=(1:size(data,2));
                        y=data(indexes(s),:);
                        line(x,y,'color',mainHandles.prefs.selcolor,'LineWidth',1);
                    end
                else % show each data item with a bar in each variable
                    for i=1:length(SelectedComponents)
                        for s=1:length(indexes)
                            line([i-.4 i-.1],[data(indexes(s),i)
data(indexes(s),i)],'Color',...
                                mainHandles.prefs.selcolor);
                        end
                    end
                end
            end
            if  get(handles.chkSelHisto,'value')
               hold on;
```

```
%boxplot(handles.boxplot,data(indexes,:),'labels',labels{1},'colors',mainHandles.prefs.se
lcolor,'plotstyle','compact')
                boxplot(handles.boxplot,data(indexes,:),'labels',labels{1},'colors',[.1
.1 .1],'plotstyle','compact')
                hold off;

            end
        end
        if  get(handles.chkMultiple,'value')
            for s=unique(gss.ClusterIndex)'
                if s>0
                    meanData=mean(data(find(gss.ClusterIndex==s),:));
                    QSelData=PercentileCalc(data(find(gss.ClusterIndex==s),:));

                    x=(1:size(data,2));
                    y=meanData;
                    line(x,y,'color',mainHandles.ClusterColors(s,:),'LineWidth',2);

                    if  get(handles.chkMultipleQuart,'value')
                        y=[QSelData(1,:) fliplr(QSelData(2,:))];
                        x=[1:size(QSelData,2) fliplr(1:size(QSelData,2))];
                        patch(x,y,mainHandles.ClusterColors(s,:), ...
                        'EdgeColor',mainHandles.ClusterColors(s,:),...
                        'UserData','Sel', 'FaceAlpha', .3);
                    end
                end
            end
        end

        geosom_ResizeFcn(1,1,handles);
        xticklabel_rotate([],90,labels{1});

    elseif strcmpi(type,'histo')  % histo
        %hist(handles.boxplot,data)

        [n,xout]=hist(handles.boxplot,data);
        bar(xout,n);


        if size(xout,1)>1
            xout=xout';
        end
        % if there is any selection plot in the box the selection and option is
        % checked
        if sum(gss.SelectedIndex)>0 && get(handles.chkShowSel,'value')
             indexes=find(gss.SelectedIndex==1);
             tmpdata=data(indexes,:);

             [n2,xout2]=hist(tmpdata,xout);
             hold on;
             bar(xout2,n2,'r')
             hold off;
             legend (labels{1},'Selection');
         else
             legend (labels{1});
         end
         geosom_ResizeFcn(1,1,handles);

    end
return

function geosom_ResizeFcn(h,evd,handles)
    if isempty(handles)==1
        return
    end
    Position = getpixelposition(handles.CreateBoxPlots);
```

```
    set(handles.boxplot,'units','pixels','Position',[50 150 Position(3)-70 Position(4)-
160]);
return

function ExportAxes2NewFig(hObject,eventdata)
    %get axes and open in new figure for export

    h=gca;
    hNewAxes=figure;
    set(hNewAxes,'color',[1 1 1]);
    copyobj(h, hNewAxes);

return

function QSelData=PercentileCalc(data)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Percentile Calculation Example
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    Nx = size(data,1);
    nVar= size(data,2);

    QSelData=zeros(2,nVar);
    % compute mean
    mx = mean(data);
    % compute the standard deviation
    sigma = std(data);
    % compute the median
    medianx = median(data);
    % STEP 1 - rank the data
    %y = sort(data);
    for v=1:nVar
        % compute 25th percentile (first quartile)
        if medianx(v)>min(data(:,v))
            QSelData(1,v)=median(data(find(data(:,v)< medianx(v)),v));
        else
            QSelData(2,v)=min(data(:,v));
        end
        if medianx(v)<max(data(:,v))
            QSelData(2,v)=median(data(find(data(:,v)> medianx(v)),v));
        else
            QSelData(2,v)=max(data(:,v));
        end
    end
    QSelData(isnan(QSelData))=0;
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_createViewToolbar(this)
%CREATETOOLBAR Create toolbar for View

selectionIcon   = makeToolbarIconFromPNG('tool_select.png');
zoomInIcon      = makeToolbarIconFromPNG('view_zoom_in.png');
zoomOutIcon     = makeToolbarIconFromPNG('view_zoom_out.png');
arrowIcon       = makeToolbarIconFromPNG('tool_arrow.png');
lineIcon        = makeToolbarIconFromPNG('tool_line.png');
textIcon        = makeToolbarIconFromPNG('tool_text.png');
fitToWindowIcon = makeToolbarIconFromPNG('view_fit_to_window.png');
datatipIcon     = makeToolbarIconFromPNG('tool_datatip.png');
panIcon         = makeToolbarIconFromPNG('tool_hand.png');
prevViewIcon    = makeToolbarIconFromPNG('view_prev.png');
selectAreaIcon  = makeToolbarIconFromPNG('tool_marquee.png');
infoToolIcon    = makeToolbarIconFromPNG('info.png');

toolbar = uitoolbar('Parent',this);
printTool = uitoolfactory(toolbar,'Standard.PrintFigure');

set(printTool,'TooltipString','Print figure');
```

```
set(printTool,'ClickedCallback',{@localPrint, this});

selectionTool = uitoggletool(toolbar,...
                             'Cdata',selectionIcon,...
                             'TooltipString','Select annotations',...
                             'Tag','select annotations',...
                             'State','on','Enable','on',...
                             'ClickedCallback',@ClickedCallback,...
                             'OnCallback',{@initEditState this},...
                             'OffCallback',{@endEditState this});

rectTool = createToolbarToggleItem(toolbar,selectAreaIcon,...
                                   {@initSelectAreaState this},...
                                   'Select area',{@endSelectAreaState this});


zoomInTool = createToolbarToggleItem(toolbar,zoomInIcon,...
                                     {@initZoomInState this},...
                                     'Zoom in',{@endZoomInState this});

zoomOutTool = createToolbarToggleItem(toolbar,zoomOutIcon,...
                                     {@initZoomOutState this},...
                                     'Zoom out',{@endZoomOutState this});

panTool = createToolbarToggleItem(toolbar,panIcon,...
                                  {@initPanState this},'Pan tool',...
                                  {@endPanState this});

insertTextItem = createToolbarToggleItem(toolbar,textIcon,...
                                       {@initInsertTextState this},...
                                       'Insert text',...
                                       {@endInsertTextState this});

dataTipTool = createToolbarToggleItem(toolbar,datatipIcon,...
                                      {@initDatatipState this},...
                                      'Datatip tool',{@endDatatipState this});

infoTool = createToolbarToggleItem(toolbar,infoToolIcon,...
                                      {@initInfoToolState this},...
                                      'Info tool',{@endInfoToolState this});

fitToWindowItem = createToolbarPushItem(toolbar,fitToWindowIcon,...
                                        {@doFitToWindow this},'Fit to window');

set(selectionTool, 'Separator','on');
set(zoomInTool,    'Separator','on');
set(insertTextItem,'Separator','on');
set(dataTipTool,   'Separator','on');
set(backTool,      'Separator','on');


%----------Callbacks----------%

% Start Tools / Initialize States

% Functions initializing a state should follow this template:
%
%  function initXYZState(hSrc,event,viewer,...)
%  delete(viewer.State);
%  viewer.setDefaultState;
%  viewer.State = PKG.XYZState(...);
%

function localPrint(hSrc,event,this)
this.printMap;

function initDatatipState(hSrc,event,viewer)
changeStateToDefault(viewer);
showUsage = true;
```

```
if ispref('MathWorks_MapViewer','ShowDatatipUsage')
  if ~getpref('MathWorks_MapViewer','ShowDatatipUsage')
    showUsage = false;
  end
end
if ~viewer.Preferences.ShowDatatipUsage
  showUsage = false;
end

if strcmpi(viewer.ActiveLayerName,'none') ||...
      isempty(viewer.ActiveLayerName)
  viewer.dataTipUsage('NoActiveLayer','Datatip Tool');
  set(hSrc,'State','off');
  viewer.setDefaultState;
  return
  %changeStateToDefault(viewer);
end
if showUsage
  viewer.dataTipUsage('Default');
end

viewer.State = MapViewer.DataTipState(viewer);
checkToolMenu(hSrc,'on');

function initSelectAreaState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.SelectAreaState(viewer);
checkToolMenu(hSrc,'on');

function initZoomInState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.ZoomInState(viewer);
checkToolMenu(hSrc,'on');

function initZoomOutState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.ZoomOutState(viewer);
checkToolMenu(hSrc,'on');

function initPanState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.PanState(viewer);
checkToolMenu(hSrc,'on');

function initEditState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.EditState(viewer);
checkToolMenu(hSrc,'on');

function initInsertTextState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.InsertTextState(viewer);
checkToolMenu(hSrc,'on');

function initInsertArrowState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.InsertArrowState(viewer);
checkToolMenu(hSrc,'on');

function initInsertLineState(hSrc,event,viewer)
changeStateToDefault(viewer);
viewer.State = MapViewer.InsertLineState(viewer);
checkToolMenu(hSrc,'on');

function initInfoToolState(hSrc,event,viewer)
changeStateToDefault(viewer);
if strcmpi(viewer.ActiveLayerName,'none') ||...
      isempty(viewer.ActiveLayerName)
  viewer.dataTipUsage('NoActiveLayer','Info Tool');
```

```
  set(hSrc,'State','off');
  viewer.setDefaultState;
  return
  %changeStateToDefault(viewer);
end
viewer.State = MapViewer.InfoToolState(viewer);
checkToolMenu(hSrc,'on');


% End State
% The "Off Callback"  functions are called in the following order:
%
% Situation 1 - Deselect a selected item
%   The callback is executed immediately.
%
% Situation 2 - Select the item. Then select a different item (mutually exclusive)
%   New item's "On Callback" is executed (state is switched).
%   Old item's "Off Callback" is executed.
%
% Because of Situation 2, the current state of the viewer must be queried before
% deleting it.  If the current state is the same as the state associated with
% the OffCallback, then it is Situation 1, and the current state should be
% deleted.  If it is Situation 2, then the state has been switched to another
% (non-default) state and the other state should handle deleting the current
% state.
%
% Functions ending a state should follow this template:
%
%  function endXYZState(hSrc,event,viewer,...)
%  if strcmp(class(viewer.State),'PGH.XYZState');
%    delete(viewer.State);
%    viewer.setDefaultState;
%  end
%

function endDatatipState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.DataTipState');
  changeStateToDefault(viewer);
end

function endSelectAreaState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.SelectAreaState');
  changeStateToDefault(viewer);
end

function endEditState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.EditState')
  changeStateToDefault(viewer);
end

function endInsertTextState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.InsertTextState');
  changeStateToDefault(viewer);
end

function endInsertArrowState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.InsertArrowState');
  changeStateToDefault(viewer);
end

function endInsertLineState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.InsertLineState');
  changeStateToDefault(viewer);
end

function endPanState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.PanState')
  changeStateToDefault(viewer);
end
```

```
function endZoomInState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.ZoomInState')
  changeStateToDefault(viewer);
end

function endZoomOutState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.ZoomOutState')
  changeStateToDefault(viewer);
end

function endInfoToolState(hSrc,event,viewer)
if strcmp(class(viewer.State),'MapViewer.InfoToolState')
  changeStateToDefault(viewer);
end

%-------------------------------------------------------------------------

function doFitToWindow(hSrc,event,viewer)
mapgate('doFitToWindow', hSrc, event, viewer);

%-------------------------------------------------------------------------

function doBackToPreviousView(hSrc,event,viewer)
viewCount = size(viewer.PreviousViews,1);
ind = viewer.ViewIndex;
ind = mod(ind-1,viewCount+1);
if ind < 1, ind = viewCount;end
lastLims = [viewer.PreviousViews(ind,1:2);...
            viewer.PreviousViews(ind,3:4)]';
% make view index -1 so as to skip store view
skipStoreView = -1;
viewer.ViewIndex = skipStoreView;
viewer.setMapLimits(lastLims);
viewer.Axis.refitAxisLimits;
viewer.ViewIndex = ind;

%-------------------------------------------------------------------------

% Toggle items that call ClickedCallback will be mutually exclusive.

function ClickedCallback(hSrc,event)
fig = ancestor(hSrc,'Figure');
uiresume(fig);
hiddenHandles = get(0,'ShowHiddenHandles');
set(0,'ShowHiddenHandles','on');
tools = findall(fig,'type','uitoggletool');
set(0,'ShowHiddenHandles',hiddenHandles);
set(tools(tools ~= hSrc),'State','off');
if (strcmp(get(hSrc,'State'),'off'))
  selAnnot = findobj(tools,'tag','select annotations');
  set(selAnnot,'State','on');
end

%--------------------------- Helper Functions --------------------------

function item = createToolbarPushItem(toolbar,icon,callback,tooltip)
item = uipushtool(toolbar,...
                  'Cdata',icon,...
                  'TooltipString',tooltip,...
                  'Tag',lower(tooltip),...
                  'ClickedCallback',callback);

%-------------------------------------------------------------------------

function item = createToolbarToggleItem(toolbar,icon,callback,...
                                        tooltip,offcallback)
item = uitoggletool(toolbar,...
                    'Cdata',icon,...
```

```
                    'TooltipString',tooltip,...
                    'Tag',lower(tooltip),...
                    'ClickedCallback',@ClickedCallback,...
                    'OnCallback',callback,...
                    'OffCallback',offcallback);

%--------------------------------------------------------------------------

function icon = makeToolbarIconFromPNG(imagefilename)

filename = fullfile(matlabroot,'toolbox','map','icons',imagefilename);

% Icon's background color is [0 1 1]
[icon,map] = imread(filename);
idx = 0;
for i=1:size(map,1)
  if all(map(i,:) == [0 1 1])
    idx = i;
    break;
  end
end
mask = icon==(idx-1); % Zero based.
[r,c] = find(mask);
icon = ind2rgb(icon,map);
for i=1:length(r)
  icon(r(i),c(i),:) = NaN;
end

%--------------------------------------------------------------------------

function changeStateToDefault(viewer)
if isa(viewer.State,'MapViewer.DataTipState')
  viewer.PreviousDataTipState = viewer.State;
end

viewer.setDefaultState;

%--------------------------------------------------------------------------

function checkToolMenu(toolbarItem, checked)
fig = ancestor(toolbarItem,'Figure');
toolmenu = findobj(fig,'type','uimenu','tag','tools menu');

set(get(toolmenu,'Children'),'Checked','off');
thisToolMenu = findobj(toolmenu,'Tag',[get(toolbarItem,'Tag'), ' menu']);

if strcmp(checked,'off')
  selectAnnotMenuItem = findobj(toolmenu,'Tag','select annotations menu');
  set(selectAnnotMenuItem,'Checked','on');
else
  if ~isempty(thisToolMenu)
    set(thisToolMenu,'Checked','on');
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function drawCluster(viewNum)
    %DRAWCLUSTER ...
    % Syntax: drawCluster(viewNum)
    % function to draw clusters on top of the U-matrix of viewNum
    %
    % INPUT PARAMETERS:
    %       viewNum - Number of the "View" we want to visualize
    % OUTPUT PARAMETERS:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 22-2009
    % Version 1.0
```

```
% Copyright (c) by the Ga2 programming team. 2009

global gss
global mainHandles

f=figure;                                % Create a figure
handles.drawClusters=f;
set(f,'position',[200 200 400 400]);

% add export capabilities
mn = uimenu('Label','Export');
uimenu(mn,'Label','To new figure','Callback',...
    {@ExportAxes2NewFig});

% add new toolbar
set(f,'Toolbar','none');
hToolbar = uitoolbar(...
                'Parent',f, ...
                'HandleVisibility','callback');

% Create the button group.
hB      =  uitoggletool(...
               'Parent',hToolbar,...
               'tag','zoomin',...
               'TooltipString','Zoom in',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\view_zoom_in.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback', {@ButtonFunction hToolbar viewNum});
hB      =  uipushtool(...
               'Parent',hToolbar,...
               'tag','zoomout',...
               'TooltipString','Zoom out',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\view_zoom_out.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback', 'zoom (0.5)');
hB      =  uipushtool(...
               'Parent',hToolbar,...
               'tag','fullextent',...
               'TooltipString','Full extent',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\webicon.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback','zoom out');
% add clusters selection
hB      =  uitoggletool(...
           'Parent',hToolbar,...
           'tag','selectclusters',...
           'TooltipString','Select Cluster',...
            'CData',iconRead(fullfile(pwd,...
               'icons\selectAreaColor.gif')),...
           'HandleVisibility','callback', ...
           'ClickedCallback',{@ButtonFunction hToolbar viewNum});

%panel 1
hp = uipanel('Title','Clusters',...
            'units','pixels',...
            'Position',[10 5 220 120]);
handles.panel1=hp;

% add list
h = uicontrol('Style', 'listbox',...
       'parent',hp,...
       'String', '',...
       'tag', 'Listbox',...
       'Position', [5 20 100 80],...
       'HorizontalAlignment','left','Max',3,'Min',1);
handles.Listbox=h;
```

```
%push buttons
h = uicontrol('Style', 'pushbutton',...
       'parent',hp,...
       'String', 'Remove',...
       'TooltipString','Remove selected cluster',...
       'tag', 'pushRemove',...
       'Position', [5 3 45 15]);
handles.pushRemove=h;
% 1 cluster neuron
h = uicontrol('Style', 'pushbutton',...
       'parent',hp,...
       'String', '1:1',...
       'TooltipString','Create 1 cluster per unit',...
       'Position', [55 3 50 15]);
handles.push1clusterunit=h;

% add a chkbox for presenting the hits
h = uicontrol('Style', 'checkbox',...
       'parent',hp,...
       'String', 'Show Hits',...
       'Position', [110 85 70 20]);
handles.checkHits=h;

% add a chkbox for presenting the hierarchical clust
h = uicontrol('Style', 'checkbox',...
       'parent',hp,...
       'String', 'Hier. Clusters',...
       'Position', [110 65 85 20]);
handles.checkHclust=h;
%textbox with number of clusters
h = uicontrol('Style', 'edit',...
       'parent',hp,...
       'String', '10',...
       'Position', [125 45 25 20]);
handles.editNclusters=h;
%label
h = uicontrol('Style', 'text',...
       'parent',hp,...
       'String', 'clusters',...
       'Position', [150 40 45 20]);
%only bmus
h = uicontrol('Style', 'checkbox',...
   'parent',hp,...
   'String', 'Only BMU',...
   'Position', [125 25 70 15]);
handles.checkBMU=h;

%show cluster labels on map
h = uicontrol('Style', 'checkbox',...
   'parent',hp,...
   'String', 'Show labels',...
   'Position', [110 5 80 15]);
handles.showClusterLabels=h;

%panel 2
hp = uipanel('Title','Output',...
            'units','pixels',...
            'Position',[220 5 115 120]);
handles.panel2=hp;

h = uicontrol('Style', 'pushbutton',...
       'parent',hp,...
       'String', 'Show Sel. Cluster',...
       'tag', 'pushSelCluster',...
       'Position', [5 75 100 20]);
handles.pushSelCluster=h;

h = uicontrol('Style', 'pushbutton',...
```

```
        'parent',hp,...
        'String', 'Show All Clusters',...
        'tag', 'pushSelAllClusters',...
        'Position', [5 45 100 20]);
    handles.pushSelAllClusters=h;

    h = uicontrol('Style', 'pushbutton',...
        'parent',hp,...
        'String', 'Export Cluster(s)',...
        'tag', 'pushExportClusters',...
        'Position', [5 15 100 20]);
    handles.pushExportClusters=h;

    %panel 3
    %flood
    hp = uipanel('Title','z-level',...
                'units','pixels',...
                'Position',[330 5 50 120]);
    handles.panel3=hp;

    h = uicontrol('Style', 'slider',...
        'parent',hp,...
        'Min',0,'Max',1,...
        'Position', [25 5 20 100]);
    handles.slider=h;

    h = uicontrol('Style', 'checkbox',...
        'parent',hp,...
        'String', '',...
        'Position', [5 85 15 15]);
    handles.chkSlide=h;

    % only for UMATS
    if strcmp(gss.ViewLabel(viewNum).Type,'UMAT')

        numColors=length(colormap);
        gss.View{viewNum}=geosom_ViewUmatCreateIValue(...
                        gss.View{viewNum},numColors );

        %check outline color
        if mainHandles.prefs.cmapOutline
            edgeColor=mainHandles.prefs.cmapOutlineColor;
        else
            edgeColor=mainHandles.prefs.cmap;
        end

        regra=makesymbolspec('Polygon',{'IValue',[1 numColors],...
                                        'FaceColor', mainHandles.prefs.cmap,...
                                        'EdgeColor', edgeColor});

        h=mapshow(gss.View{viewNum},'SymbolSpec',regra);

        set(gca,'units','pixels');
        handles.map=gca;
        Position = getpixelposition(handles.drawClusters);
        % map
        setpixelposition(handles.map,[10 150 Position(3)-20 Position(4)-150]);

        %Remove labels
        set(gca,'XTickLabel',{},'yTickLabel',{})
    end

    %SET CALLBACKS
    set(handles.drawClusters,'ResizeFcn',{@ResizeFcn handles});
    set(handles.Listbox,'callback',{@Listbox_Callback handles});
    set(handles.pushRemove,'callback',{@pushRemove_Callback handles});
    set(handles.push1clusterunit,'callback',{@push1clusterunit_Callback handles
viewNum});
```

```
    set(handles.pushExportClusters,'callback',{@pushExportClusters_Callback handles
viewNum});
    set(handles.checkHclust,'callback',{@showHclust handles viewNum});
    set(handles.checkHits,'callback',{@showHits viewNum 'umat'});
    set(handles.pushSelCluster,'callback',{@pushSelCluster_Callback handles viewNum});
    set(handles.pushSelAllClusters,'callback',{@pushSelAllClusters_Callback handles
viewNum});
    set(handles.pushExportClusters,'callback',{@pushExportClusters_Callback handles
viewNum});
    set(handles.slider,'callback',{@slider_Callback handles viewNum});
    set(handles.chkSlide,'callback',{@chkSlide_Callback handles viewNum});

    %set(gcf,'windowbuttondownfcn',{@FigMouseClick_Callback handles viewNum});
    set(gcf,'CloseRequestFcn',{@ViewClose,viewNum});
    set(gcf,'windowbuttondownfcn',{@FigMouseClick_Callback handles viewNum});

    %load existing clusters
    loadClusters(handles,viewNum)
return

function loadClusters(handles,viewNum)
    %load clusters from gss.ViewClusters
    global gss
    global mainHandles

    somId=gss.ViewLabel(viewNum).SomOrigin;

    if isfield(gss, 'ViewClusters')
        if size(gss.ViewClusters,2)>=somId
            if isempty(gss.ViewClusters{somId})==0
                for i=1:size(gss.ViewClusters{somId},2)
                    Coords=gss.ViewClusters{somId}{i};
                    %plot line
                    line(Coords(:,1),Coords(:,2),...
                    'UserData','cluster','color',mainHandles.prefs.ClusterColor,...
                    'linewidth',2);
                    items=get(handles.Listbox,'string');
                    if isempty(items)
                        items=[strcat('Cluster',num2str(size(items,1)+1))];
                    else
                        items=cellstr(items);
                        items=[items;strcat('Cluster',num2str(size(items,1)+1))];
                    end
                    set(handles.Listbox,'string',items);
                end
            end
        end
    end
return

function ResizeFcn(h,evd,handles)
    if isempty(handles)==1
        return
    end

    % new figure position -> x y width height
    Position = getpixelposition(handles.drawClusters);

    % panel1
    setpixelposition(handles.panel1,[10 5 200 120]);

    % panel2
    setpixelposition(handles.panel2,[220 5 115 120]);

    % panel3
    setpixelposition(handles.panel3,[330 5 50 120]);

    % map
    setpixelposition(handles.map,[10 150 Position(3)-20 Position(4)-150]);
```

```
return

function pushSelCluster_Callback (hObject,eventdata,handles,viewNumber)
    %make selection on data items using the selected cluster(s)

    global gss

    %get selection
    SelectedComponents = get(handles.Listbox,'Value');
    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    p1=p(a);
    pSel=p1(SelectedComponents);

    %clean selection
    gss.SelectedIndex(:)=0;

    %get data items
    for i=pSel'   % for each sel cluster
        indice=geosom_ClosestItem([get(i,'X'); get(i,'Y')]',gss,viewNumber);
        for j=indice
            sel=find(gss.ViewIndexData(:,viewNumber)==j);
            if isempty(sel)~=1
                gss.SelectedIndex(sel)=1;
            end
        end
    end
    geosom_MakeSelectionAllOpenViews;
    geosom_TableDraw();
return

function pushSelAllClusters_Callback (hObject,eventdata,handles,viewNumber)
    %make selection on data items using all cluster(s).
    %for this we have to use different colors for each cluster.
    global gss
    global mainHandles

    % Create the colors
    Items=get(handles.Listbox,'String');
    numbClusters=size(Items,1);

    %%%%%%%%%%%%%% Alternative color coding %%%%%%%%%%%%%
    % define the colors according to the colormap and the position of
    % the cluster in the umatrix
    % we are using a SOM to project the 3D color data to 2D

    %train the SOM
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    sD=som_data_struct(mainHandles.prefs.clusterscmap);
    %get SOM size
    sM=som_map_struct(3,...
        'msize',gss.Som(gss.ViewLabel(viewNumber).SomOrigin).topol.msize,...
        'lattice','hexa',...
        'shape','sheet',...
        'neigh','gaussian');

    sM = som_batchtrain(sM,sD,...

'radius_ini',floor(sqrt(prod(gss.Som(gss.ViewLabel(viewNumber).SomOrigin).topol.msize))),
...
    'radius_fin',0,...
    'trainlen',100);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % for each defined cluster get the closest unit in the output space
    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    p1=p(a);
```

```
    counter=0;
    clustCentr=zeros(length(p1),2);
    for i=p1'   % for each cluster
        counter=counter+1;
        clustCentr(counter,:)=mean([get(i,'X'); get(i,'Y')]');
    end
    %get bounding box
    bBox=[min(clustCentr(:,1)) min(clustCentr(:,2)); max(clustCentr(:,1))
max(clustCentr(:,2))];
    % adjust SOM output space
    Delta=diff(bBox);
    DeltaX=Delta(1)/(sM.topol.msize(2)-1);
    DeltaY=Delta(2)/(sM.topol.msize(1)-1);
    SomOutput=zeros(prod(sM.topol.msize),2);
    counter=0;
    xPos=bBox(1,1);
    yPos=bBox(2,2);
    for i=1:sM.topol.msize(2)%each column
        for j=1: sM.topol.msize(1)% each line
            counter=counter+1;
            SomOutput(counter,:)=[xPos yPos];
            yPos=yPos-DeltaY;
        end
        xPos=xPos+DeltaX;
        yPos=bBox(2,2);
    end
    % distance between cluster centroids and SOM output centroids
    Dists=zeros(size(clustCentr,1),size(SomOutput,1));
    for i=1:size(clustCentr,1); %each cluster
        for j=1:size(SomOutput,1); % each SOM output
            Dists(i,j)=norm(clustCentr(i,:)-SomOutput(j,:));
        end
    end
    [a ind]=min(Dists');
    clustColors=sM.codebook(ind,:);

    colorItems=cell(numbClusters,1);
    for j=1:numbClusters
         colorItems{j}=['<HTML><FONT color="' rgbconv(clustColors(j,:)) '">Cluster'
num2str(j) '</FONT>'];
    end

    set (handles.Listbox,'String',colorItems);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % edicao cores manual
    %clustColors=[0 0 1; 0 1 0; 1 0 0];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %SHOW LABELS
    showClusterLabels(handles,numbClusters,clustColors)

    CreateClusters(handles,viewNumber)
    geosom_MakeSelectionAllOpenViewsByCluster(viewNumber,clustColors);

    %save colors in global
    mainHandles.ClusterColors=clustColors;
return

function showClusterLabels(handles,numbClusters,clustColors)
    %delete group of labels
    p = get(gca,'children');
    delInd=strcmpi(get(p,'userdata'),'LabelsGroup');
    delete(p(delInd==1));

    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    p1=p(a);
    xClust=get(p1,'X');
```

```
    yClust=get(p1,'Y');

    gh = hggroup;
    set(gh,'userdata','LabelsGroup');

    for i=1:numbClusters
        if numbClusters==1
            centroidX=mean(xClust);
            centroidY=mean(yClust);
        else
            centroidX=mean(xClust{i});
            centroidY=mean(yClust{i});
        end
        if get(handles.showClusterLabels,'value') % show LABELS
            text(centroidX,centroidY,num2str(i),...
            'parent',gh,'FontSize',10,'FontWeight','bold',...
            'color',[0 0 0],'BackgroundColor', [1 1 1 ]);
        end
        set(p1(i),'color',clustColors(i,:));
    end
return

function chkSlide_Callback(hObject,eventdata,handles,viewNumber)
    %turn on and off the flood
    if get(hObject,'value')
        slider_Callback(hObject,eventdata,handles,viewNumber)
    else
        %get patchs with userdata=sel and delete them
        p = get(gca,'children');
        delInd=strcmpi(get(p,'userdata'),'flood');
        delete(p(delInd==1));
    end

return

function slider_Callback(hObject,eventdata,handles,viewNumber)
    %flood umat
    global mainHandles

    if get(handles.chkSlide,'value')
        global gss

        %get patchs with userdata=sel and delete them
        p = get(gca,'children');
        delInd=strcmpi(get(p,'userdata'),'flood');
        delete(p(delInd==1));

        slideVal=get(handles.slider,'value');

        %get the umat
        umat=gss.View(viewNumber);
        breakval=slideVal*64;

        %create group
        gh = hggroup;
        set(gh,'userdata','flood');
        for i=1:length(umat{1})
            if umat{1}(i).IValue<breakval;
                x=umat{1}(i).X;
                x=[x(1:end-1) x(1)];
                y=umat{1}(i).Y;
                y=[y(1:end-1) y(1)];
                patch(x,y,mainHandles.prefs.floodcolor,...
                'parent',gh);
            end
        end
    end
return
```

```
function pushExportClusters_Callback(hObject,eventdata,handles,viewNumber)
    % function to export the clusters

    % Create the Clusters
    CreateClusters(handles,viewNumber)

    geosom_ExportClusters();
return

function CreateClusters(handles,viewNumber)
    % Create the Clusters
    % this will create gss.ClusterIndex and define for each data item its
    % cluster

    global gss

    SelectedComponents = get(handles.Listbox,'Value');

    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    p1=p(a);

    gss.ClusterIndex=zeros(length(gss.SelectedIndex),1);
    counter=0;
    for i=p1'   % for each cluster
        counter=counter+1;
        indice=geosom_ClosestItem([get(i,'X'); get(i,'Y')]',gss,viewNumber);
        if isempty(indice)
            break
        end

        for j=indice
            sel=find(gss.ViewIndexData(:,viewNumber)==j);
            if isempty(sel)~=1
                gss.ClusterIndex(sel)=counter;
            end
        end

    end
return

function push1clusterunit_Callback(hObject,eventdata,handles,viewNum)
    global gss
    global mainHandles

    button = questdlg(['Are you shure you want to create 1 cluster',...
        ' per unit? This will erase all defined clusters!'],'1 cluster per
unit','Yes','No','No');

    if strcmpi(button,'yes')
        somId=gss.ViewLabel(viewNum).SomOrigin;

        %delete all clusters drawn
        p =flipud(get(gca,'children'));
        a=find(strcmpi(get(p,'Type'),'line')==1);
        p1=p(a);
        if isempty(p1)==0
            delete(p1);
        end
        set(handles.Listbox,'string','');

        %create a cluster for each unit
        %delInd=strcmpi(get(p,'UserData'),'clusterdraw');
        IValues=[];
        for i=1:size(gss.View{viewNum},1)
            if gss.View{viewNum}(i).SomIndex>0 %unit
                IValues=[IValues;gss.View{viewNum}(i).IValue];
                xCoord= [gss.View{viewNum}(i).X(1)-.1; ...
                         gss.View{viewNum}(i).X(1)-.1; ...
```

```
                        gss.View{viewNum}(i).X(1)+1;...
                        gss.View{viewNum}(i).X(1)+1;...
                        gss.View{viewNum}(i).X(1)-.1];
                yCoord= [gss.View{viewNum}(i).Y(1)-.1; ...
                        gss.View{viewNum}(i).Y(1)+1; ...
                        gss.View{viewNum}(i).Y(1)+1; ...
                        gss.View{viewNum}(i).Y(1)-.1; ...
                        gss.View{viewNum}(i).Y(1)-.1];
                line('X',xCoord,'Y',yCoord,...
                'UserData','cluster','color',mainHandles.prefs.ClusterColor,...
                'linewidth',2);
                drawnow

                %add item to the list
                items=get(handles.Listbox,'string');
                if isempty(items)
                    items=[strcat('Cluster',num2str(size(items,1)+1))];
                else
                    items=cellstr(items);
                    items=[items;strcat('Cluster',num2str(size(items,1)+1))];
                end
                set(handles.Listbox,'string',items);
            end
        end
        %as alternative present the on the map, the colors of each unit in the
        % the BMU
        button = questdlg('Do you want to present clusters on all open views using as
color code the same each BMU has on the U-mat?',...
                  '1 cluster per unit','Yes','No','No');
        if strcmpi(button,'yes')
            clustColors=mainHandles.prefs.cmap(IValues,:);
            geosom_MakeSelectionAllOpenViewsByCluster(viewNum,clustColors);
        end
    end
return

function pushRemove_Callback(hObject,eventdata,handles)

    SelectedComponents = get(handles.Listbox,'Value');
    if size(SelectedComponents,2)>1
        msgbox('Please select only one cluster to remove!','Remove Cluster','help')
        return
    end

    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    p1=p(a);

    if isempty(p1)
        return
    end
    pSel=p1(SelectedComponents);
    %pNotSel=p1(find(SelItems==0));

    stringV=get(handles.Listbox,'string');
    stringV(size(stringV,1)-size(SelectedComponents,2)+1:end,:)=[];
    set(handles.Listbox,'value',1);
    set(handles.Listbox,'string',stringV);
    delete(pSel)
return

function Listbox_Callback(hObject,eventdata,handles)
    %list_entries = get(handles.Listbox,'String');
    global mainHandles

    SelectedComponents = get(handles.Listbox,'Value');
    SelItems=zeros(size(get(handles.Listbox,'string'),1),1);
    SelItems(SelectedComponents)=1;
```

```
    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    %a=find(strcmpi(get(p,'Type'),'patch')==1);
    p1=p(a);
    pSel=p1(SelectedComponents);
    pNotSel=p1(find(SelItems==0));

    set (pSel,'Color',mainHandles.prefs.SelClusterColor);
    set (pNotSel,'Color',mainHandles.prefs.ClusterColor);
return

function FigMouseClick_Callback(hObject,eventdata,handles,viewNum)
    %function to deal with clicking in the figure
    global gss
    global mainHandles

    seltype=get(gcf,'selectiontype');

    %check the selected  button
    if strcmp(get(gcf,'Pointer'),'crosshair')==1
        mouse =get(gca,'currentpoint');
        mouse=mouse(1,1:2);
        mousestate = get(gcbf,'SelectionType');
    else
        return
    end

    %get line draw before
    p = get(gca,'children');
    delInd=strcmpi(get(p,'UserData'),'clusterdraw');

    if delInd==0
        mouseCoords= mouse;
    else
        mouseCoords=[get(p(delInd==1),'X')' get(p(delInd==1),'Y')'];
        mouseCoords=[mouseCoords; mouse];
    end

    if strcmp(mousestate,'open') %double click
        set(p(delInd==1),'X',[mouseCoords(:,1);mouseCoords(1,1)],...
            'Y',[mouseCoords(:,2);mouseCoords(1,2)],...
            'UserData','cluster','color',mainHandles.prefs.ClusterColor,...
            'linewidth',2);

        drawnow
        %add item to the list
        items=get(handles.Listbox,'string');
        if isempty(items)
            items=[strcat('Cluster',num2str(size(items,1)+1))];
        else
            items=cellstr(items);
            items=[items;strcat('Cluster',num2str(size(items,1)+1))];
        end
        set(handles.Listbox,'string',items);
    else
        %line(mouseCoords(:,1),mouseCoords(:,2),'UserData','clusterdraw',...
        %    'color',mainHandles.prefs.DrawClusterColor,'linewidth',2);
        if size(mouseCoords,1)==1
            line(mouseCoords(:,1),mouseCoords(:,2),'UserData','clusterdraw',...
            'color',mainHandles.prefs.DrawClusterColor,'linewidth',2);
        else
            set(p(delInd==1),'X',mouseCoords(:,1),'Y',mouseCoords(:,2));
        end
    end
return

function ButtonFunction(hObject,eventdata,hToolbar,viewNum)

    buttontag=get(hObject,'tag');
```

```
    bt=get(hToolbar,'children');
    % uncheck all the toogle buttons
    for i=1:length(bt)
        try
        set (bt(i),'state','off');
        catch
        end
    end
    set (hObject,'state','on');

    if strcmpi(buttontag,'zoomin')
        zoom on;
    elseif strcmpi(buttontag,'selectclusters')
        zoom off;
        set(gcf,'Pointer','crosshair');
    end
return

function showHclust(hObject,eventdata, handles,viewNum)
    global gss

    %get number of clusters
    nClust=get(hObject,'value');
    nClust = str2double(get(handles.editNclusters,'string'));
    if isnan(nClust)
        helpdlg('Please insert a numeric max. number of clusters!','Training GeoSOM');
        return
    end

    if get(hObject,'value'); % show the labels
        % build clusters
        SOM=gss.Som(gss.ViewLabel(viewNum).SomOrigin);
        codebook=SOM.codebook(:,find(SOM.mask==1));
        %if geosom remove x and y
        if strcmpi(SOM.method,'geosom')
            codebook=codebook(:,3:end);
        end

        %get umat neuron centroids for text adding
        [Xcoords Ycoords]=getUmatNeuronCentroids(gss.View(viewNum),SOM.topol.msize);

        if get(handles.checkBMU,'value')  %only bmus
            %get hits view
            HitsViewIndex=0;
            for i=1:length(gss.ViewLabel)
                if strcmpi(gss.ViewLabel(i).Type,'hits')
                    % is this the Umat or the CP view?
                    if strcmpi(gss.ViewLabel(i).Name,strcat('umat',' hits'))
                        if gss.ViewLabel(i).SomOrigin==gss.ViewLabel(viewNum).SomOrigin
                            HitsViewIndex=i;
                            break;
                        end
                    end
                end
            end
            % based on each hit bounding box find BMUs
            BmuNeurons=[];
            for j=1:length(gss.View{HitsViewIndex})
                xDif=gss.View{HitsViewIndex}(1,j).BoundingBox(1,1)...
                    -gss.View{HitsViewIndex}(1,j).BoundingBox(2,1);
                if xDif
                    BmuNeurons=[BmuNeurons;j];
                end
            end
            %include only bmus
            codebook=codebook(BmuNeurons,:);
            %text only for bmus
            Xcoords=Xcoords(BmuNeurons,:);
```

```
            Ycoords=Ycoords(BmuNeurons,:);
        end
        %make clutering
        cls = clusterdata([Xcoords Ycoords codebook],'maxclust',nClust);

        % add labels
        h=text(Xcoords, Ycoords,num2cell(cls));
        set(h,'userdata','HclustGroup');

        %disable chk only bmu
        set(handles.checkBMU,'enable','off');
    else % remove the hits
        %get patchs with userdata=sel and delete them
        p = get(gca,'children');
        delInd=strcmpi(get(p,'userdata'),'HclustGroup');
        delete(p(delInd==1));

        %enable chk only bmu
        set(handles.checkBMU,'enable','on');
    end
return

function [Xcoords Ycoords]=getUmatNeuronCentroids(umat,msize)
    %get centroids
    Xcoords=zeros(msize(1)*msize(2),1);
    Ycoords=zeros(msize(1)*msize(2),1);
    counter=0;
    for i=1:length(umat{1})
        if isempty(umat{1}(i).SomIndex)==0
            counter=counter+1;
            Xcoords(counter)=mean(umat{1}(i).X(1:end-1));
            Ycoords(counter)=mean(umat{1}(i).Y(1:end-1));
        end
    end

return

function showHits(hObject,eventdata, viewNum, source)
    % show the hits on the top of the UMAT
    global gss
    global mainHandles

    if get(hObject,'value'); % show the Hits
        SomIndex=gss.ViewLabel(viewNum).SomOrigin;
        HitsViewIndex=0;


        for i=1:length(gss.ViewLabel)
            if strcmpi(gss.ViewLabel(i).Type,'hits')
                % is this the Umat or the CP view?
                if strcmpi(gss.ViewLabel(i).Name,strcat(source,' hits'))
                    if gss.ViewLabel(i).SomOrigin==SomIndex
                        HitsViewIndex=i;
                        break;
                    end
                end
            end
        end
        if HitsViewIndex>0  % found the view
            %create group
            gh = hggroup;
            set(gh,'userdata','HitsGroup');
            for j=1:length(gss.View{HitsViewIndex})
                x=gss.View{HitsViewIndex}(j).X;
                x=[x(1:end-1) x(1)];
                y=gss.View{HitsViewIndex}(j).Y;
                y=[y(1:end-1) y(1)];
                if mainHandles.prefs.onlyhitsoutline
                    line(x,y,'Color',mainHandles.prefs.hitscolor ...
```

```
                    ,'parent',gh ...
                    ,'LineWidth',1 ...
                    ,'UserData','hits');
                else
                    patch(x,y,mainHandles.prefs.hitscolor,...
                        'parent',gh,...,...
                        'UserData','hits');
                end
            end
        end
    else % remove the hits
        %get patchs with userdata=sel and delete them
        p = get(gca,'children');

        delInd=strcmpi(get(p,'userdata'),'HitsGroup');
        delete(p(delInd==1));
    end
return

function ViewClose(hObject,eventdata,viewNum)
    %closes the view and saves

    saveClusters(viewNum);
    %close
    delete (gcf);
return

function saveClusters(viewNum)
    %add clusters to gss.ClusterClusters

    global gss

    p =flipud(get(gca,'children'));
    a=find(strcmpi(get(p,'Type'),'line')==1);
    p1=p(a);

    counter=0;
    clust=[];
    for i=p1'   % for each cluster
        counter=counter+1;
        clustcoords=[get(i,'X'); get(i,'Y')]';
        clust{counter}=clustcoords;
    end
    gss.ViewClusters{gss.ViewLabel(viewNum).SomOrigin}=clust;
return

function ExportAxes2NewFig(hObject,eventdata)
    % get axes and open in new figure for export

    h=gca;
    hNewAxes=figure;
    set(hNewAxes,'color',[1 1 1]);
    copyobj(h, hNewAxes);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function distances=geosom_Eucdist(sM, sD, DataInd);
    %geosom_Eucdist
    %
    % Syntax: geosom_Eucdist
    % input - sM, sD
    % output - distances
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
```

```
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: October 06-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    distances=zeros(size(DataInd,2),1);

    for i=1:size(DataInd,2)
        X=sD.data(DataInd(i),find(sM.mask==1));
        Y=sM.codebook(bmus(DataInd(i)), find(sM.mask==1));
        U=~isnan(Y);
        V=~isnan(X);
        distances(i)=abs(X.^2*U'+V*Y'.^2-2*X*Y');
    end

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_export2KML(path,file,viewNum)
    %GEOSOM_EXPORT2KML ...
    % Export the selected features to kml format
    % Syntax: export2KML(path,file,viewNum)
    % path - path where to save the file
    % file - filename
    % viewNum - SOM view
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 22-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    global gss
    global mainHandles

    filename=[path,file];
    s=gss.View{viewNum};
    s=s(gss.SelectedIndex);
    % We have to change all empty SomIndex to 0
    for i=1:length(s)

        if isfield(s,'SomIndex')
            if isempty(s(i).SomIndex)
                s(i).SomIndex=0;
            end
        end
        % add selection field
        s(i).Sel=double(gss.SelectedIndex(i));
         % add clusters field
        %s(i).Clust=double(gss.ClusterIndex(i));
    end

    fid = fopen(filename, 'wt');
    fwrite(fid,'<?xml version="1.0" encoding="UTF-8" ?>');
    fwrite(fid,'<kml xmlns="http://earth.google.com/kml/2.0">');
    fwrite(fid,'<Document>');
    fwrite(fid,strcat('<name>',file,'</name>'));
    fwrite(fid,'<Style id="Mapit">');
    fwrite(fid,'<IconStyle>');
    fwrite(fid,'<color>FFFFFFFF</color>');
    fwrite(fid,'<scale>1.0</scale>');
    fwrite(fid,'<Icon>');
    fwrite(fid,'<href>root://icons/palette-5.png</href>');
    fwrite(fid,'<x>192</x>');
    fwrite(fid,'<y>192</y>');
    fwrite(fid,'<w>32</w>');
    fwrite(fid,'<h>32</h>');
    fwrite(fid,'</Icon>');
    fwrite(fid,'</IconStyle>');
```

```
    fwrite(fid,'<LabelStyle>');
    fwrite(fid,'<scale>0</scale>');
    fwrite(fid,'</LabelStyle>');
    fwrite(fid,'<BalloonStyle>');
    fwrite(fid,'<text>$[description]</text>');
    fwrite(fid,'<color>FFFFFFFF</color>');
    fwrite(fid,'</BalloonStyle>');
    fwrite(fid,'</Style>');
    fwrite(fid,'<Style id="symbol">');
    fwrite(fid,'<IconStyle>');
    fwrite(fid,'<color>7FFFCA95</color>');
    fwrite(fid,'<scale>1.0</scale>');
    fwrite(fid,'<Icon>');
    fwrite(fid,'</Icon>');
    fwrite(fid,'</IconStyle>');
    fwrite(fid,'<LabelStyle>');
    fwrite(fid,'<scale>0</scale>');
    fwrite(fid,'</LabelStyle>');
    fwrite(fid,'<BalloonStyle>');
    fwrite(fid,'<text>$[description]</text>');
    fwrite(fid,'<color>FFFFFFFF</color>');
    fwrite(fid,'</BalloonStyle>');
    fwrite(fid,'<LineStyle>');

fwrite(fid,strcat('<color>',dec2hex(255),rgbconv(fliplr(mainHandles.prefs.selcolor)),'</c
olor>'));
    fwrite(fid,'<width>3</width>');
    fwrite(fid,'</LineStyle>');
    fwrite(fid,'<PolyStyle>');
    % not rgb but abgr a is transparency

fwrite(fid,strcat('<color>',dec2hex(127),rgbconv(fliplr(mainHandles.prefs.selcolor)),'</c
olor>'));
    fwrite(fid,'</PolyStyle>');
    fwrite(fid,'</Style>');
    fwrite(fid,'<Folder>');
    fwrite(fid,'<name>Features</name>');
    fwrite(fid,'<Snippet>Legend: Single Symbol</Snippet>');
    fwrite(fid,'<open>0</open>');
    fwrite(fid,'<visibility>1</visibility>');

    %progress bar
    h = waitbar(0,'Exporting kml...');
    for i=1:length(s)
        % for each element
        fwrite(fid,'<Placemark>');
        fwrite(fid,strcat('<name>',s(i).DTCCFRSEC0,'</name>'));
        fwrite(fid,'<Snippet />');
        fwrite(fid,'<description>');
        fwrite(fid,'<![CDATA[ <table border=0 cellpadding=0 cellspacing=0  width=250
style="FONT-SIZE: 11px; FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;"><tr><td
bgcolor="#E3E1CA" align="right"><font COLOR="#000000"><b>DISTRITO</b></font></td><td
bgcolor="#E4E6CA"> <font COLOR="#008000">LISBOA</font></td></tr></table>]]>');
        fwrite(fid,'</description>');
        fwrite(fid,'<styleUrl>#symbol</styleUrl>');
        fwrite(fid,'<Polygon>');
        fwrite(fid,'<tessellate>0</tessellate>');
        fwrite(fid,'<extrude>0</extrude>');
        fwrite(fid,'<altitudeMode>clampedToGround</altitudeMode>');
        fwrite(fid,'<outerBoundaryIs>');
        fwrite(fid,'<LinearRing>');
        fwrite(fid,'<coordinates>');

        X = getfield(s(i),'X')';
        Y = getfield(s(i),'Y')';
        Z=[X,Y];
        Z(isnan(Z(:,1)),:)=[];
        %Z=num2str(Z,'%10.5g,');
        %Z(:,end)=' ';(1:end-1,:)
```

```
        a=num2str(Z(1:end-1,:),'%g,');
        for j=1:length(Z)-1
            fprintf(fid,'%s\n',[a(j,:) '0']);
        end

        fwrite(fid,'</coordinates>');
        fwrite(fid,'</LinearRing>');
        fwrite(fid,'</outerBoundaryIs>');
        fwrite(fid,'</Polygon>');
        fwrite(fid,'</Placemark>');
        waitbar(i/length(s));
    end
    fwrite(fid,'</Folder>');
    fwrite(fid,'<Folder>');
    fwrite(fid,'<name>Info</name>');
    fwrite(fid,'<Snippet />');
    fwrite(fid,'<Style>');
    fwrite(fid,'<BalloonStyle>');
    fwrite(fid,'<text>$[description]</text>');
    fwrite(fid,'</BalloonStyle>');
    fwrite(fid,'</Style>');
    fwrite(fid,'<description>');
    fwrite(fid,'<![CDATA[ Source : GeoSOM suite<br>Legend: Single Symbol<br>Label:
DTCCFRSEC0<br><br><hr>GeoSOM to Google Earth<p><font color="blue"><b>Powered by GeoSOM
suite</b><br> <i>http://www.isegi.unl.pt/labnt/geosom/</i></font></p>]]>');
    fwrite(fid,'</description>');
    fwrite(fid,'</Folder>');
    fwrite(fid,'</Document>');
    fwrite(fid,'</kml>');

    fclose(fid);
    close(h);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_exportClusters2KML(path,file,viewNum)
    %GEOSOM_EXPORTCLUSTERS2KML ...
    % Syntax: geosom_exportClusters2KML(path,file,viewNum)
    % Export the CLUSTERS to kml format
    % path - path where to save the file
    % file - filename
    % viewNum - SOM view
    %
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 22-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    global gss
    global mainHandles

    filename=[path,file];
    s=gss.View{viewNum};
    s=s(find(gss.ClusterIndex>0));
    cl=find(gss.ClusterIndex>0);
    % We have to change all empty SomIndex to 0
    for i=1:length(s)
        if isfield(s,'SomIndex')
            if isempty(s(i).SomIndex)
                s(i).SomIndex=0;
            end
        end
        % add selection field
        %s(i).Sel=double(gss.SelectedIndex(i));
         % add clusters field
        s(i).Clust=double(gss.ClusterIndex(cl(i)));
    end
```

```
    fid = fopen(filename, 'wt');
    fwrite(fid,'<?xml version="1.0" encoding="UTF-8" ?>');
    fwrite(fid,'<kml xmlns="http://earth.google.com/kml/2.0">');
    fwrite(fid,'<Document>');
    fwrite(fid,strcat('<name>',file,'</name>'));
    fwrite(fid,'<Style id="Mapit">');
    fwrite(fid,'<IconStyle>');
    fwrite(fid,'<color>FFFFFFFF</color>');
    fwrite(fid,'<scale>1.0</scale>');
    fwrite(fid,'<Icon>');
    fwrite(fid,'<href>root://icons/palette-5.png</href>');
    fwrite(fid,'<x>192</x>');
    fwrite(fid,'<y>192</y>');
    fwrite(fid,'<w>32</w>');
    fwrite(fid,'<h>32</h>');
    fwrite(fid,'</Icon>');
    fwrite(fid,'</IconStyle>');
    fwrite(fid,'<LabelStyle>');
    fwrite(fid,'<scale>0</scale>');
    fwrite(fid,'</LabelStyle>');
    fwrite(fid,'<BalloonStyle>');
    fwrite(fid,'<text>$[description]</text>');
    fwrite(fid,'<color>FFFFFFFF</color>');
    fwrite(fid,'</BalloonStyle>');
    fwrite(fid,'</Style>');

    for i =unique(gss.ClusterIndex)'
        if i~=0
            fwrite(fid,strcat('<Style id="symbol',num2str(i),'">'));
            fwrite(fid,'<IconStyle>');
            fwrite(fid,'<color>7FFFCA95</color>');
            fwrite(fid,'<scale>1.0</scale>');
            fwrite(fid,'<Icon>');
            fwrite(fid,'</Icon>');
            fwrite(fid,'</IconStyle>');
            fwrite(fid,'<LabelStyle>');
            fwrite(fid,'<scale>0</scale>');
            fwrite(fid,'</LabelStyle>');
            fwrite(fid,'<BalloonStyle>');
            fwrite(fid,'<text>$[description]</text>');
            fwrite(fid,'<color>FFFFFFFF</color>');
            fwrite(fid,'</BalloonStyle>');
            fwrite(fid,'<LineStyle>');

fwrite(fid,strcat('<color>',dec2hex(255),rgbconv(fliplr(mainHandles.ClusterColors(i,:))),
'</color>'));
            fwrite(fid,'<width>1</width>');
            fwrite(fid,'</LineStyle>');
            fwrite(fid,'<PolyStyle>');
            % not rgb but abgr a is transparency

fwrite(fid,strcat('<color>',dec2hex(127),rgbconv(fliplr(mainHandles.ClusterColors(i,:))),
'</color>'));
            fwrite(fid,'</PolyStyle>');
            fwrite(fid,'</Style>');
        end
    end
    fwrite(fid,'<Folder>');
    fwrite(fid,'<name>Features</name>');
    fwrite(fid,'<Snippet>Legend: Single Symbol</Snippet>');
    fwrite(fid,'<open>0</open>');
    fwrite(fid,'<visibility>1</visibility>');

    %progress bar
    h = waitbar(0,'Exporting kml...');
    for i=1:length(s)
        % for each element
        fwrite(fid,'<Placemark>');
```

```
        fwrite(fid,strcat('<name>',s(i).DTCCFRSEC0,'</name>'));
        fwrite(fid,'<Snippet />');
        fwrite(fid,'<description>');
        fwrite(fid,'<![CDATA[ <table border=0 cellpadding=0 cellspacing=0  width=250
style="FONT-SIZE: 11px; FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;"><tr><td
bgcolor="#E3E1CA" align="right"><font COLOR="#000000"><b>DISTRITO</b></font></td><td
bgcolor="#E4E6CA"> <font COLOR="#008000">LISBOA</font></td></tr></table>]]>');
        fwrite(fid,'</description>');
        fwrite(fid,strcat('<styleUrl>#symbol',num2str(s(i).Clust),'</styleUrl>'));
        fwrite(fid,'<Polygon>');
        fwrite(fid,'<tessellate>0</tessellate>');
        fwrite(fid,'<extrude>0</extrude>');
        fwrite(fid,'<altitudeMode>clampedToGround</altitudeMode>');
        fwrite(fid,'<outerBoundaryIs>');
        fwrite(fid,'<LinearRing>');
        fwrite(fid,'<coordinates>');

        X = getfield(s(i),'X')';
        Y = getfield(s(i),'Y')';
        Z=[X,Y];
        Z(isnan(Z(:,1)),:)=[];
        %Z=num2str(Z,'%10.5g,');
        %Z(:,end)=' ';(1:end-1,:)
        a=num2str(Z(1:end-1,:),'%g,');
        for j=1:length(Z)-1
            fprintf(fid,'%s\n',[a(j,:) '0']);
        end
        fwrite(fid,'</coordinates>');
        fwrite(fid,'</LinearRing>');
        fwrite(fid,'</outerBoundaryIs>');
        fwrite(fid,'</Polygon>');
        fwrite(fid,'</Placemark>');
        waitbar(i/length(s));
    end
    fwrite(fid,'</Folder>');
    fwrite(fid,'<Folder>');
    fwrite(fid,'<name>Info</name>');
    fwrite(fid,'<Snippet />');
    fwrite(fid,'<Style>');
    fwrite(fid,'<BalloonStyle>');
    fwrite(fid,'<text>$[description]</text>');
    fwrite(fid,'</BalloonStyle>');
    fwrite(fid,'</Style>');
    fwrite(fid,'<description>');
    fwrite(fid,'<![CDATA[ Source : GeoSOM suite<br>Legend: Single Symbol<br>Label:
DTCCFRSEC0<br><br><hr>GeoSOM to Google Earth<p><font color="blue"><b>Powered by GeoSOM
suite</b><br> <i>http://www.isegi.unl.pt/labnt/geosom/</i></font></p>]]>');
    fwrite(fid,'</description>');
    fwrite(fid,'</Folder>');
    fwrite(fid,'</Document>');
    fwrite(fid,'</kml>');

    fclose(fid);
    close(h);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_ExportClusters()
%GEOSOM_EXPORTCLUSTERS ...
    % Syntax: geosom_ExportClusters()
    %function to show the form to export clusters
    %
    % Subfunctions: pushbuttonOk_Callback
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 22-2009
    % Version 1.0
```

```
    % Copyright (c) by the Ga2 programming team. 2009

    global gss

    %center in screen
    scrsz = get(0,'ScreenSize');
    fh=figure('Position',[scrsz(3)/2-180 scrsz(3)/2-100 370 100]);

    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off','name',...
        'Export Clusters','Resize','off');
    handles.exportclusters=fh;

    % % add controls
    hb = uibuttongroup('Title','Export options','tag','uibtgroupExport',...
                 'units','pixels','Position',[10 50 350 40]);
    handles.uibtgroupExport=hb;

    %option buttons (export all or selected data)
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'All data',...
           'tag', 'chkAll',...
           'Position', [50 5 120 15],...
           'Callback', '');
    handles.chkAll=h;
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'Only clusters',...
           'tag', 'chkSelection',...
           'Position', [200 5 120 15],...
           'Callback', '');
    handles.chkSelection=h;

    % delimiter
    h = uicontrol('Style', 'text',...
           'String', 'Delimiter',...
           'tag', 'textdelimiter',...
           'Position', [10 25 50 15]);
    h = uicontrol('Style', 'edit',...
           'String', ';',...
           'tag', 'editdelimiter',...
           'Position', [60 25 50 20],...
           'BackgroundColor',[1 1 1],...
           'Callback', '');
    handles.delimiter=h;

    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'Ok',...
           'tag', 'pushOK',...
           'Position', [310 10 50 25]);
    handles.pushOK=h;

    %%% CALLBACKS with handles %%%
     set(handles.pushOK,'Callback',{@pushbuttonOk_Callback handles});

     % Make the GUI modal
    set(handles.exportclusters,'WindowStyle','modal')
    % UIWAIT makes geosom_querydata wait for user response (see UIRESUME)
    uiwait(handles.exportclusters);
return

function pushbuttonOk_Callback(hObject, eventdata, handles)
    % export data
    global gss

    delimiter=get(handles.delimiter,'string');
    if get(handles.chkAll,'value')==1 % export all data
```

```
        strMsg='Export Data and clusters As';
        dataCell =[gss.NumDataLabel gss.CellDataLabel 'Clust' ;...
            num2cell(gss.NumData) gss.CellData num2cell(gss.ClusterIndex)];
    else
        strMsg='Export only clusters As';
        dataCell =['Clust';num2cell(gss.ClusterIndex)];
    end

    % get file
    [file,path] = uiputfile('*.txt',strMsg);

    %open file for writing
    fid = fopen(strcat(path,file), 'wt');

    %progress bar
    h = waitbar(0,'Exporting data...');
    for l=1:size(dataCell,1) %each line
        for r=1:size(dataCell,2) %each row
            fprintf(fid,['%s' delimiter],mat2str(cell2mat(dataCell(l,r))));
        end
        fprintf(fid,'\n');
        waitbar(l/size(dataCell,1));
    end
    fclose(fid);
    close(h);

    % Use UIRESUME instead of delete because the OutputFcn needs
    % to get the updated handles structure.
    uiresume(handles.exportclusters);
    close(handles.exportclusters);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_ExportData()
    %geosom_ExportData
    %function to show the form to export data
    %
    % Syntax: geosom_ExportData()
    % input - none
    % output - none
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 16-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    %center in screen
    scrsz = get(0,'ScreenSize');
    fh=figure('Position',[scrsz(3)/2-180 scrsz(3)/2-100 370 100]);

    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off','name',...
        'Export data','Resize','off');
    handles.exportdata=fh;

    % % add controls

    hb = uibuttongroup('Title','Export options','tag','uibtgroupExport',...
                 'units','pixels','Position',[10 50 350 40]);
    handles.uibtgroupExport=hb;
```

```
    %option buttons (export all or selected data)
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'All data',...
           'tag', 'chkAll',...
           'Position', [50 5 120 15],...
           'Callback', '');
    handles.chkAll=h;
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'Only selected data',...
           'tag', 'chkSelection',...
           'Position', [200 5 120 15],...
           'Callback', '');
    handles.chkSelection=h;

    % check if any selection exist
    if sum(gss.SelectedIndex)==0
        set(handles.chkSelection,'enable','off');
    end

    % delimiter
    h = uicontrol('Style', 'text',...
           'String', 'Delimiter',...
           'tag', 'textdelimiter',...
           'Position', [10 25 50 15]);
    h = uicontrol('Style', 'edit',...
           'String', ';',...
           'tag', 'editdelimiter',...
           'Position', [60 25 50 15],...
           'BackgroundColor',[1 1 1],...
           'Callback', '');
    handles.delimiter=h;


    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'Ok',...
           'tag', 'pushOK',...
           'Position', [310 10 50 25]);
    handles.pushOK=h;

    %%% CALLBACKS with handles %%%
     set(handles.pushOK,'Callback',{@pushbuttonOk_Callback handles});

     % Make the GUI modal
    set(handles.exportdata,'WindowStyle','modal')
    % UIWAIT makes geosom_querydata wait for user response (see UIRESUME)
    uiwait(handles.exportdata);
return

function pushbuttonOk_Callback(hObject, eventdata, handles)
    % export data
    global gss

    delimiter=get(handles.delimiter,'string');

    if get(handles.chkAll,'value')==1 % export all data
        strMsg='Export all Data As';
        dataCell =[gss.NumDataLabel gss.CellDataLabel ;...
            num2cell(gss.NumData) gss.CellData];
    else
        strMsg='Export selected Data As';
        if isempty(gss.CellDataLabel)
            dataCell =[gss.NumDataLabel;...
            num2cell(gss.NumData(gss.SelectedIndex,:))];
        else
            dataCell =[gss.NumDataLabel gss.CellDataLabel ;...
            num2cell(gss.NumData(gss.SelectedIndex,:)) ...
```

```
            gss.CellData(gss.SelectedIndex,:)];
        end
    end

    % get file
    [file,path] = uiputfile('*.txt',strMsg);

    %open file for writing
    fid = fopen(strcat(path,file), 'wt');

    %progress bar
    h = waitbar(0,'Exporting data...');
    for l=1:size(dataCell,1) %each line
        for r=1:size(dataCell,2) %each row
            fprintf(fid,['%s' delimiter],mat2str(cell2mat(dataCell(l,r))));
        end
        fprintf(fid,'\n');
        waitbar(l/size(dataCell,1));
    end
    fclose(fid);
    close(h);

    % Use UIRESUME instead of delete because the OutputFcn needs
    % to get the updated handles structure.
    uiresume(handles.exportdata);
    close(handles.exportdata);
return

function popupVariables_Callback(hObject, eventdata, handles)
    global gss
    sel = get(hObject,'Value');
    if length(gss.NumDataLabel)>=sel
        set(handles.popupValues,'String',sort(gss.NumData(:,sel)));
    else
        set(handles.popupValues,'String',sort(gss.CellData(:,sel-
length(gss.NumDataLabel))));
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_GeoSomTrainingParameters()
%GEOSOM_GEOSOMTRAININGPARAMETERS ...
% Syntax: geosom_GeoSomTrainingParameters()
% input - none
% output - none
%
% Subfunctions: trainSOM_CloseRequest, changeTrain, changeGeoComp,
%               chkSelGeoCom_Callback, pushOK_Callback
%   See also:
%
% Author: Roberto Henriques, Victor Lobo, Fernando Bação
% Created: September 04-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2009

    global gss

    fh=figure('Position',[200 200 400 500]);
    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
        'Resize','off','name','Train GEOSOM');
    handles.trainGeoSOM=fh;

    % % add controls
    % text - title
    h = uicontrol('Style', 'text',...
           'String', 'GeoSOM training parameters',...
           'tag', 'textViews',...
```

```
       'Position', [10 480 380 15],...
       'HorizontalAlignment','Center',...
       'fontweight','bold');
handles.textViews=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% panel map initialization
hp = uipanel('Title','Map Initialization',...
             'units','pixels','Position',[10 400 380 80]);
% size X
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'X',...
       'tag', 'textX',...
       'Position', [10 40 40 15],...
       'HorizontalAlignment','left');
handles.textX=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '5',...
       'tag', 'editX',...
       'background','white',...
       'Position', [50 40 50 20]);
handles.editX=h;

% size Y
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Y',...
       'tag', 'textY',...
       'Position', [130 40 40 15],...
       'HorizontalAlignment','left');
handles.textY=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '5',...
       'tag', 'editY',...
       'background','white',...
       'Position', [170 40 50 20]);
handles.editY=h;

% lattice
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Lattice',...
       'tag', 'textLattice',...
       'Position', [10 15 40 15],...
       'HorizontalAlignment','left');
handles.textLattice=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'hexa';'rect'},...
       'tag', 'popupLattice',...
       'Position', [50 15 70 20]);
handles.popupLattice=h;

% shape
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Shape',...
       'tag', 'textShape',...
       'Position', [130 15 40 15],...
       'HorizontalAlignment','left');
handles.textShape=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'sheet';'cyl';'toroid'},...
       'tag', 'popupShape',...
       'Position', [170 15 70 20]);
```

```
    handles.popupShape=h;

    % type
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Type',...
           'tag', 'textType',...
           'Position', [250 15 40 15],...
           'HorizontalAlignment','left');
    handles.textType=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp,...
           'String', {'geo dist';'linear';'random'},...
           'tag', 'popupType',...
           'Position', [280 15 70 20]);
    handles.popupType=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % components to select (non-geo)
    h = axes('tag', 'TableComponents','AmbientLightColor','white',...
           'units','pixels','Position', [20 290 200 100]);
    handles.TableComponents=h;

    % replace the character _ by space
    Comp = gss.NumDataLabel';
    for i=1:length(Comp)
        Comp{i}=strrep(Comp{i}, '_', ' ');
    end

    columninfo.titles={'Non-geo Components'};
    columninfo.formats = {'%4.6g'};
    columninfo.weight =      [1];
    columninfo.multipliers = [1];
    columninfo.isEditable =  [0];
    columninfo.isNumeric =   [0];
    columninfo.withCheck = true; % optional to put checkboxes along left side
    columninfo.chkLabel = 'Use'; % optional col header for checkboxes
    rowHeight = 12;
    gFont.size=8;
    gFont.name='Arial';
    geosom_MlTable(handles.trainGeoSOM, handles.TableComponents, 'CreateTable',
columninfo,rowHeight, Comp, gFont);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % components to select (geo)
    hb = uibuttongroup('Title','Geo Components','tag','uibtgroupTrain', ...
                'units','pixels','Position',[230 340 160 50]);
    handles.SelectGeoComp=hb;

    %option buttons
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'value',1,...
           'String', 'Use geo from spatial data',...
           'Position', [5 20 150 15],...
           'Callback', '');
    handles.optSpatialFromSHP=h;
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'Select geo components',...
           'Position', [5 0 150 15],...
           'Callback', '');
    handles.optSpatialFromFields=h;

    hp = uipanel('Title','',...
                'units','pixels','Position',[230 280 160 60]);
    h = uicontrol('Style', 'text',...
           'parent', hp,...
           'String', 'X comp.',...
```

```
       'tag', 'textType',...
       'Position', [10 30 60 15],...
       'HorizontalAlignment','left');
h = uicontrol('Style', 'popup',...
       'parent', hp,...
       'String', Comp,...
       'tag', 'popupType',...
       'enable','off',...
       'Position', [60 30 80 20]);
handles.popupXcomp=h;

h = uicontrol('Style', 'text',...
       'parent', hp,...
       'String', 'Y comp.',...
       'tag', 'textType',...
       'Position', [10 5 60 15],...
       'HorizontalAlignment','left');
h = uicontrol('Style', 'popup',...
       'parent', hp,...
       'String', Comp,...
       'tag', 'popupType',...
       'enable','off',...
       'Position', [60 5 80 20]);
handles.popupYcomp=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize data
h = uicontrol('Style', 'text',...
       'String', 'Normalize data',...
       'tag', 'textNormalize',...
       'Position', [10 260 80 15],...
       'HorizontalAlignment','left');
handles.textNormalize=h;
h = uicontrol('Style', 'popup',...
       'String', {'none';'var';'range';'log';'logistic';'histD';'histC'},...
       'tag', 'popupNormalize',...
       'Position', [100 260 80 20]);
handles.popupNormalize=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Map train
hp = uipanel('Title','Map Train',...
             'units','pixels','Position',[10 170 380 80]);

% batch or sequential
% at this time only sequential is implemented
hb = uibuttongroup('Title','','tag','uibtgroupTrain','parent',hp,...
             'units','pixels','Position',[10 40 360 20]);
handles.uibtgroupTrain=hb;

%option buttons
h = uicontrol('Style', 'radiobutton',...
       'parent',hb,...
       'enable','off',...
       'String', 'Batch train',...
       'tag', 'chkbatch',...
       'Position', [50 0 100 15],...
       'Callback', '');
handles.chkbatch=h;
h = uicontrol('Style', 'radiobutton',...
       'parent',hb,...
       'value',1,...
       'String', 'Sequential train',...
       'tag', 'chkSequential',...
       'Position', [200 0 100 15],...
       'Callback', '');
handles.chkSequential=h;

% sequential trainning parameters panel
```

```
hp2 = uipanel('Title','','parent',hp,...
              'tag','panelTrain',...
              'units','pixels',...
              'visible','off',...
              'Position',[100 5 270 35]);
handles.panelTrain=hp2;
set(handles.panelTrain,'visible','on')

% Iterations type
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Iter.',...
       'tag', 'textIter',...
       'Position', [5 5 40 15],...
       'HorizontalAlignment','left');
handles.textIter=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'epochs';'samples'},...
       'tag', 'popupIter',...
       'Position', [25 5 60 20]);
handles.popupIter=h;

% order
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Order',...
       'tag', 'textOrder',...
       'Position', [85 5 40 15],...
       'HorizontalAlignment','left');
handles.textOrder=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'random';'ordered'},...
       'tag', 'popupOrder',...
       'Position', [115 5 60 20]);
handles.popupOrder=h;

% Length function
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Length',...
       'tag', 'textLength',...
       'Position', [175 5 40 15],...
       'HorizontalAlignment','left');
handles.textLength=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'inv';'linear';'power'},...
       'tag', 'popupLength',...
       'Position', [210 5 50 20]);
handles.popupLength=h;

% Neigh function
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Neigh',...
       'tag', 'textNeigh',...
       'Position', [10 10 40 15],...
       'HorizontalAlignment','left');
handles.textNeigh=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'gaussian';'cutgauss';'ep';'bubble'},...
       'tag', 'popupNeigh',...
       'Position', [40 10 60 20]);
handles.popupNeigh=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Rough parameters
hp = uipanel('Title','Rough',...
         'tag','panelRough',...
         'units','pixels',...
         'Position',[10 80 190 80]);
handles.panelRough=hp;

% Iterations_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 1',...
       'tag', 'textIterations1',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editIterations1',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations1=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 1',...
       'tag', 'textRadio1',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editRadio1',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio1=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 1',...
       'tag', 'textAlpha1',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '0.3',...
       'tag', 'editAlpha1',...
       'background','white',...
       'Position', [100 5 40 20]);
handles.editAlpha1=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finetune parameters
hp = uipanel('Title','Finetune',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[200 80 190 80]);
handles.panelFinetune=hp;

% Iterations_2
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 2',...
       'tag', 'textIterations2',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations2=h;
```

```
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editIterations2',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations2=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 2',...
       'tag', 'textRadio2',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editRadio2',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio2=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 2',...
       'tag', 'textAlpha2',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '0.1',...
       'tag', 'editAlpha2',...
       'background','white',...
       'Position', [100 5 40 20]);
handles.editAlpha2=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%chk only finetune
h = uicontrol('Style', 'checkbox',...
       'String', 'Use only finetune',...
       'tag', 'chkFinetune',...
       'Position', [250 60 105 15]);
handles.chkFinetune=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%chk only finetune
h = uicontrol('Style', 'checkbox',...
       'String', 'Train only selected data',...
       'tag', 'chkTrainSelection',...
       'Position', [250 40 140 15]);
handles.chkTrainSelection=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GeoSOM specific parameters
hp = uipanel('Title','GeoSOM specific parameters',...
             'tag','panelRough',...
             'units','pixels',...
             'Position',[10 10 190 70]);
handles.panelRough=hp;

% k
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Geo radius(k)',...
       'tag', 'textk',...
       'Position', [30 30 100 15],...
       'HorizontalAlignment','left');
```

```
    handles.textK=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String', '0',...
           'tag', 'editK',...
           'background','white',...
           'Position', [100 30 40 20]);
    handles.editK=h;

    %f
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Update(f)',...
           'tag', 'textf',...
           'Position', [30 10 55 15],...
           'HorizontalAlignment','left');
    handles.textK=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp,...
           'String', {'all';'nongeo'},...
           'tag', 'popupF',...
           'Position', [100 10 60 20]);
    handles.popupF=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'OK',...
           'tag', 'pushOK',...
           'Position', [340 5 50 15],...
           'Callback', 'setmap');
    handles.pushOK=h;

    %%% CALLBACKS with handles %%%
    set(handles.trainGeoSOM,'CloseRequestFcn',{@trainSOM_CloseRequest handles});
    set(handles.uibtgroupTrain,'SelectionChangeFcn',{@changeTrain handles});
    set(handles.SelectGeoComp,'SelectionChangeFcn',{@changeGeoComp handles});
    set(handles.pushOK,'Callback',{@pushOK_Callback handles});

    % Make the GUI modal
    set(handles.trainGeoSOM ,'WindowStyle','modal')
    % UIWAIT
    uiwait(handles.trainGeoSOM);
return

function trainSOM_CloseRequest(hObject, eventdata,handles)
    uiresume(handles.trainGeoSOM);
    delete (handles.trainGeoSOM);
return

function changeTrain(hObject, eventdata,handles)
    if get(handles.chkbatch,'value');
        set(handles.panelTrain,'visible','off');
    else get(handles.chkSequential,'value');
        set(handles.panelTrain,'visible','on');
    end
return

function changeGeoComp(hObject, eventdata,handles)
    if get(handles.optSpatialFromSHP,'value');
        set(handles.popupXcomp,'enable','off');
        set(handles.popupYcomp,'enable','off');
    elseif get(handles.optSpatialFromFields,'value');
        set(handles.popupXcomp,'enable','on');
        set(handles.popupYcomp,'enable','on');
    end
return

function chkSelGeoCom_Callback(hObject, eventdata, handles)
```

```
    % allow to select geo components from a list of fields
    % if get(hObject,'value') % selection on
    %     set(handles.popupXcomp,'enable','on');
    %     set(handles.popupYcomp,'enable','on');
    % else
    %     set(handles.popupXcomp,'enable','off');
    %     set(handles.popupYcomp,'enable','off');
    % end
return

function pushOK_Callback(hObject, eventdata, handles)
    global gss
    % X
    user_entry = str2double(get(handles.editX,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric X map size!','Training GeoSOM');
        return
    else
        parameters.xdim = user_entry;
    end

    % Y
    user_entry = str2double(get(handles.editY,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Y map size!','Training GeoSOM');
        set(h,'string','');
        return
    else
       parameters.ydim = user_entry;
    end

    % Lattice
    Lattice=get(handles.popupLattice,'string');
    parameters.Lattice=cell2mat(Lattice(get(handles.popupLattice,'value'),:));

    % Shape
    Shape=get(handles.popupShape,'string');
    parameters.Shape=cell2mat(Shape(get(handles.popupShape,'value'),:));

    % map type
    MapType=get(handles.popupType,'string');
    parameters.MapType=cell2mat(MapType(get(handles.popupType,'value'),:));

    % normalization
    Norm=get(handles.popupNormalize,'string');
    parameters.Normalization=cell2mat(Norm(get(handles.popupNormalize,'value'),:));

    %non geo components
    comp = get(handles.TableComponents, 'userdata');
    parameters.SelectedComponents=find(comp.isChecked==1);

    if isempty(parameters.SelectedComponents)
        helpdlg('Please select a non geo component!','Training GeoSOM');
        return
    end
    parameters.ComponentsNames=comp.data;

    %geo components
    parameters.chkSelGeoCom = 0;
    if get(handles.optSpatialFromFields, 'value') % X and Y came from the field list
        % get the fields from the popups
        parameters.chkSelGeoCom = 1;
        parameters.SelectedComponentsGeo=[get(handles.popupXcomp,'value')...
                                          get(handles.popupYcomp, 'value')];
    else % in this case we are calculating the centroids from spatial data
        %chech if the first view correspont to geographic data
        if isfield(gss,'ViewLabel')
            if strcmpi(gss.ViewLabel(1).Type,'geographicmap')==0
```

```
                helpdlg('Since the data you are using is not from a shapfile, you must
select the geo components!','Training GeoSOM');
                return
            end
        else
            helpdlg('Since the data you are using is not from a shapefile, you must
select the geo components!','Training GeoSOM');
            return
        end
    end

    % Neigh
    Neigh=get(handles.popupNeigh,'string');
    parameters.Neigh=cell2mat(Neigh(get(handles.popupNeigh,'value'),:));

    % batch or sequential train?
    if get(handles.chkSequential,'value')
        %IterType
        IterType=get(handles.popupIter,'string');
        parameters.IterType=cell2mat(IterType(get(handles.popupIter,'value'),:));
        % order
        Order=get(handles.popupOrder,'string');
        parameters.Order=cell2mat(Order(get(handles.popupOrder,'value'),:));
        % Length function
        Length_funct=get(handles.popupLength,'string');

parameters.Length_funct=cell2mat(Length_funct(get(handles.popupLength,'value'),:));
    end

    % rough and finetune parameters
    user_entry = str2double(get(handles.editIterations1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a number of Iterations 1!','Training GeoSOM');
        return
    else
        parameters.niterations_1 = user_entry;
    end

    user_entry = str2double(get(handles.editIterations2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a number of Iterations 2!','Training GeoSOM');
        return
    else
        parameters.niterations_2 = user_entry;
    end

    user_entry = str2double(get(handles.editRadio1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Radio 1!','Training GeoSOM');
        return
    else
        parameters.radius_ini_1 = user_entry;
    end

    user_entry = str2double(get(handles.editRadio2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Radio 2!','Training GeoSOM');
        return
    else
        parameters.radius_ini_2 = user_entry;
    end

    user_entry = str2double(get(handles.editAlpha1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Alpha 1!','Training GeoSOM');
        return
    else
        parameters.alpha_ini_1 = user_entry;
    end
```

```
    user_entry = str2double(get(handles.editAlpha2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Alpha 2!','Training GeoSOM');
        return
    else
        parameters.alpha_ini_2 = user_entry;
    end

    parameters.batchTrain= get(handles.chkbatch,'value');

    % k
    user_entry = str2double(get(handles.editK ,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric k!','Training GeoSOM');
        return
    else
        parameters.k = user_entry;
    end

    % f
    f=get(handles.popupF,'string');
    parameters.f=cell2mat(f(get(handles.popupType,'value'),:));

    %chkFinetune
    parameters.chkFinetune=get(handles.chkFinetune,'value');

    % only selection?
    parameters.TrainSelection=get(handles.chkTrainSelection,'value');

    % disableButtons
    set(handles.trainGeoSOM,'Pointer','watch');
    set(handles.pushOK,'Enable','off');
    refresh(gcf) %redraws the GUI to reflect changes

    %train GeoSOM and make new view
    %we are using also geosom_struct and gss because geosom_struct.NumData has
    %in the geosom two new columns (x and y)
    [somIndex,geosom_struct] = geosom_BuildGeoSom(parameters);

    % build umat
    geosom_BuildViewUmat_Geo(geosom_struct,somIndex);
    %geosom_BuildViewUmat(somIndex);

    % % build component planes
    geosom_BuildComponentPlanes_Geo(geosom_struct,somIndex);

    % % build Hits
    geosom_BuildHits_Geo(geosom_struct,somIndex)

    %resume
    uiresume(handles.trainGeoSOM);
    close(handles.trainGeoSOM);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_DataIndex=geosom_Geostruct2Index2DataIndex(geosomstruct2,
CurrentViewIndex, InputIndex);
% geosom_NCIndex =
%    geosom_GetNumCellIndexFromViewIndex(ViewIndex, ViewStructIndex);
%
% NAME: geosom_GetNumCellIndexFromViewIndex
%
% OBJECTIVE: Get the indexes of the NumData and CellData from a given
% ViewIndex and a given ViewStructIndex
%
% INPUT PARAMETERS:
%       ViewIndex - The index of the view. Ex.: View{1}, View{2}
```

```
%       ViewStructIndex - The indexes in the ViewStruct (may be more than
%       one)
%
% OUTPUT PARAMETERS:
%       geosom_NCIndex - the index
% V.0.1 2007/01/23 By V.Lobo & M.Loureiro

    tmp_NCIndex=zeros(1,length(geosomstruct2.NumData));
    for i=1:length(InputIndex)
        tmp = find (geosomstruct2.ViewIndexData(:,CurrentViewIndex)==InputIndex(i));
        tmp_NCIndex(tmp)=1;
    end
    geosom_DataIndex=find(tmp_NCIndex==1)';
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gs2=geosom_getSelectionBasedItemCount(View,numberofItem,topol)
    % geosom_getSelectionBasedItemCount - this function will give a feature
    % composed by xn and yn vertex which represent selection based on item count
    %
    % Syntax: geosom_getSelectionBasedItemCount()
    % input
    % output
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: October 30-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    % create polygon tmp structure
    for i=1:(size(numberofItem,1));
        tmp{i}='Polygon';
    end;

    gs2 = struct('Geometry',tmp,'BoundingBox',[0 0 0 0],'X',0,'Y',0,...
            'Value',0);

    if strcmpi(topol,'rect') % rect lattice
        SelSize = numberofItem(:,2)./max(numberofItem(:,2));
        for j=1:size(numberofItem,1)
            xtmp=View(numberofItem(j,1)).X;
            ytmp=View(numberofItem(j,1)).Y;

            xCentroid=mean(xtmp(1:end-1));
            yCentroid=mean(ytmp(1:end-1));

            gs2(j).X = [xCentroid-1*SelSize(j)...
                        xCentroid+1*SelSize(j)...
                        xCentroid+1*SelSize(j)...
                        xCentroid-1*SelSize(j) NaN];
            gs2(j).Y = [yCentroid-1*SelSize(j)...
                        yCentroid-1*SelSize(j)...
                        yCentroid+1*SelSize(j)...
                        yCentroid+1*SelSize(j) NaN];
        end;

    elseif strcmpi(topol,'hexa') % hexa lattice
        SelSize = numberofItem(:,2).*1.5./max(numberofItem(:,2));

        for j=1:size(numberofItem,1)
            if numberofItem(j,1)>0
                xtmp=View(numberofItem(j,1)).X;
                ytmp=View(numberofItem(j,1)).Y;
```

```
                xCentroid=mean(xtmp(1:end-1));
                yCentroid=mean(ytmp(1:end-1));

                C=SelSize(j);
                A=C*.5;
                B=sind(60)*C;

                gs2(j).X = [xCentroid-B xCentroid-B xCentroid xCentroid+B xCentroid+B
xCentroid NaN];
                gs2(j).Y = [yCentroid-1/2*C yCentroid+1/2*C yCentroid+A+1/2*C
yCentroid+1/2*C yCentroid-1/2*C yCentroid-A-1/2*C NaN];
            end
        end;
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_HSomTrainingParameters()
    %geosom_HSomTrainingParameters
    % form to define the HSOM trainning parameters
    %
    % Syntax: geosom_HSomTrainingParameters()
    % input - none
    % output - none
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 04-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    fh=figure('Position',[200 200 400 500]);
    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
        'Resize','off','name','Train Hierarchical SOM');
    handles.trainSOM=fh;

    % % add controls
    % text - title
    h = uicontrol('Style', 'text',...
           'String', 'Hierarchical SOM training parameters',...
           'tag', 'textViews',...
           'Position', [10 480 380 15],...
           'HorizontalAlignment','Center',...
           'fontweight','bold');
    handles.textViews=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % panel map initialization
    hp = uipanel('Title','Map Initialization',...
                 'units','pixels','Position',[10 400 380 80]);
    % size X
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'X',...
           'tag', 'textX',...
           'Position', [10 40 40 15],...
           'HorizontalAlignment','left');
    handles.textX=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
```

```
       'String', '5',...
       'tag', 'editX',...
       'background','white',...
       'Position', [50 40 50 20]);
handles.editX=h;

% size Y
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Y',...
       'tag', 'textY',...
       'Position', [130 40 40 15],...
       'HorizontalAlignment','left');
handles.textY=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '5',...
       'tag', 'editY',...
       'background','white',...
       'Position', [170 40 50 20]);
handles.editY=h;

% lattice
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Lattice',...
       'tag', 'textLattice',...
       'Position', [10 15 40 15],...
       'HorizontalAlignment','left');
handles.textLattice=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'hexa';'rect'},...
       'tag', 'popupLattice',...
       'Position', [50 15 70 20]);
handles.popupLattice=h;

% shape
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Shape',...
       'tag', 'textShape',...
       'Position', [130 15 40 15],...
       'HorizontalAlignment','left');
handles.textShape=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'sheet';'cyl';'toroid'},...
       'tag', 'popupShape',...
       'Position', [170 15 70 20]);
handles.popupShape=h;

% type
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Type',...
       'tag', 'textType',...
       'Position', [250 15 40 15],...
       'HorizontalAlignment','left');
handles.textType=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'linear';'random'},...
       'tag', 'popupType',...
       'Position', [280 15 70 20]);
handles.popupType=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize data
```

```
    h = uicontrol('Style', 'text',...
           'String', 'Normalize data',...
           'tag', 'textNormalize',...
           'Position', [10 380 80 15],...
           'HorizontalAlignment','left');
    handles.textNormalize=h;
    h = uicontrol('Style', 'popup',...
           'String', {'none';'var';'range';'log';'logistic';'histD';'histC'},...
           'tag', 'popupNormalize',...
           'Position', [10 360 80 20]);
    handles.popupNormalize=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %type of HSOM
    h = uicontrol('Style', 'text',...
        'String', 'Input of the HSOM',...
        'tag', 'textNormalize',...
        'Position', [10 340 100 15],...
        'HorizontalAlignment','left');

    if checkTypeofDataset()
    Inputs={'Umats´ coordinates'; ...
        'Quantization error '; ...
        'All pattern activations';'Spatial coordinates'};
    else
        Inputs={'Umats´ coordinates'; ...
        'Quantization error '; ...
        'All pattern activations'};
    end

    h = uicontrol('Style', 'listbox',...
        'String', Inputs,...
        'tag', 'Listbox',...
        'Position', [10 250 150 90],...
        'HorizontalAlignment','left','Max',3,'Min',1);
    handles.Listbox=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % components to select
    h = axes('tag', 'TableComponents','AmbientLightColor','white',...
           'units','pixels','Position', [170 250 210 130]);
    handles.TableComponents=h;

    % replace the character _ by space
    SOMnames =cell(length(gss.Som),1);
    for i=1:length(gss.Som)
        SOMnames{i}=strrep(gss.Som(i).name, '_', ' ');
    end
    columninfo.titles={'SOMs'};
    columninfo.formats = {'%4.6g'};
    columninfo.weight =      [1];
    columninfo.multipliers = [1];
    columninfo.isEditable =  [0];
    columninfo.isNumeric =   [0];
    columninfo.withCheck = true; % optional to put checkboxes along left side
    columninfo.chkLabel = 'All'; % optional col header for checkboxes
    rowHeight = 12;
    gFont.size=8;
    gFont.name='Arial';
    geosom_MlTable(handles.trainSOM, handles.TableComponents, 'CreateTable',
columninfo,rowHeight, SOMnames, gFont);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Map train
    hp = uipanel('Title','Map Train',...
                 'units','pixels','Position',[10 160 380 80]);

    % batch or sequential button group
    hb = uibuttongroup('Title','','tag','uibtgroupTrain','parent',hp,...
```

```
            'units','pixels','Position',[10 40 360 20]);
handles.uibtgroupTrain=hb;

%option buttons
h = uicontrol('Style', 'radiobutton',...
       'parent',hb,...
       'String', 'Batch train',...
       'tag', 'chkbatch',...
       'Position', [50 0 100 15],...
       'Callback', '');
handles.chkbatch=h;
h = uicontrol('Style', 'radiobutton',...
       'parent',hb,...
       'String', 'Sequential train',...
       'tag', 'chkSequential',...
       'Position', [200 0 100 15],...
       'Callback', '');
handles.chkSequential=h;

% sequential trainning parameters panel
hp2 = uipanel('Title','','parent',hp,...
             'tag','panelTrain',...
             'units','pixels',...
             'visible','off',...
             'Position',[100 5 270 35]);
handles.panelTrain=hp2;

% Iterations type
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Iter.',...
       'tag', 'textIter',...
       'Position', [5 5 40 15],...
       'HorizontalAlignment','left');
handles.textIter=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'epochs';'samples'},...
       'tag', 'popupIter',...
       'Position', [25 5 60 20]);
handles.popupIter=h;

% order
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Order',...
       'tag', 'textOrder',...
       'Position', [85 5 40 15],...
       'HorizontalAlignment','left');
handles.textOrder=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'random';'ordered'},...
       'tag', 'popupOrder',...
       'Position', [115 5 60 20]);
handles.popupOrder=h;

% Length function
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Length',...
       'tag', 'textLength',...
       'Position', [175 5 40 15],...
       'HorizontalAlignment','left');
handles.textLength=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'inv';'linear';'power'},...
       'tag', 'popupLength',...
```

```
       'Position', [210 5 50 20]);
handles.popupLength=h;

% Neigh function
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Neigh',...
       'tag', 'textNeigh',...
       'Position', [10 10 40 15],...
       'HorizontalAlignment','left');
handles.textNeigh=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'gaussian';'cutgauss';'ep';'bubble'},...
       'tag', 'popupNeigh',...
       'Position', [40 10 60 20]);
handles.popupNeigh=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rough parameters
hp = uipanel('Title','Rough',...
             'tag','panelRough',...
             'units','pixels',...
             'Position',[10 80 190 80]);
handles.panelRough=hp;

% Iterations_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 1',...
       'tag', 'textIterations1',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editIterations1',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations1=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 1',...
       'tag', 'textRadio1',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editRadio1',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio1=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 1',...
       'tag', 'textAlpha1',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '0.3',...
       'tag', 'editAlpha1',...
       'background','white',...
```

```
       'enable','off',...
       'Position', [100 5 40 20]);
handles.editAlpha1=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finetune parameters
hp = uipanel('Title','Finetune',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[200 80 190 80]);
handles.panelFinetune=hp;

% Iterations_2
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 2',...
       'tag', 'textIterations2',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editIterations2',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations2=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 2',...
       'tag', 'textRadio2',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editRadio2',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio2=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 2',...
       'tag', 'textAlpha2',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '0.1',...
       'tag', 'editAlpha2',...
       'background','white',...
       'enable','off',...
       'Position', [100 5 40 20]);
handles.editAlpha2=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% description parameters
hp = uipanel('Title','Description',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[10 10 190 70]);
handles.panelDescription=hp;
% name
h = uicontrol('Style', 'text',...
       'Parent',hp,...
```

```
            'String', 'Name HSOM',...
            'Position', [5 35 65 15],...
            'HorizontalAlignment','left');
    handles.textName=h;
    h = uicontrol('Style', 'edit',...
            'Parent',hp,...
            'String', [date],...
            'tag', 'editName',...
            'background','white',...
            'HorizontalAlignment','left',...
            'Position', [70 35 110 20]);
    handles.editName=h;
    %obs
    h = uicontrol('Style', 'text',...
            'Parent',hp,...
            'String', 'Obs.',...
            'Position', [10 20 55 15],...
            'HorizontalAlignment','left');
    handles.textObs=h;
    h = uicontrol('Style', 'edit',...
            'Parent',hp,...
            'String', '',...
            'tag', 'editObs',...
            'max',2,'min',0,...
            'background','white',...
            'HorizontalAlignment','left',...
            'Position', [40 5 140 30]);
    handles.editObs=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %chk only finetune
    h = uicontrol('Style', 'checkbox',...
            'String', 'Use only finetune',...
            'tag', 'chkFinetune',...
            'Position', [210 55 105 15]);
    handles.chkFinetune=h;

    %train only selected data
    h = uicontrol('Style', 'checkbox',...
            'String', 'Train only selected data',...
            'tag', 'chkTrainSelection',...
            'Position', [210 35 140 15]);
    handles.chkTrainSelection=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %push buttons
    h = uicontrol('Style', 'pushbutton',...
            'String', 'OK',...
            'tag', 'pushOK',...
            'Position', [340 5 50 15],...
            'Callback', 'setmap');
    handles.pushOK=h;

    %%% CALLBACKS with handles %%%
    set(handles.trainSOM,'CloseRequestFcn',{@trainSOM_CloseRequest handles});
    set(handles.uibtgroupTrain,'SelectionChangeFcn',{@changeTrain handles});
    set(handles.pushOK,'Callback',{@pushOK_Callback handles});

    % Make the GUI modal
    set(handles.trainSOM ,'WindowStyle','modal')
    % UIWAIT
    uiwait(handles.trainSOM);
return

function trainSOM_CloseRequest(hObject, eventdata,handles)
    uiresume(handles.trainSOM);
    delete (handles.trainSOM);
return
```

```
function changeTrain(hObject, eventdata,handles)
    if get(handles.chkbatch,'value');
        set(handles.panelTrain,'visible','off');
        set(handles.editAlpha1,'enable','off');
        set(handles.editAlpha2,'enable','off');
    else get(handles.chkSequential,'value');
        set(handles.panelTrain,'visible','on');
        set(handles.editAlpha1,'enable','on');
        set(handles.editAlpha2,'enable','on');
    end
return

function pushOK_Callback(hObject, eventdata, handles)
    global gss
    % X
    user_entry = str2double(get(handles.editX,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric X map size!','Training SOM');
        return
    else
        parameters.xdim = user_entry;
    end

    % Y
    user_entry = str2double(get(handles.editY,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Y map size!','Training SOM');
        set(h,'string','');
        return
    else
       parameters.ydim = user_entry;
    end

    % Lattice
    Lattice=get(handles.popupLattice,'string');
    parameters.Lattice=cell2mat(Lattice(get(handles.popupLattice,'value'),:));

    % Shape
    Shape=get(handles.popupShape,'string');
    parameters.Shape=cell2mat(Shape(get(handles.popupShape,'value'),:));

    % map type
    MapType=get(handles.popupType,'string');
    parameters.MapType=cell2mat(MapType(get(handles.popupType,'value'),:));

    % normalization
    Norm=get(handles.popupNormalize,'string');
    parameters.Normalization=cell2mat(Norm(get(handles.popupNormalize,'value'),:));

    % components
    comp = get(handles.TableComponents, 'userdata');
    parameters.SelectedComponents=find(comp.isChecked==1);

    if isempty(parameters.SelectedComponents)
        helpdlg('Please select a component!','Training SOM');
        return
    end
    parameters.ComponentsNames=comp.data;

    % Neigh
    Neigh=get(handles.popupNeigh,'string');
    parameters.Neigh=cell2mat(Neigh(get(handles.popupNeigh,'value'),:));

    % batch or sequential train?
    if get(handles.chkSequential,'value')
        %IterType
        IterType=get(handles.popupIter,'string');
        parameters.IterType=cell2mat(IterType(get(handles.popupIter,'value'),:));
        % order
```

```
        Order=get(handles.popupOrder,'string');
        parameters.Order=cell2mat(Order(get(handles.popupOrder,'value'),:));
        % Length function
        Length_funct=get(handles.popupLength,'string');

parameters.Length_funct=cell2mat(Length_funct(get(handles.popupLength,'value'),:));
    end

    % rough and finetune parameters
    parameters.chkFinetune=get(handles.chkFinetune,'Value');

    user_entry = str2double(get(handles.editIterations1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a number of Iterations 1!','Training HSOM');
        return
    else
        parameters.niterations_1 = user_entry;
    end

    user_entry = str2double(get(handles.editIterations2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a number of Iterations 2!','Training HSOM');
        return
    else
        parameters.niterations_2 = user_entry;
    end

    user_entry = str2double(get(handles.editRadio1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Radio 1!','Training HSOM');
        return
    else
        parameters.radius_ini_1 = user_entry;
    end

    user_entry = str2double(get(handles.editRadio2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Radio 2!','Training HSOM');
        return
    else
        parameters.radius_ini_2 = user_entry;
    end

    user_entry = str2double(get(handles.editAlpha1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Alpha 1!','Training HSOM');
        return
    else
        parameters.alpha_ini_1 = user_entry;
    end

    user_entry = str2double(get(handles.editAlpha2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Alpha 2!','Training HSOM');
        return
    else
        parameters.alpha_ini_2 = user_entry;
    end

    %name
    parameters.SOMname = ['HSOM ' get(handles.editName,'string')];

    %Obs.
    parameters.Obs = get(handles.editObs,'string');

    %train
    parameters.batchTrain= get(handles.chkbatch,'value');

    % only selected data?
```

```
    parameters.TrainSelection=get(handles.chkTrainSelection,'value');

    parameters.SelectedInputs = get(handles.Listbox,'Value');

    if isempty(parameters.SelectedInputs)
        helpdlg('Please select at leat one input to the HSOM!','Training HSOM');
        return
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %train SOM
    % disableButtons
    set(handles.trainSOM,'Pointer','watch');
    set(handles.pushOK  ,'Enable','off');
    refresh(gcf) %redraws the GUI to reflect changes

    %Hierarchical SOM
    sD=geosom_BuildHierarqSom(parameters);
    somIndex = length(gss.Som);
    geosom_BuildViewUmatHSOM(somIndex,sD);

    % % build component planes
    geosom_BuildComponentPlanesHSOM(sD,somIndex);
    % % build Hits
    geosom_BuildHitsHSOM(sD,somIndex)

    %resume
    uiresume(handles.trainSOM);
    close(handles.trainSOM);
return

function chkHSOM_Callback(hObject, eventdata, handles)
    %check if using SOM or HSOM
    global gss

    if get(hObject,'value') % HSOM

        if isfield(gss,'Som')
            if length(gss.Som)<2
                msgbox('To use Hierarchical SOM you need 2 or more trainned SOMs'...
                    ,'Cannot use Hierarchical SOM','warn')
                set(hObject,'value',0)
                return
            end
        else
            msgbox('To use Hierarchical SOM you need 2 or more trainned SOMs'...
                ,'Cannot use Hierarchical SOM','warn')
            set(hObject,'value',0)
            return
        end

        geosom_MlTable(handles.trainSOM, handles.TableComponents, 'DestroyTable');

        % replace the character _ by space
        SOMnames =cell(length(gss.Som),1);
        for i=1:length(gss.Som)
            SOMnames{i}=strrep(gss.Som(i).name, '_', ' ');
        end
        columninfo.titles={'SOMs'};
        columninfo.formats = {'%4.6g'};
        columninfo.weight =      [1];
        columninfo.multipliers = [1];
        columninfo.isEditable =  [0];
        columninfo.isNumeric =   [0];
        columninfo.withCheck = true; % optional to put checkboxes along left side
        columninfo.chkLabel = 'All'; % optional col header for checkboxes
        rowHeight = 12;
        gFont.size=8;
        gFont.name='Arial';
```

```
        geosom_MlTable(handles.trainSOM, handles.TableComponents, 'CreateTable',
columninfo,rowHeight, SOMnames, gFont);

    else %SOM
        geosom_MlTable(handles.trainSOM, handles.TableComponents, 'DestroyTable');

        % replace the character _ by space
        Comp = gss.NumDataLabel';
        for i=1:length(Comp)
            Comp{i}=strrep(Comp{i}, '_', ' ');
        end
        columninfo.titles={'Component'};
        columninfo.formats = {'%4.6g'};
        columninfo.weight =      [1];
        columninfo.multipliers = [1];
        columninfo.isEditable =  [0];
        columninfo.isNumeric =   [0];
        columninfo.withCheck = true; % optional to put checkboxes along left side
        columninfo.chkLabel = 'All'; % optional col header for checkboxes
        rowHeight = 12;
        gFont.size=8;
        gFont.name='Arial';
        geosom_MlTable(handles.trainSOM, handles.TableComponents, 'CreateTable',
columninfo,rowHeight, Comp, gFont);

    end
return

function TypeofDataset=checkTypeofDataset()
    % this function tests if the dataset has a geopatial component
    % checkTypeofDataset=0 --> has no spatial attributes
    % checkTypeofDataset=1 --> has spatial attributes

    global gss
    if strcmpi(gss.ViewLabel(1).Type,'GeographicMap')
        TypeofDataset=1;
    else
        TypeofDataset=0;
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_loadPrefs()
    %GEOSOM_LOADPREFS ...
    % Syntax: geosom_loadPrefs()
    % function to load prefs from prefs.mat file
    %   located in the geosom root
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 30-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    global mainHandles

    try %to load prefs.m
        load prefs.mat
        mainHandles.prefs=prefs;
    catch % if there is no file, create it
        global mainHandles
        mainHandles.prefs.cmap=colormap;
        mainHandles.prefs.selcolor=[1 0 0];
        mainHandles.prefs.onlyoutline=0;
        mainHandles.prefs.hitscolor=[1 0 0];
        mainHandles.prefs.onlyhitsoutline=0;
        mainHandles.prefs.cmapOutline=0;
        mainHandles.prefs.cmapOutlineColor=0;
        mainHandles.prefs.clusterscmap=colormap;
```

```
        mainHandles.prefs.DrawClusterColor=[0 0 1];
        mainHandles.prefs.ClusterColor=[1 0 0];
        mainHandles.prefs.SelClusterColor=[1 0 0];
        mainHandles.prefs.floodcolor=[1 1 0];

        prefs=mainHandles.prefs;
        save 'prefs.mat' prefs
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_MakeSelectionAllOpenViews()
    % geosom_MakeSelectionAllOpenViews - this function represents in each
    % view the current selection based on gss.selectedindex
    %
    % Syntax: geosom_MakeSelectionAllOpenViews()
    % input
    % output
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 04-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss
    global mainHandles

    %get the selection color
    onlyOut = mainHandles.prefs.onlyoutline;
    selColor=mainHandles.prefs.selcolor;

    if isfield(gss,'ViewLabel')==0 % no views to represent
        return
    end

    for i=1:length(gss.ViewLabel)

        if strcmp(gss.ViewLabel(i).Type,'GeographicMap')
            if ishandle(gss.ViewLabel(i).WindowHandle)

                figure(gss.ViewLabel(i).WindowHandle);

                %get patchs with userdata=sel and delete them
                p = get(gss.ViewLabel(i).WindowHandle,'children');
                for a=1:length(p)
                    if strcmpi(get(p(a),'Type'),'axes')
                        break
                    end
                end
                p = get(p(a),'children');
                delInd=zeros(1,length(p));
                for j=1:length(p)
                    if strcmpi(get(p(j),'userdata'),'Sel')
                        delInd(j)=1;
                    end
                end
                delete(p(delInd==1));

                if sum(gss.SelectedIndex)>0
                    Sel=unique(gss.ViewIndexData(find(gss.SelectedIndex==1),i));
                    if strcmpi(gss.View{i}(1).Geometry,'point')
                        for j=Sel'
                            x=gss.View{i}(j).X;
```

```
                            y=gss.View{i}(j).Y;


line(x,y,selColor,'Marker','o','MarkerFaceColor',selColor,'UserData','Sel');
                        end
                    elseif strcmpi(gss.View{i}(1).Geometry,'line')
                        for j=Sel'
                            x=gss.View{i}(j).X;
                            y=gss.View{i}(j).Y;

                            line(x,y,selColor,'UserData','Sel');
                        end
                    elseif strcmpi(gss.View{i}(1).Geometry,'polygon')
                        for j=Sel'
                            x=gss.View{i}(j).X;
                            y=gss.View{i}(j).Y;

                            % check for multi parts features
                            indexNan=[0;find(isnan(x)==1)'];
                            for part=1:length(indexNan)-1
                                beginFeat=indexNan(part);
                                endFeat=indexNan(part+1);

                                x1=[x(beginFeat+1:endFeat-1) x(beginFeat+1)];
                                y1=[y(beginFeat+1:endFeat-1) y(beginFeat+1)];
                                patch(x1,y1,selColor,'UserData','Sel');
                            end
                            %x=[x(1:end-1) x(1)];
                            %y=[y(1:end-1) y(1)];
                        end
                    end
                end
            end

        elseif strcmp(gss.ViewLabel(i).Type,'UMAT')
            %check if figure is visible. If yes then represent features.
            % else, dont do nothing
            if ishandle(gss.ViewLabel(i).WindowHandle)
                figure(gss.ViewLabel(i).WindowHandle)


                %get patchs with userdata=sel and delete them
                p = get(gss.ViewLabel(i).WindowHandle,'children');

                for a=1:length(p)
                    if strcmpi(get(p(a),'Type'),'axes')
                        break
                    end
                end

                p = get(p(a),'children');
                delInd=zeros(1,length(p));
                for j=1:length(p)
                    if strcmpi(get(p(j),'userdata'),'Sel')
                        delInd(j)=1;
                    end
                end
                delete(p(delInd==1));

                %check if selection is based on item count or not (access the
                %chkbox on the figure and check if it is on...
                selectionBasedOnCount=0;
                hChild=get(gss.ViewLabel(i).WindowHandle,'children');
                tags=get(hChild,'tag');
                index=find(strcmpi(tags,'selBasedOnCount')==1);

                if isempty(index)==0
                    if get(hChild(index),'Value')
                        selectionBasedOnCount=1;
```

```
                end
            end

            if sum(gss.SelectedIndex)>0
                tmpSel=gss.ViewIndexData(find(gss.SelectedIndex==1),i);
                Sel=unique(tmpSel);

                if selectionBasedOnCount==0 % normal selection
                    for j=Sel'
                        if j>0
                            x=gss.View{i}(j).X;
                            x=[x(1:end-1) x(1)];
                            y=gss.View{i}(j).Y;
                            y=[y(1:end-1) y(1)];

                            if onlyOut % only outline

line(x,y,'Color',selColor,'LineWidth',2,'UserData','Sel');
                            else
                                patch(x,y,selColor,'UserData','Sel');
                            end
                        end
                    end
                else % selection based on item count
                    numberofItem=[Sel zeros(length(Sel),1)];
                    for j=1:length(Sel)
                        numberofItem(j,2)=sum(tmpSel==Sel(j));
                    end

                    gs2=geosom_getSelectionBasedItemCount(gss.View{i},numberofItem
...
                        ,gss.Som(gss.ViewLabel(i).SomOrigin).topol.lattice);

                    for j=1:length(gs2)
                        x=gs2(j).X;
                        x=[x(1:end-1) x(1)];
                        y=gs2(j).Y;
                        y=[y(1:end-1) y(1)];
                        if onlyOut % only outline

line(x,y,'Color',selColor,'LineWidth',2,'UserData','Sel');
                        else
                            patch(x,y,selColor,'UserData','Sel');
                        end
                    end

                end
            end
        end
    elseif strcmpi(gss.ViewLabel(i).Type,'Cp')
        %check if figure is visible. If yes then represent features.
        % else, dont do nothing
        if ishandle(gss.ViewLabel(i).WindowHandle)
            figure(gss.ViewLabel(i).WindowHandle)

            if gss.ViewLabel(i).GraphicHandle>0
                %h=gco(gss.ViewLabel(i).GraphicHandle);
                if ishandle(gss.ViewLabel(i).GraphicHandle)
                   try
                       axes(gss.ViewLabel(i).GraphicHandle);
                   end
                end
            end

            %get patchs with userdata=sel and delete them
            p = get(gss.ViewLabel(i).WindowHandle,'children');
            for a=1:length(p)
                if strcmpi(get(p(a),'Type'),'axes')
                    break
```

```
                    end
                end
                p = get(p(a),'children');
                delInd=zeros(1,length(p));
                for j=1:length(p)
                    if strcmpi(get(p(j),'userdata'),'Sel')
                        delInd(j)=1;
                    end
                end
                delete(p(delInd==1));

                %check if selection is based on item count or not (access the
                %chkbox on the figure and check if it is on...
                selectionBasedOnCount=0;
                hChild=get(gss.ViewLabel(i).WindowHandle,'children');
                tags=get(hChild,'tag');
                index=find(strcmpi(tags,'selBasedOnCount')==1);

                if isempty(index)==0
                    if get(hChild(index),'Value')
                        selectionBasedOnCount=1;
                    end
                end

                if sum(gss.SelectedIndex)>0 % normal selection
                    tmpSel=gss.ViewIndexData(find(gss.SelectedIndex==1),i);
                    Sel=unique(tmpSel);

                    if selectionBasedOnCount==0 % normal selection
                        for j=Sel'
                            x=gss.View{i}(j).X;
                            x=[x(1:end-1) x(1)];
                            y=gss.View{i}(j).Y;
                            y=[y(1:end-1) y(1)];

                            if onlyOut % only outline

line(x,y,'Color',selColor,'LineWidth',2,'UserData','Sel');
                            else
                                patch(x,y,selColor,'UserData','Sel');
                            end
                            %patch(x,y,selColor,'UserData','Sel');
                        end
                    else % selection based on item count
                        numberofItem=[Sel zeros(length(Sel),1)];
                        for j=1:length(Sel)
                            numberofItem(j,2)=sum(tmpSel==Sel(j));
                        end

                        gs2=geosom_getSelectionBasedItemCount(gss.View{i},numberofItem
...
                            ,gss.Som(gss.ViewLabel(i).SomOrigin).topol.lattice);

                        for j=1:length(gs2)
                            x=gs2(j).X;
                            x=[x(1:end-1) x(1)];
                            y=gs2(j).Y;
                            y=[y(1:end-1) y(1)];
                            if onlyOut % only outline

line(x,y,'Color',selColor,'LineWidth',2,'UserData','Sel');
                            else
                                patch(x,y,selColor,'UserData','Sel');
                            end
                        end
                    end
                end
            end
    elseif strcmpi(gss.ViewLabel(i).Type,'pcp')
```

```
            %check if figure is visible. If yes then represent features.
            % else, dont do nothing
            if ishandle(gss.ViewLabel(i).WindowHandle)
                figure(gss.ViewLabel(i).WindowHandle)

                %get patchs with userdata=sel and delete them
                p = get(gss.ViewLabel(i).WindowHandle,'children');
                for a=1:length(p)
                    if strcmpi(get(p(a),'Type'),'axes')
                        break
                    end
                end
                p = get(p(a),'children');
                delInd=zeros(1,length(p));
                for j=1:length(p)
                    if strcmpi(get(p(j),'userdata'),'Sel')
                        delInd(j)=1;
                    end
                end
                delete(p(delInd==1));

                if sum(gss.SelectedIndex)>0
                    Sel=unique(gss.ViewIndexData(find(gss.SelectedIndex==1),i));

                    for j=Sel'
                        x=gss.View{i}(j).X;
                        %x=[x(1:end-1) x(1)];
                        y=gss.View{i}(j).Y;
                        %y=[y(1:end-1) y(1)];

                        line(x,y,'Color',selColor,'UserData','Sel');
                    end
                end
            end
        end
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_MakeSelectionAllOpenViewsByCluster(viewNumber,clstColors)
    % geosom_MakeSelectionAllOpenViewsByCluster - this function represents in each view
    % the current selection based on gss.selectedindex, using a different color
    % for each group of individual belonging to the same bmu
    %
    % Syntax: geosom_MakeSelectionAllOpenViewsByCluster()
    % input     viewNumber - the viewNumber of the caller view
    % output
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 05-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss
    global mainHandles

    %get the selection color
    onlyOut = mainHandles.prefs.onlyoutline;
    selColormap=mainHandles.prefs.cmap;

    if isfield(gss,'ViewLabel')==0 % no views to represent
        return
    end
```

```
% based on the viewNumber, lets us update all the views related
SomOrigin=gss.ViewLabel(viewNumber).SomOrigin;

% get the umat correspondent
for i=1:length(gss.ViewLabel)
    if strcmpi(gss.ViewLabel(i).Type,'umat')
        if gss.ViewLabel(i).SomOrigin==SomOrigin
            umatView=i;
            break;
        end
    end
end

for i=1:length(gss.ViewLabel)

    if strcmp(gss.ViewLabel(i).Type,'GeographicMap')
        if ishandle(gss.ViewLabel(i).WindowHandle)

            figure(gss.ViewLabel(i).WindowHandle);
            p = get(gca,'children');
            delInd=zeros(1,length(p));
            for j=1:length(p)
                if strcmpi(get(p(j),'userdata'),'Sel')
                    delInd(j)=1;
                end
            end
            delete(p(delInd==1));


            if sum(gss.ClusterIndex)>0
                clst=unique(gss.ClusterIndex)';
                clst(find(clst==0))=[];  %remove zero if exists
                counter=0;
                for cl=clst  %for each cluster
                    counter=counter+1;
                    Sel=unique(gss.ViewIndexData(find(gss.ClusterIndex==cl),i));

                    %depending on geometry...
                    if strcmpi(gss.View{i}(1).Geometry,'point')
                        for j=Sel'
                            x=gss.View{i}(j).X;
                            y=gss.View{i}(j).Y;

                            line(x,y,clstColors(counter,:),...
                                'Marker','o',...
                                'MarkerEdgeColor',clstColors(counter,:),...
                                'MarkerFaceColor',clstColors(counter,:),...
                                'UserData','Sel');
                        end
                    elseif strcmpi(gss.View{i}(1).Geometry,'line')
                        for j=Sel'
                            x=gss.View{i}(j).X;
                            y=gss.View{i}(j).Y;

                            line(x,y,clstColors(counter,:),...
                                'Marker','o',...
                                'MarkerEdgeColor',clstColors(counter,:),...
                                'MarkerFaceColor',clstColors(counter,:),...
                                'UserData','Sel');
                        end
                    elseif strcmpi(gss.View{i}(1).Geometry,'polygon')
                        for j=Sel'
                            x=gss.View{i}(j).X;
                            y=gss.View{i}(j).Y;

                            % check for multi parts features
                            indexNan=[0;find(isnan(x)==1)'];
                            for part=1:length(indexNan)-1
```

```
                                    beginFeat=indexNan(part);
                                    endFeat=indexNan(part+1);

                                    x1=[x(beginFeat+1:endFeat-1) x(beginFeat+1)];
                                    y1=[y(beginFeat+1:endFeat-1) y(beginFeat+1)];

                                    patch(x1,y1,clstColors(counter,:), ...
                                        'EdgeColor',clstColors(counter,:),...
                                        'UserData','Sel');
                                end
                            end
                        end
                    end
                end
            end
        elseif strcmp(gss.ViewLabel(i).Type,'UMAT')

            %check if figure is visible. If yes then represent features.
            % else, dont do nothing
            if ishandle(gss.ViewLabel(i).WindowHandle)
                figure(gss.ViewLabel(i).WindowHandle)

                %get patchs with userdata=sel and delete them
                p = get(gss.ViewLabel(i).WindowHandle,'children');

                for a=1:length(p)
                    if strcmpi(get(p(a),'Type'),'axes')
                        break
                    end
                end

                p = get(p(a),'children');
                delInd=zeros(1,length(p));
                for j=1:length(p)
                    if strcmpi(get(p(j),'userdata'),'Sel')
                        delInd(j)=1;
                    end
                end
                delete(p(delInd==1));

                % draw pie according to distribution
                if sum(gss.ClusterIndex)>0
                    %check if selection is based on item count or not (access the
                    %chkbox on the figure and check if it is on...
                    hChild=get(gss.ViewLabel(i).WindowHandle,'children');
                    tags=get(hChild,'tag');
                    index=find(strcmpi(tags,'selBasedOnCount')==1);

                    NumIndMaxPerCluster=0;
                    if isempty(index)==0
                        if get(hChild(index),'Value')
                            for j=unique(gss.ViewIndexData(:,i))'; % each unit
                                if j>0

clusters=gss.ClusterIndex(find(gss.ViewIndexData(:,i)==j));
                                    Contagem=max(histc(clusters,1:max(clusters)));
                                    if Contagem>NumIndMaxPerCluster
                                        NumIndMaxPerCluster=Contagem;
                                    end
                                end
                            end
                        end
                    end

                    %get bmus from this umat
                    for j=unique(gss.ViewIndexData(:,i))'; % each unit
                        if j>0
                            clusters=gss.ClusterIndex(find(gss.ViewIndexData(:,i)==j))';
                            histograma=[];
```

```
                            for cl=unique(clusters)
                                if cl~=0
                                    histograma=[histograma; cl sum(clusters==cl)];
                                end
                            end

                            if isempty(histograma)==0
                                %draw pie using the histogram
                                x=gss.View{i}(j).X;
                                xM=mean(x(1:end-1));
                                y=gss.View{i}(j).Y;
                                yM=mean(y(1:end-1));

                                v=histograma(:,2)';
                                v(v==0) = eps; % Matlab 5.2 version of linspace doesn't
work otherwise
                                N=25;
                                phi=[0 2*pi*cumsum(v./sum(v))];
                                xP=[];
                                yP=[];

                                pSize=1;
                                if NumIndMaxPerCluster>0
                                    pSize=sum(v)/NumIndMaxPerCluster*1.5;
                                end

                                for k=2:length(phi),
                                  [xi,yi]=pol2cart(linspace(phi(k-1),phi(k),N),pSize);
                                  xP(:,k-1)=[0 xi 0]';
                                  yP(:,k-1)=[0 yi 0]';
                                end
                                xP=xP+xM;
                                yP=yP+yM;

                                tcolor=[];
                                for t=1:size(histograma,1);
                                    tcolor(1,t,1:3) = clstColors(histograma(t,1),:);
                                end

                                patch(xP,yP,tcolor,...
                                    'UserData','Sel');
                            end
                        end
                    end
                end
            elseif strcmpi(gss.ViewLabel(i).Type,'Cp')
                %check if figure is visible. If yes then represent features.
                % else, dont do nothing
                if ishandle(gss.ViewLabel(i).WindowHandle)
                    figure(gss.ViewLabel(i).WindowHandle)


                    %get patchs with userdata=sel and delete them
                    p = get(gss.ViewLabel(i).WindowHandle,'children');
                    for a=1:length(p)
                        if strcmpi(get(p(a),'Type'),'axes')
                            break
                        end
                    end
                    p = get(p(a),'children');
                    delInd=zeros(1,length(p));
                    for j=1:length(p)
                        if strcmpi(get(p(j),'userdata'),'Sel')
                            delInd(j)=1;
                        end
                    end
                    delete(p(delInd==1));
```

```
            if sum(gss.ClusterIndex)>0
                clst=unique(gss.ClusterIndex)';
                clst(find(clst==0))=[];  %remove zero if exists
                counter=0;
                for cl=clst  %for each cluster
                    counter=counter+1;
                    Sel=unique(gss.ViewIndexData(find(gss.ClusterIndex==cl),i));

                    for j=Sel'
                        x=gss.View{i}(j).X;
                        x=[x(1:end-1) x(1)];
                        y=gss.View{i}(j).Y;
                        y=[y(1:end-1) y(1)];

                        patch(x,y,clstColors(counter,:),...
                            'EdgeColor',clstColors(counter,:),...
                            'FaceColor','none',...
                            'LineWidth',2,...
                            'UserData','Sel');
                    end
                end
            end
        end
elseif strcmpi(gss.ViewLabel(i).Type,'pcp')
        %check if figure is visible. If yes then represent features.
        % else, dont do nothing
        if ishandle(gss.ViewLabel(i).WindowHandle)
            figure(gss.ViewLabel(i).WindowHandle)

            %get patchs with userdata=sel and delete them
            p = get(gss.ViewLabel(i).WindowHandle,'children');
            for a=1:length(p)
                if strcmpi(get(p(a),'Type'),'axes')
                    break
                end
            end
            p = get(p(a),'children');
            delInd=zeros(1,length(p));
            for j=1:length(p)
                if strcmpi(get(p(j),'userdata'),'Sel')
                    delInd(j)=1;
                end
            end
            delete(p(delInd==1));

            if sum(gss.ClusterIndex)>0
                clst=unique(gss.ClusterIndex)';
                clst(find(clst==0))=[];  %remove zero if exists
                counter=0;

                for cl=clst  %for each cluster
                    counter=counter+1;
                    Sel=unique(gss.ViewIndexData(find(gss.ClusterIndex==cl),i));

                    for j=Sel'
                        x=gss.View{i}(j).X;
                        y=gss.View{i}(j).Y;

                        x=x(1:end-1);
                        y=y(1:end-1);

                        line(x,y,...
                            'Color',clstColors(counter,:),...
                            'UserData','Sel', 'LineWidth',3);
                    end
                end
            end
        end
    end
```

```
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_MapClick(h,evd,viewNumber)
    % geosom_MapClick - Click in a geosom geographic view
    % Syntax: geosom_MapClick(h,evd, viewNumber)
    % input
    % output
    %           h - handle
    %           evd -
    %           viewNumber - view number
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 04-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    seltype=get(gcf,'selectiontype');

    %check if the cross pointer is selected indicating that the select tool
    %is on
    if strcmp(get(gcf,'Pointer'),'cross')==1
        coord=get(gca,'CurrentPoint');
        coord=[coord(1,1),coord(1,2)];
    elseif strcmp(get(gcf,'Pointer'),'crosshair')==1
        %select the rectangle
        point1 = get(gca,'CurrentPoint');    % button down detected
        finalRect = rbbox;                   % return figure units
        point2 = get(gca,'CurrentPoint');    % button up detected
        point1 = point1(1,1:2);              % extract x and y
        point2 = point2(1,1:2);
        p1 = min(point1,point2);             % calculate locations
        offset = abs(point1-point2);         % and dimensions
        x = [p1(1) p1(1)+offset(1) p1(1)+offset(1) p1(1) p1(1)];
        y = [p1(2) p1(2) p1(2)+offset(2) p1(2)+offset(2) p1(2)];
        coord=[x' y'];
    elseif strcmp(get(gcf,'Pointer'),'topl')==1
        %select the rectangle
        point1 = get(gca,'CurrentPoint');    % button down detected
        finalRect = rbbox;                   % return figure units
        point2 = get(gca,'CurrentPoint');    % button up detected
        point1 = point1(1,1:2);              % extract x and y
        point2 = point2(1,1:2);
        p1 = min(point1,point2);             % calculate locations
        offset = abs(point1-point2);         % and dimensions
        x = [p1(1) p1(1)+offset(1) p1(1)+offset(1) p1(1) p1(1)];
        y = [p1(2) p1(2) p1(2)+offset(2) p1(2)+offset(2) p1(2)];
        coord=[x' y'];
    else
        return
    end

    indice=geosom_ClosestItem(coord,gss,viewNumber);
    %indice=geosom_ClosestItem([coord(1,1),coord(1,2)],gss,viewNumber);

    hh = gss.ViewLabel(viewNumber).GraphicHandle;
    if strcmpi(seltype,'normal')==1, %normal
        gss.SelectedIndex(:)=0;
        gss.SelectedIndex(indice)=1;
    elseif strcmpi(seltype,'Extend')==1, %shift - adds features to current selection
        gss.SelectedIndex(indice)=1;
```

```
    elseif strcmpi(seltype,'alt')==1, %control - rmeoves features to current selection
        gss.SelectedIndex(indice)=0;
    end


    %show selection in all the views
    if strcmp(get(gcf,'Pointer'),'cross')==1    % select using 1 color
        geosom_MakeSelectionAllOpenViews
    elseif strcmp(get(gcf,'Pointer'),'crosshair')==1    % select using 1 color
        geosom_MakeSelectionAllOpenViews
    elseif strcmp(get(gcf,'Pointer'),'topl')==1    % select using 1 color per BMU
        geosom_MakeSelectionAllOpenViewsByColor(viewNumber)
    end

    %show selection in table
    gss = geosom_TableDraw();
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function data = geosom_MlTable(fig, hObj, action, columnInfo, rowHeight, cell_data,
gFont, varargin)
% function data = geosom_MlTable(fig, hObj, action, columnInfo, rowHeight, cell_data,
gFont)
%
% Author: Morris Maynard
% Based on code by Gregory Gershanok
% Last update: 27 Jan 2005
%
% Manages a table with editing and scrolling capability
% Features: varying column widths, entry formatting, insert/delete rows,
%           editable or read-only cells, font control, scaled numeric
%           display, row selection highlighing, multiple tables per figure,
%           optional checkboxes on left-hand side
%
% Usage:
% Supply the parent figure, the handle of the axes object to use, the
% 'CreateTable' action, info about columns, and the cell data
%
% Example usage: (also just run with no arguments to see result)
%
% fig = nf;
% tbl = axes('units', 'pixels','position', [10 10 400 100]);
% cell_data = {...
%           'Alpha',   1, 2, 3,'';...
%           'Bravo',   4, 5, 6,'';...
%           'Charlie', 7, 8, 9,'';...
%           'Dog',    10,11,12,'';...
%           'Echo',   13,14,15,'';...
%           'Foxtrot',16,17,18,'';...
%           'Golf',   19,20,21,'';...
%           'Hotel',  26,27,28,'';...
%           };
%
% columninfo.titles={'Param','Lower Limit','Upper Limit','Initial Value','Result'};
% columninfo.formats = {'%4.6g','%4.6g','%4.6g','%4.6g', '%4.6g'};
% columninfo.weight =      [ 1, 1, 1, 1, 1];
% columninfo.multipliers = [ 1, 1, 1, 1, 1];
% columninfo.isEditable =  [ 1, 1, 1, 1, 0];
% columninfo.isNumeric =   [ 0  1, 1, 1, 1];
% columninfo.withCheck = true; % optional to put checkboxes along left side
% columninfo.chkLabel = 'Use'; % optional col header for checkboxes
% rowHeight = 16;
% gFont.size=9;
% gFont.name='Helvetica';
%
% geosom_MlTable(fig, tbl, 'CreateTable', columninfo, rowHeight, cell_data, gFont);
%
% To use in a GUIDE-created figure:
```

```
%
% Create a figure including a blank "axes" object with the tag 'tblParams'
% Put the lines starting with the "cell_data" line above into your figure's
% OpeningFcn, but replace the geosom_MlTable line with:
%
% geosom_MlTable(gcf, handles.tblParams, 'CreateTable', columninfo, rowHeight, cell_data,
gFont);
%% so clicking outside the table will finish edit in progress...
% endfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''SetCellValue'');', hObject,
handles.tblParams);
% set(hObject,'buttondownfcn',endfcn);
%
% To access the data edited by the table:
%
% info = get(tbl, 'userdata');
% data = info.data;
%

%--------------------------------------------------------------------------
% All functions dispatched from here.
% If necessary to call from figure use: geosom_MlTable(fig, hObj, 'Action',...)

global MINROWS;
MINROWS = 2;
if ~exist('action', 'var')
    data = mltest;
    return
end

switch(action)
  case 'CreateTable'
    data = createTable(fig, hObj, columnInfo, rowHeight, cell_data, gFont);
  case 'DestroyTable'
    data = destroyTable(fig, hObj);
  case 'ResizeTable'
    fig = resizeTable(fig, hObj);
  case 'ScrollData'
    fig = scrollData(fig, hObj);
  case 'EditCell'
    editCell(fig, hObj);
  case 'SetCellValue'
    setCellValue(fig, hObj);
  case 'EndEdit'
    setCellValue(fig, hObj);
  case 'AddRow'
    addRow(fig, hObj);
  case 'DelRow'
    delRow(fig, hObj);
  case 'SetOnSetCell'
    setOnSetCell(fig, hObj, varargin{:});
  case 'OnSetCell'
    data = onSetCell(fig, hObj, varargin{:});
  case 'SetDblClick'
    setDblClick(fig, hObj, varargin{:});
  case 'OnDblClick'
    onDblClick(fig, hObj, varargin{:});
  case 'OnCheck'
    onCheck(fig, hObj, varargin{1});
  case 'SetCheck'
    setCheck(fig, hObj, varargin{1}, varargin{2});
   case 'UnCheckAll'
    UnCheckAll(hObj);
end

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = createTable(fig, hObj, columnInfo, rowHeight, cell_data, gFont)
% Initially creates the table
%--------------------------------------------------------------------------
```

```
global MINROWS
data.figure = fig;
% get axes position in pixel coordinates
set(hObj, 'units', 'pixels');
set(hObj, 'visible', 'on');
pos_ax = get(hObj, 'position');
% set up grid info structure
ds = size(cell_data);
data.maxRows = ds(1);
if data.maxRows < MINROWS
    blanks = cell(1, ds(2));
    for ii = data.maxRows+1:MINROWS
        cell_data = [cell_data; blanks];
    end
    data.maxRows = MINROWS;
end
data.data = cell_data;
data.isChecked = zeros(1,size(cell_data, 1));
data.axes = hObj;
data.userModified = zeros(ds);
data.rowHeight = rowHeight;
data.columnInfo = columnInfo;
data.numCols= length(columnInfo.titles);
data.ltGray = [92 92 92]/255;
data.OffscreenPos = [-1000 -1000 30 20];
data.selectedRow = 0;
data.selectedCol = 0;
data.gFont = gFont;

data.doCheck = false;
if isfield(data.columnInfo, 'withCheck') && ...
    data.columnInfo.withCheck ~= 0
    data.doCheck = true;
end

% use 0...1 scaling on table x and y positions
set(fig, 'CurrentAxes', data.axes);
set(data.axes, 'box', 'on', 'DrawMode', 'fast');
set(data.axes, 'xlimmode', 'manual', 'xlim', [0 1], 'ylim', [0 1], ...
               'xtick', [], 'ytick', [], 'xticklabelmode', 'manual', 'xticklabel', []);

if data.doCheck % shrink on left for checkboxes column
    data.checkdx = pos_ax(3) * 20 * (1/pos_ax(3)); % chkbox offset
    pos_ax(1) = pos_ax(1) + data.checkdx;
    pos_ax(3) = pos_ax(3) - data.checkdx;
end
pos_ax(3) = pos_ax(3) - 10; % width of slider
set(data.axes, 'position', pos_ax, 'LineWidth', 2);
% callback for starting editing
editfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''EditCell'');',fig, hObj);
set(data.axes, 'ButtonDownFcn', editfcn);
% callback for scrolling table
scrfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''ScrollData'');',fig, hObj);
data.slider = uicontrol('style', 'slider', 'units', 'pixels',...
    'position', [pos_ax(1)+pos_ax(3)+2 pos_ax(2) 16 pos_ax(4)],...
    'Callback', scrfcn);

% Add buttons for addrow/delrow
if sum(columnInfo.isEditable) > 0 && (~isfield(columnInfo,'rowsFixed') ||...
        ~columnInfo.rowsFixed)
    btnw = 19; btnh = 15;
    btnx = pos_ax(1) + pos_ax(3) - btnw - 2;
    btny = pos_ax(2) + pos_ax(4) + 2;
    btnfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''AddRow'');',fig, hObj);
    data.btnAdd = uicontrol('style', 'pushbutton', 'units', 'pixels',...
        'position', [btnx, btny, btnw, btnh],...
        'string',' + ','fontsize',12,'Callback', btnfcn,...
        'TooltipString','Click to add a row');
    btnfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''DelRow'');',fig, hObj);
```

```
    data.btnDel = uicontrol('style', 'pushbutton', 'units', 'pixels',...
        'position', [btnx + btnw + 2, btny, btnw, btnh],...
        'string',' - ','fontsize',12,'Callback', btnfcn,...
        'TooltipString','Click to remove selected row');

    set(data.btnAdd,'Units','normalized');
    set(data.btnDel,'Units','normalized');
else
    data.btnAdd = [];
    data.btnDel = [];
end

set(hObj, 'UserData', data);
% so clicking outside the table will finish edit in progress
endfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''SetCellValue'');', fig, hObj);
set(fig,'buttondownfcn',endfcn);

resizeTable(fig, hObj);

return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = resizeTable(fig, hObj)
% fit table within boundaries and update scrollbar
% at this time doesn't handle figure resize
%--------------------------------------------------------------------------
data = get(hObj, 'UserData');
if isempty(data)
    return
end
% zap checkboxes if any
if data.doCheck && isfield(data, 'checks')
    delete(data.checks(:));
    data.checks = [];
end

cla(hObj);

set(hObj, 'units', 'pixels');
set(fig,'CurrentAxes',hObj);

pos_ax = get(hObj,'position');
data.numRows = floor((pos_ax(4)-(2*data.rowHeight))/data.rowHeight);
if data.numRows > data.maxRows
    data.numRows = data.maxRows;
end

unit_d_h = 1/pos_ax(3);
unit_d_v = 1/(pos_ax(4) - data.rowHeight);

if(data.numRows < data.maxRows)
    set(data.slider,'Units','pixels');
    set(data.slider, 'visible', 'on',...
                    'position', [pos_ax(1)+pos_ax(3)+1 pos_ax(2) 16 pos_ax(4)], ...
                    'min', 0,...
                    'max', data.maxRows - data.numRows, 'value', data.maxRows -
data.numRows, ...
                    'sliderstep', [1/(data.maxRows-data.numRows)
data.numRows/(data.maxRows-data.numRows)]);
else
  set(data.slider, 'visible', 'off');
end

% get gui units for rows and columns
% average column width and row height
d_x = 1/sum(data.columnInfo.weight);
d_y = 1/(data.numRows + 1);
% minimum adjust unit
```

```
unit_d_h = 1/(pos_ax(3));
unit_d_v = 1/(pos_ax(4) - data.rowHeight);

% Horizontal line positions
lx_h = ones(2, data.numRows+1);
lx_h(1, :) = 0;
ly_h = [1:data.numRows+1; 1:data.numRows+1]/(data.numRows+1);

% Vertical line positions
ly_v = ones(2, data.numCols);
ly_v(1, :) = 0;
lx_v = [d_x*data.numCols 2:data.numCols; d_x*data.numCols 2:data.numCols]/data.numCols;
for i = 2:data.numCols
  lx_v(:, i) = lx_v(:, i-1)+d_x*data.columnInfo.weight(i);
end
% draw initial grid
data.vertLines  = line(lx_v, ly_v);
data.vertLines1  = line(lx_v(:, 1:(data.numCols-1)), ly_v(:, 1:(data.numCols-1)));
data.horizLines = line(lx_h, ly_h);
set(data.horizLines, 'color', data.ltGray);
set(data.vertLines, 'color', data.ltGray, 'LineWidth', 2);
set(data.vertLines1, 'color', [1 1 1], 'LineWidth', 0.5);

% now display text in grid
txt_x = [0:data.numCols-1]/data.numCols + 4*unit_d_h;
for i = 2:data.numCols
  txt_x(i) = txt_x(i-1) + d_x*data.columnInfo.weight(i-1);
end
data.txt_x = txt_x;
data.txtCells = zeros(data.numRows, data.numCols);
uictx = get(hObj,'UIContextMenu');

chkdx = (pos_ax(3) * 20 * unit_d_h); % chkbox offset
chkdy = pos_ax(4) * d_y;

for j = 1:data.numRows
    txt_y = (data.numRows-j+1)/(data.numRows+1) * ones(1, data.numCols);
    txt_y = txt_y - (0.9*d_y); % reduce by (almost?) one row (title row)
    data.txtCells(j, :) = text(txt_x, txt_y, 'a','Clipping','on');
    if data.doCheck % put checkboxes to left of row
        poschk = [ pos_ax(1) - chkdx...
                 pos_ax(2) + 3 + chkdy * (data.numRows - j)...
                 10 10 ];
        data.checks(j) = ...
          uicontrol('style','checkbox','units','pixels','position', poschk);
        chkfcn = sprintf('geosom_MlTable(%14.13f, %14.13f,''OnCheck'',[], [], [], [],
%d);',...
          fig, hObj, j);
        set(data.checks(j), 'Units','normalized','Callback', chkfcn);
    end
    for i = 1:data.numCols
        if data.columnInfo.isNumeric(i)
            nums = data.data{j, i}/data.columnInfo.multipliers(i);
            nums = num2str(nums, data.columnInfo.formats{i});
            set(data.txtCells(j, i), 'string', nums);
        else
            set(data.txtCells(j, i), 'string', StripChars(data.data{j, i}));
        end
        set(data.txtCells(j, i), 'Position', [txt_x(i), txt_y(i)]);
        set(data.txtCells(j, i), 'UIContextMenu', uictx);
    end
end
set(data.txtCells(:, :), 'FontSize', data.gFont.size, 'FontName', data.gFont.name,
'FontWeight', 'normal', ...
                         'HorizontalAlignment','left','VerticalAlignment', 'bottom');
set(data.txtCells(1:data.numRows, 1:data.numCols), 'buttondownfcn', get(data.axes,
'ButtonDownFcn'));

% do title cells
```

```
titleCellsy = ones(1, data.numCols);
data.titleCells = text(txt_x, titleCellsy, data.columnInfo.titles);
set(data.titleCells(:, :), 'FontSize', data.gFont.size, 'FontName', data.gFont.name,
'FontWeight', 'bold', ...
                        'HorizontalAlignment','Center','VerticalAlignment',
'top','Editing','off');
for i = 1:length(data.titleCells)
    if i > 1
        titleOffset = 0.92 * (lx_v(1,i) - lx_v(1,i-1)) / 2;
    else
        titleOffset = (lx_v(1,i)) / 2;
    end
    pos = get(data.titleCells(i), 'position');
    pos(1) = pos(1) + titleOffset;
    set(data.titleCells(i), 'position', pos);
    if(data.columnInfo.isEditable(i) == 0)
        set(data.titleCells(i, :), 'FontWeight', 'normal');
        set(data.txtCells(:, i), 'FontWeight', 'normal');
    end
end
if data.doCheck && isfield(data.columnInfo,'chkLabel')
    %chkLbl = text(-25*unit_d_h, titleCellsy(1), data.columnInfo.chkLabel);
    %set(chkLbl, 'FontSize', data.gFont.size, 'FontName', data.gFont.name, 'FontWeight',
'bold', ...
    %           'HorizontalAlignment','left','VerticalAlignment', 'top','Editing','off');
    chkfcn = sprintf('geosom_MlTable(%14.13f, %14.13f,''UnCheckAll'',[], [], [], [],
%d);',...
    fig, hObj, j);
    data.selbutton = uicontrol('Style',
'pushbutton','units','pixels','position',[pos_ax(1) - 25 ...
                pos_ax(2)+15+ chkdy * (data.numRows - 1) 22 20], 'String',
data.columnInfo.chkLabel,...
                'Callback', chkfcn);
end

row = 1;
x1 = 0; x2 = 1;
y1 = (data.numRows - row + 1)/(data.numRows+1);
y2 = y1 + (1/(data.numRows+1)) + .01;
makepatch(hObj,x1,x2,y1,y2, 0.753);

set(hObj, 'UserData', data);

scrollData(fig, hObj);
set(hObj, 'units', 'normalized');
return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = scrollData(fig, hObj)
% handle scrollbar (slider) callback
%--------------------------------------------------------------------------
data = get(hObj, 'UserData');
if isempty(data)
    return
end

set(data.slider,'Units','pixels');
if isfield(data,'editBox') && ishandle(data.editBox)
    delete(data.editBox);
end

% handle non-scroll case in case slider was switched off
if(strcmp(get(data.slider, 'visible'), 'off') == 1)
    ind0 = 0;
    for i = 1:data.numRows
        if data.doCheck
            set(data.checks(i),'value',data.isChecked(i+ind0));
        end
```

```
        for j = 1:data.numCols
            if data.columnInfo.isNumeric(j)
                nums = data.data{i, j}/data.columnInfo.multipliers(j);
                nums = num2str(nums, data.columnInfo.formats{j});
                set(data.txtCells(i, j), 'string', nums);
            else
                set(data.txtCells(i, j), 'string', data.data{i, j});
            end
        end
    end
else
    val = get(data.slider, 'Value');
    max_val = get(data.slider, 'Max');
    min_val = get(data.slider, 'Min');

    val0 = data.maxRows - data.numRows;
    ind0 = round(val0-val);
    % move the text to give illusion of scrolling
    for i = ind0+1:(ind0+data.numRows)
      if data.doCheck
          set(data.checks(i-ind0),'value',data.isChecked(i));
      end
      for j = 1:data.numCols
            if data.columnInfo.isNumeric(j)
                nums = data.data{i, j}/data.columnInfo.multipliers(j);
                nums = num2str(nums, data.columnInfo.formats{j});
                set(data.txtCells(i-ind0, j), 'string', nums);
            else
                set(data.txtCells(i-ind0, j), 'string', data.data{i, j});
            end
      end
    end
end
% save scroll position
data.ind0 = ind0;

data.hpatch = remakepatch(data.selectedRow, data, hObj);

set(data.slider,'Units','normalized');
set(hObj, 'UserData', data);
return

% --------------------------------------------------------------------
% --------------------------------------------------------------------
function [hpatch] = makepatch(hObj, x1, x2, y1, y2, co)
% --------------------------------------------------------------------
    if ~exist('co','var')
        co = 0.88;
    end
    hpatch = ...
        patch([x1 x2 x2 x1 x1],[y1 y1 y2 y2 y1],[co co co],...
        'HitTest','off','FaceAlpha',0.2);
    ch=get(hObj,'children');
    % put new patch at bottom layer
    c1 = ch(1); % latest object
    ch = ch(2:end); % shift up
    ch(size(ch,1)+1) = c1; % append new
   set(hObj,'children',ch);
return

% --------------------------------------------------------------------
% --------------------------------------------------------------------
function [hpatch] = remakepatch(row, data, hObj)
% --------------------------------------------------------------------
hpatch = []; % empty return if nothing to do
% see if selected row is visible
if ((row - data.ind0) <= (data.numRows) && (row - data.ind0) > 0)
    % yes, compute row coordinates
    row = row - data.ind0 + 1;
```

```
    x1 = 0; x2 = 1;
    y1 = (data.numRows - row + 1)/(data.numRows+1);
    y2 = y1 + (1/(data.numRows+1)) + .005;
    % see if a previous patch exists
    if isfield(data,'hpatch') && ~isempty(data.hpatch) && ishandle(data.hpatch)
        % yes, see if it is on same row
        yp = get(data.hpatch,'ydata');
        if y1 ~= yp(1)
            % no, delete old patch and make new one
            delete(data.hpatch);
            data.hpatch = makepatch(hObj,x1,x2,y1,y2);
        end
    else % no previous patch exists, make new one
        data.hpatch = makepatch(hObj,x1,x2,y1,y2);
    end
    hpatch = data.hpatch; % return new or previous patch
else % if patch is no longer visible, delete it
    if isfield(data,'hpatch') && ~isempty(data.hpatch) && ishandle(data.hpatch)
        delete(data.hpatch);
    end
end
return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = destroyTable(fig, hObj)
% Destroys the table
%--------------------------------------------------------------------------
    data = get(hObj, 'UserData');
    if ~isempty(data)
        set(hObj, 'visible','off');
        cla(hObj);

        if isfield(data.columnInfo, 'withCheck') && ...
            data.columnInfo.withCheck ~= 0 && ...
            isfield(data, 'checks')
            delete(data.checks(:));
            data.checks = [];
        end
         % restore orig size
        set(data.axes, 'Units', 'pixels');
        pos_ax = get(data.axes, 'position');
        if data.doCheck % restore on left for checkboxes column
            pos_ax(1) = pos_ax(1) - data.checkdx;
            pos_ax(3) = pos_ax(3) + data.checkdx;
        end
        pos_ax(3) = pos_ax(3) + 10; % width of slider;
        set(data.axes, 'position', pos_ax);
        set(data.axes, 'Units', 'normalized');
        delete(data.slider);
        delete(data.btnAdd);
        delete(data.btnDel);
        %roberto
        delete(data.selbutton);

        data = [];
        set(hObj, 'UserData', data);
    end
return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = editCell(fig, hObj)
% put an edit control over the selected cell
%--------------------------------------------------------------------------
persistent lasttime;

data = get(hObj, 'UserData');
if isempty(data)
```

```
    return
end

if isempty(lasttime)
    tic;
    lasttime = toc - .400;
end
thistime = toc;
if (thistime - lasttime) < .350
    geosom_MlTable(fig, hObj, 'OnDblClick');
    return
end
lasttime = thistime;

set(hObj, 'units', 'pixels');
pt = get(hObj, 'CurrentPoint');
pt=pt(1,:); % strip out 2nd axis info
pos_ax = get(hObj, 'position');
pt(1) = pos_ax(1) + (pt(1) * pos_ax(3));
pt(2) = pt(2) .* pos_ax(4);

d_x = pos_ax(3)/sum(data.columnInfo.weight);
d_y = pos_ax(4)/(data.numRows+1);

% find column index
col = -1;
p1 = 0;
p2 = 0;
for i = 1:data.numCols
   p2 = p1 + d_x*data.columnInfo.weight(i);
   if((p1 <= (pt(1)-pos_ax(1))) & (p2 >= (pt(1)-pos_ax(1))))
     col = i;
     break;
   else
     p1 = p2;
   end;
end
if(col == -1)
  set(hObj, 'units', 'normalized');
  return;
end

% find row index
row = data.numRows - (floor(pt(2) / d_y));

if(row < 1) % could be header row
  set(hObj, 'units', 'normalized');
  return;
end

data.selectedCol = col;
data.selectedRow = row + data.ind0;

if isfield(data, 'editBox') && ishandle(data.editBox)
    delete(data.editBox);
end

data.hpatch = remakepatch(data.selectedRow, data, hObj);

% continue only if editable
if(0 ~= data.columnInfo.isEditable(col) && row <= data.numRows)
    unit_d_h = 1/pos_ax(3);
    unit_d_v = 1/pos_ax(4);
    % handle numeric (or not) data
    if data.columnInfo.isNumeric(col)
        ebtxt = data.data{row + data.ind0, col}/data.columnInfo.multipliers(col);
        ebtxt = num2str(ebtxt, data.columnInfo.formats{col});
    else
        ebtxt = UnStripChars(data.data{row + data.ind0, col});
```

```
    end
    % set the edt control contents and position
    % callback for entering cell data
    endfcn = sprintf('geosom_MlTable(%14.13f, %14.13f, ''SetCellValue'');',fig, hObj);
    data.editBox = uicontrol('style', 'edit', 'units', 'pixels',...
        'Callback', endfcn);
    set(data.editBox, 'FontSize', data.gFont.size, 'FontName', data.gFont.name,...
        'FontWeight', 'normal');
    set(data.editBox, 'string', ebtxt, 'UserData', [row col]);
    ext_eb = get(data.editBox, 'extent');
    ext_eb(4) = d_y + 3;
    pos = [(pos_ax(1)-unit_d_h+p1) ,...
            pos_ax(2) + ...                              % start of table
              ceil((data.numRows - row) * d_y) + ...    % cvt index to row #, get offset
from ystart
                  (ceil(d_y) - ext_eb(4))/2, ...          % add half of the ctrl height??
            floor(d_x*data.columnInfo.weight(col))-unit_d_h, ...
            ext_eb(4)];
    % fprintf(1, 'Click at point (%3.2f, %3.2f) on cell (%d, %d) coordinates [%3.2f %3.2f
%3.2f %3.2f]\n', ...
    %            pt(1), pt(2), col, row, pos(1), pos(2), pos(3), pos(4));
    set(data.editBox, 'Position', pos, 'HorizontalAlignment' ,'Left');
    set(fig, 'CurrentObject', data.editBox);
    %set(data.editBox,'Editing','true');
end
set(hObj, 'UserData', data);
set(hObj, 'units', 'normalized');
return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = setCellValue(fig, hObj)
% when edit control calls back, update data in cell
%--------------------------------------------------------------------------
data = get(hObj, 'UserData');
if isempty(data)
    return
end

if ~isfield(data,'editBox') || ~ishandle(data.editBox)
    return
end

ind = get(data.editBox, 'UserData');
if isempty(ind)
    return;
end;

nums = StripChars(get(data.editBox, 'string'));
row = ind(1) + data.ind0; col = ind(2);
d_old = data.data{row, col};
if data.columnInfo.isNumeric(col)
    num = sscanf(nums, '%f');
    if(isempty(num))
      errordlg('Please enter a valid number', 'Error', 'modal');
      return;
    end
    d_new = num*data.columnInfo.multipliers(col);
    if(d_old == d_new)
       delete(data.editBox);
       return;
    end
    if geosom_MlTable(fig, hObj, 'OnSetCell', [], [], [], [], d_old, d_new(1))
        nums = num2str(num, data.columnInfo.formats{col});
        data.data{row, col} = d_new;
    else
        nums = num2str(d_old/data.columnInfo.multipliers(col),
data.columnInfo.formats{col});
    end
```

```
else
    if geosom_MlTable(fig, hObj, 'OnSetCell', [], [], [], [], data.data{row, col}, nums)
        data.data{row, col} = nums;
    else
        nums = data.data{row, col};
    end
end

% need to check handles, since OnSetCell callback may have refreshed table
if ishandle(data.editBox)
    delete(data.editBox);
end
if ishandle(data.txtCells(row - data.ind0, col))
    set(data.txtCells(row - data.ind0, col), 'string', nums);
    set(hObj,'UserData',data);
end

return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = addRow(fig, hObj)
% insert a row into the table
%--------------------------------------------------------------------------
data = get(hObj, 'UserData');
if isempty(data)
    return
end

% increase the number of rows
data.maxRows = data.maxRows + 1;
selRow = data.selectedRow;
if selRow < 1
    selRow = 1;
end

% move data from new row and following down one row
dtmp = data.data;
ctmp = data.isChecked;

for jj = size(dtmp, 1):-1:selRow % for subsequent rows
    ctmp(jj+1) = ctmp(jj);
    for ii = 1:size(dtmp,2) % for all columns
        dtmp{jj+1, ii} = data.data{jj, ii};
    end
end
% zap data in new row
ctmp(selRow) = 0;
for ii = 1:size(dtmp,2) % for all columns
    dtmp{selRow, ii} = '';
end

data.data = dtmp;
data.isChecked = ctmp;

set(hObj, 'UserData', data);
resizeTable(fig, hObj);
return

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
function data = delRow(fig, hObj)
% delete the currently selected row
%--------------------------------------------------------------------------
global MINROWS

data = get(hObj, 'UserData');
if isempty(data)
    return
```

```
end

selRow = data.selectedRow;
if selRow < 1
    selRow = data.maxRows;
end

if data.maxRows < 2
    return
end

% decrease the number of rows
data.maxRows = data.maxRows - 1;
mr = data.maxRows;
cols = size(data.data,2);
dtmp = {};
ctmp = [];
% remove the selected row
% copy previous data
for ii = 1:selRow-1
    ctmp(ii) = data.isChecked(ii);
    for jj = 1:cols
        dtmp{ii, jj} = data.data{ii, jj};
    end
end
% copy following data
for ii = selRow+1:size(data.data,1)
    ctmp(ii-1) = data.isChecked(ii);
    for jj = 1:cols
        dtmp{ii-1, jj} = data.data{ii, jj};
    end
end

if data.maxRows < MINROWS
    blanks = cell(1, size(dtmp, 2));
    for ii = data.maxRows+1:MINROWS
        dtmp = [dtmp; blanks];
        ctmp(ii) = 0;
    end
    data.maxRows = MINROWS;
end

data.data = dtmp;
data.isChecked = ctmp;

set(hObj, 'UserData', data);
resizeTable(fig, hObj);
return


% --------------------------------------------------------------------------
% --------------------------------------------------------------------------
function [] = setOnSetCell(fig, hObj, varargin);
% establishes an action for double-clicking on a cell
% call as "geosom_MlTable(fig, hobj, 'SetDblClick', 'Myfunc', 'fmt_str', args..."
% where "fmt_str" and args are optional
% function name supplied will be called as
% [] = Myfunc(hObject, [], handles, arg1, arg2, ...)
% --------------------------------------------------------------------------
  sfig  = varargin{1};
  sfunc = varargin{2};
  data = get(hObj, 'UserData');
  data.OnSetCellFcn{1} = sfig;
  data.OnSetCellFcn{2} = sfunc;
%  if nargin < -3
  if nargin > 4
      data.SetCellFmt = varargin{3};
  end
  set(hObj, 'UserData', data);
```

```
return

% --------------------------------------------------------------------
% --------------------------------------------------------------------
function [res] = onSetCell(fig, hObj, varargin);
% establishes an action for finising a cell edit
% --------------------------------------------------------------------
    res = true;
    data = get(hObj, 'UserData');
    if ~isfield(data,'OnSetCellFcn')
        return
    end
    % start with func name, hObject (fig), evdata ([]), and handles
    sfmt  = '%s(''%s'', %14.13f, [], handles';
    sfig  = data.OnSetCellFcn{1};
    sfunc = data.OnSetCellFcn{2};
    if isfield(data,'SetCellFmt')
        sfmt = [sfmt ', ' data.SetCellFmt];
    end
    handles = guidata(fig);
    fcn = sprintf(sfmt, sfig, sfunc, hObj, varargin{:});
    fcn = [fcn ');'];
    try
        res = eval(fcn);
    catch
        res = [];
        disp(['error in OnSetCell callback:\n' lasterr]);
    end
return

% --------------------------------------------------------------------
% --------------------------------------------------------------------
function [] = setDblClick(fig, hObj, varargin);
% establishes an action for double-clicking on a cell
% call as "geosom_MlTable(fig, hobj, 'SetDblClick', 'Myfunc', 'fmt_str', args..."
% where "fmt_str" and args are optional
% function name supplied will be called as
% [] = Myfunc(hObject, [], handles, arg1, arg2, ...)
% --------------------------------------------------------------------
  sfig  = varargin{1};
  sfunc = varargin{2};
  data = get(hObj, 'UserData');
  data.OnDblClickFcn{1} = sfig;
  data.OnDblClickFcn{2} = sfunc;
  if nargin < -3
      data.DblClickFmt = varargin{2};
  end
  set(hObj, 'UserData', data);
return

% --------------------------------------------------------------------
% --------------------------------------------------------------------
function [] = onDblClick(fig, hObj, varargin);
% establishes an action for double-clicking on a cell
% --------------------------------------------------------------------
    data = get(hObj, 'UserData');
    if ~isfield(data,'OnDblClickFcn')
        return
    end
    % start with func name, hObject (fig), evdata ([]), and handles
    sfmt  = '%s(''%s'', %14.13f, [], handles';
    sfig  = data.OnDblClickFcn{1};
    sfunc = data.OnDblClickFcn{2};
    if isfield(data,'DblClickFmt')
        sfmt = [sfmt ', ' data.DblClickFmt];
    end
    sfmt = [sfmt ');'];
    handles = guidata(fig);
    fcn = sprintf(sfmt, sfig, sfunc, hObj, varargin{:});
```

```
    try
        eval(fcn);
    end
return

% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
function [] = onCheck(fig, hObj, row);
% What happens when a user clicks on a row's checkbox
% ------------------------------------------------------------------------
    data = get(hObj, 'UserData');
    rowchk = row + data.ind0;
    vchk = get(data.checks(row),'value');
    data.isChecked(rowchk) = vchk;
    set(hObj,'UserData',data);
return

% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
function [] = setCheck(fig, hObj, row, vchk);
% allows parent to set the data row's checked state
% ------------------------------------------------------------------------
    data = get(hObj, 'UserData');
    if row > data.maxRows
        return
    end
    if row - data.ind0 <= data.numRows
        set(data.checks(row - data.ind0),'value', vchk);
    end
    data.isChecked(row) = vchk;
    set(hObj,'UserData',data);
return


% ------------------------------------------------------------------------
% ---------------------------------------------------------------------
function [sout] = StripChars(sin)
    sbad = '_';
    sout = sin;
    stmp = [];
    [sf, sr] = strtok(sin, sbad);
    while ~isempty(sr) % found a '_' char
        stmp = [stmp sf '\_']; % replace with '\_'
        [sf, sr] = strtok(sr, sbad);
    end
    if ~isempty(stmp)
        stmp = [stmp sf];
        sout = stmp;
    end
return

% ---------------------------------------------------------------------
function [sout] = UnStripChars(sin)
    sbad = '\\';
    sout = sin;
    stmp = [];
    [sf, sr] = strtok(sin, sbad);
    while ~isempty(sr) % found a '\' char
        stmp = [stmp sf ]; % remove '\'
        [sf, sr] = strtok(sr, sbad);
    end
    if ~isempty(stmp)
        stmp = [stmp sf];
        sout = stmp;
    end
return

function result = trim(string)
[nR, nC] = size(string);
```

```
indexStart = 1;
indexEnd   = nC;

for i = 1:nC
  if(string(1, indexStart) == ' ')
    indexStart = indexStart + 1;
  else
    break;
  end
end

for i = nC:-1:1
  if(string(1, indexEnd) == ' ')
    indexEnd = indexEnd - 1;
  else
    break;
  end
end

result = string(indexStart:indexEnd);

% --------------------------------------------------------------------
% --------------------------------------------------------------------
function [data] = mltest
% function to test the table
% --------------------------------------------------------------------
    fig = figure('renderer','zbuffer');
    pos = get(fig, 'position');
    pos(3) = 440;pos(4)=160;
    set(fig,'position',pos);

    tbl = axes('units', 'pixels','position', [20 20 400 120]);
    cell_data = {...
              'Alpha',   1, 2, 3,'';...
              'Bravo',   4, 5, 6,'';...
              'Charlie', 7, 8, 9,'';...
              'Dog',    10,11,12,'';...
              'Echo',   13,14,15,'';...
              'Foxtrot',16,17,18,'';...
              'Golf',   19,20,21,'';...
              'Hotel',  26,27,28,'';...
              };

    columninfo.titles={'Param','Lower Limit','Upper Limit','Initial Value','Result'};
    columninfo.formats = {'%4.6g','%4.6g','%4.6g','%4.6g', '%4.6g'};
    columninfo.weight =      [ 1, .85, .85, .85, .85];
    columninfo.multipliers = [ 1, 1, 1, 1, 1];
    columninfo.isEditable =  [ 1, 1, 1, 1, 0];
    columninfo.isNumeric =   [ 0  1, 1, 1, 1];
    columninfo.withCheck = true;
    columninfo.chkLabel = 'Use';
    rowHeight = 16;
    gFont.size=9;
    gFont.name='Helvetica';

    data = geosom_MlTable(fig, tbl, 'CreateTable', columninfo, rowHeight, cell_data,
gFont);
return

function UnCheckAll(hObj)
%function created by Roberto Henriques 23/06/2008
% this function will (un)check all the checkboxes

    data = get(hObj, 'UserData');
    vchk = get(data.checks(1),'value');
    data.isChecked=ones(length(data.isChecked),1)*not(vchk);
    set(data.checks, 'Units','normalized','Value', not(vchk));
    set(hObj, 'UserData', data);
```

```
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function output=geosom_NewParallelCoordinatePlot()
    %GEOSOM_NEWPARALLELCOORDINATEPLOT ...
    % Syntax: output=geosom_NewParallelCoordinatePlot()
    %
    %
    % Subfunctions: NewPCP_CloseRequest, untitled_OutputFcn, pushOK_Callback
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 30-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

    global gss

    fh=figure('Position',[300 300 300 200]);
    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
        'Resize','off','name','Create a new PCP');
    handles.NewPCP=fh;

    % text - title
    h = uicontrol('Style', 'text',...
           'String', 'Please select the variables to include:',...
           'tag', 'textViews',...
           'Position', [10 180 280 15],...
           'HorizontalAlignment','Center',...
           'fontweight','bold');
    handles.textViews=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % components to select
    h = axes('tag', 'TableComponents','AmbientLightColor','white',...
           'units','pixels','Position', [30 50 250 100]);
    handles.TableComponents=h;

    % replace the character _ by space
    Comp = gss.NumDataLabel';
    for i=1:length(Comp)
        Comp{i}=strrep(Comp{i}, '_', ' ');
    end

    columninfo.titles={'Component'};
    columninfo.formats = {'%4.6g'};
    columninfo.weight =      [1];
    columninfo.multipliers = [1];
    columninfo.isEditable =  [0];
    columninfo.isNumeric =   [0];
    columninfo.withCheck = true; % optional to put checkboxes along left side
    columninfo.chkLabel = 'Use'; % optional col header for checkboxes
    rowHeight = 12;
    gFont.size=8;
    gFont.name='Arial';
    geosom_MlTable(handles.NewPCP, handles.TableComponents, 'CreateTable',
columninfo,rowHeight, Comp, gFont);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'OK',...
           'tag', 'pushOK',...
           'Position', [220 10 50 15],...
           'Callback', 'setmap');
    handles.pushOK=h;
```

```
    %%% CALLBACKS with handles %%%
    set(handles.NewPCP,'CloseRequestFcn',{@NewPCP_CloseRequest handles});
    set(handles.pushOK,'Callback',{@pushOK_Callback handles});

    % Make the GUI modal
    set(handles.NewPCP ,'WindowStyle','modal')
    % UIWAIT
    uiwait(handles.NewPCP);

    % check if any variable was selected.
    if ishandle(handles.TableComponents)
        comp = get(handles.TableComponents, 'userdata');
        output=find(comp.isChecked==1);

        % close form
        close(handles.NewPCP);
    else
        output=[];
    end
return

function NewPCP_CloseRequest(hObject, eventdata,handles)
    uiresume(handles.NewPCP);
    delete (handles.NewPCP);
return

function varargout = untitled_OutputFcn(hObject, eventdata, handles)
    % varargout  cell array for returning output args (see VARARGOUT);
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Get default command line output from handles structure
    varargout{1} = handles.output;

    % The figure can be deleted now
    delete(handles.figure1);
return

function pushOK_Callback(hObject, eventdata, handles)
    % components
    comp = get(handles.TableComponents, 'userdata');
    SelectedComponents=find(comp.isChecked==1);

    if isempty(SelectedComponents)
        helpdlg('Please select at least one variable!','New PCP');
        return
    end

    %resume
    uiresume(handles.NewPCP);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_Preferences()
%GEOSOM_PREFERENCES ...
% Syntax: geosom_Preferences()
%
% Subfunctions: pushClustCmap_Callback, pushCmap_Callback, pushSetCmap_Callback,
pushSetClusterCmap_Callback, SelColor, pushcmapOutlineColor_Callback,
chkOutline_Callback, chkHitsOutline_Callback, chkcmapOutline_Callback, pushSave_Callback
%   See also:
%
% Author: Roberto Henriques, Victor Lobo, Fernando Bação
% Created: December 30-2009
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2009
```

```
global mainHandles

fh=figure('Position',[300 300 400 250]);
% clear default menu and toolbars and add title
set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
    'Resize','off','name','GeoSom Preferences');
handles.Prefs=fh;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% panel colormaps
hp = uipanel('Title','Preferences',...
             'units','pixels','Position',[10 50 380 180]);
%colormap
a(:,:,1) = repmat(mainHandles.prefs.cmap(:,1)',16,1);
a(:,:,2) = repmat(mainHandles.prefs.cmap(:,2)',16,1);
a(:,:,3) = repmat(mainHandles.prefs.cmap(:,3)',16,1);
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Colormap',...
       'tag', 'textUmatCmap',...
       'Position', [10 140 80 15],...
       'HorizontalAlignment','left');
%colormap sel.
h = uicontrol('Style', 'pushbutton',...
        'cdata', a,...
        'Parent',hp,...
        'tag', 'pushCmap',...
        'Position', [100 140 64 20]);
handles.pushCmap=h;
%set colormap
h = uicontrol('Style', 'pushbutton',...
       'Parent',hp,...
       'String', 'Set',...
       'Position', [165 140 20 20]);
handles.pushSetCmap=h;

%add outline
h = uicontrol('Style', 'checkbox',...
       'Parent',hp,...
       'value',mainHandles.prefs.cmapOutline,...
       'String', 'Add outline',...
       'tag', 'chkOutline',...
       'Position', [190 140 150 20]);
handles.chkcmapOutline=h;

if mainHandles.prefs.cmapOutline==0
    enb='off';
    mainHandles.prefs.cmapOutlineColor=[.7 .7 .7];
else
    enb='on';
end

h = uicontrol('Style', 'pushbutton',...
       'Parent',hp,...
        'backgroundcolor', mainHandles.prefs.cmapOutlineColor,...
       'String', '...',...
       'tag', 'pushCmap',...
       'enable',enb,...
       'Position', [280 140 20 20]);
handles.pushcmapOutlineColor=h;

%selection color
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Selection color',...
       'Position', [10 120 80 15],...
       'HorizontalAlignment','left');
h = uicontrol('Style', 'pushbutton',...
       'Parent',hp,...
```

```
       'backgroundcolor', mainHandles.prefs.selcolor,...
       'String', '...',...
       'tag', 'selcolor',...
       'Position', [100 120 20 20]);
handles.pushSelCol=h;

%check only outline
h = uicontrol('Style', 'checkbox',...
       'Parent',hp,...
       'value',mainHandles.prefs.onlyoutline,...
       'String', 'Only outline selection',...
       'tag', 'chkOutline',...
       'Position', [190 120 150 20]);
handles.chkOutline=h;

%hits color
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Hits color',...
       'tag', 'textSelColor',...
       'Position', [10 100 80 15],...
       'HorizontalAlignment','left');

h = uicontrol('Style', 'pushbutton',...
       'Parent',hp,...
       'backgroundcolor', mainHandles.prefs.hitscolor,...
       'String', '...',...
       'tag', 'hitscolor',...
       'Position', [100 100 20 20],...
       'Callback', {@pushHitsCol_Callback handles});
handles.pushHitColor=h;

%check only outline for hits
h = uicontrol('Style', 'checkbox',...
       'Parent',hp,...
       'value',mainHandles.prefs.onlyhitsoutline,...
       'String', 'Only outline selection',...
       'tag', 'chkHitsOutline',...
       'Position', [190 100 150 20]);
handles.chkHitsOutline=h;

%%%%%%%% clusters %%%%%%%%
% clusters colormap
a(:,:,1) = repmat(mainHandles.prefs.clusterscmap(:,1)',16,1);
a(:,:,2) = repmat(mainHandles.prefs.clusterscmap(:,2)',16,1);
a(:,:,3) = repmat(mainHandles.prefs.clusterscmap(:,3)',16,1);
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Clust. Colormap',...
       'tag', 'textUmatCmap',...
       'Position', [10 80 100 15],...
       'HorizontalAlignment','left');
%colormap sel. button
h = uicontrol('Style', 'pushbutton',...
       'Parent',hp,...
       'cdata', a,...
       'Position', [100 80 64 20]);
handles.pushClustCmap=h;
%set colormap
h = uicontrol('Style', 'pushbutton',...
       'Parent',hp,...
       'String', 'Set',...
       'Position', [165 80 20 20]);
handles.pushSetClusterCmap=h;
% draw clusters color
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Draw Clust. color',...
       'Position', [10 60 90 15],...
```

```
        'HorizontalAlignment','left');
    h = uicontrol('Style', 'pushbutton',...
        'Parent',hp,...
        'backgroundcolor', mainHandles.prefs.DrawClusterColor,...
        'String', '...',...
        'tag', 'DrawClusterColor',...
        'Position', [100 60 20 20]);
    handles.pushDrawClusterColor=h;
    % clusters color
    h = uicontrol('Style', 'text',...
        'Parent',hp,...
        'String', 'Clusters color',...
        'Position', [10 40 90 15],...
        'HorizontalAlignment','left');
    h = uicontrol('Style', 'pushbutton',...
        'Parent',hp,...
        'String', '...',...
        'backgroundcolor', mainHandles.prefs.ClusterColor,...
        'tag', 'ClusterColor',...
        'Position', [100 40 20 20]);
    handles.pushClusterColor=h;
    % sel. clusters color
    h = uicontrol('Style', 'text',...
        'Parent',hp,...
        'String', 'Sel. Clust. color',...
        'Position', [10 20 90 15],...
        'HorizontalAlignment','left');
    h = uicontrol('Style', 'pushbutton',...
        'Parent',hp,...
        'backgroundcolor', mainHandles.prefs.SelClusterColor,...
        'String', '...',...
        'tag', 'SelClusterColor',...
        'Position', [100 20 20 20]);
    handles.pushSelClusterColor=h;

    %%%%%%%% flood %%%%%%%%
    % flood color
    h = uicontrol('Style', 'text',...
        'Parent',hp,...
        'String', 'z-level color',...
        'Position', [10 1 80 15],...
        'HorizontalAlignment','left');
    h = uicontrol('Style', 'pushbutton',...
        'Parent',hp,...
        'backgroundcolor', mainHandles.prefs.floodcolor,...
        'tag','floodcolor',...
        'String', '...',...
        'Position', [100 1 20 20]);
    handles.pushFloodColor=h;

    %save button
    h = uicontrol('Style', 'pushbutton',...
        'String', 'Save Prefs',...
        'tag', 'pushSave',...
        'Position', [330 20 60 20]);
    handles.pushSave=h;

    % callbacks
    set(handles.pushCmap,'Callback',{@pushCmap_Callback handles});
    set(handles.pushSelCol,'Callback',{@SelColor});
    set(handles.pushSetCmap,'Callback',{@pushSetCmap_Callback handles});
    set(handles.chkOutline,'Callback',{@chkOutline_Callback handles});
    set(handles.pushHitColor,'Callback',{@SelColor});
    set(handles.chkHitsOutline,'Callback',{@chkHitsOutline_Callback handles});
    set(handles.chkcmapOutline,'Callback',{@chkcmapOutline_Callback handles});
    set(handles.pushcmapOutlineColor,'Callback',{@pushcmapOutlineColor_Callback
handles});
    set(handles.pushClustCmap,'Callback',{@pushClustCmap_Callback handles});
    set(handles.pushSetClusterCmap,'Callback',{@pushSetClusterCmap_Callback handles});
```

```
    set(handles.pushDrawClusterColor,'Callback',{@SelColor});
    set(handles.pushClusterColor,'Callback',{@SelColor});
    set(handles.pushSelClusterColor,'Callback',{@SelColor});
    set(handles.pushFloodColor,'Callback',{@SelColor});
    set(handles.pushSave,'Callback',{@pushSave_Callback handles});

    % Make the GUI modal
    set(handles.Prefs ,'WindowStyle','modal')
    % UIWAIT
    uiwait(handles.Prefs);
return

function pushClustCmap_Callback(hObject, eventdata, handles)
    global mainHandles

    set(handles.Prefs,'Colormap',mainHandles.prefs.clusterscmap);

    % open colormap editor
    colormapeditor (handles.Prefs);
    %mainHandles.prefs.cmap=colormap;
return

function pushCmap_Callback(hObject, eventdata, handles)
    global mainHandles

    set(handles.Prefs,'Colormap',mainHandles.prefs.cmap);
    % open colormap editor
    colormapeditor (handles.Prefs);
return

function pushSetCmap_Callback(hObject, eventdata, handles)
    global mainHandles
    % this button only exists because colormapeditor does not work properly.

    %so we have to:
    %1 define the figure colormap as the handles.prefs.cmap
    %2 give a color to the colormap selection button

    mainHandles.prefs.cmap=colormap;
    %set(hObject,'CData',a)
    a(:,:,1) = repmat(mainHandles.prefs.cmap(:,1)',16,1);
    a(:,:,2) = repmat(mainHandles.prefs.cmap(:,2)',16,1);
    a(:,:,3) = repmat(mainHandles.prefs.cmap(:,3)',16,1);
    set(handles.pushCmap,'CData', a);
return

function pushSetClusterCmap_Callback(hObject, eventdata, handles)
    global mainHandles
    % this button only exists because colormapeditor does not work properly.

    %so we have to:
    %1 define the figure colormap as the handles.prefs.clustercmap
    %2 give a color to the colormap selection button

    mainHandles.prefs.clusterscmap=colormap;
    a(:,:,1) = repmat(mainHandles.prefs.clusterscmap(:,1)',16,1);
    a(:,:,2) = repmat(mainHandles.prefs.clusterscmap(:,2)',16,1);
    a(:,:,3) = repmat(mainHandles.prefs.clusterscmap(:,3)',16,1);
    set(handles.pushClustCmap,'CData', a);
return

function SelColor(hObject, eventdata)
    global mainHandles

    color=get(hObject,'backgroundcolor');
    prefsField=get(hObject,'tag');
    mainHandles.prefs.(prefsField)=uisetcolor(color);
    set(hObject,'backgroundcolor',mainHandles.prefs.(prefsField));
return
```

```
function pushcmapOutlineColor_Callback(hObject, eventdata, handles)
    % save Prefs to prefs.mat
    global mainHandles

    color=get(handles.pushcmapOutlineColor,'backgroundcolor');
    mainHandles.prefs.cmapOutlineColor = uisetcolor(color);
    set(handles.pushcmapOutlineColor,'backgroundcolor',...
        mainHandles.prefs.cmapOutlineColor);
return

function chkOutline_Callback(hObject, eventdata, handles)
    % save Prefs to prefs.mat
    global mainHandles

    mainHandles.prefs.onlyoutline = get(hObject,'Value');
return

function chkHitsOutline_Callback(hObject, eventdata, handles)
    % save Prefs to prefs.mat
    global mainHandles

    mainHandles.prefs.onlyhitsoutline = get(hObject,'Value');
return

function chkcmapOutline_Callback(hObject, eventdata, handles)
    % save Prefs to prefs.mat
    global mainHandles

    mainHandles.prefs.cmapOutline = get(hObject,'Value');
    if mainHandles.prefs.cmapOutline==1
        set(handles.pushcmapOutlineColor,'enable','on');
    else
        set(handles.pushcmapOutlineColor,'enable','off');
    end
return

function pushSave_Callback(hObject, eventdata, handles)
    % save Prefs to prefs.mat
    global mainHandles

    prefs = mainHandles.prefs;
    save 'prefs.mat' prefs

    %resume
    uiresume(handles.Prefs);
    close(handles.Prefs);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_querydata()
    %geosom_querydata
    %function to show the form to query data
    %
    % Syntax: geosom_querydata()
    % input - none
    % output - none
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 09-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008
```

```
scrsz = get(0,'ScreenSize');

fh=figure('Position',[scrsz(3)/2-180 scrsz(3)/2-100 400 200]);
% clear default menu and toolbars and add title
set(fh,'MenuBar','none','toolbar','none','NumberTitle','off','name',...
    'Query data','Resize','off');
handles.querydata=fh;

% panel option
hb = uibuttongroup('Title','','tag','uibtgroupTrain',...
             'units','pixels','Position',[10 40 20 150]);
%opt1
h = uicontrol('Style', 'radiobutton',...
       'tag', 'Batch train',...
       'parent',hb,...
       'Position',[2 110 15 15]);
handles.opt1=h;
%opt2
h = uicontrol('Style', 'radiobutton',...
        'tag', 'train',...
       'parent',hb,...
       'Position',[2 40 15 15]);
handles.opt2=h;

% panel query1
hp = uipanel('Title','',...
             'units','pixels','Position',[30 130 360 60]);

% % add controls
% text
h = uicontrol('Style', 'text',...
       'parent',hp,...
       'String', 'Select * from Data where:',...
       'tag', 'textSelect',...
       'Position', [10 30 150 15],...
       'HorizontalAlignment','left');
handles.textSelect=h;

% popups
h = uicontrol('Style', 'popup',...
       'parent',hp,...
       'tag', 'popupVariables',...
       'Position', [10 15 150 15],...
       'Callback', 'setmap');
handles.popupVariables=h;

% popups
h = uicontrol('Style', 'popup',...
       'parent',hp,...
       'tag', 'popupOperator',...
       'Position', [160 15 50 15]);
handles.popupOperator=h;

% popups
h = uicontrol('Style', 'popup',...
       'parent',hp,...
       'tag', 'popupValues',...
       'Position', [210 15 140 15]);
handles.popupValues=h;

% panel query2
hp = uipanel('Title','',...
             'units','pixels','Position',[30 40 360 90]);
h = uicontrol('Style', 'edit',...
       'parent',hp,...
       'max',10,'min',1,...
       'backgroundcolor','white',...
       'HorizontalAlignment','left',...
       'Position', [5 30 240 50]);
```

```
    handles.editQuery=h;

    % popups
    h = uicontrol('Style', 'popup',...
           'parent',hp,...
           'Position', [250 65 100 15]);
    handles.popupVariables2=h;

    %add field
    h = uicontrol('Style', 'pushbutton',...
           'String', 'Add',...
           'parent',hp,...
           'Position', [250 40 50 15]);
    handles.pushAddVariables=h;

    %example
    h = uicontrol('Style', 'text',...
           'String', {'Example: find(data(:,1)>0)',
             'find(data(:,1)>0 & data(:,1)<1)'},...
           'HorizontalAlignment','left',...
           'parent',hp,...
           'Position', [5 1 200 28]);

    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'Ok',...
           'tag', 'pushOK',...
           'Position', [340 10 50 20]);
    handles.pushOK=h;

    %%% CALLBACKS with handles %%%
    set(handles.popupVariables,'Callback',{@popupVariables_Callback handles});
    set(handles.pushAddVariables,'Callback',{@pushAddVariables_Callback handles});
    set(handles.pushOK,'Callback',{@pushbuttonOk_Callback handles});

    % LOAD POPUPS
    global gss
    set(handles.popupVariables,'String',[gss.NumDataLabel gss.CellDataLabel])
    set(handles.popupOperator,'String',{'=';'>';'<';'>=';'<='});
    set(handles.popupValues,'String',sort((gss.NumData(:,1))))
    set(handles.popupVariables2,'String',[gss.NumDataLabel])

    % Make the GUI modal
    set(handles.querydata,'WindowStyle','modal')
    % UIWAIT makes geosom_querydata wait for user response (see UIRESUME)
    uiwait(handles.querydata);
return %function geosom_querydata_v2()

function pushAddVariables_Callback(hObject, eventdata, handles)
    %adds the variable number to the text field
    txt=get(handles.editQuery,'string');
    var=num2str(get(handles.popupVariables2,'value'));

    set(handles.editQuery,'string',[txt '(:,' var ')']);
return

% --- Executes on button press in pushbuttonOk.
function pushbuttonOk_Callback(hObject, eventdata, handles)
    % make selection
    global gss

    if get(handles.opt1,'value')
        operator=get(handles.popupOperator,'value');
        indsel=get(handles.popupValues,'value');
        sel=get(handles.popupVariables,'value');

        if sel<=length(gss.NumDataLabel)
            data=[gss.NumData(:,sel) (1:length(gss.NumData))'];
            sorted=sortrows(data,1);
```

```
            switch operator
            case 1  % =
                 %resultInd= sorted(indsel,2);
                 resultInd=find(gss.NumData(:,sel)==sorted(indsel,1));
             case 2  % >
                 resultInd=find(gss.NumData(:,sel)>sorted(indsel,1));
             case 3  % <
                 resultInd=find(gss.NumData(:,sel)<sorted(indsel,1));
             case 4  % >=
                 resultInd=find(gss.NumData(:,sel)>=sorted(indsel,1));
             case 5  % <=
                 resultInd=find(gss.NumData(:,sel)<=sorted(indsel,1));
             end


            gss.SelectedIndex(:)=0;
            gss.SelectedIndex(resultInd)=1;
        else
            sel=sel-length(gss.NumDataLabel);
            data=[gss.CellData(:,sel) num2cell((1:length(gss.CellData))')];
            sorted=sortrows(data,1);

            switch operator
            case 1  % =
                resultInd= sorted(indsel,2);
            case 2  % >
                if indsel>length(sorted)
                    resultInd=sorted(indsel+1:end,2);
                else
                    resultInd=sorted(indsel:end,2);
                end
            case 3  % <
                if indsel>1
                    resultInd=sorted(1:indsel-1,2);
                else
                    resultInd=sorted(1:indsel,2);
                end
            case 4  % >=
                resultInd=sorted(indsel:end,2);
             case 5  % <=
                resultInd=sorted(1:indsel,2);
            end
            gss.SelectedIndex(:)=0;
            gss.SelectedIndex(cell2mat(resultInd))=1;
        end
    else
        if selectBasedOnString(handles)==0
            msgbox('Cannot make selection based on your query!');
            return
        end
    end

    % Use UIRESUME instead of delete because the OutputFcn needs
    % to get the updated handles structure.
    uiresume(handles.querydata);
    close(handles.querydata);
return

function out=selectBasedOnString(handles)
    global gss

    try
        %get query
        txt=get(handles.editQuery,'string');
        data=gss.NumData;
        result=eval(txt);
        gss.SelectedIndex(:)=0;
        gss.SelectedIndex(result)=1;
        out=1;
```

```
    catch
        out=0;
    end
return

function popupVariables_Callback(hObject, eventdata, handles)
    global gss
    sel = get(hObject,'Value');
    if length(gss.NumDataLabel)>=sel
        set(handles.popupValues,'String',sort(gss.NumData(:,sel)));
    else
        set(handles.popupValues,'String',sort(gss.CellData(:,sel-
length(gss.NumDataLabel))));
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [cellData,numData,FieldNames]=geosom_readCsvFile(pathname,filename)
    % geosom_struct=geosom_readCsvFile(pathname,filename)
    %
    % NAME: GEOSOM_READCSVMATFILE
    %
    % OBJECTIVE: Read a 'csv' or 'mat' file to a GeoSomStruct
    %
    % INPUT PARAMETERS:
    %       pathname - path to the file
    %       filename - Name of the file
    %
    % OUTPUT PARAMETERS:
    %
    %
    % COMMENTS:
    %
    %
    %
    % V.0.1 2008/07/02 By R.Henriques

    % Open the csv file
    fid=fopen(strcat(pathname,filename));

    % Start reading the data
    tline = fgetl(fid); % Read in the first line only (text headers)
    tline = tline(tline~=' '); %Get rid of any spaces
    commaLocs=findstr(',',tline); % find the commas
    FieldNames=cell(1,(length(commaLocs)+1));
    start=1;
    for colIdx=1:length(commaLocs)
        FieldNames{colIdx}=tline(start:commaLocs(colIdx)-1);
        start=commaLocs(colIdx)+1;
    end
    FieldNames{colIdx+1}=tline(start:end);

    % add 100 rows
    cellData=cell(100,(length(commaLocs)+1));

    line=0;
    while 1
        % exit if EOF
        tline = fgetl(fid);
        if ~ischar(tline),   break,   end

        line=line+1;
        if line>size(cellData,1)
            % add more 100 rows
            cellData=[cellData;cell(100,(length(commaLocs)+1))];
        end

        commaLocs=findstr(',',tline); % find the commas
```

```
        start=1;
        for colIdx=1:length(commaLocs)
            cellData{line,colIdx}=tline(start:commaLocs(colIdx)-1);
            start=commaLocs(colIdx)+1;
        end
        cellData{line,colIdx+1}=tline(start:end);

    end
    fclose(fid);

    %remove extra rows
    cellData(line+1:end,:)=[];

    % check if there are numeric fiedls
    numColumns=size(cellData,2);
    numData=[];
    removeCols=[];
    for i=1:numColumns % for each column
    %    try
            if ~isempty(str2num(str2mat(cellData(:,i))));
                %numCols=[numCols i];
                numData=[numData str2num(str2mat(cellData(:,i)))];
                removeCols=[removeCols,i];
            end
    %    catch
    %        continue
    %    end
    end
    cellData(:,removeCols)=[];
    % change the fields order. First the numeric and then the cell
    FieldNamesTmp=FieldNames;
    %numeric
    FieldNames=FieldNamesTmp(removeCols);
    %not numeric
    AlphaCols= setdiff((1:numColumns), removeCols);
    FieldNames=[FieldNames FieldNamesTmp(AlphaCols)];
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [geosom_struct, ok]=geosom_ReadCsvMatFile(pathname,filename);
    % geosom_struct=geosom_ReadCsvMatFile(filename)
    %
    % NAME: GEOSOM_READCSVMATFILE
    %
    % OBJECTIVE: Read a 'csv' or 'mat' file to a GeoSomStruct
    %
    % INPUT PARAMETERS:
    %       pathname - path to the file
    %       filename - Name of the file
    %
    % OUTPUT PARAMETERS:
    %       geosom_struct - Structure used by GeoSOM (see Lobo et al 2007)
    %
    % COMMENTS:
    %       This is the basic fuction used to import data into the GeoSOM
    %       programs.
    %

    % V.0.1 2008/07/02 By R.Henriques

    FieldNames=[];
    file=filename(end-2:end);
    cellData=[];
    numData=[];

    if strcmpi(file,'csv') % csv
        [cellData,numdata,FieldNames]=geosom_readCsvFile(pathname,filename);
```

```
    elseif strcmpi(file,'mat') % mat

        try % to load the header file
            header = load(strcat(pathname, 'header_',filename), '-mat');
            FieldNames=header.(strcat('header_',filename(1:end-4)));
        catch
            button=questdlg({'Cannot find the file with the header information!';
                strcat('This file should be named header_',filename);
                'and exist in the selected directory.';''
                'Do you and to continue?';
                'In this case the variables will be named variable1 to n.'},...
                'No header found','Continue','Cancel','default');
            if strcmp(button,'Cancel')
                ok=0;
                return
            end

        end

        try % to load the mat file
            numdata = load(strcat(pathname,filename), '-mat');
            names = fieldnames(numdata);
            numdata=numdata.(char(names(1)));
            %numdata=numdata.(filename(1:end-4));
        catch
            msgbox('Cannot read the mat file!'...
            ,'Problem reading file','warn');
            ok=0;
            return
        end

        if isempty(FieldNames)  % no header file defined -->var1,var2,var3
            FieldNames=cell(1,size(numdata,2));
            for i=1:length(FieldNames)
                FieldNames(i)=cellstr(strcat('var',num2str(i)));
            end
        end
    end


    % We can now create our GeoSomStruct, and fill it with the new data.

    numElements=length(numdata);
    geosom_struct = struct('NumData',[],'NumDataLabel',[],...
                    'CellData',[],'CellDataLabel',[]);

    %numeric fields
    geosom_struct.NumData = numdata;
    geosom_struct.NumDataLabel = FieldNames(1:size(numdata,2));

    %alpha fields
    geosom_struct.CellData = cellData;
    geosom_struct.CellDataLabel = FieldNames(size(numdata,2)+1:end);


    geosom_struct.ViewIndexData = [1:numElements]';
    geosom_struct.SelectedIndex=false(numElements,1);
    ok=1;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_struct =geosom_ReadShapeFile(filename)
    % geosom_struct=geosom_ReadShapeFile(filename)
    %
    % NAME: GEOSOM_READSHAPEFILE
    %
    % OBJECTIVE: Read a 'shp' file to a GeoSomStruct
    %
```

```
% INPUT PARAMETERS:
%       filename - Name of the 'shp' file
%
% OUTPUT PARAMETERS:
%       geosom_struct - Structure used by GeoSOM (see Lobo et al 2007)
%
% COMMENTS:
%       This is the basic fuction used to import data into the GeoSOM
%       programs.
%

% V.0.1 2007/01/22 By V.Lobo & M.Loureiro
% V.0.2 2007/01/23 By V.Lobo & M.Loureiro
% V.0.3 2007/02/15 By V.Lobo & M.Loureiro, minor bug fixes


tmpStruct=shaperead(filename);
tmpStructBackup = tmpStruct;
                % This generates a matrix with n lines, each of which is
                % a structure (named geostruct2 by MAPTOOLBOX)
                % Each of these structures has necessarily the fields:
                    % geometry - 'polygon' | 'line' | 'point'
                    % X - Vector of x-coordinates
                    % Y - Vector of y-coordinates

% We can now create our GeoSomStruct, and fill it with the new data.
tmpFieldNames=fieldnames(tmpStruct);        % Read the NAMES of the fields
numOriginalFields=length(tmpFieldNames);    % Find out how many there are
numElements=length(tmpStruct);

geosom_struct = struct('NumData',[],'NumDataLabel',[],...
                'CellData',[],'CellDataLabel',[]);
geosom_struct.TableIndex=1:numElements;
geosom_struct.TableIndex=geosom_struct.TableIndex';
numFieldIndex=0;
cellFieldIndex=0;
firstcellfield=true;    %necessary for the first column of cells
for i=1:numOriginalFields
                        %if the field belongs to the "view"...let it be!
    if ( strcmp(tmpFieldNames{i},'X')||...
            strcmp(tmpFieldNames{i},'Y')||...
            strcmp(tmpFieldNames{i},'Geometry') || ...
            strcmp(tmpFieldNames{i},'BoundingBox'))
        continue;
    end

                        %get ONE field at a time andprocess it
                        %if the field is num
    if isnumeric(tmpStruct(1).(tmpFieldNames{i}))
        numFieldIndex=numFieldIndex+1;
        tmp=zeros(numElements,1);
        for j=1:numElements
            tmp(j)=tmpStruct(j).(tmpFieldNames{i});
        end;
        geosom_struct.NumData = [geosom_struct.NumData tmp];
        geosom_struct.NumDataLabel{numFieldIndex}=tmpFieldNames{i};
        tmpStruct=rmfield(tmpStruct,tmpFieldNames{i});
    else
                        %If the field is cell
        if ischar(tmpStruct(1).(tmpFieldNames{i}))
            cellFieldIndex=cellFieldIndex+1;
            tmp=cell(numElements,1);
            for j=1:numElements
                tmp{j}=tmpStruct(j).(tmpFieldNames{i});
            end;
            if firstcellfield
                geosom_struct.CellData = tmp;
                firstcellfield=false;
            else
```

```
                    geosom_struct.CellData = [geosom_struct.CellData, tmp];
                end
                geosom_struct.CellDataLabel{cellFieldIndex}=tmpFieldNames{i};
                tmpStruct=rmfield(tmpStruct,tmpFieldNames{i});
            end;
        end;
    end;

    % for i=1:numElements,
    %     tmpStructBackup(i,1).CORR = i;
    % end

    geosom_struct.View{1}=tmpStructBackup;
    geosom_struct.ViewLabel.Name = filename;
    geosom_struct.ViewLabel.Type = 'GeographicMap';
    geosom_struct.ViewIndexData = [1:numElements]';
    geosom_struct.SelectedIndex=false(numElements,1);
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_struct=geosom_Selected2Table(geosom_struct,selectedData)
    % geosom_struct=geosom_Selected2Table(geosom_struct,selectedData)
    %
    % NAME: geosom_Selected2Table
    %
    % OBJECTIVE: Generate the field "TableIndex" so that the "selected" data
    %               appears first in that table
    %
    % INPUT PARAMETERS:
    %       geosom_struct - Structure used by GeoSOM (see Lobo et al 2007)
    %       selectedData - line array with indexes of the selected data
    %
    % OUTPUT PARAMETERS:
    %       geosom_struct - Structure used by GeoSOM (see Lobo et al 2007)
    %
    % COMMENTS:
    %
    %

    % V.0.1 2007/02/15 By V.Lobo & M.Loureiro & R.Henriques

    numSelected=length(selectedData);
    [numData,NumFeatures]=size(geosom_struct.NumData);
    notSelected=setdiff([1:numData],selectedData);

    geosom_struct.TableIndex(selectedData)=[1:numSelected]';
    geosom_struct.TableIndex(notSelected)=[numSelected+1:numData]';
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_showProperties(somId)
    %geosom_showProperties
    % show SOM properties
    %
    % Syntax: geosom_showProperties()
    % input - none
    % output - none
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 16-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008
```

```
    global gss
    SOM=gss.Som(somId);

    if strcmpi(SOM.method,'som')
        showSOMproperties(SOM)
    elseif strcmpi(SOM.method,'geosom')
        showGeoSOMproperties(SOM)
    elseif strcmpi(SOM.method,'hsom')
        showHSOMproperties(SOM)
    end

return

function showSOMproperties(SOM)
    %show SOM properties
    global gss

    fh=figure('Position',[200 200 400 450]);
    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
        'Resize','off','name',[SOM.name ' properties']);
    handles.trainSOM=fh;

    % % add controls
    % text - title
    h = uicontrol('Style', 'text',...
           'String', 'SOM properties',...
           'tag', 'textViews',...
           'Position', [10 430 380 15],...
           'HorizontalAlignment','Center',...
           'fontweight','bold');
    handles.textViews=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % panel map initialization
    hp = uipanel('Title','Map Initialization',...
                 'units','pixels','Position',[10 350 380 80]);
    % size X
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'X',...
           'tag', 'textX',...
           'Position', [10 40 40 15],...
           'HorizontalAlignment','left');
    handles.textX=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String',SOM.topol.msize(2),...
           'tag', 'editX',...
           'enable','off',...
           'background','white',...
           'Position', [50 40 50 20]);
    handles.editX=h;

    % size Y
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Y',...
           'tag', 'textY',...
           'Position', [130 40 40 15],...
           'HorizontalAlignment','left');
    handles.textY=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String', SOM.topol.msize(1),...
           'tag', 'editY',...
           'background','white',...
           'enable','off',...
```

```
       'Position', [170 40 50 20]);
handles.editY=h;

% lattice
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Lattice',...
       'tag', 'textLattice',...
       'Position', [10 15 40 15],...
       'HorizontalAlignment','left');
handles.textLattice=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {SOM.topol.lattice},...
       'tag', 'popupLattice',...
       'Position', [50 15 70 20]);
handles.popupLattice=h;

% shape
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Shape',...
       'tag', 'textShape',...
       'Position', [130 15 40 15],...
       'HorizontalAlignment','left');
handles.textShape=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {SOM.topol.shape},...
       'tag', 'popupShape',...
       'Position', [170 15 70 20]);
handles.popupShape=h;

% type
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Type',...
       'tag', 'textType',...
       'Position', [250 15 40 15],...
       'HorizontalAlignment','left');
handles.textType=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', SOM.trainhist(1).algorithm,...
       'tag', 'popupType',...
       'Position', [280 15 70 20]);
handles.popupType=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize data
h = uicontrol('Style', 'text',...
       'String', 'Normalize data',...
       'tag', 'textNormalize',...
       'Position', [10 320 80 15],...
       'HorizontalAlignment','left');
handles.textNormalize=h;
h = uicontrol('Style', 'popup',...
       'String', {SOM.topol.Normalization},...
       'tag', 'popupNormalize',...
       'Position', [10 300 80 20]);
handles.popupNormalize=h;

% else % normal SOM
h = uicontrol('Style', 'listbox',...
       'String', gss.NumDataLabel(find(SOM.mask==1)),...
       'tag', 'listComponents',...
       'Position', [130 240 250 100]);
handles.listComponents=h;
% end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Map train
hp = uipanel('Title','Map Train',...
             'units','pixels','Position',[10 160 380 80]);

% batch or sequential button group
hb = uibuttongroup('Title','','tag','uibtgroupTrain','parent',hp,...
             'units','pixels','Position',[10 40 360 20]);
handles.uibtgroupTrain=hb;

%option buttons
h = uicontrol('Style', 'radiobutton',...
       'parent',hb,...
       'String', 'Batch train',...
       'tag', 'chkbatch',...
       'Position', [50 0 100 15],...
       'Callback', '');
handles.chkbatch=h;
h = uicontrol('Style', 'radiobutton',...
       'parent',hb,...
       'String', 'Sequential train',...
       'tag', 'chkSequential',...
       'Position', [200 0 100 15],...
       'Callback', '');
handles.chkSequential=h;

%select type of train
if strcmpi(SOM.trainhist(2).algorithm,'seq')
    set(handles.chkSequential,'value',1);

    % sequential trainning parameters panel
    hp2 = uipanel('Title','','parent',hp,...
                  'tag','panelTrain',...
                  'units','pixels',...
                  'Position',[100 5 270 35]);
    handles.panelTrain=hp2;

    % Iterations type
    h = uicontrol('Style', 'text',...
           'Parent',hp2,...
           'String', 'Iter.',...
           'tag', 'textIter',...
           'Position', [5 5 40 15],...
           'HorizontalAlignment','left');
    handles.textIter=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp2,...
           'String', {''},...
           'tag', 'popupIter',...
           'Position', [25 5 60 20]);
    handles.popupIter=h;

    % order
    h = uicontrol('Style', 'text',...
           'Parent',hp2,...
           'String', 'Order',...
           'tag', 'textOrder',...
           'Position', [85 5 40 15],...
           'HorizontalAlignment','left');
    handles.textOrder=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp2,...
           'String', {''},...
           'tag', 'popupOrder',...
           'Position', [115 5 60 20]);
    handles.popupOrder=h;

    % Length function
    h = uicontrol('Style', 'text',...
```

```
            'Parent',hp2,...
            'String', 'Length',...
            'tag', 'textLength',...
            'Position', [175 5 40 15],...
            'HorizontalAlignment','left');
    handles.textLength=h;
    h = uicontrol('Style', 'popup',...
            'Parent',hp2,...
            'String', SOM.trainhist(2).alpha_type,...
            'tag', 'popupLength',...
            'Position', [210 5 50 20]);
    handles.popupLength=h;
else
    set(handles.chkbatch,'value',1);
end

% Neigh function
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Neigh',...
       'tag', 'textNeigh',...
       'Position', [10 10 40 15],...
       'HorizontalAlignment','left');
handles.textNeigh=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', SOM.trainhist(2).neigh,...
       'tag', 'popupNeigh',...
       'Position', [40 10 60 20]);
handles.popupNeigh=h;

%check if 2 trains or 1
if length(SOM.trainhist)==3 %2 train sets
    % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    trainHist=SOM.trainhist(2);
    % Rough parameters
    hp = uipanel('Title','Rough',...
                 'tag','panelRough',...
                 'units','pixels',...
                 'Position',[10 80 190 80]);
    handles.panelRough=hp;

    % Iterations_1
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Iterations 1',...
           'tag', 'textIterations1',...
           'Position', [30 45 55 15],...
           'HorizontalAlignment','left');
    handles.textIterations1=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String', trainHist.trainlen,...
           'tag', 'editIterations1',...
           'background','white',...
           'enable','off',...
           'Position', [100 45 40 20]);
    handles.editIterations1=h;
    % Radio_1
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Radio 1',...
           'tag', 'textRadio1',...
           'Position', [30 25 55 15],...
           'HorizontalAlignment','left');
    handles.textRadio1=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String',trainHist.radius_ini,...
```

```
            'enable','off',...
            'tag', 'editRadio1',...
            'background','white',...
            'Position', [100 25 40 20]);
    handles.editRadio1=h;
    % Alpha 1
    h = uicontrol('Style', 'text',...
            'Parent',hp,...
            'String', 'Alpha 1',...
            'tag', 'textAlpha1',...
            'Position', [30 5 55 15],...
            'HorizontalAlignment','left');
    handles.textAlpha1=h;
    h = uicontrol('Style', 'edit',...
            'Parent',hp,...
            'String', trainHist.alpha_ini,...
            'enable','off',...
            'tag', 'editAlpha1',...
            'background','white',...
            'Position', [100 5 40 20]);
    handles.editAlpha1=h;
    trainHist=SOM.trainhist(3);
else
    trainHist=SOM.trainhist(2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finetune parameters
hp = uipanel('Title','Finetune',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[200 80 190 80]);
handles.panelFinetune=hp;

% Iterations_2
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 2',...
       'tag', 'textIterations2',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', trainHist.trainlen,...
       'tag', 'editIterations2',...
       'background','white',...
       'enable','off',...
       'Position', [100 45 40 20]);
handles.editIterations2=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 2',...
       'tag', 'textRadio2',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', trainHist.radius_ini,...
       'tag', 'editRadio2',...
       'enable','off',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio2=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
```

```
       'String', 'Alpha 2',...
       'tag', 'textAlpha2',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', trainHist.alpha_ini,...
       'enable','off',...
       'tag', 'editAlpha2',...
       'background','white',...
       'Position', [100 5 40 20]);
handles.editAlpha2=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% description parameters
hp = uipanel('Title','Description',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[10 10 190 70]);
handles.panelDescription=hp;
% name
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Name',...
       'Position', [10 35 65 15],...
       'HorizontalAlignment','left');
handles.textName=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', SOM.name,...
       'tag', 'editName',...
       'background','white',...
       'HorizontalAlignment','left',...
       'enable','off',...
       'Position', [70 35 110 20]);
handles.editName=h;
%obs
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Obs.',...
       'Position', [10 20 55 15],...
       'HorizontalAlignment','left');
handles.textObs=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', SOM.topol.Obs,...
       'tag', 'editObs',...
       'enable','off',...
       'max',2,'min',0,...
       'background','white',...
       'HorizontalAlignment','left',...
       'Position', [40 5 140 30]);
handles.editObs=h;

%Qerror
h = uicontrol('Style', 'text',...
       'String', 'Q.error',...
       'Position', [210 50 50 15],...
       'HorizontalAlignment','left');
h = uicontrol('Style', 'edit',...
       'String', SOM.topol.qe,...
       'background','white',...
       'HorizontalAlignment','left',...
       'enable','on',...
       'Position', [260 50 100 20]);
% Terror
h = uicontrol('Style', 'text',...
       'String', 'T.error',...
```

```
           'Position', [210 30 50 15],...
           'HorizontalAlignment','left');
    h = uicontrol('Style', 'edit',...
           'String', SOM.topol.te,...
           'background','white',...
           'HorizontalAlignment','left',...
           'enable','on',...
           'Position', [260 30 100 20]);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'OK',...
           'tag', 'pushOK',...
           'Position', [340 5 50 15],...
           'Callback', 'close');
    handles.pushOK=h;

    % Make the GUI modal
    set(handles.trainSOM ,'WindowStyle','modal')
    % UIWAIT
    %uiwait(handles.trainSOM);
return

function showGeoSOMproperties(SOM)
    %show geoSOM properties
    global gss

    fh=figure('Position',[200 200 400 500]);
    % clear default menu and toolbars and add title
    set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
        'Resize','off','name',[SOM.name ' properties']);
    handles.trainGeoSOM=fh;

    % % add controls
    % text - title
    h = uicontrol('Style', 'text',...
           'String', 'GeoSOM training parameters',...
           'tag', 'textViews',...
           'Position', [10 480 380 15],...
           'HorizontalAlignment','Center',...
           'fontweight','bold');
    handles.textViews=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % panel map initialization
    hp = uipanel('Title','Map Initialization',...
                 'units','pixels','Position',[10 400 380 80]);
    % size X
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'X',...
           'tag', 'textX',...
           'Position', [10 40 40 15],...
           'HorizontalAlignment','left');
    handles.textX=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String', SOM.topol.msize(2),...
           'tag', 'editX',...
           'enable','off',...
           'background','white',...
           'Position', [50 40 50 20]);
    handles.editX=h;

    % size Y
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Y',...
```

```
       'tag', 'textY',...
       'Position', [130 40 40 15],...
       'HorizontalAlignment','left');
handles.textY=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', SOM.topol.msize(1),...
       'enable','off',...
       'tag', 'editY',...
       'background','white',...
       'Position', [170 40 50 20]);
handles.editY=h;

% lattice
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Lattice',...
       'tag', 'textLattice',...
       'Position', [10 15 40 15],...
       'HorizontalAlignment','left');
handles.textLattice=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {SOM.topol.lattice},...
       'tag', 'popupLattice',...
       'Position', [50 15 70 20]);
handles.popupLattice=h;

% shape
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Shape',...
       'tag', 'textShape',...
       'Position', [130 15 40 15],...
       'HorizontalAlignment','left');
handles.textShape=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {SOM.topol.shape},...
       'tag', 'popupShape',...
       'Position', [170 15 70 20]);
handles.popupShape=h;

% type
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Type',...
       'tag', 'textType',...
       'Position', [250 15 40 15],...
       'HorizontalAlignment','left');
handles.textType=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', SOM.trainhist(1).algorithm,...
       'tag', 'popupType',...
       'Position', [280 15 70 20]);
handles.popupType=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% show in a list the selected non geo components
h = uicontrol('Style', 'listbox',...
       'String', gss.NumDataLabel(find(SOM.mask(3:end)==1)),...
       'tag', 'listComponents',...
       'Position', [20 290 200 100]);
handles.listComponents=h;

h = uicontrol('Style', 'text',...
       'String', 'Geo components',...
       'tag', 'textNormalize',...
```

```
           'Position', [230 375 100 15],...
           'HorizontalAlignment','left');

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % show in a list the selected geo components
    h = uicontrol('Style', 'listbox',...
           'String',{'X' 'Y'},...     %'String',
gss.NumDataLabel(find(SOM.mask(1:2)==1)),...
           'tag', 'listComponents',...
           'Position', [230 320 140 50]);
    handles.listGeoComponents=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % normalize data
    h = uicontrol('Style', 'text',...
           'String', 'Normalize data',...
           'tag', 'textNormalize',...
           'Position', [10 260 80 15],...
           'HorizontalAlignment','left');
    handles.textNormalize=h;
    h = uicontrol('Style', 'popup',...
           'String', {SOM.topol.Normalization},...
           'tag', 'popupNormalize',...
           'Position', [100 260 80 20]);
    handles.popupNormalize=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Map train

    hp = uipanel('Title','Map Train',...
                 'units','pixels','Position',[10 170 380 80]);

    % batch or sequential
    % at this time only sequential is implemented
    hb = uibuttongroup('Title','','tag','uibtgroupTrain','parent',hp,...
                 'units','pixels','Position',[10 40 360 20]);
    handles.uibtgroupTrain=hb;

    %option buttons
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'enable','off',...
           'String', 'Batch train',...
           'tag', 'chkbatch',...
           'Position', [50 0 100 15],...
           'Callback', '');
    handles.chkbatch=h;
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'value',1,...
           'String', 'Sequential train',...
           'tag', 'chkSequential',...
           'Position', [200 0 100 15],...
           'Callback', '');
    handles.chkSequential=h;


    % sequential trainning parameters panel
    hp2 = uipanel('Title','','parent',hp,...
                 'tag','panelTrain',...
                 'units','pixels',...
                 'visible','off',...
                 'Position',[100 5 270 35]);
    handles.panelTrain=hp2;
    set(handles.panelTrain,'visible','on')

    % Iterations type
    h = uicontrol('Style', 'text',...
           'Parent',hp2,...
```

```
       'String', 'Iter.',...
       'tag', 'textIter',...
       'Position', [5 5 40 15],...
       'HorizontalAlignment','left');
handles.textIter=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {''},...
       'tag', 'popupIter',...
       'Position', [25 5 60 20]);
handles.popupIter=h;

% order
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Order',...
       'tag', 'textOrder',...
       'Position', [85 5 40 15],...
       'HorizontalAlignment','left');
handles.textOrder=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {''},...
       'tag', 'popupOrder',...
       'Position', [115 5 60 20]);
handles.popupOrder=h;

% Length function
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Length',...
       'tag', 'textLength',...
       'Position', [175 5 40 15],...
       'HorizontalAlignment','left');
handles.textLength=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', SOM.trainhist(2).alpha_type,...
       'tag', 'popupLength',...
       'Position', [210 5 50 20]);
handles.popupLength=h;

% Neigh function
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Neigh',...
       'tag', 'textNeigh',...
       'Position', [10 10 40 15],...
       'HorizontalAlignment','left');
handles.textNeigh=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', SOM.trainhist(2).neigh,...
       'tag', 'popupNeigh',...
       'Position', [40 10 60 20]);
handles.popupNeigh=h;

%check if 2 trains or 1
if length(SOM.trainhist)==3 %2 train sets
    % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    trainHist=SOM.trainhist(2);
    % Rough parameters
    hp = uipanel('Title','Rough',...
                 'tag','panelRough',...
                 'units','pixels',...
                 'Position',[10 80 190 80]);
    handles.panelRough=hp;

    % Iterations_1
```

```
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Iterations 1',...
           'tag', 'textIterations1',...
           'Position', [30 45 55 15],...
           'HorizontalAlignment','left');
    handles.textIterations1=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String',  trainHist.trainlen,...
           'enable','off',...
           'tag', 'editIterations1',...
           'background','white',...
           'Position', [100 45 40 20]);
    handles.editIterations1=h;
    % Radio_1
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Radio 1',...
           'tag', 'textRadio1',...
           'Position', [30 25 55 15],...
           'HorizontalAlignment','left');
    handles.textRadio1=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String', trainHist.radius_ini,...
           'enable','off',...
           'tag', 'editRadio1',...
           'background','white',...
           'Position', [100 25 40 20]);
    handles.editRadio1=h;
    % Alpha 1
    h = uicontrol('Style', 'text',...
           'Parent',hp,...
           'String', 'Alpha 1',...
           'tag', 'textAlpha1',...
           'Position', [30 5 55 15],...
           'HorizontalAlignment','left');
    handles.textAlpha1=h;
    h = uicontrol('Style', 'edit',...
           'Parent',hp,...
           'String', trainHist.alpha_ini,...
           'enable','off',...
           'tag', 'editAlpha1',...
           'background','white',...
           'Position', [100 5 40 20]);
    handles.editAlpha1=h;
    trainHist=SOM.trainhist(3);
else
    trainHist=SOM.trainhist(2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finetune parameters
hp = uipanel('Title','Finetune',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[200 80 190 80]);
handles.panelFinetune=hp;

% Iterations_2
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 2',...
       'tag', 'textIterations2',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations2=h;
h = uicontrol('Style', 'edit',...
```

```
       'Parent',hp,...
       'String', trainHist.trainlen,...
       'enable','off',...
       'tag', 'editIterations2',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations2=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 2',...
       'tag', 'textRadio2',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', trainHist.radius_ini,...
       'enable','off',...
       'tag', 'editRadio2',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio2=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 2',...
       'tag', 'textAlpha2',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', trainHist.alpha_ini,...
       'enable','off',...
       'tag', 'editAlpha2',...
       'background','white',...
       'Position', [100 5 40 20]);
handles.editAlpha2=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GeoSOM specific parameters
hp = uipanel('Title','GeoSOM specific parameters',...
             'tag','panelRough',...
             'units','pixels',...
             'Position',[10 10 190 70]);
handles.panelRough=hp;

% k
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Geo radius(k)',...
       'tag', 'textk',...
       'Position', [30 30 100 15],...
       'HorizontalAlignment','left');
handles.textK=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', SOM.topol.k,...
       'enable','off',...
       'tag', 'editK',...
       'background','white',...
       'Position', [100 30 40 20]);
handles.editK=h;

%f
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Update(f)',...
```

```
           'tag', 'textf',...
           'Position', [30 10 55 15],...
           'HorizontalAlignment','left');
    handles.textK=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp,...
           'String', {SOM.topol.f},...
           'tag', 'popupF',...
           'Position', [100 10 60 20]);
    handles.popupF=h;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %push buttons
    h = uicontrol('Style', 'pushbutton',...
           'String', 'OK',...
           'tag', 'pushOK',...
           'Position', [340 5 50 15],...
           'Callback', 'close');
    handles.pushOK=h;
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_struct=geosom_ShowView(viewNum,cmap)
    %GEOSOM_SHOWVIEW ...
    % Syntax: geosom_struct=geosom_ShowView(viewNum,cmap)
    % OBJECTIVE: Create a window with a View of the data
    % viewNum - number of the view
    % cmap - colormap
    % geosom_struct - gs struc
    %
    % Subfunctions: Umati_Callback, geosom_showHits, geosom_showViewClose, ExportSel2kml,
ExportClusters2kml, ExportAxes2NewFig, ExportSHP, showCPsMax, showCPsMin, showCPs,
ButtonFunction, IDW
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 30-2009
    % Version 1.0
    % V.0.1 2007/01/28 By V.Lobo & M.Loureiro
    % Copyright (c) by the Ga2 programming team. 2009
    % geosom_struct=geosom_ShowView(geosom_struct,viewNum,colormap)

    global gss
    global mainHandles

    % check if the handle saved for this view is already displayed.
    % if no then create a new figure
    % if yes two things can occur:
    % 1. this view is already displayed -->just send focus to it
    % 2. another view with the same handle exist in the gss struct
    % So we have to verify if the displayed handle view is the same

    if isfield(gss.ViewLabel(viewNum), 'WindowHandle')
        if ishandle(gss.ViewLabel(viewNum).WindowHandle)
            figure(gss.ViewLabel(viewNum).WindowHandle);
            %check view NUmber in the userdata property of the figure
            if get (gcf,'UserData')==viewNum %  it is the same view so just leave
                return
            end
        end
    end

    f=figure;                                % Create a figure
    set(f,'position',[200 200 400 400]);
    set (f,'UserData',viewNum); % add the viewNumber to the figure

    % add export capabilities
    mn = uimenu('Label','Export');
```

```
uimenu(mn,'Label','To shapefile','Callback',...
    {@ExportSHP mn viewNum});
uimenu(mn,'Label','To new figure','Callback',...
    {@ExportAxes2NewFig});

% add new toolbar
set(f,'Toolbar','none');
hToolbar = uitoolbar(...
                'Parent',f, ...
                'HandleVisibility','callback');

% Create the button group.
%hg = uibuttongroup('Parent',hToolbar);

hB      =  uitoggletool(...
               'Parent',hToolbar,...
               'tag','zoomin',...
               'TooltipString','Zoom in',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\view_zoom_in.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback', {@ButtonFunction hToolbar viewNum});
hB      =  uipushtool(...
               'Parent',hToolbar,...
               'tag','zoomout',...
               'TooltipString','Zoom out',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\view_zoom_out.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback', 'zoom (0.5)');
hB      =  uipushtool(...
               'Parent',hToolbar,...
               'tag','fullextent',...
               'TooltipString','Full extent',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\webicon.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback','zoom out');
hB      =  uitoggletool(...
               'Parent',hToolbar,...
               'tag','selectpoint',...
               'TooltipString','Select',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\select.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback',{@ButtonFunction hToolbar viewNum});
hB      =  uitoggletool(...
               'Parent',hToolbar,...
               'tag','selectarea',...
               'TooltipString','Select Area',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\selectArea.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback',{@ButtonFunction hToolbar viewNum});

% define the figure handle in the gss structure
gss.ViewLabel(viewNum).WindowHandle=f;

if strcmp(gss.ViewLabel(viewNum).Type,'GeographicMap')
    % kml features
    uimenu(mn,'Label','Selection to kml','Callback',...
        {@ExportSel2kml viewNum});
    uimenu(mn,'Label','Clusters to kml','Callback',...
        {@ExportClusters2kml viewNum});

    if strcmpi(gss.View{1}(1).Geometry,'Polygon')
        %blueRoads = makesymbolspec('Polygon',{'Default','FaceColor',summer(1)});
        blueRoads = makesymbolspec('Polygon',{'Default','FaceColor',[.9 .9 .9]});
    elseif strcmpi(gss.View{1}(1).Geometry,'Point')
```

```
            blueRoads = makesymbolspec('Point',{'Default','MarkerEdgeColor',...
                [0 0 1],'Marker','.'});
        end

        %h=geoshow(gss.View{viewNum}, 'SymbolSpec', blueRoads);
        h=mapshow(gss.View{viewNum}, 'SymbolSpec', blueRoads);
        %h=geoshow(gss.View{viewNum}, 'SymbolSpec', blueRoads);
        gss.ViewLabel(viewNum).GraphicHandle=h;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%here is added the listener to the "click event" over the map
        set(gcf,'windowbuttondownfcn',strcat({@geosom_MapClick, viewNum}));

    elseif strcmp(gss.ViewLabel(viewNum).Type,'UMAT')
        title(['UMAT: ',gss.Som(gss.ViewLabel(viewNum).SomOrigin).name]);
        numColors=length(colormap);
        gss.View{viewNum}=geosom_ViewUmatCreateIValue(...
                        gss.View{viewNum},numColors );

        %check outline color
        if mainHandles.prefs.cmapOutline
            edgeColor=mainHandles.prefs.cmapOutlineColor;
        else
            edgeColor=cmap;
        end

        regra=makesymbolspec('Polygon',{'IValue',[1 numColors],...
                                        'FaceColor', cmap,...
                                        'EdgeColor', edgeColor});

        %h=geoshow(gss.View{viewNum},'SymbolSpec',regra);
        h=mapshow(gss.View{viewNum},'SymbolSpec',regra);
        gss.ViewLabel(viewNum).GraphicHandle=h;
        %Remove labels
        set(gca,'XTickLabel',{},'yTickLabel',{});
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%here is added the listener to the "click event"
        set(gcf,'windowbuttondownfcn',strcat({@geosom_UMATClick, viewNum}));

        % add a chkbox for presenting the hits
        h = uicontrol('Style', 'checkbox',...
           'String', 'Show Hits',...
           'Position', [20 5 80 20],...
           'Callback', {@geosom_showHits viewNum 'umat'});

        % add a chkbox for selection based on item number
        h = uicontrol('Style', 'checkbox',...
           'String', 'Show sel. based on item count',...
           'tag','selBasedOnCount',...
           'Position', [120 5 200 20],...
           'Callback', {});

        % add export capabilities
        mn = uimenu('Label','Visualize');
        uimenu(mn,'Label','CP max.','Callback',...
            {@showCPsMax viewNum});
        uimenu(mn,'Label','CP min.','Callback',...
            {@showCPsMin viewNum });
        uimenu(mn,'Label','CPs','Callback',...
            {@showCPs viewNum });
        uimenu(mn,'Label','Interpolated U-Mat','Callback',...
            {@Umati_Callback viewNum });

    elseif strcmp(gss.ViewLabel(viewNum).Type,'CP')


title([gss.ViewLabel(viewNum).Name,gss.Som(gss.ViewLabel(viewNum).SomOrigin).name]);
        %title(gss.ViewLabel(viewNum).Name);
        numColors=length(colormap);
        gss.View{viewNum}=geosom_ViewUmatCreateIValue(...
```

```
                    gss.View{viewNum},numColors );
    regra=makesymbolspec('Polygon',{'IValue',[1 numColors],...
                                    'FaceColor', cmap,...
                                    'EdgeColor', cmap});
    h=mapshow(gss.View{viewNum},'SymbolSpec',regra);
    gss.ViewLabel(viewNum).GraphicHandle=h;
    %Remove labels
    set(gca,'XTickLabel',{},'yTickLabel',{})

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%here is added the listener to the "click event"
    set(gcf,'windowbuttondownfcn',strcat({@geosom_UMATClick, viewNum}));

    % add a chkbox for presenting the hits
    h = uicontrol('Style', 'checkbox',...
       'String', 'Show Hits',...
       'Position', [20 5 80 20],...
       'Callback', {@geosom_showHits viewNum 'cp'});

    % add a chkbox for selection based on item number
    h = uicontrol('Style', 'checkbox',...
       'String', 'Show sel. based on item count',...
       'tag','selBasedOnCount',...
       'Position', [120 5 200 20],...
       'Callback', {});

elseif strcmpi(gss.ViewLabel(viewNum).Type,'Hits')
    title(gss.ViewLabel(viewNum).Name);
    numColors=length(colormap);
    gss.View{viewNum}=geosom_ViewUmatCreateIValue(...
                    gss.View{viewNum},numColors );
    regra=makesymbolspec('Polygon',{'IValue',[1 numColors],...
                                    'FaceColor', cmap,...
                                    'EdgeColor', cmap});
    h=mapshow(gss.View{viewNum},'SymbolSpec',regra);
    gss.ViewLabel(viewNum).GraphicHandle=h;
    %Remove labels
    set(gca,'XTickLabel',{},'yTickLabel',{})

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%here is added the listener to the "click event"
    set(gcf,'windowbuttondownfcn',strcat({@geosom_UMATClick, viewNum}));

elseif strcmpi(gss.ViewLabel(viewNum).Type,'PCP')
    %title(gss.ViewLabel(viewNum).Name);
    numColors=length(colormap);
    %gss.View{viewNum}=geosom_ViewUmatCreateIValue(...
    %                gss.View{viewNum},numColors );
    %regra=makesymbolspec('Line',{'Default','Color',colormap});
    %makesymbolspec('Line',{'IValue',[1 numColors],...
    %                         'Color', colormap});
    blueRoads = makesymbolspec('Line',{'Default','Color',[.5 .5 .5]});

    h=mapshow(gss.View{viewNum},'SymbolSpec',blueRoads);
    gss.ViewLabel(viewNum).GraphicHandle=h;

    set(gca,'XTick',(1:length(gss.ViewLabel(viewNum).SelVariables)));
    set(gca,'XTickLabel',gss.NumDataLabel(gss.ViewLabel(viewNum).SelVariables));
    grid;
    %stretch to fill
    set(gca,'DataAspectRatioMode','auto');
    set(gca,'PlotBoxAspectRatioMode','auto');
    set(gca,'CameraViewAngleMode','auto');

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%here is added the listener to the "click event"
    set(gcf,'windowbuttondownfcn',strcat({@geosom_UMATClick, viewNum}));

end
```

```
    set(gcf,'CloseRequestFcn',{@geosom_showViewClose, viewNum});
return

function Umati_Callback(hObject, eventdata, viewNum)
    % present umati
    global gss
    global mainHandles
    SomIndex=gss.ViewLabel(viewNum).SomOrigin;
    sM=gss.Som(SomIndex);
    sM = rmfield(sM, 'method');

    figure('Colormap',mainHandles.prefs.cmap);
    som_show (sM,'umati',find(sM.mask==1));
return

function geosom_showHits(hObject,eventdata, viewNum, source)
    % show the hits on the top of the UMAT
    global gss
    global mainHandles

    if get(hObject,'value'); % show the Hits
        SomIndex=gss.ViewLabel(viewNum).SomOrigin;
        HitsViewIndex=0;

        for i=1:length(gss.ViewLabel)
            if strcmpi(gss.ViewLabel(i).Type,'hits')
                % is this the Umat or the CP view?
                if strcmpi(gss.ViewLabel(i).Name,strcat(source,' hits'))
                    if gss.ViewLabel(i).SomOrigin==SomIndex
                        HitsViewIndex=i;
                        break;
                    end
                end
            end
        end
        if HitsViewIndex>0  % found the view
            %create group
            gh = hggroup;
            set(gh,'userdata','HitsGroup');
            for j=1:length(gss.View{HitsViewIndex})
                x=gss.View{HitsViewIndex}(j).X;
                x=[x(1:end-1) x(1)];
                y=gss.View{HitsViewIndex}(j).Y;
                y=[y(1:end-1) y(1)];
                if mainHandles.prefs.onlyhitsoutline
                    line(x,y,'Color',mainHandles.prefs.hitscolor ...
                    ,'parent',gh ...
                    ,'LineWidth',1 ...
                    ,'UserData','hits');
                else
                    patch(x,y,mainHandles.prefs.hitscolor,...
                        'parent',gh,...,...
                        'UserData','hits');
                end
            end
        end
    else % remove the hits
        %get patchs with userdata=sel and delete them
        p = get(gca,'children');

        delInd=strcmpi(get(p,'userdata'),'HitsGroup');
        delete(p(delInd==1));
    end

return

function geosom_showViewClose(hObject,eventdata,viewNum)
    global gss
```

```
    if isfield(gss,'ViewLabel');
        gss.ViewLabel(viewNum).WindowHandle=[];
        delete (gcf);
    end
return

function ExportSel2kml(hObject,eventdata,viewNum)
    %kml
    global gss

    [file,path] = uiputfile('*.kml','Save View As');
    geosom_export2KML(path,file,viewNum);
return

function ExportClusters2kml(hObject,eventdata,viewNum)
    %kml
    global gss

    [file,path] = uiputfile('*.kml','Save View As');
    geosom_exportClusters2KML(path,file,viewNum);
return

function ExportAxes2NewFig(hObject,eventdata)
    %get axes and open in new figure for export
    h=gca;
    hNewAxes=figure;
    set(hNewAxes,'color',[1 1 1]);
    copyobj(h, hNewAxes);
    set(gca,'PlotBoxAspectRatioMode','auto');
    set(gca,'CameraViewAngleMode','auto');
return

function ExportSHP(hObject,eventdata,hToolbar,viewNum)
    % function to export the view to shp
    global gss
    global mainHandles

    [file,path] = uiputfile('*.shp','Save View
As',strcat(mainHandles.prefs.DataFolder,'\'));
    s=gss.View{viewNum};
    % We have to change all empty SomIndex to 0
    for i=1:length(s)
        if isfield(s,'SomIndex')
            if isempty(s(i).SomIndex)
                s(i).SomIndex=0;
            end
        end
        % add selection field
        if isfield(gss,'SelectedIndex')
            s(i).Sel=double(gss.SelectedIndex(i));
        end
        % add clusters field
        if isfield(gss,'ClusterIndex')
            s(i).Clust=double(gss.ClusterIndex(i));
        end
    end

    try
        shapewrite(s ,strcat(path, file));
        if path~=0
            mainHandles.prefs.DataFolder=path;
        end
    catch
        disp('Error exporting view to shp');
    end
return

function showCPsMax(hObject,eventdata,viewNum)
    % draw in for each variable its name on the unit where it is maximum
```

```
    global gss

    if strcmpi(get(hObject,'checked'),'on')
        set(hObject,'checked','off')
    else
        set(hObject,'checked','on')
    end

    % delete text
    %get patchs with userdata=textCP and delete them
    p = get(gca,'children');
    delInd=strcmpi(get(p,'userdata'),'textCPMax');
    delete(p(delInd==1));

    %if get(hObject,'value')
    if strcmpi(get(hObject,'checked'),'on')

        %get CPs based on the viewNum
        SomOrigin=gss.ViewLabel(viewNum).SomOrigin;

        counter=0;
        for i=find(gss.Som(SomOrigin).mask==1)'
            counter=counter+1;
            [mx ind]=max(gss.Som(SomOrigin).codebook(:,i));
            for j=1:size(gss.View{viewNum},1)
                if gss.View{viewNum}(j).SomIndex==ind
                    Xc(counter)=mean(gss.View{viewNum}(j).X(1:end-1))+ (-.5+(.5+.5) *
rand);
                    Yc(counter)=mean(gss.View{viewNum}(j).Y(1:end-1))+ (-.5+(.5+.5) *
rand);
                    sText(counter)=gss.Som(SomOrigin).comp_names(i);
                    continue
                end
            end
        end

        if isempty(Xc)==0
            %create group
            gh = hggroup;
            set(gh,'userdata','textCPMax');
            for j=1:length(Xc)
                text(Xc(j),Yc(j),sText(j),...
                        'parent',gh,'FontSize',8,'color',[.7 0 0]);
            end
        end
    end
return

function showCPsMin(hObject,eventdata,viewNum)
    % draw in for each variable its name on the unit where it is maximum
    global gss

    if strcmpi(get(hObject,'checked'),'on')
        set(hObject,'checked','off')
    else
        set(hObject,'checked','on')
    end

    % delete text
    %get patchs with userdata=textCP and delete them
    p = get(gca,'children');
    delInd=strcmpi(get(p,'userdata'),'textCPMin');
    delete(p(delInd==1));

    %if get(hObject,'value')
    if strcmpi(get(hObject,'checked'),'on')

        %get CPs based on the viewNum
        SomOrigin=gss.ViewLabel(viewNum).SomOrigin;
```

```
        counter=0;
        for i=find(gss.Som(SomOrigin).mask==1)'
            counter=counter+1;
            [mx ind]=min(gss.Som(SomOrigin).codebook(:,i));
            for j=1:size(gss.View{viewNum},1)
                if gss.View{viewNum}(j).SomIndex==ind
                    Xc(counter)=mean(gss.View{viewNum}(j).X(1:end-1))+ (-.5+(.5+.5) *
rand);
                    Yc(counter)=mean(gss.View{viewNum}(j).Y(1:end-1))+ (-.5+(.5+.5) *
rand);
                    sText(counter)=gss.Som(SomOrigin).comp_names(i);
                    continue
                end
            end
        end

        if isempty(Xc)==0
            %create group
            gh = hggroup;
            set(gh,'userdata','textCPMin');
            for j=1:length(Xc)
                text(Xc(j),Yc(j),sText(j),...
                        'parent',gh,'FontSize',8,'color',[0 0 .7]);
            end
        end
    end
return

function showCPs(hObject,eventdata,viewNum)
    % draw in for each variable its name on the unit where it is maximum
    global gss
    global mainHandles

    if strcmpi(get(hObject,'checked'),'on')
        set(hObject,'checked','off')
    else
        set(hObject,'checked','on')
    end

    % delete text
    %get patchs with userdata=textCP and delete them
    p = get(gca,'children');
    delInd=strcmpi(get(p,'userdata'),'textCP');
    delete(p(delInd==1));

    %if get(hObject,'value')
    if strcmpi(get(hObject,'checked'),'on')
        %get CPs based on the viewNum
        SomOrigin=gss.ViewLabel(viewNum).SomOrigin;

        counter=0;
        Xc=zeros(1,2);
        Yc=zeros(1,2);
        for i=find(gss.Som(SomOrigin).mask==1)'
            counter=counter+1;
            [mx indM]=max(gss.Som(SomOrigin).codebook(:,i));
            [mn indm]=min(gss.Som(SomOrigin).codebook(:,i));
            for j=1:size(gss.View{viewNum},1)
                if gss.View{viewNum}(j).SomIndex==indm
                    Xc(counter,1)=mean(gss.View{viewNum}(j).X(1:end-1))+ (-.5+(.5+.5) *
rand);
                    Yc(counter,1)=mean(gss.View{viewNum}(j).Y(1:end-1))+ (-.5+(.5+.5) *
rand);
                    sText(counter)=gss.Som(SomOrigin).comp_names(i);
                end
                if gss.View{viewNum}(j).SomIndex==indM
                    Xc(counter,2)=mean(gss.View{viewNum}(j).X(1:end-1))+ (-.5+(.5+.5) *
rand);
```

```
                    Yc(counter,2)=mean(gss.View{viewNum}(j).Y(1:end-1))+ (-.5+(.5+.5) *
rand);
                end
            end
        end

        if isempty(Xc)==0
            %create group
            gh = hggroup;
            set(gh,'userdata','textCP');
            cmap=colormap(hsv(length(Xc)));
            for j=1:length(Xc)
                line(Xc(j,:),Yc(j,:),...
                        'parent',gh,'color',cmap(j,:));
                text(mean(Xc(j,:)),mean(Yc(j,:)),sText(j),...
                        'parent',gh,'FontSize',8,'color',cmap(j,:));
            end
        end
    end

return

function ButtonFunction(hObject,eventdata,hToolbar,viewNum)
    buttontag=get(hObject,'tag');

    bt=get(hToolbar,'children');
    % uncheck all the toogle buttons
    for i=1:length(bt)
        try
        set (bt(i),'state','off');
        catch
        end
    end
    set (hObject,'state','on');

    if strcmpi(buttontag,'zoomin')
        zoom on;
    elseif strcmpi(buttontag,'selectpoint')
        zoom off;
        set(gcf,'Pointer','cross');
    elseif strcmpi(buttontag,'selectarea')
        zoom off;
        set(gcf,'Pointer','crosshair');
    elseif strcmpi(buttontag,'selectareacolor')
        zoom off;
        set(gcf,'Pointer','topl');
    end
return

function[Vint]=IDW(xc,yc,vc,x,y,e,r1,r2)
    %%%%%%%%%%%%%%%%%%
    %%% INPUTS
    %xc = stations x coordinates (columns) [vector]
    %yc = stations y coordinates (rows) [vector]
    %vc = variable values on the point [xc yc]
    %x = interpolation points  x coordinates [vector]
    %y = interpolation points y coordinates [vector]
    %e = distance weight
    %r1 --- 'fr' = fixed radius ;  'ng' = neighbours
    %r2 --- radius lenght if r1 == 'fr' / number of neighbours if  r1 =='ng'
    %%% OUTPUTS
    %Vint --- Matrix [length(y),length(x)] with interpolated  variable values
    %%% EXAMPLES
    %%% --> V_spa=IDW(x1,y1,v1,x,y,-2,'ng',length(x1));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %   Simone Fatichi -- simonef@dicea.unifi.it
    %   Copyright 2009
    %   $Date: 2009/06/19 $
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Vint=zeros(length(y),length(x));
    xc=reshape(xc,1,length(xc));
    yc=reshape(yc,1,length(yc));
    vc=reshape(vc,1,length(vc));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if  strcmp(r1,'fr')
        if  (r2<=0)
            disp('Error: Radius must be positive')
            return
        end
        for i=1:length(x)
            for j=1:length(y)
                D=[]; V=[]; wV =[]; vcc=[];
                D= sqrt((x(i)-xc).^2 +(y(j)-yc).^2);
                D=D(D<r2); vcc=vc(D<r2);
                V = vcc.*(D.^e);
                wV = D.^e;
                if isempty(D)
                    V=NaN;
                else
                    V=sum(V)/sum(wV);
                end
                Vint(j,i)=V;
            end
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    else
        if (r2 > length(vc)) || (r2<1)
            disp('Error: Number of neighbours not congruent with data')
            return
        end
        for i=1:length(x)
            for j=1:length(y)
                D=[]; V=[]; wV =[];vcc=[];
                D= sqrt((x(i)-xc).^2 +(y(j)-yc).^2);
                [D,I]=sort(D);
                vcc=vc(I);
                V = vcc(1:r2).*(D(1:r2).^e);
                wV = D(1:r2).^e;
                V=sum(V)/sum(wV);
                Vint(j,i)=V;
            end
        end
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function geosom_ShowViewsOneFigure(type,SomIndex,colormap,UsedOnSOM);
    % geosom_ShowViewsOneFigure(viewNum,colormap)
    % NAME: geosom_ShowViewsOneFigure
    %
    % OBJECTIVE: Create a window with several Views (e.g. Component planes)
    % INPUT PARAMETERS:
    % OUTPUT PARAMETERS:
    %
    % COMMENTS:
    %
    % V.0.1 2008/07/17 By R. Henriques & V.Lobo

    global gss

    f=figure;                                % Create a figure
    set(f,'position',[100 100 600 600]);
    set(f,'Toolbar','none');
    hToolbar = uitoolbar(...
                    'Parent',f, ...
                    'HandleVisibility','callback');
```

```
% add export capabilities
mn = uimenu('Label','Export');
uimenu(mn,'Label','To new figure','Callback',...
    {@ExportSeveralAxes2NewFig});

% Create the button group.
%hg = uibuttongroup('Parent',hToolbar);

hB      =  uitoggletool(...
               'Parent',hToolbar,...
               'tag','zoomin',...
               'TooltipString','Zoom in',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\view_zoom_in.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback', {@ButtonFunction hToolbar});
hB      =  uipushtool(...
               'Parent',hToolbar,...
               'tag','zoomout',...
               'TooltipString','Zoom out',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\view_zoom_out.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback', 'zoom (0.5)');
hB      =  uipushtool(...
               'Parent',hToolbar,...
               'tag','fullextent',...
               'TooltipString','Full extent',...
                'CData',iconRead(fullfile(pwd,...
                   'icons\webicon.gif')),...
               'HandleVisibility','callback', ...
               'ClickedCallback','zoom out');
if strcmpi(type,'cp')   % component planes
    set(f,'Name','Component Planes');

    % get the views
    tempViews={};
    IdViews=[];
    names={};
    for i=1:length(gss.ViewLabel)
        if gss.ViewLabel(i).SomOrigin==SomIndex
            if UsedOnSOM
                if iscell(gss.ViewLabel(i).Name)
                    firstChar=cell2mat(gss.ViewLabel(i).Name);
                    firstChar=firstChar(1,1);
                else
                    firstChar='';
                end
                if strcmpi('*',firstChar)
                    if strcmpi(gss.ViewLabel(i).Type,'cp')
                        tempViews(length(tempViews)+1)= gss.View(i);
                        IdViews=[IdViews; i];
                        names(length(names)+1)=gss.ViewLabel(i).Name;
                    end
                end
            else
                if strcmpi(gss.ViewLabel(i).Type,'cp')
                    tempViews(length(tempViews)+1)= gss.View(i);
                    IdViews=[IdViews; i]
                    names(length(names)+1)=gss.ViewLabel(i).Name;
                end
            end
        end
    end

    % plot the tempViews
    nCols=ceil(sqrt(length(tempViews)));
    nRows=ceil(length(tempViews)/nCols);
    margin=0.005;
```

```
        pos=zeros(length(tempViews),4);
        pos(:,3)=1/nCols-(margin*(nCols+1)/nCols);
        pos(:,4)=1/nRows-(margin*(nRows+1)/nRows);

        a=repmat((0+margin:pos(1,4)+margin:1-margin)',nCols,1);
        a=a(1:length(pos));
        pos(:,1)=a;

        counter=0;
        a=flipud((0+margin:pos(1,3)+margin:1-margin)');
        for i=1:nRows
            for j=1:nCols
                counter=counter+1;
                pos(counter,2)=a(i);
            end
        end

        for i=1:length(tempViews)
            subplot(nRows,nCols,i)

            %hs=subplot('Position',pos(i,:));
            set(gca,'PlotBoxAspectRatioMode','auto');
            set(gca,'CameraViewAngleMode','auto');
            set(gca,'XColor',[1 1 1]);
            set(gca,'YColor',[1 1 1]);

            numColors=length(colormap);
            tempViews{i}=geosom_ViewUmatCreateIValue(...
                        tempViews{i},numColors );
            regra=makesymbolspec('Polygon',{'IValue',[1 numColors],...
                                        'FaceColor', colormap});
            h=mapshow(tempViews{i}, 'SymbolSpec', regra);
            text(pos(i,1),5,names{i}(5:end));
            %Remove labels
            set(gca,'XTickLabel',{},'yTickLabel',{});

             % define the figure handle in the gss structure
             gss.ViewLabel(IdViews(i)).WindowHandle=gcf;
             % define the graphic handle in the gss structure (in this case
             % represents the subplot
             gss.ViewLabel(IdViews(i)).GraphicHandle=gca;

        end
        % add a chkbox for selection based on item number
        h = uicontrol('Style', 'checkbox',...
           'String', 'Show sel. based on item count',...
           'tag','selBasedOnCount',...
           'Position', [120 5 200 20],...
           'Callback', {});
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%here is added the listener to the "click event" over the map
        set(gcf,'windowbuttondownfcn','');

        %%here is added the listener to the "close event"
        % when closing a view, the gss.viewlabel.windowHandle should be deleted
        set(gcf,'CloseRequestFcn',{@geosom_showViewClose, IdViews});
    end
return

function geosom_showViewClose(hObject,eventdata,IdViews)
    global gss

    if isfield(gss,'ViewLabel');
        %if length(gss.ViewLabel)>=viewNum
            for i=IdViews'
                gss.ViewLabel(i).WindowHandle=[];
            end
            delete (gcf);
```

```
        %end
    end
return

function ButtonFunction(hObject,eventdata,hToolbar)
    buttontag=get(hObject,'tag');

    bt=get(hToolbar,'children');
    % uncheck all the toogle buttons
    for i=1:length(bt)
        try
        set (bt(i),'state','off');
        catch
        end
    end
    set (hObject,'state','on');

    if strcmpi(buttontag,'zoomin')
        zoom on
    elseif strcmpi(buttontag,'selectpoint')
        zoom off
        set(gcf,'Pointer','cross')
    elseif strcmpi(buttontag,'selectarea')
        zoom off
        set(gcf,'Pointer','crosshair')
    end
return

function ExportSeveralAxes2NewFig(hObject,eventdata)
    %get axes and open in new figure for export

    h=gcf;
    a=get(h,'children');
    b=get(a,'type');
    h=gcf;

    numPlots=sum(ismember(b,'axes'));
    nCols=ceil(sqrt(numPlots));
    nRows=ceil(numPlots/nCols);

    %get each plot into a new figure
    hNewFig=figure;
    set(hNewFig,'color',[1 1 1]);

    for i=1:numPlots
        copyobj(a(i), hNewFig);
        set(gca,'PlotBoxAspectRatioMode','auto');
        set(gca,'CameraViewAngleMode','auto');
        set(gca,'XColor',[1 1 1]);
        set(gca,'YColor',[1 1 1]);
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_SomTrainingParameters()
    %geosom_SomTrainingParameters
    % form to define the SOM trainning parameters
    %
    % Syntax: geosom_SomTrainingParameters()
    % input - none
    % output - none
    %
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
```

```
% Created: September 04-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008

global gss

fh=figure('Position',[200 200 400 450]);
% clear default menu and toolbars and add title
set(fh,'MenuBar','none','toolbar','none','NumberTitle','off',...
    'Resize','off','name','Train SOM');
handles.trainSOM=fh;

% % add controls
% text - title
h = uicontrol('Style', 'text',...
       'String', 'SOM training parameters',...
       'tag', 'textViews',...
       'Position', [10 430 380 15],...
       'HorizontalAlignment','Center',...
       'fontweight','bold');
handles.textViews=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% panel map initialization
hp = uipanel('Title','Map Initialization',...
             'units','pixels','Position',[10 350 380 80]);
% size X
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'X',...
       'tag', 'textX',...
       'Position', [10 40 40 15],...
       'HorizontalAlignment','left');
handles.textX=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '5',...
       'tag', 'editX',...
       'background','white',...
       'Position', [50 40 50 20]);
handles.editX=h;

% size Y
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Y',...
       'tag', 'textY',...
       'Position', [130 40 40 15],...
       'HorizontalAlignment','left');
handles.textY=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '5',...
       'tag', 'editY',...
       'background','white',...
       'Position', [170 40 50 20]);
handles.editY=h;

% lattice
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Lattice',...
       'tag', 'textLattice',...
       'Position', [10 15 40 15],...
       'HorizontalAlignment','left');
handles.textLattice=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'hexa';'rect'},...
```

```
       'tag', 'popupLattice',...
       'Position', [50 15 70 20]);
handles.popupLattice=h;

% shape
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Shape',...
       'tag', 'textShape',...
       'Position', [130 15 40 15],...
       'HorizontalAlignment','left');
handles.textShape=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'sheet';'cyl';'toroid'},...
       'tag', 'popupShape',...
       'Position', [170 15 70 20]);
handles.popupShape=h;

% type
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Type',...
       'tag', 'textType',...
       'Position', [250 15 40 15],...
       'HorizontalAlignment','left');
handles.textType=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'linear';'random'},...
       'tag', 'popupType',...
       'Position', [280 15 70 20]);
handles.popupType=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize data
h = uicontrol('Style', 'text',...
       'String', 'Normalize data',...
       'tag', 'textNormalize',...
       'Position', [10 320 80 15],...
       'HorizontalAlignment','left');
handles.textNormalize=h;
h = uicontrol('Style', 'popup',...
       'String', {'none';'var';'range';'log';'logistic';'histD';'histC'},...
       'tag', 'popupNormalize',...
       'Position', [10 300 80 20]);
handles.popupNormalize=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% components to select
h = axes('tag', 'TableComponents','AmbientLightColor','white',...
       'units','pixels','Position', [130 240 250 100]);
handles.TableComponents=h;

% replace the character _ by space
Comp = gss.NumDataLabel';
for i=1:length(Comp)
    Comp{i}=strrep(Comp{i}, '_', ' ');
end

columninfo.titles={'Component'};
columninfo.formats = {'%4.6g'};
columninfo.weight =      [1];
columninfo.multipliers = [1];
columninfo.isEditable =  [0];
columninfo.isNumeric =   [0];
columninfo.withCheck = true; % optional to put checkboxes along left side
columninfo.chkLabel = 'All'; % optional col header for checkboxes
rowHeight = 12;
```

```
    gFont.size=8;
    gFont.name='Arial';
    geosom_MlTable(handles.trainSOM, handles.TableComponents, 'CreateTable',
columninfo,rowHeight, Comp, gFont);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Map train
    hp = uipanel('Title','Map Train',...
                 'units','pixels','Position',[10 160 380 80]);

    % batch or sequential button group
    hb = uibuttongroup('Title','','tag','uibtgroupTrain','parent',hp,...
                 'units','pixels','Position',[10 40 360 20]);
    handles.uibtgroupTrain=hb;

    %option buttons
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'Batch train',...
           'tag', 'chkbatch',...
           'Position', [50 0 100 15],...
           'Callback', '');
    handles.chkbatch=h;
    h = uicontrol('Style', 'radiobutton',...
           'parent',hb,...
           'String', 'Sequential train',...
           'tag', 'chkSequential',...
           'Position', [200 0 100 15],...
           'Callback', '');
    handles.chkSequential=h;

    % sequential trainning parameters panel
    hp2 = uipanel('Title','','parent',hp,...
                 'tag','panelTrain',...
                 'units','pixels',...
                 'visible','off',...
                 'Position',[100 5 270 35]);
    handles.panelTrain=hp2;

    % Iterations type
    h = uicontrol('Style', 'text',...
           'Parent',hp2,...
           'String', 'Iter.',...
           'tag', 'textIter',...
           'Position', [5 5 40 15],...
           'HorizontalAlignment','left');
    handles.textIter=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp2,...
           'String', {'epochs';'samples'},...
           'tag', 'popupIter',...
           'Position', [25 5 60 20]);
    handles.popupIter=h;

    % order
    h = uicontrol('Style', 'text',...
           'Parent',hp2,...
           'String', 'Order',...
           'tag', 'textOrder',...
           'Position', [85 5 40 15],...
           'HorizontalAlignment','left');
    handles.textOrder=h;
    h = uicontrol('Style', 'popup',...
           'Parent',hp2,...
           'String', {'random';'ordered'},...
           'tag', 'popupOrder',...
           'Position', [115 5 60 20]);
    handles.popupOrder=h;
```

```
% Length function
h = uicontrol('Style', 'text',...
       'Parent',hp2,...
       'String', 'Length',...
       'tag', 'textLength',...
       'Position', [175 5 40 15],...
       'HorizontalAlignment','left');
handles.textLength=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp2,...
       'String', {'inv';'linear';'power'},...
       'tag', 'popupLength',...
       'Position', [210 5 50 20]);
handles.popupLength=h;

% Neigh function
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Neigh',...
       'tag', 'textNeigh',...
       'Position', [10 10 40 15],...
       'HorizontalAlignment','left');
handles.textNeigh=h;
h = uicontrol('Style', 'popup',...
       'Parent',hp,...
       'String', {'gaussian';'cutgauss';'ep';'bubble'},...
       'tag', 'popupNeigh',...
       'Position', [40 10 60 20]);
handles.popupNeigh=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rough parameters
hp = uipanel('Title','Rough',...
             'tag','panelRough',...
             'units','pixels',...
             'Position',[10 80 190 80]);
handles.panelRough=hp;

% Iterations_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 1',...
       'tag', 'textIterations1',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editIterations1',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations1=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 1',...
       'tag', 'textRadio1',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editRadio1',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio1=h;
% Alpha 1
```

```
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 1',...
       'tag', 'textAlpha1',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha1=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '0.3',...
       'tag', 'editAlpha1',...
       'background','white',...
       'enable','off',...
       'Position', [100 5 40 20]);
handles.editAlpha1=h;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finetune parameters
hp = uipanel('Title','Finetune',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[200 80 190 80]);
handles.panelFinetune=hp;

% Iterations_2
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Iterations 2',...
       'tag', 'textIterations2',...
       'Position', [30 45 55 15],...
       'HorizontalAlignment','left');
handles.textIterations2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editIterations2',...
       'background','white',...
       'Position', [100 45 40 20]);
handles.editIterations2=h;
% Radio_1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Radio 2',...
       'tag', 'textRadio2',...
       'Position', [30 25 55 15],...
       'HorizontalAlignment','left');
handles.textRadio2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '10',...
       'tag', 'editRadio2',...
       'background','white',...
       'Position', [100 25 40 20]);
handles.editRadio2=h;
% Alpha 1
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Alpha 2',...
       'tag', 'textAlpha2',...
       'Position', [30 5 55 15],...
       'HorizontalAlignment','left');
handles.textAlpha2=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '0.1',...
       'tag', 'editAlpha2',...
       'background','white',...
       'enable','off',...
```

```
       'Position', [100 5 40 20]);
handles.editAlpha2=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% description parameters
hp = uipanel('Title','Description',...
             'tag','panelFinetune',...
             'units','pixels',...
             'Position',[10 10 190 70]);
handles.panelDescription=hp;
% name
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Name  SOM',...
       'Position', [10 35 65 15],...
       'HorizontalAlignment','left');
handles.textName=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', [date],...
       'tag', 'editName',...
       'background','white',...
       'HorizontalAlignment','left',...
       'Position', [70 35 110 20]);
handles.editName=h;
%obs
h = uicontrol('Style', 'text',...
       'Parent',hp,...
       'String', 'Obs.',...
       'Position', [10 20 55 15],...
       'HorizontalAlignment','left');
handles.textObs=h;
h = uicontrol('Style', 'edit',...
       'Parent',hp,...
       'String', '',...
       'tag', 'editObs',...
       'max',2,'min',0,...
       'background','white',...
       'HorizontalAlignment','left',...
       'Position', [40 5 140 30]);
handles.editObs=h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%chk only finetune
h = uicontrol('Style', 'checkbox',...
       'String', 'Use only finetune',...
       'tag', 'chkFinetune',...
       'Position', [210 55 105 15]);
handles.chkFinetune=h;

%train only selected data
h = uicontrol('Style', 'checkbox',...
       'String', 'Train only selected data',...
       'tag', 'chkTrainSelection',...
       'Position', [210 35 140 15]);
handles.chkTrainSelection=h;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%push buttons
h = uicontrol('Style', 'pushbutton',...
       'String', 'OK',...
       'tag', 'pushOK',...
       'Position', [340 5 50 15],...
       'Callback', 'setmap');
handles.pushOK=h;

%%% CALLBACKS with handles %%%
set(handles.trainSOM,'CloseRequestFcn',{@trainSOM_CloseRequest handles});
```

```
    set(handles.uibtgroupTrain,'SelectionChangeFcn',{@changeTrain handles});
    set(handles.pushOK,'Callback',{@pushOK_Callback handles});


    % Make the GUI modal
    set(handles.trainSOM ,'WindowStyle','modal')
    % UIWAIT
    uiwait(handles.trainSOM);
return

function trainSOM_CloseRequest(hObject, eventdata,handles)
    uiresume(handles.trainSOM);
    delete (handles.trainSOM);
return

function changeTrain(hObject, eventdata,handles)

    if get(handles.chkbatch,'value');
        set(handles.panelTrain,'visible','off');
        set(handles.editAlpha1,'enable','off');
        set(handles.editAlpha2,'enable','off');
    else get(handles.chkSequential,'value');
        set(handles.panelTrain,'visible','on');
        set(handles.editAlpha1,'enable','on');
        set(handles.editAlpha2,'enable','on');
    end
return

function pushOK_Callback(hObject, eventdata, handles)
    global gss
    % X
    user_entry = str2double(get(handles.editX,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric X map size!','Training SOM');
        return
    else
        parameters.xdim = user_entry;
    end

    % Y
    user_entry = str2double(get(handles.editY,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Y map size!','Training SOM');
        set(h,'string','');
        return
    else
       parameters.ydim = user_entry;
    end

    % Lattice
    Lattice=get(handles.popupLattice,'string');
    parameters.Lattice=cell2mat(Lattice(get(handles.popupLattice,'value'),:));

    % Shape
    Shape=get(handles.popupShape,'string');
    parameters.Shape=cell2mat(Shape(get(handles.popupShape,'value'),:));

    % map type
    MapType=get(handles.popupType,'string');
    parameters.MapType=cell2mat(MapType(get(handles.popupType,'value'),:));

    % normalization
    Norm=get(handles.popupNormalize,'string');
    parameters.Normalization=cell2mat(Norm(get(handles.popupNormalize,'value'),:));

    % components
    comp = get(handles.TableComponents, 'userdata');
    parameters.SelectedComponents=find(comp.isChecked==1);
```

```
    if isempty(parameters.SelectedComponents)
        helpdlg('Please select a component!','Training SOM');
        return
    end
    parameters.ComponentsNames=comp.data;

    % Neigh
    Neigh=get(handles.popupNeigh,'string');
    parameters.Neigh=cell2mat(Neigh(get(handles.popupNeigh,'value'),:));

    % batch or sequential train?
    if get(handles.chkSequential,'value')
        %IterType
        IterType=get(handles.popupIter,'string');
        parameters.IterType=cell2mat(IterType(get(handles.popupIter,'value'),:));
        % order
        Order=get(handles.popupOrder,'string');
        parameters.Order=cell2mat(Order(get(handles.popupOrder,'value'),:));
        % Length function
        Length_funct=get(handles.popupLength,'string');

parameters.Length_funct=cell2mat(Length_funct(get(handles.popupLength,'value'),:));
    end

    % rough and finetune parameters
    parameters.chkFinetune=get(handles.chkFinetune,'Value');

    user_entry = str2double(get(handles.editIterations1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a number of Iterations 1!','Training SOM');
        return
    else
        parameters.niterations_1 = user_entry;
    end

    user_entry = str2double(get(handles.editIterations2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a number of Iterations 2!','Training SOM');
        return
    else
        parameters.niterations_2 = user_entry;
    end

    user_entry = str2double(get(handles.editRadio1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Radio 1!','Training SOM');
        return
    else
        parameters.radius_ini_1 = user_entry;
    end

    user_entry = str2double(get(handles.editRadio2,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Radio 2!','Training SOM');
        return
    else
        parameters.radius_ini_2 = user_entry;
    end

    user_entry = str2double(get(handles.editAlpha1,'string'));
    if isnan(user_entry)
        helpdlg('Please insert a numeric Alpha 1!','Training SOM');
        return
    else
        parameters.alpha_ini_1 = user_entry;
    end

    user_entry = str2double(get(handles.editAlpha2,'string'));
    if isnan(user_entry)
```

```
        helpdlg('Please insert a numeric Alpha 2!','Training SOM');
        return
    else
        parameters.alpha_ini_2 = user_entry;
    end

    %name
    parameters.SOMname = ['SOM ' get(handles.editName,'string')];

    %Obs.
    parameters.Obs = get(handles.editObs,'string');

    %train
    parameters.batchTrain= get(handles.chkbatch,'value');

    % only selected data?
    parameters.TrainSelection=get(handles.chkTrainSelection,'value');

    %train SOM
    % disableButtons
    set(handles.trainSOM,'Pointer','watch');
    set(handles.pushOK  ,'Enable','off');
    refresh(gcf) %redraws the GUI to reflect changes

    %train SOM and make new view
    geosom_BuildSom (parameters);

    somIndex = length(gss.Som);
    geosom_BuildViewUmat(somIndex);

    % % build component planes
    geosom_BuildComponentPlanes(somIndex);
    % % build Hits
    geosom_BuildHits(somIndex)

    %resume
    uiresume(handles.trainSOM);
    close(handles.trainSOM);
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gss = geomsom_TableDrawn()
    %GEOSOM_TABLEDRAW ...
    % Syntax: gss = geomsom_TableDrawn()
    % gss -
    % function to represent in the table the selection
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: December 30-2009
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2009

  global gss
    global mainHandles  % handles of main window

    numrows=size(gss.NumData,1);
    cols_num = size(gss.NumData,2);
    cols_cell = size(gss.CellData,2);

    data=[num2cell(gss.SelectedIndex) num2cell(gss.NumData) gss.CellData];
    if get(mainHandles.chkSel,'value')==0 % see all records
        set(mainHandles.uitable1,'Data',data);
        columneditable =[true(1,1) false(1,cols_num+cols_cell)];
    else % see only the selection
        set(mainHandles.uitable1,'Data',data(find(gss.SelectedIndex==1),:));
        columneditable =false(1,cols_num+cols_cell+1);
    end
```

```
    % write in the check selected data label the number of selected features
    if sum(gss.SelectedIndex)>0
        set (mainHandles.chkSel,'string',strcat('See only selected data (' ...
            ,num2str(sum(gss.SelectedIndex)), ' records)'));
    else
        set (mainHandles.chkSel,'string','See only selected data')
    end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function geosom_UMATClick(h,evd, viewNumber)
    % geosom_UMATClick - Click in a geosom view
    % Syntax: geosom_UMATClick(h,evd, viewNumber)
    % input
    % output
    %           h - handle
    %           evd -
    %           viewNumber - view number
    % Example
    %
    % Subfunctions:
    %   See also:
    %
    % Author: Roberto Henriques, Victor Lobo, Fernando Bação
    % Created: September 04-2008
    % Version 1.0
    % Copyright (c) by the Ga2 programming team. 2008

    global gss

    seltype=get(gcf,'selectiontype');

    %check if the cross pointer is selected indicating that the select tool
    %is on
    if strcmp(get(gcf,'Pointer'),'cross')==1
        coord=get(gca,'CurrentPoint');
        coord=[coord(1,1),coord(1,2)];
    elseif strcmp(get(gcf,'Pointer'),'crosshair')==1
        %select the rectangle
        point1 = get(gca,'CurrentPoint');    % button down detected
        finalRect = rbbox;                   % return figure units
        point2 = get(gca,'CurrentPoint');    % button up detected
        point1 = point1(1,1:2);              % extract x and y
        point2 = point2(1,1:2);
        p1 = min(point1,point2);             % calculate locations
        offset = abs(point1-point2);         % and dimensions
        x = [p1(1) p1(1)+offset(1) p1(1)+offset(1) p1(1) p1(1)];
        y = [p1(2) p1(2) p1(2)+offset(2) p1(2)+offset(2) p1(2)];
        coord=[x' y'];
    elseif strcmp(get(gcf,'Pointer'),'topl')==1
        %select the rectangle
        point1 = get(gca,'CurrentPoint');    % button down detected
        finalRect = rbbox;                   % return figure units
        point2 = get(gca,'CurrentPoint');    % button up detected
        point1 = point1(1,1:2);              % extract x and y
        point2 = point2(1,1:2);
        p1 = min(point1,point2);             % calculate locations
        offset = abs(point1-point2);         % and dimensions
        x = [p1(1) p1(1)+offset(1) p1(1)+offset(1) p1(1) p1(1)];
        y = [p1(2) p1(2) p1(2)+offset(2) p1(2)+offset(2) p1(2)];
        coord=[x' y'];
    else
        return
    end

    indice=geosom_ClosestItem(coord,gss,viewNumber);
    if isempty(indice)
        return
```

```
    end

    hh = gss.ViewLabel(viewNumber).GraphicHandle;

    if strcmpi(seltype,'normal')==1, %normal
        gss.SelectedIndex(:)=0;
        for j=indice
            sel=find(gss.ViewIndexData(:,viewNumber)==j);
            if isempty(sel)~=1
                gss.SelectedIndex(sel)=1;
            end
        end
    elseif strcmpi(seltype,'Extend')==1, %shift - adds features to current selection
        for j=indice
            sel=find(gss.ViewIndexData(:,viewNumber)==j);
            if isempty(sel)~=1
                gss.SelectedIndex(sel)=1;
            end
        end
    elseif strcmpi(seltype,'alt')==1, %control - rmeoves features to current selection
        for j=indice
            sel=find(gss.ViewIndexData(:,viewNumber)==j);
            if isempty(sel)~=1
                gss.SelectedIndex(sel)=1;
            end
        end
        gss.SelectedIndex(sel)=0;
    end

    %show selection in all the views
    if strcmp(get(gcf,'Pointer'),'cross')==1    % select using 1 color
        geosom_MakeSelectionAllOpenViews
    elseif strcmp(get(gcf,'Pointer'),'crosshair')==1
        geosom_MakeSelectionAllOpenViews
    elseif strcmp(get(gcf,'Pointer'),'topl')==1    % select using 1 color per BMU
        geosom_MakeSelectionAllOpenViewsByColor(viewNumber)
    end

    %show selection in table
    gss = geosom_TableDraw();
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function umatView=geosom_ViewUmatCreateIValue( umatView, numColors )
    % geosom_ViewUmatCreateIValue( umatView, numColors )
    %
    % NAME: geosom_ViewUmatCreateIValue
    %
    % OBJECTIVE: Create a field, named IValue, in UMAT type views. This field
    % exists for each element of the UMAT, and acts as index into a colormap.
    % It may then be used to create a rule, using the 'makeSymbolSpec'
    %
    % INPUT PARAMETERS:
    %       umatView - structure of the type 'geostruct2', with 'Type'='UMAT',
    %                  and thus with a field named Value
    %       numColors - Number of different colors to use
    %
    % OUTPUT PARAMETERS:
    %       umatView - modified structure of the type 'geostruct2'
    %
    % COMMENTS:
    %       This is an auxiliary function for visualizing UMAT
    %

    % V.0.1 2007/01/28 By V.Lobo & M.Loureiro

    min=umatView(1).Value;                      % First find max and Min values
    max=min;
```

```
    nelements=length(umatView);
    for i=2:nelements;
        if umatView(i).Value > max;
            max=umatView(i).Value;
        elseif umatView(i).Value < min;
            min=umatView(i).Value;
        end;
    end;
    umatView(1).IValue=1;                       %...then create the indexes
    for i=1:nelements;
        umatView(i).IValue=...
            floor((umatView(i).Value-min)/(max-min)*(numColors-1)+1);
    end;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Bmus,qerror_geo,qerror_dta] = som_bmus_geo2(sMap, sData, kradius)
    % function [Bmus,qerror_geo,qerror_dta] = som_bmus_geo2(sMap, sData, kradius)
    %
    % Find the BMU for each data pattern, together with the geographical error,
    % and the error in the NON-Geographic variables
    %
    % INPUTS
    %   sMAP    - SOM map
    %   sData   - SOM data
    %   kradius - Geographic tolerance (squared)
    %
    % OUTPUTS
    %   Bmus        - Index of the BMU for each pattern
    %   qerror_geo  - Geographic quantization error (BMU-Pattern)
    %   qerror_dta  - Data (NON-Geographic) quantization error (BMU-Pattern)

    % V1.0 by V.Lobo & F.Bacao, 31/10/2004


    % SOM_BMUS Find the best-matching units from the map for the given vectors.
    % check arguments and initialize

    % ---- GEO initializations
    geo=0;Ud = som_unit_dists(sMap.topol).^2;
    % kradius2=kradius^2;
    % ---- End GEO initializations;
    error(nargchk(1, 3, nargin));  % check no. of input args is correct

    if kradius == -1
        kradius = inf;
    end;

    % sMap
    if isstruct(sMap),
      switch sMap.type,
       case 'som_map', M = sMap.codebook;
       case 'som_data', M = sMap.data;
       otherwise, error('Invalid 1st argument.');
      end
    else
      M = sMap;
    end
    [munits dim] = size(M);
    if any(any(isnan(M))),
      error ('Map codebook must not have missing components.');
    end

    % data
    if isstruct(sData),
      switch sData.type,
       case 'som_map', D = sData.codebook;
       case 'som_data', D = sData.data;
```

```
       otherwise, error('Invalid 2nd argument.');
      end
    else
      D = sData;
    end
    [dlen ddim] = size(D);
    if dim ~= ddim,
      error('Data and map dimensions do not match.')
    end

    qerror_geo=zeros(1,dlen);
    qerror_dta=qerror_geo;
    Bmus=qerror_geo;

    for i=1:dlen        % search each data point independently
        Dst=dist( D(i,1:2),M(:,1:2)');
        [qerror_geo(i) bmu_geo]=min(Dst);

        non_searchable_units = find(Ud(bmu_geo,1:munits)>kradius);

        Dst=dist( D(i,3:end),M(:,3:end)');
        Dst(non_searchable_units)=inf;
        [qerror_dta(i) bmu_dta]=min(Dst);

        Bmus(i)=bmu_dta;
    end;
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [hits] = som_hits_geo(sMap, sData,kradius,mode)
    % GEOSOM som_hits function
    %SOM_HITS Calculate the response of the given data on the map.
    %
    % hits = som_hits(sMap, sData, [mode])
    %
    %   h = som_hits(sMap,sData);
    %   h = som_hits(sMap,sData,'fuzzy');
    %
    %  Input and output arguments ([]'s are optional):
    %   sMap     (struct) map struct
    %            (matrix) codebook matrix, size munits x dim
    %   sData    (struct) data struct
    %            (matrix) data matrix, size dlen x dim
    %   [mode]   (string) 'crisp' (default), 'kernel', 'fuzzy'
    %
    %   hits     (vector) the number of hits in each map unit, length = munits
    %
    % The response of the data on the map can be calculated e.g. in
    % three ways, selected with the mode argument:
    %  'crisp'    traditional hit histogram
    %  'kernel'   a sum of dlen neighborhood kernels, where kernel
    %             is positioned on the BMU of each data sample. The
    %             neighborhood function is sMap.neigh and the
    %             neighborhood width is sMap.trainhist(end).radius_fin
    %             or 1 if this is empty or NaN
    %  'fuzzy'    fuzzy response calculated by summing 1./(1+(q/a)^2)
    %             for each data sample, where q is a vector containing
    %             distance from the data sample to each map unit and
    %             a is average quantization error
    %
    % For more help, try 'type som_hits' or check out online documentation.
    % See also SOM_AUTOLABEL, SOM_BMUS.

    %%%%%%%%%%%%%% DETAILED DESCRIPTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % som_hits
    %
    % PURPOSE
```

```
%
% Calculate the response of the given data on the map.
%
% SYNTAX
%
%  hits = som_hits(sMap, sData)
%  hits = som_hits(M, D)
%  hits = som_hits(..., mode)
%
% DESCRIPTION
%
% Returns a vector indicating the response of the map to the data.
% The response of the data on the map can be calculated e.g. in
% three ways, selected with the mode argument:
%  'crisp'    traditional hit histogram: how many times each map unit
%             was the BMU for the data set
%  'kernel'   a sum of neighborhood kernels, where a kernel
%             is positioned on the BMU of each data sample. The
%             neighborhood function is sMap.neigh and the
%             neighborhood width is sMap.trainhist(end).radius_fin
%             or 1 if this is not available
%  'fuzzy'    fuzzy response calculated by summing
%
%                          1
%                      ------------
%                      1 +  (q/a)^2
%
%             for each data sample, where q is a vector containing
%             distance from the data sample to each map unit and
%             a is average quantization error
%
% REQUIRED INPUT ARGUMENTS
%
%   sMap              The vectors from among which the BMUs are searched
%                     for. These must not have any unknown components (NaNs).
%            (struct) map struct
%            (matrix) codebook matrix, size munits x dim
%
%   sData             The data vector(s) for which the BMUs are searched.
%            (struct) data struct
%            (matrix) data matrix, size dlen x dim
%
% OPTIONAL INPUT ARGUMENTS
%
%   mode     (string) The respond mode: 'crisp' (default), 'kernel'
%                     or 'fuzzy'. 'kernel' can only be used if
%                     the first argument (sMap) is a map struct.
%
% OUTPUT ARGUMENTS
%
%   hits     (vector) The number of hits in each map unit.
%
% EXAMPLES
%
%  hits = som_hits(sM,D);
%  hits = som_hits(sM,D,'kernel');
%  hits = som_hits(sM,D,'fuzzy');
%
% SEE ALSO
%
%  som_bmus      Find BMUs and quantization errors for a given data set.

% Copyright (c) 1997-2000 by the SOM toolbox programming team.
% http://www.cis.hut.fi/projects/somtoolbox/

% Version 1.0beta juuso 220997
% Version 2.0beta juuso 161199

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% check arguments

error(nargchk(3, 4, nargin));  % check no. of input args is correct

if isstruct(sMap),
  switch sMap.type,
   case 'som_map', munits = prod(sMap.topol.msize);
   case 'som_data', munits = size(sMap.data,1);
   otherwise,
    error('Illegal struct for 1st argument.')
  end
else
  munits = size(sMap,1);
end
hits = zeros(munits,1);

if nargin<4, mode = 'crisp'; end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% action

% calculate Geo BMUs
[bmus,qerrs] = som_bmus_geo(sMap,sData,kradius);

switch mode,
case 'crisp',

% for each unit, check how many hits it got
for i=1:munits, hits(i) = sum(bmus == i); end

case 'kernel',

 % check that sMap really is a map
 if ~isstruct(sMap) & ~strcmp(sMap.type,'som_map'),
   error('Kernel mode can only be used for maps.');
 end

 % calculate neighborhood kernel
 Ud = som_unit_dists(sMap.topol).^2;
 sTrain = sMap.trainhist(end);
 if ~isempty(sTrain),
   rad = sTrain.radius_fin;
   if isempty(rad) | isnan(rad), rad = 1; end
 else
   rad = 1;
 end
 rad = rad^2;
 if rad==0, rad = eps; end % to avoid divide-by-0 errors
 switch sTrain.neigh,
  case 'bubble',   H = (Ud<=rad);
  case 'gaussian', H = exp(-Ud/(2*rad));
  case 'cutgauss', H = exp(-Ud/(2*rad)) .* (Ud<=rad);
  case 'ep',       H = (1-Ud/rad) .* (Ud<=rad);
 end

 % weight hits with neighborhood kernel
 hits = sum(H(bmus,:),1)';

case 'fuzzy',

 % extract the two matrices (M, D) and the mask
 mask = [];
 if isstruct(sMap),
   if strcmp(sMap.type,'som_data'), M = sMap.data;
   else M = sMap.codebook; mask = sMap.mask;
   end
 else M = sMap;
 end
 if any(isnan(M(:))),
```

```
      error('Data in first argument must not have any NaNs.');
    end

    if isstruct(sData),
      switch sData.type,
       case 'som_map',
        D = sData.codebook;
        if isempty(mask), mask = sData.mask; end
       case 'som_data', D = sData.data;
       otherwise, error('Illegal 2nd argument.');
      end
    else D = sData;
    end
    [dlen dim] = size(D);
    if isempty(mask), mask = ones(dim,1); end

    % scaling factor
    a = mean(qerrs).^2;

    % calculate distances & bmus
    % (this is better explained in som_batchtrain and som_bmus)
    Known = ~isnan(D); D(find(~Known)) = 0; % unknown components
    blen = min(munits,dlen); % block size
    W1 = mask*ones(1,blen); W2 = ones(munits,1)*mask'; D = D'; Known = Known';
    i0 = 0;
    while i0+1<=dlen,
      inds = [(i0+1):min(dlen,i0+blen)]; i0 = i0+blen; % indeces
      Dist = (M.^2)*(W1(:,1:length(inds)).*Known(:,inds)) ...
         + W2*(D(:,inds).^2) ...
         - 2*M*diag(mask)*D(:,inds); % squared distances
      hits = hits + sum(1./(1+Dist/a),2);
    end

    otherwise,
     error(['Unknown mode: ' mode]);

   end
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [mqe_geo,mqe_dta,uqe_geo,uqe_dta] = som_quality_geo(sMap, D, bmu_ind,want_q_err)
    % SOM_QUALITY_GEO Calculate the mean quantization for GEO problems
    %
    % [mqe_geo,mqe_dta,uqe_geo,uqe_dta] = som_quality_geo(sMap, D, bmu_ind,want_q_err)
    %
    %  qe = som_quality(sMap,D);
    %
    %  INPUTS
    %   sMap     (struct) a map struct
    %   D                 the data
    %            (struct) a data struct
    %            (matrix) a data matrix, size dlen x dim
    %   bmu_ind  (vector) a vector with the indexes of the bmus
    %   want_q_error (bool) a boolean value indicating
    %
    % OUTPUTS
    %   mqe_geo       (scalar) mean quantization error in geo-coords
    %   mqe_dta       (scalar) mean quantization error in non-geo coords
    %   uqe_geo       (scalar) quantization error in geo-coords for EACH UNIT
    %   uqe_dta       (scalar) quantization error in non-geo coords for E.U.

    % Version 1.0   12/6/2004

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% check arguments

    % input arguments
    if nargin < 2, error('Not enough input arguments.'); end
```

```
    if nargin < 4, want_q_err =-1; end;

    % data
    if isstruct(D), D = D.data; end
    [dlen dim] = size(D);

    %define geo and non-geo indeces
    ind_geo=[1 2];
    ind_dta=[3:dim];
    qe_geo=zeros(dlen,1);
    qe_dta=qe_geo;

    for i=1:dlen
        qe_geo(i)=dist(D(i,ind_geo),sMap.codebook(bmu_ind(i),ind_geo)');
        qe_dta(i)=dist(D(i,ind_dta),sMap.codebook(bmu_ind(i),ind_dta)');
    end;

    mqe_geo=mean(qe_geo);
    mqe_dta=mean(qe_dta);

    uqe_geo=[];
    uqe_dta=[];
    if want_q_err ~= -1
        [nunits x]=size(sMap.codebook);
        uqe_geo=zeros(nunits,1);
        uqe_dta=uqe_geo;
        for i=1:nunits
            ind = find( bmu_ind == i);
            uqe_geo(i)=mean(qe_geo(ind));
            uqe_dta(i)=mean(qe_dta(ind));
        end;
    end;
return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sMap = som_randinit_geo(D, varargin)
    %SOM_RANDINIT Initialize a Self-Organizing Map with random values.
    %
    % sMap = som_randinit(D, [[argID,] value, ...])
    %
    %  sMap = som_randinit(D);
    %  sMap = som_randinit(D,sMap);
    %  sMap = som_randinit(D,'munits',100,'hexa');
    %
    %  Input and output arguments ([]'s are optional):
    %    D                 The training data.
    %              (struct) data struct
    %              (matrix) data matrix, size dlen x dim
    %   [argID,    (string) Parameters affecting the map topology are given
    %    value]    (varies) as argument ID - argument value pairs, listed below.
    %
    %   sMap       (struct) map struct
    %
    % Here are the valid argument IDs and corresponding values. The values
    % which are unambiguous (marked with '*') can be given without the
    % preceeding argID.
    %  'munits'       (scalar) number of map units
    %  'msize'        (vector) map size
    %  'lattice'     *(string) map lattice: 'hexa' or 'rect'
    %  'shape'       *(string) map shape: 'sheet', 'cyl' or 'toroid'
    %  'topol'       *(struct) topology struct
    %  'som_topol','sTopol'    = 'topol'
    %  'map'         *(struct) map struct
    %  'som_map','sMap'        = 'map'
    %
    % For more help, try 'type som_randinit' or check out online documentation.
    % See also SOM_MAP_STRUCT, SOM_LININIT, SOM_MAKE.
```

```
%%%%%%%%%%%%% DETAILED DESCRIPTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% som_randinit
%
% PURPOSE
%
% Initializes a SOM with random values.
%
% SYNTAX
%
%  sMap = som_randinit(D)
%  sMap = som_randinit(D,sMap);
%  sMap = som_randinit(D,'munits',100,'hexa');
%
% DESCRIPTION
%
% Initializes a SOM with random values. If necessary, a map struct
% is created first. For each component (xi), the values are uniformly
% distributed in the range of [min(xi) max(xi)].
%
% REQUIRED INPUT ARGUMENTS
%
%  D                The training data.
%           (struct) Data struct. If this is given, its '.comp_names' and
%                    '.comp_norm' fields are copied to the map struct.
%           (matrix) data matrix, size dlen x dim
%
% OPTIONAL INPUT ARGUMENTS
%
%  argID (string) Argument identifier string (see below).
%  value (varies) Value for the argument (see below).
%
%  The optional arguments can be given as 'argID',value -pairs. If an
%  argument is given value multiple times, the last one is used.
%
%  Here are the valid argument IDs and corresponding values. The values
%  which are unambiguous (marked with '*') can be given without the
%  preceeding argID.
%  'dlen'         (scalar) length of the training data
%  'data'         (matrix) the training data
%                *(struct) the training data
%  'munits'       (scalar) number of map units
%  'msize'        (vector) map size
%  'lattice'     *(string) map lattice: 'hexa' or 'rect'
%  'shape'       *(string) map shape: 'sheet', 'cyl' or 'toroid'
%  'topol'       *(struct) topology struct
%  'som_topol','sTopol'    = 'topol'
%  'map'          *(struct) map struct
%  'som_map','sMap'        = 'map'
%
% OUTPUT ARGUMENTS
%
%  sMap     (struct) The initialized map struct.
%
% EXAMPLES
%
%  sMap = som_randinit(D);
%  sMap = som_randinit(D,sMap);
%  sMap = som_randinit(D,sTopol);
%  sMap = som_randinit(D,'msize',[10 10]);
%  sMap = som_randinit(D,'munits',100,'hexa');
%
% SEE ALSO
%
%  som_map_struct   Create a map struct.
%  som_lininit      Initialize a map using linear initialization algorithm.
%  som_make         Initialize and train self-organizing map.
```

```
% Copyright (c) 1997-2000 by the SOM toolbox programming team.
% http://www.cis.hut.fi/projects/somtoolbox/

% Version 1.0beta ecco 100997
% Version 2.0beta juuso 101199

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% check arguments

% data
if isstruct(D),
  data_name = D.name;
  comp_names = D.comp_names;
  comp_norm = D.comp_norm;
  D = D.data;
  struct_mode = 1;
else
  data_name = inputname(1);
  struct_mode = 0;
end
[dlen dim] = size(D);

% varargin
sMap = [];
sTopol = som_topol_struct;
sTopol.msize = 0;
munits = NaN;
i=1;
while i<=length(varargin),
  argok = 1;
  if ischar(varargin{i}),
    switch varargin{i},
     case 'munits',     i=i+1; munits = varargin{i}; sTopol.msize = 0;
     case 'msize',      i=i+1; sTopol.msize = varargin{i};
                               munits = prod(sTopol.msize);
     case 'lattice',    i=i+1; sTopol.lattice = varargin{i};
     case 'shape',      i=i+1; sTopol.shape = varargin{i};
     case {'som_topol','sTopol','topol'}, i=i+1; sTopol = varargin{i};
     case {'som_map','sMap','map'}, i=i+1; sMap = varargin{i}; sTopol = sMap.topol;
     case {'hexa','rect'},          sTopol.lattice = varargin{i};
     case {'sheet','cyl','toroid'}, sTopol.shape = varargin{i};
     otherwise argok=0;
    end
  elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
    switch varargin{i}.type,
     case 'som_topol',
      sTopol = varargin{i};
     case 'som_map',
      sMap = varargin{i};
      sTopol = sMap.topol;
     otherwise argok=0;
    end
  else
    argok = 0;
  end
  if ~argok,
    disp(['(som_topol_struct) Ignoring invalid argument #' num2str(i)]);
  end
  i = i+1;
end

if ~isempty(sMap),
  [munits dim2] = size(sMap.codebook);
  if dim2 ~= dim, error('Map and data must have the same dimension.'); end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% create map
```

```
    % map struct
    if ~isempty(sMap),
      sMap = som_set(sMap,'topol',sTopol);
    else
      if ~prod(sTopol.msize),
        if isnan(munits),
          sTopol = som_topol_struct('data',D,sTopol);
        else
          sTopol = som_topol_struct('data',D,'munits',munits,sTopol);
        end
      end
      sMap = som_map_struct(dim, sTopol);
    end

    if struct_mode,
      sMap = som_set(sMap,'comp_names',comp_names,'comp_norm',comp_norm);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% initialization

    % train struct
    sTrain = som_train_struct('algorithm','randinit');
    sTrain = som_set(sTrain,'data_name',data_name);

    munits = prod(sMap.topol.msize);
    %%%
    %%% Modified by V.Lobo, 04/04/06 (yy/mm/dd)
    %%% Modified by R.Henriques, 09/03/26 (yy/mm/dd)
    %%%
    sMap.codebook = zeros([munits dim]);
                    % set interval of GEO component to correct value
    ma_x = max(D(:,1)); mi_x = min(D(:,1));
    ma_y = max(D(:,2)); mi_y = min(D(:,2));
    intervalx=(ma_x-mi_x)/(sMap.topol.msize(2)-1);
    if intervalx==0
        x=ones(1,sMap.topol.msize(2)).*mi_x;
    elseif intervalx==inf
        x=ones(1,sMap.topol.msize(2)).*mi_x;
    else
        x=mi_x:intervalx:ma_x;
    end
    intervaly=(ma_y-mi_y)/(sMap.topol.msize(1)-1);
    if intervaly==0
        y=ones(1,sMap.topol.msize(1)).*mi_y;
    elseif intervaly==inf
        y=ones(1,sMap.topol.msize(1)).*mi_y;
    else
        y=ma_y:-intervaly:mi_y;
    end

    k=1;
    for i=1:sMap.topol.msize(2)
        for j=1:sMap.topol.msize(1)
            sMap.codebook(k,1:2)=[ x(i) y(j) ];
            k=k+1;
        end;
    end;
    %%

    % Atribuir os outros par^ametros do vizinho mais proximo
    Dist_total = dist( sMap.codebook(:,1:2),D(:,1:2)');
    [lixo, bmp ] = min( Dist_total,[], 2 );
    sMap.codebook(:,3:end)=D(bmp,3:end);

    % training struct
    sTrain = som_set(sTrain,'time',datestr(now,0));
    sMap.trainhist = sTrain;
return;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sMap, sTrain] = som_seqtrain_geo(sMap, D, varargin)
    %SOM_SEQTRAIN_GEO  Use sequential algorithm to train the GEOGRAPHICAL Self-Organizing
Map.
    %
    % [sM,sT] = som_seqtrain(sM, D, [[argID,] value, ...])
    %
    %  GEOGRAPHICAL SOM options:
    %  (ALL geographical variants assume geographical coordinates are cartesian and
stores in
    %   in the first two components of the pattern vector)
    %
    %   'geo_match'    (scalar)    Defines the radius of the geographical BMU that should
be
    %                              searched for the final BMU. A radius of ZERO implies
that
    %                              only geo-coordinates are used, thus originating a
Hypermap.
    %                              A radius of 1 or greater with originate a spatial-
kangas map.
    %   'geo_update'   (string)    Defines which components should be updated:
    %                               'nongeo' - Updata only non-geographic components
(DEFAULT)
    %                               'all' - all components (including GEO) will be
updated
    %
    %
    %
    %  sM     = som_seqtrain(sM,D);
    %  sM     = som_seqtrain(sM,sD,'alpha_type','power','tracking',3);
    %  [M,sT] = som_seqtrain(M,D,'ep','trainlen',10,'inv','hexa');
    %
    %  Input and output arguments ([]'s are optional):
    %   sM      (struct) map struct, the trained and updated map is returned
    %           (matrix) codebook matrix of a self-organizing map
    %                    size munits x dim or  msize(1) x ... x msize(k) x dim
    %                    The trained map codebook is returned.
    %   D       (struct) training data; data struct
    %           (matrix) training data, size dlen x dim
    %   [argID, (string) See below. The values which are unambiguous can
    %    value] (varies) be given without the preceeding argID.
    %
    %   sT      (struct) learning parameters used during the training
    %
    % Here are the valid argument IDs and corresponding values. The values which
    % are unambiguous (marked with '*') can be given without the preceeding argID.
    %   'mask'        (vector) BMU search mask, size dim x 1
    %   'msize'       (vector) map size
    %   'radius'      (vector) neighborhood radiuses, length 1, 2 or trainlen
    %   'radius_ini'  (scalar) initial training radius
    %   'radius_fin'  (scalar) final training radius
    %   'alpha'       (vector) learning rates, length trainlen
    %   'alpha_ini'   (scalar) initial learning rate
    %   'tracking'    (scalar) tracking level, 0-3
    %   'trainlen'    (scalar) training length
    %   'trainlen_type' *(string) is the given trainlen 'samples' or 'epochs'
    %   'train'      *(struct) train struct, parameters for training
    %   'sTrain','som_train '  = 'train'
    %   'alpha_type' *(string) learning rate function, 'inv', 'linear' or 'power'
    %   'sample_order'*(string) order of samples: 'random' or 'ordered'
    %   'neigh'      *(string) neighborhood function, 'gaussian', 'cutgauss',
    %                          'ep' or 'bubble'
    %   'topol'      *(struct) topology struct
    %   'som_topol','sTopo l'  = 'topol'
    %   'lattice'    *(string) map lattice, 'hexa' or 'rect'
    %   'shape'      *(string) map shape, 'sheet', 'cyl' or 'toroid'
    %
```

```
    % For more help, try 'type som_seqtrain' or check out online documentation.
    % See also  SOM_MAKE, SOM_BATCHTRAIN, SOM_TRAIN_STRUCT.

M,D); % sM.trainhist{end}==sTrain
    %
    % SEE ALSO
    %
    %  som_make         Initialize and train a SOM using default parameters.
    %  som_batchtrain   Train SOM with batch algorithm.
    %  som_train_struct Determine default training parameters.

    % Copyright (c) 1997-2000 by the SOM toolbox programming team.
    % http://www.cis.hut.fi/projects/somtoolbox/

    % Version 1.0beta juuso 220997
    % Version 2.0beta juuso 101199

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Check arguments

    error(nargchk(2, Inf, nargin));  % check the number of input arguments

    % map
    struct_mode = isstruct(sMap);
    if struct_mode,
      sTopol = sMap.topol;
    else
      orig_size = size(sMap);
      if ndims(sMap) > 2,
        si = size(sMap); dim = si(end); msize = si(1:end-1);
        M = reshape(sMap,[prod(msize) dim]);
      else
        msize = [orig_size(1) 1];
        dim = orig_size(2);
      end
      sMap   = som_map_struct(dim,'msize',msize);
      sTopol = sMap.topol;
    end
    [munits dim] = size(sMap.codebook);

    % data
    if isstruct(D),
      data_name = D.name;
      D = D.data;
    else
      data_name = inputname(2);
    end
    D = D(find(sum(isnan(D),2) < dim),:); % remove empty vectors from the data
    [dlen ddim] = size(D);                % check input dimension
    if dim ~= ddim, error('Map and data input space dimensions disagree.'); end

    % varargin
    sTrain = som_set('som_train','algorithm','seq','neigh', ...
             sMap.neigh,'mask',sMap.mask,'data_name',data_name);
    radius     = [];
    alpha      = [];
    tracking   = 1;
    sample_order_type = 'random';
    tlen_type  = 'epochs';
    geo_match=0;

    i=1;
    while i<=length(varargin),
      argok = 1;
      if ischar(varargin{i}),
        switch varargin{i},
         % argument IDs
         case 'msize', i=i+1; sTopol.msize = varargin{i};
         case 'lattice', i=i+1; sTopol.lattice = varargin{i};
```

```
        case 'shape', i=i+1; sTopol.shape = varargin{i};
        case 'mask', i=i+1; sTrain.mask = varargin{i};
        case 'neigh', i=i+1; sTrain.neigh = varargin{i};
        case 'trainlen', i=i+1; sTrain.trainlen = varargin{i};
        case 'trainlen_type', i=i+1; tlen_type = varargin{i};
        case 'tracking', i=i+1; tracking = varargin{i};
        case 'sample_order', i=i+1; sample_order_type = varargin{i};
        case 'radius_ini', i=i+1; sTrain.radius_ini = varargin{i};
        case 'radius_fin', i=i+1; sTrain.radius_fin = varargin{i};
        case 'radius',
         i=i+1;
         l = length(varargin{i});
         if l==1,
           sTrain.radius_ini = varargin{i};
         else
           sTrain.radius_ini = varargin{i}(1);
           sTrain.radius_fin = varargin{i}(end);
           if l>2, radius = varargin{i}; tlen_type = 'samples'; end
         end
        case 'alpha_type', i=i+1; sTrain.alpha_type = varargin{i};
        case 'alpha_ini', i=i+1; sTrain.alpha_ini = varargin{i};
        case 'alpha',
         i=i+1;
         sTrain.alpha_ini = varargin{i}(1);
         if length(varargin{i})>1,
           alpha = varargin{i}; tlen_type = 'samples';
           sTrain.alpha_type = 'user defined';
         end
        case {'sTrain','train','som_train'}, i=i+1; sTrain = varargin{i};
        case {'topol','sTopol','som_topol'},
         i=i+1;
         sTopol = varargin{i};
         if prod(sTopol.msize) ~= munits,
           error('Given map grid size does not match the codebook size.');
         end
    %---------------------------------------
    % MODIFIED by V.Lobo, 04/04/06
        case 'geo_match',                   % Defines the RADIUS of the GEOgraphic BMU
that must be
           i=i+1;                           % searched to find the true BMU
           geo_match_radius=varargin{i};

       case 'geo_update',                   % Defines if we should update 'all'
components, or only
           i=i+1;                           % 'nongeo' than the GEO. default is 'nongeo'
           geo_update=varargin{i};

    % END modified
    %---------------------------------------
         % unambiguous values
        case {'inv','linear','power'}, sTrain.alpha_type = varargin{i};
        case {'hexa','rect'}, sTopol.lattice = varargin{i};
        case {'sheet','cyl','toroid'}, sTopol.shape = varargin{i};
        case {'gaussian','cutgauss','ep','bubble'}, sTrain.neigh = varargin{i};
        case {'epochs','samples'}, tlen_type = varargin{i};
        case {'random', 'ordered'}, sample_order_type = varargin{i};
        otherwise argok=0;
       end
     elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
       switch varargin{i}(1).type,
        case 'som_topol',
         sTopol = varargin{i};
         if prod(sTopol.msize) ~= munits,
           error('Given map grid size does not match the codebook size.');
         end
        case 'som_train', sTrain = varargin{i};
        otherwise argok=0;
       end
     else
```

```
    argok = 0;
  end
  if ~argok,
    disp(['(som_seqtrain) Ignoring invalid argument #' num2str(i+2)]);
  end
  i = i+1;
end

% training length
if ~isempty(radius) | ~isempty(alpha),
  lr = length(radius);
  la = length(alpha);
  if lr>2 | la>1,
    tlen_type = 'samples';
    if     lr> 2 & la<=1, sTrain.trainlen = lr;
    elseif lr<=2 & la> 1, sTrain.trainlen = la;
    elseif lr==la,        sTrain.trainlen = la;
    else
      error('Mismatch between radius and learning rate vector lengths.')
    end
  end
end
if strcmp(tlen_type,'samples'), sTrain.trainlen = sTrain.trainlen/dlen; end

% check topology
if struct_mode,
  if ~strcmp(sTopol.lattice,sMap.topol.lattice) | ...
    ~strcmp(sTopol.shape,sMap.topol.shape) | ...
    any(sTopol.msize ~= sMap.topol.msize),
    warning('Changing the original map topology.');
  end
end
sMap.topol = sTopol;

% complement the training struct
sTrain = som_train_struct(sTrain,sMap,'dlen',dlen);
if isempty(sTrain.mask), sTrain.mask = ones(dim,1); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% initialize

M        = sMap.codebook;
mask     = sTrain.mask;
trainlen = sTrain.trainlen*dlen;

% neighborhood radius
if length(radius)>2,
  radius_type = 'user defined';
else
  radius = [sTrain.radius_ini sTrain.radius_fin];
  rini = radius(1);
  rstep = (radius(end)-radius(1))/(trainlen-1);
  radius_type = 'linear';
end

% learning rate
if length(alpha)>1,
  sTrain.alpha_type ='user defined';
  if length(alpha) ~= trainlen,
    error('Trainlen and length of neighborhood radius vector do not match.')
  end
  if any(isnan(alpha)),
    error('NaN is an illegal learning rate.')
  end
else
  if isempty(alpha), alpha = sTrain.alpha_ini; end
  if strcmp(sTrain.alpha_type,'inv'),
    % alpha(t) = a / (t+b), where a and b are chosen suitably
    % below, they are chosen so that alpha_fin = alpha_ini/100
```

```
    b = (trainlen - 1) / (100 - 1);
    a = b * alpha;
  end
end

% initialize random number generator
rand('state',sum(100*clock));

% distance between map units in the output space
%  Since in the case of gaussian and ep neighborhood functions, the
%  equations utilize squares of the unit distances and in bubble case
%  it doesn't matter which is used, the unitdistances and neighborhood
%  radiuses are squared.
Ud = som_unit_dists(sTopol).^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Action

update_step = 100;
mu_x_1 = ones(munits,1);
samples = ones(update_step,1);
r = samples;
alfa = samples;

qe = 0;
start = clock;
if tracking >  0, % initialize tracking
  track_table = zeros(update_step,1);
  qe = zeros(floor(trainlen/update_step),1);
end

for t = 1:trainlen,

  % Every update_step, new values for sample indeces, neighborhood
  % radius and learning rate are calculated. This could be done
  % every step, but this way it is more efficient. Or this could
  % be done all at once outside the loop, but it would require much
  % more memory.
  ind = rem(t,update_step); if ind==0, ind = update_step; end
  if ind==1,
    steps = [t:min(trainlen,t+update_step-1)];
    % sample order
    switch sample_order_type,
     case 'ordered', samples = rem(steps,dlen)+1;
     case 'random',  samples = ceil(dlen*rand(update_step,1)+eps);
    end

    % neighborhood radius
    switch radius_type,
     case 'linear',       r = rini+(steps-1)*rstep;
     case 'user defined', r = radius(steps);
    end
    r=r.^2;        % squared radius (see notes about Ud above)
    r(r==0) = eps; % zero radius might cause div-by-zero error

    % learning rate
    switch sTrain.alpha_type,
     case 'linear',       alfa = (1-steps/trainlen)*alpha;
     case 'inv',          alfa = a ./ (b + steps-1);
     case 'power',        alfa = alpha * (0.005/alpha).^((steps-1)/trainlen);
     case 'user defined', alfa = alpha(steps);
    end
  end

  % find BMU
  x = D(samples(ind),:);                % pick one sample vector
  known = ~isnan(x);                    % its known components

%-----------------------------------------------------------
```

```
    % MODIFIED by V.Lobo in 03/11/29 to implement kangas-type map
    % V.2 by V.Lobo, 04/04/06 (yy/mm/dd)

      geocoord=known-known;                  % build a 0-logical vector with the same
size as x
      geocoord(1)=1;                         % "activate" the geographical coordinates
      geocoord(2)=1;
      geocoord=logical(geocoord);
                                                     % Find the BMU considering ONLY the
GEO coordinates
      Dx = M(:,geocoord) - x(mu_x_1,geocoord);       % each map unit minus the vector
      [qerr bmu_initial] = min((Dx.^2)*mask([1 2])); % minimum distance(^2) and the
BMU_initial

      if geo_match_radius==0,
          bmu=bmu_initial;
          Dx = M(:,known) - x(mu_x_1,known);
      else
          non_searchable_units=find(Ud(:,bmu_initial)>geo_match_radius);

                                                 % NOW, find the distances considering
all components
          Dx = M(:,known) - x(mu_x_1,known);     % each map unit minus the vector
          tmp=(Dx.^2)*mask(known);
          tmp(non_searchable_units)=inf;         % Exclude units that are not in the
KANGAS Neighborhood
                                                 % NOW find the true BMU
          [qerr bmu] = min(tmp); % minimum distance(^2) and the BMU
      end;
    % END modification for GEO-SOM
    %-------------------------------------------------------------
      % tracking
      if tracking>0,
        track_table(ind) = sqrt(qerr);
        if ind==update_step,
          n = ceil(t/update_step);
          qe(n) = mean(track_table);
          trackplot(M,D,tracking,start,n,qe);
        end
      end

      % neighborhood & learning rate
      % notice that the elements Ud and radius have been squared!
      % (see notes about Ud above)
      switch sTrain.neigh,
      case 'bubble',   h = (Ud(:,bmu)<=r(ind));
      case 'gaussian', h = exp(-Ud(:,bmu)/(2*r(ind)));
      case 'cutgauss', h = exp(-Ud(:,bmu)/(2*r(ind))) .* (Ud(:,bmu)<=r(ind));
      case 'ep',       h = (1-Ud(:,bmu)/r(ind)) .* (Ud(:,bmu)<=r(ind));
      end
      h = h*alfa(ind);

      % update M
      %if not(issame(geo_update,'all'))
       if strcmpi(geo_update,'all')==0
          known(1:2)=0;
      end;
      M(:,known) = M(:,known) - h(:,ones(sum(known),1)).*Dx(:,known);

    end; % for t = 1:trainlen

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Build / clean up the return arguments

    if tracking, fprintf(1,'\n'); end

    % update structures
    sTrain = som_set(sTrain,'time',datestr(now,0));
    if struct_mode,
```

```
      sMap = som_set(sMap,'codebook',M,'mask',sTrain.mask,'neigh',sTrain.neigh);
      tl = length(sMap.trainhist);
      sMap.trainhist(tl+1) = sTrain;
    else
      sMap = reshape(M,orig_size);
    end

    return;

function [] = trackplot(M,D,tracking,start,n,qe)

  l = length(qe);
  elap_t = etime(clock,start);
  tot_t = elap_t*l/n;
  fprintf(1,'\rTraining: %3.0f/ %3.0f s',elap_t,tot_t)
  switch tracking
   case 1,
   case 2,
    plot(1:n,qe(1:n),(n+1):l,qe((n+1):l))
    title('Quantization errors for latest samples')
    drawnow
   otherwise,
    subplot(2,1,1), plot(1:n,qe(1:n),(n+1):l,qe((n+1):l))
    title('Quantization error for latest samples');
    subplot(2,1,2), plot(M(:,1),M(:,2),'ro',D(:,1),D(:,2),'b.');
    title('First two components of map units (o) and data vectors (+)');
    drawnow
  end
  % end of trackplot
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sU=som_umat_geo(sM, varargin )
% SOM_UMAT_GEO - U-MAT for geographical applications
%
% This function in only a wrapper for the standard som_umat, that
% removes the geographical coordinates
% The u-mat is calculated ignoring the geographical references (first two
% components of the data matrix

% V1.0 by V.Lobo, F.Bacao, 31/10/04

    mascara_reserva=sM.mask(1:2);
    sM.mask(1:2)=0;                 %% retirar coord geograficas
    sU=som_umat(sM, varargin);
    sM.mask(1:2)=mascara_reserva;
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function xticklabel_rotate90(XTick,varargin)
    %XTICKLABEL_ROTATE90 - Rotate numeric Xtick labels by 90 degrees
    %
    % Syntax: xticklabel_rotate90(XTick)
    %
    % Input:  XTick - vector array of XTick positions & values (numeric)
    %
    % Output:  none
    %
    % Example 1:  Set the positions of the XTicks and rotate them
    %    figure;  plot([1960:2004],randn(45,1)); xlim([1960 2004]);
    %    xticklabel_rotate90([1960:2:2004]);
    %    %If you wish, you may set a few text "Property-value" pairs
    %    xticklabel_rotate90([1960:2:2004],'Color','m','Fontweight','bold');
    %
    % Example 2:  %Rotate XTickLabels at their current position
    %    XTick = get(gca,'XTick');
    %    xticklabel_rotate90(XTick);
    %
```

```
    % Other m-files required: none
    % Subfunctions: none
    % MAT-files required: none
    %
    % See also: TEXT,  SET

    % Author: Denis Gilbert, Ph.D., physical oceanography
    % Maurice Lamontagne Institute, Dept. of Fisheries and Oceans Canada
    % email: gilbertd@dfo-mpo.gc.ca  Web: http://www.qc.dfo-mpo.gc.ca/iml/
    % February 1998; Last revision: 24-Mar-2003

    if ~isnumeric(XTick)
       error('XTICKLABEL_ROTATE90 requires a numeric input argument');
    end

    %Make sure XTick is a column vector
    XTick = XTick(:);

    %Set the Xtick locations and set XTicklabel to an empty string
    set(gca,'XTick',XTick,'XTickLabel','')

    % Define the xtickLabels
    xTickLabels = num2str(XTick);

    % Determine the location of the labels based on the position
    % of the xlabel
    hxLabel = get(gca,'XLabel');  % Handle to xlabel
    xLabelString = get(hxLabel,'String');

    if ~isempty(xLabelString)
       warning('You may need to manually reset the XLABEL vertical position')
    end

    set(hxLabel,'Units','data');
    xLabelPosition = get(hxLabel,'Position');
    y = xLabelPosition(2);

    %CODE below was modified following suggestions from Urs Schwarz
    y=repmat(y,size(XTick,1),1);
    % retrieve current axis' fontsize
    fs = get(gca,'fontsize');

    % Place the new xTickLabels by creating TEXT objects
    hText = text(XTick, y, xTickLabels,'fontsize',fs);

    % Rotate the text objects by 90 degrees
    set(hText,'Rotation',90,'HorizontalAlignment','right',varargin{:})
return
```

Appendix 4. Themes used in the Hierarchical SOM tests

Theme	Variable (%)	Description (%)
Lodgings	*AFCRH1_2D*	Classical lodgings with 1 or 2 divisions
	AFCRH3_4D	Classical lodgings with 3 or 4 divisions
	AFCRHPO	Classical lodgings occupied by owner
	AFCRHARR	Classical lodgings rented
	AC	Collective lodgings
	AFV	Vague/empty family lodgings
Buildings	*E1919*	Buildings built before 1919
	E1945	Buildings built between 1919 and 1945
	E1960	Buildings built between 1946 and 1960
	E1970	Buildings built between 1961 and 1970
	E1980	Buildings built between 1971 and 1980
	E1985	Buildings built between 1981 and 1985
	E1990	Buildings built between 1986 and 1990
	E1995	Buildings built between 1991 and 1995
	E2001	Buildings built between 1996 and 2001
	PV2	Buildings with 1 or 2 floors
	PV4	Buildings with 3 or 4 floors
	PV5	Buildings with 5 or more floors
	EBAR	Buildings with resistant elements of concrete
	EARG	Buildings with walls of masonry cemented
	EPAT	Buildings with stone masonry walls
	EORE	Buildings with others elements (wood, metallic)
Families	*FCR1_2*	Classical families with 1 or 2 persons
	FCR3_4	Classical families with 3 or 4 persons
	FCD_0	Classical families without unemployed
	FCD_1	Classical families with 1 unemployed
	FCPME15	Classical families with less than 15 years people
	FCPMA65	Classical families with more than 65 years people
	NF_1FNC	Nucleus with 1 not married son
	NF_2FNC	Nucleus with 2 not married sons
	NF_1NNC	Nucleus with 1 not married grandson
	NF_2NNC	Nucleus with 2 not married grandsons
	NFF6	Nucleus with sons less than 6 years old
	NGN6	Nucleus with grandsons less than 6 years old
Age structure	*HR0_4*	Resident men with age between 0 and 4 years
	MR0_4	Resident women with age between 0 and 4 years

Theme	Variable (%)	Description (%)
	HR5_9	Resident men with age between 5 and 9 years
	MR5_9	Resident women with age between 5 and 9 years
	HR10_13	Resident men with age between 10 and 13 years
	MR10_13	Resident women with age between 10 and 13 years
	HR14_19	Resident men with age between 14 and 19 years
	MR14_19	Resident women with age between 14 and 19 years
	HR20_24	Resident men with age between 20 and 24 years
	MR20_24	Resident women with age between 20 and 24 years
	HR25_64	Resident men with age between 25 and 64 years
	MR25_64	Resident women with age between 25 and 64 years
	HR65	Resident men with 65 years old or more
	MR65	Resident women with 65 years old or more
Education level	*IRQA_001*	Resident individuals which do not know to read nor write
	IRQA_110	Resident individuals with the 1st degree of the basic education
	IRQA_120	Resident individuals with the 2nd degree of the basic education
	IRQA_130	Resident individuals with the 3rd degree of the basic education
	IRQA_200	Resident individuals with a *k*-12 education level
	IRQA_300	Resident individuals with a professional degree
	IRQA_400	Resident individuals with an undergraduate university degree
	IRNI_413	Resident individuals attending the 1st degree of the basic education
	IRNI_423	Resident individuals attending the 2nd degree of the basic education
	IRNI_433	Resident individuals attending the 3rd degree of the basic education
	IRNI_513	Resident individuals that frequent the *k*-12 education level
	IRNI_713	Resident individuals that frequent the university
	IRP_ECR	Resident individuals that study in the local area
Employment	*IR_SP*	Resident individuals employed at the first sector
	IR_SS	Resident individuals employed at the second sector
	IR_ST	Resident individuals employed at the third sector
	IR_PR	Resident individuals pensioners or retired
	IR_EP	Employed resident individuals
	IRD1E	Resident unemployed looking for the first job
	IRDNE	Resident unemployed looking for a job
	IRSAC	Resident without economic activity
	IRP_TCR	Resident individuals that work in the local area

Appendix 5. UAV path definition SOM based tool code

Functions summary


```
% batch file to test ship sensor in ocean
clear;
close all;

%% inputs
%1 knot = 1 naut mile/hour =  0.8684 km/h = 0.241 m/s
% ships
Ships.FishShips=25;
Ships.FishShipsMnVel=.5;
Ships.FishShipsMxVel=1;

Ships.TransportShips=25;
Ships.TransportShipsMnVel=1;
Ships.TransportShipsMxVel=4;

Ships.SportsShips=5;
Ships.SportsShipsMnVel=10;
Ships.SportsShipsMxVel=50;

%Uav
numSensor=9;
sensorRange=500; % sensor range radius
sensorVelMn=1;
sensorVelMx=20;
initialPoint=[5000 5000];

%field
xmax=10000;
ymax=10000;

%iterations
t=1;
numIter=10000;

%som parameters
niterations_1 = 1;
niterations_2 = 20;
radius_ini_1  = .5;
radius_ini_2  = 1;
alpha_ini_1   = 0.2;
```

```
alpha_ini_2   = 0.2;

%% 10 iteration test
OutputFinal=zeros(10,1);
for i=1:10 % 10 testes

    Output=shipS_2d_UAV(Ships,...
            'numSensor',numSensor,...
            'sensorRange',sensorRange,...
            'sensorVelMn',sensorVelMn,...
            'sensorVelMx',sensorVelMx,...
            'xmax',xmax,...
            'ymax',ymax,...
            'numIter',numIter,...
            'niterations_1',niterations_1,...
            'radius_ini_1',radius_ini_1,...
            'alpha_ini_1',radius_ini_1,...
            'plot',1);

    OutputFinal.(strcat('t',num2str(i)))=Output;
end

%% changing the number of sensors
Sensores=[4;9;16;25;36;49;64;81];
OutputFinal=zeros(10,1);
for i=1:10 % 10 tests
    numSensor=Sensores(i);

    Output=shipS_2d_UAV(Ships,...
            'numSensor',numSensor,...
            'sensorRange',sensorRange,...
            'sensorVelMn',sensorVelMn,...
            'sensorVelMx',sensorVelMx,...
            'xmax',xmax,...
            'ymax',ymax,...
            'numIter',numIter,...
            'niterations_1',niterations_1,...
            'radius_ini_1',radius_ini_1,...
            'alpha_ini_1',radius_ini_1,...
            'plot',0);

    OutputFinal.(strcat('t',num2str(i)))=Output;
end

%% changing the number of ships
numFishShips=[5;10;20;30;40;50];
numTransportShips=[5;10;20;30;40;50];

OutputVariacaoSensores=zeros(10,1);
for i=1:6
    Ships.FishShips=numFishShips(i);
    Ships.TransportShips=numTransportShips(i);

    Output=shipS_2d_UAV(Ships,...
            'numSensor',numSensor,...
            'sensorRange',sensorRange,...
            'sensorVelMn',sensorVelMn,...
            'sensorVelMx',sensorVelMx,...
            'xmax',xmax,...
            'ymax',ymax,...
            'numIter',numIter,...
            'niterations_1',niterations_1,...
            'radius_ini_1',radius_ini_1,...
            'alpha_ini_1',radius_ini_1,...
            'plot',0);

    OutputVariacaoSensores.(strcat('t',num2str(i)))=Output;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Output=ShipS_2d_UAV(Ships,varargin)
%SHIPS_2D_UAV ...
% Syntax: output=ShipS_2d_UAV(Ships,varargin)
% Ships -
% varargin -
% output -
%
% Example
% Line 1 of example
%
% Subfunctions:
%   See also:
%
% Author: Roberto Henriques, Victor Lobo, Fernando Bação
% Created: May 28-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008

i=1;
while i<=length(varargin),
    if ischar(varargin{i}),
        switch varargin{i},
            % argument IDs
            case 'numSensor', i=i+1; numSensor = varargin{i};
            case 'sensorRange', i=i+1; sensorRange = varargin{i};
            case 'sensorVelMn', i=i+1; sensorVelMn = varargin{i};
            case 'sensorVelMx', i=i+1; sensorVelMx = varargin{i};
            case 'xmax', i=i+1; xmax = varargin{i};
            case 'ymax', i=i+1; ymax = varargin{i};
            case 'numIter', i=i+1; numIter = varargin{i};
            case 'niterations_1', i=i+1; niterations_1 = varargin{i};
            case 'radius_ini_1', i=i+1; radius_ini_1 = varargin{i};
            case 'alpha_ini_1', i=i+1; alpha_ini_1 = varargin{i};
            case 'plot', i=i+1; plot = varargin{i};
        end
    end
    i = i+1;
end


%% Initialize Ships
Ships=Ships_CreateInitial(Ships,xmax, ymax);

%% SOM
% Initialize Som Map based on geoSOM technique
%create Som Data based on the initial point
%sD=som_data_struct([initialPoint;initialPoint-xmax/4]);
sD=som_data_struct([0 0; 0 xmax; 0 ymax; xmax ymax]);

% create Som Map
sMap = som_randinit_geo(sD,'msize',[sqrt(numSensor) sqrt(numSensor)],'rect','sheet');
%sMap = som_randinit(sD,'msize',[1 numSensor],'rect','sheet');
predict=[];

%% Create Ghost points (study area matrix)
x=xmax/(sMap.topol.msize(1)-1);
ghostX=repmat((0:x:xmax)',sMap.topol.msize(1),1);
y=ymax/(sMap.topol.msize(2)-1);
ghost=repmat((0:y:ymax),sMap.topol.msize(2),1);
ghostY=ghost(:,1);
for i=2:sMap.topol.msize(2)
    ghostY=[ghostY;ghost(:,i)];
end
ghost=[ghostX,ghostY];
clear ghostX;
clear ghostY;
```

```
%% Calculate Fix Sensor Position
x=xmax/(sMap.topol.msize(1)+1);
FixSensorPosX=repmat((x:x:xmax-1)',sMap.topol.msize(1),1);
y=ymax/(sMap.topol.msize(2)+1);
FixSensorPos=repmat((y:y:ymax-1),sMap.topol.msize(2),1);
FixSensorPosY=FixSensorPos(:,1);
for i=2:sMap.topol.msize(2)
    FixSensorPosY=[FixSensorPosY;FixSensorPos(:,i)];
end
FixSensorPos=[FixSensorPosX,FixSensorPosY];
clear FixSensorPosX;
clear FixSensorPosY;


%% Calculate Zigzag Sensor Position and future movement
[Movement,zSensorsPos]= ShipS_GetZigzagSensorsMovement(xmax, ymax,...
    numSensor,sensorRange,sensorVelMx,numIter);

Err=zeros(numIter,3);
NumDifShips=zeros(numIter,3);
AverageTime=zeros(numIter,3);
ShipsIDs.UAV=[];
ShipsIDs.UAVtime=[0 0];
ShipsIDs.FixedSens=[];
ShipsIDs.FixedSenstime=[0 0];
ShipsIDs.ZigZagSens=[];
ShipsIDs.ZigZagSenstime=[0 0];

%% for each iteration in time
for i=1:numIter
    % advance in time
    Ships=ShipS_AdvanceTime(Ships,xmax, ymax);

    % calculate ideal sensors position based on SOM
    [sMap ShipsSensed, predict]=ShipS_sensorManagment(xmax,ymax,Ships, sMap,...
    sD,numSensor,sensorRange,sensorVelMx,ghost,niterations_1,...
    radius_ini_1,alpha_ini_1);

    % calculate detected ships using fix sensors
    ShipsFixSensed=ShipS_DetectionWithFixSensors(FixSensorPos,Ships,sensorRange);

    % calculate zigzag sensors and its detected ships
    [zSensorsPos, zShipsSensed]= ShipS_ZigzagSensors(i,...
        zSensorsPos,Movement,sensorVelMx,sensorRange, Ships,numSensor);

    %% Calculate ERRORS
    % percentage of seen ships

Err(i,1)=length(ShipsSensed)/(Ships.FishShips+Ships.TransportShips+Ships.SportsShips);

Err(i,2)=length(ShipsFixSensed)/(Ships.FishShips+Ships.TransportShips+Ships.SportsShips);

Err(i,3)=length(zShipsSensed)/(Ships.FishShips+Ships.TransportShips+Ships.SportsShips);

    % number of diferent ships seen
    ShipsIDs.UAV=unique([ShipsIDs.UAV; ShipsSensed(:,5)]);
    ShipsIDs.FixedSens=unique([ShipsIDs.FixedSens; ShipsFixSensed(:,5)]);
    ShipsIDs.ZigZagSens=unique([ShipsIDs.ZigZagSens; zShipsSensed(:,5)]);

    NumTotalShipsSinceBeg=(max(Ships.FishShipsPos(:,5))-10000 +
max(Ships.SportsShipsPos(:,5))-30000 + max(Ships.TransportShipsPos(:,5))-20000);

    NumDifShips(i,1)=length(ShipsIDs.UAV)/NumTotalShipsSinceBeg;
    NumDifShips(i,2)=length(ShipsIDs.FixedSens)/NumTotalShipsSinceBeg;
    NumDifShips(i,3)=length(ShipsIDs.ZigZagSens)/NumTotalShipsSinceBeg;

    % Average time on diferent ships seen
    [AverageTime,ShipsIDs]= ShipS_GetAverageTime(ShipsIDs,i,AverageTime);
```

```
    if plot==1
        % plot
        shipS_Plot(Ships,xmax,ymax,sMap,ShipsSensed,sensorRange,predict,ghost...
            ,Err,FixSensorPos,zSensorsPos,i,NumDifShips,AverageTime);
        pause(.0001)
    elseif mod(i,100)==0
        disp(i);
    end

end

%% Output
Output.Err=Err;
Output.ShipsIDs=ShipsIDs;
Output.NumDifShips=NumDifShips;
Output.AverageTime=AverageTime;

% plot
shipS_Plot(Ships,xmax,ymax,sMap,ShipsSensed,sensorRange,predict,ghost...
    ,Err,FixSensorPos,zSensorsPos,i,NumDifShips,AverageTime);
pause(.1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function M=ShipS_AdvanceTime(Ships,xMax,yMax)
%SHIPS_ADVANCETIME Update ship positions
%
% Syntax: M=ShipS_AdvanceTime(M,t,vmin,vmax,xmax)
% Ships - struct array with the number of each ship type, min and max
%         velocity, and the position matrix
%
% Author: Roberto Henriques, Victor Lobo, Fernando Bação
% Created: May 08-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008


%% FishShips
% FishShips will circle around and sometimes moveaway
% ships with zero in the 3rd column are circling around
% ships with a P number in the 3rd column means that are moving away for P
% iterations

% those who are Moving Around
MvAround=find(Ships.FishShipsPos(:,3)==0);
vel=rand(length(MvAround),1).*(Ships.FishShipsMxVel-
Ships.FishShipsMnVel)+Ships.FishShipsMnVel;

if isempty(MvAround)~=1
    %x = x1 + r * cos (rand(radians)) & y = y1 + r * sin (rand())
    Ships.FishShipsPos(MvAround,1)=Ships.FishShipsPos(MvAround,1)+...
        vel.*cos(rand(length(MvAround),1)*(2*pi));
    Ships.FishShipsPos(MvAround,2)=Ships.FishShipsPos(MvAround,2)+...
        vel.*sin(rand(length(MvAround),1)*(2*pi));
end

% those who are Moving Away
MvAway=find(Ships.FishShipsPos(:,3)>0);
if isempty(MvAway)~=1
    Ships.FishShipsPos(MvAway,1)=Ships.FishShipsPos(MvAway,1)+ ...
        Ships.FishShipsMxVel.*cos(Ships.FishShipsPos(MvAway,4));
    Ships.FishShipsPos(MvAway,2)=Ships.FishShipsPos(MvAway,2)+ ...
        Ships.FishShipsMxVel.*sin(Ships.FishShipsPos(MvAway,4));
    Ships.FishShipsPos(MvAway,3)=Ships.FishShipsPos(MvAway,3)-1;
end


% if some leave the field, we must remove them and add some new
indices=find(Ships.FishShipsPos(:,1)>xMax | ...
```

```
    Ships.FishShipsPos(:,1)< 0 |...
    Ships.FishShipsPos(:,2)>yMax | ...
    Ships.FishShipsPos(:,2)< 0);

if isempty(indices)~=1
    MaxID=max(Ships.FishShipsPos(:,5));
    Ships.FishShipsPos(indices,:)=[rand(length(indices),1).*xMax,...
    rand(length(indices),1).*yMax,zeros(length(indices),1)...
    ,zeros(length(indices),1),(MaxID+1:MaxID+length(indices))'];
end

% randomly select some ships to move away in the next iterations
% we define in the column 3 the number of iterations it will move
% this is done randomly between 5 and 20 iterations
% in the column 4 we randomly select an angle for each ship to follow in
% its move away trip

MvAway=find(rand(Ships.FishShips,1)>.95);
Ships.FishShipsPos(MvAway,3)= round(rand()*(20-5)+5);
Ships.FishShipsPos(MvAway,4)= rand(length(MvAway),1)*(2*pi);


%% TransportShips
% TransportShips will follow a straitgh line
vel=rand(Ships.TransportShips,1).*(Ships.TransportShipsMxVel-...
    Ships.TransportShipsMnVel)+Ships.TransportShipsMnVel;

Ships.TransportShipsPos(:,1)=Ships.TransportShipsPos(:,1)+ ...
    vel.*cos(Ships.TransportShipsPos(:,4));
Ships.TransportShipsPos(:,2)=Ships.TransportShipsPos(:,2)+ ...
    vel.*sin(Ships.TransportShipsPos(:,4));

% if some leave the field, we must remove them and new some new
indices=find(Ships.TransportShipsPos(:,1)>xMax | ...
    Ships.TransportShipsPos(:,1)< 0 |...
    Ships.TransportShipsPos(:,2)>yMax | ...
    Ships.TransportShipsPos(:,2)< 0);

if isempty(indices)~=1
    MaxID=max(Ships.TransportShipsPos(:,5));
    Ships.TransportShipsPos(indices,:)=[rand(length(indices),1).*xMax,...
    rand(length(indices),1).*yMax,zeros(length(indices),1),...
    rand(length(indices),1)*(2*pi),(MaxID+1:MaxID+length(indices))'];
end

M=Ships;
return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function newsMap=MWS_CalculateSensorNewPositions(SenPos, sMap, SenVel,sensorRange,t)

% NAME: MWS_CalculateSensorNewPositions
%
% OBJECTIVE:
% Calculate sensor new positions based on the SOM units and sensors' restrictions
%
% Restrictions are:
% 1. Sensor velocity
% 2. Sensor must not intersect other sensors cover area
%
% INPUT PARAMETERS:
%     SenPos        = current sensor positions
%     sMap          = Som Map (ideal sensor position)
%     SenVel        = sensor velocity
%     sensorRange   = sensor range
%     t             = time delayed

% COMMENTS:
```

```
%
% V.0.1 2007/03/22 By R.Henriques,V.Lobo,F.Bacao

%% version based on the velocity, and covered area by each sensor
% updating the sensors is made from 1->n or n->1
% this criterion is defined based on:
% 1. if the sensor is moving away from other sensors
%   1.1 We choose the cloosest sensor to the center to update first
% 2. if sensor 1 or sensor n does not move away from the other sensors then:
%   2.1. we choose to update first the sensor that moves least towards the center

sensorCentroid=(max(SenPos)+ min(SenPos))/2;
deslocamentoMaximo=SenVel*t;
numSensor=length(sMap.codebook);
distancia=sMap.codebook-SenPos; %distance each sensor should run

distSensor1Centro=sqrt((sMap.codebook(1,1)-sensorCentroid(1,1))^2+(sMap.codebook(1,2)-
sensorCentroid(1,2))^2);
distSensorNCentro=sqrt((sMap.codebook(numSensor,1)-
sensorCentroid(1,1))^2+(sMap.codebook(numSensor,2)-sensorCentroid(1,2))^2);

%check if sensor 1 or n is moving away
if distancia(1,1)<0 | distancia(length(distancia),1)>0,%sensor 1 or sensor n move away
    if distancia(1,1)<0 & distancia(length(distancia),1)>0 %sensor 1 and sensor n move
away
        if distSensor1Centro<distSensorNCentro,
            ordemActualiza=1;
        else
            ordemActualiza=-1;
        end
    elseif distancia(1,1)<0,%sensor 1 moves away
        ordemActualiza=1;
    else %sensor n%sensor 1 and sensor n move away
        ordemActualiza=-1;
    end
else  %sensor 1 and sensor n dont move away
    if distSensor1Centro>distSensorNCentro,
        ordemActualiza=1;
    else
        ordemActualiza=-1;
    end
end

%% calculate the real distance for each sensor
% moving to thre right
indices=find(distancia>0&distancia>deslocamentoMaximo);
distancia(indices)=deslocamentoMaximo;

% moving to thre left
indices=find(distancia<0&distancia<(-deslocamentoMaximo));
distancia(indices)=-deslocamentoMaximo;

%% updating based on the order
newSenPos=[];
if ordemActualiza==1,
    for i = 1:1:length(sMap.codebook)
        %update
        newSenPos=SensorUpdate(i,ordemActualiza,SenPos,distancia,newSenPos,sensorRange);
    end

elseif ordemActualiza==-1,
    for i = length(sMap.codebook):-1:1
        %update
        newSenPos=SensorUpdate(i,ordemActualiza,SenPos,distancia,newSenPos,sensorRange);
    end
else
    disp('erro na actualiaçao dos sensores');
end
```

```
sMap.codebook= [newSenPos zeros(length(newSenPos),1)];
newsMap=sMap;


%% version only based on the velocity
% %deslocamento=V*T
% deslocamentoMaximo=SenVel*t;
%
% distancia=sMap.codebook-SenPos;
% indices=find(distancia>0&distancia>deslocamentoMaximo);
% distancia(indices)=deslocamentoMaximo;
%
% indices=find(distancia<0&distancia<(-deslocamentoMaximo));
% distancia(indices)=-deslocamentoMaximo;
%
% sMap.codebook=sMap.codebook+distancia;
% newsMap=sMap;
return

function
updtSensorPosition=SensorUpdate(i,ordemActualiza,SenPos,distancia,newSenPos,sensorRange)
    % function to actualize the sensor position considering other sensors
    sensorRangeTolerance=1.2;

    if ordemActualiza==1,
        if i==1,
            newSenPos(i,1)=SenPos(i,1)+ distancia(i,1);
        else
            if SenPos(i,1)+ distancia(i,1)>newSenPos(i-1,1)+ sensorRange...
                    *sensorRangeTolerance, % this sensor is out of the range of other
sensors
                newSenPos(i,1)=SenPos(i,1)+ distancia(i,1);
            else % this sensor in in the range of other sensors, must move it away
                newSenPos(i,1)=newSenPos(i-1,1)+ sensorRange*sensorRangeTolerance;
            end
        end
    end
    if ordemActualiza==-1,
        if i==length(distancia),
            newSenPos(i,1)=SenPos(i,1)+ distancia(i,1);
        else
            if SenPos(i,1)+ distancia(i,1)<newSenPos(i+1,1)- sensorRange...
                    *sensorRangeTolerance, % this sensor is out of the range of other
sensors
                newSenPos(i,1)=SenPos(i,1)+ distancia(i,1);
            else % this sensor in in the range of other sensors, must move it away
                newSenPos(i,1)=newSenPos(i+1,1)- sensorRange*sensorRangeTolerance;
            end
        end
    end
    updtSensorPosition=newSenPos;

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function S=MWS_CreateInitial(Ships, xMax, yMax)
%SHIPS_CREATEINITIAL Create a struct with 3 matrix with the data
% necessary to simulate different type of ships positions in the ocean
%
% Syntax: S=MWS_CreateInitial(Ships)
% Ships - struct array with:
%               FishShips: Number of fish ships
%          FishShipsMnVel: fish ships min vel
%          FishShipsMxVel: fish ships max vel
%          TransportShips: Number of Transport Ships
%     TransportShipsMnVel: Transport ships min vel
%     TransportShipsMxVel: Transport ships max vel
%             SportsShips: Number of Sports Ships
```

```
%        SportsShipsMnVel: Sports ships min vel
%        SportsShipsMxVel: Sports ships max vel
%            FishShipsPos: [10x4 double] x,y,i,a,id
%          SportsShipsPos: [5x4 double]  x,y,i,a,id
%       TransportShipsPos: [5x4 double]  x,y,i,a,id

%       where:
%                       x: coordinate x
%                       y: coordinate y
%                       i: number of iterations moving with an angle a
%                       a: angle used to move
%                       id: ship id. fish start with 10000
%                                    Transport start with 20000
%                                    Sports start with 30000
%
%       NOTE: in the transportation ships the i is always 0 and the a is
%       defined in the ships creation, while in the fishing ships the a
%       and i are randomly created, randomly in time
%
% Author: Roberto Henriques, Victor Lobo, Fernando Bacao
% Created: May 08-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008

%rand()*(b-a)+a

%initialize ships
Ships.FishShipsPos=[rand(Ships.FishShips,1).*xMax,...
    rand(Ships.FishShips,1).*yMax,zeros(Ships.FishShips,1),...
    zeros(Ships.FishShips,1),(10000:10000+Ships.FishShips-1)'];
Ships.TransportShipsPos=[rand(Ships.TransportShips,1).*xMax...
    ,rand(Ships.TransportShips,1).*yMax,...
    zeros(Ships.TransportShips,1),rand(Ships.TransportShips,1)*(2*pi)...
    ,(20000:20000+Ships.TransportShips-1)'];
Ships.SportsShipsPos =[rand(Ships.SportsShips,1).*xMax, ...
    rand(Ships.SportsShips,1).*yMax,zeros(Ships.SportsShips,1),...
    zeros(Ships.SportsShips,1),(30000:30000+Ships.SportsShips-1)'];
S=Ships;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ShipsSensed]= ShipS_DetectionWithFixSensors(SensorPos,Ships,sensorRange)
%SHIPS_DETECTIONWITHFIXSENSORS ...
% Syntax: [ShipsSensed]= ShipS_DetectionWithFixSensors(SensorPos,Ships)
% SensorPos -
% Ships -
% ShipsSensed -
%
% Example
% Line 1 of example
%
% Subfunctions:
%   See also:
%
% Author: Roberto Henriques, Victor Lobo
% Created: May 23-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008

indices=[];
ShipsPosition=[Ships.FishShipsPos; ...
               Ships.TransportShipsPos; ...
               Ships.SportsShipsPos];
NumberOfShips=length(ShipsPosition);

% calculate the distance between each sensor and each ship
indicesG=[];
for i=1:length(SensorPos)
    dist=(repmat(SensorPos(i,:),NumberOfShips,1)-ShipsPosition(:,1:2)).^2;
```

```
    dist=sqrt(dist(:,1)+dist(:,2));
    indices=find(dist<sensorRange);
    indicesG=[indicesG;indices];
end
%indices=unique(indicesG);
ShipsSensed=ShipsPosition(unique(indicesG),:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [AverageTime,ShipsIDs]= ShipS_GetAverageTime(...
    ShipsIDs,i,AverageTime);
%SHIPS_GETAVERAGETIME ...
% Syntax: [AverageTime,ShipsIDs]= ShipS_GetAverageTime(...
%    ShipsIDs,i,AverageTime);
% ShipsIDs -
% i -
% AverageTime -
% AverageTime -
% ShipsIDs -
%
%
% Author: Roberto Henriques, Victor Lobo
% Created: May 26-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008

%% UAV
    %get new ships
    [c, ia, ib]= setxor(ShipsIDs.UAVtime(:,1),ShipsIDs.UAV);
    ShipsIDs.UAVtime=[ShipsIDs.UAVtime;[ShipsIDs.UAV(ib) ones(length(ib),1)]];
    %increment time on repetead ships
    [c, ia, ib]= intersect(ShipsIDs.UAVtime(:,1),ShipsIDs.UAV);
    ShipsIDs.UAVtime(ia,2)= ShipsIDs.UAVtime(ia,2)+1;

%% Fixed
    %get new ships
    [c, ia, ib]= setxor(ShipsIDs.FixedSenstime(:,1),ShipsIDs.FixedSens);
    ShipsIDs.FixedSenstime=[ShipsIDs.FixedSenstime;[ShipsIDs.FixedSens(ib)
ones(length(ib),1)]];
    %increment time on repetead ships
    [c, ia, ib]= intersect(ShipsIDs.FixedSenstime(:,1),ShipsIDs.FixedSens);
    ShipsIDs.FixedSenstime(ia,2)= ShipsIDs.FixedSenstime(ia,2)+1;

%% Zigzag
    %get new ships
    [c, ia, ib]= setxor(ShipsIDs.ZigZagSenstime(:,1),ShipsIDs.ZigZagSens);
    ShipsIDs.ZigZagSenstime=[ShipsIDs.ZigZagSenstime;[ShipsIDs.ZigZagSens(ib)
ones(length(ib),1)]];
    %increment time on repetead ships
    [c, ia, ib]= intersect(ShipsIDs.ZigZagSenstime(:,1),ShipsIDs.ZigZagSens);
    ShipsIDs.ZigZagSenstime(ia,2)= ShipsIDs.ZigZagSenstime(ia,2)+1;

AverageTime(i,1)=mean(ShipsIDs.UAVtime(2:end,2));
AverageTime(i,2)=mean(ShipsIDs.FixedSenstime(2:end,2));
AverageTime(i,3)=mean(ShipsIDs.ZigZagSenstime(2:end,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Movement,zSensorsPos]= ShipS_GetZigzagSensorsMovement(xmax, ymax,...
    numSensor,sensorRange,sensorVelMx,NumberOfTotalIterations);
%SHIPS_GETZIGZAGSENSORSMOVEMENT ...
% Syntax: [Movement]= ShipS_GetZigzagSensorsMovement(xmax, ymax,...
%    numSensor,sensorVelMx,NumberOfTotalIterations);
% xmax -
% ymax -
% numSensor -
% sensorVelMx -
% NumberOfTotalIterations -
% Movement -
```

```
%
% Example
% Line 1 of example
%
% Subfunctions:
%   See also:
%
% Author: Roberto Henriques, Victor Lobo
% Created: May 23-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008


xDistance = xmax/sqrt(numSensor);
yDistance = ymax/sqrt(numSensor);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% numIterForward--------->
%         numIterCurve   |
% <---------------------- numIterForward
% |         numIterCurve
% numIterForward--------->
%         numIterCurve   |
%
% numCurves2End= 2


numIterForward=ceil((xDistance-2*sensorRange)/sensorVelMx);
numCurves2End=ceil((yDistance-2*sensorRange)/(sensorRange*2));

numCurves2End=ceil((yDistance-2*sensorRange)/(sensorRange*2));
if mod(numCurves2End,2)==0
    numCurves2End=numCurves2End-1;
end

distPerCurve=ceil((yDistance-2*sensorRange)/numCurves2End);
numIterCurve=ceil(distPerCurve/sensorVelMx);

% Moving forward
Movement=[ones(numIterForward,1), zeros(numIterForward,1)];
% Making the 1st curve
Movement=[Movement;...
         [zeros(numIterCurve,1), ones(numIterCurve,1)*(-1)]];
% Moving backwards
Movement=[Movement;...
         [ones(numIterForward,1)*(-1), zeros(numIterForward,1)]];
% Making the 2nd curve
Movement=[Movement;...
         [zeros(numIterCurve,1), ones(numIterCurve,1)*(-1)]];
% Replicate until reach the y bottom
Movement = repmat(Movement,round(numCurves2End/2),1);

% number of curves is odd -> remove the last curve and go to begining
%remove last curve
Movement(length(Movement)-numIterCurve+1:end,:)=[];

%return to begining
Movement=[Movement;...
    [zeros(numIterCurve*numCurves2End,1), ...
    ones(numIterCurve*numCurves2End,1)]];
rep=ceil(NumberOfTotalIterations/length(Movement));
Movement=repmat(Movement,rep,1);


%% zigzag initial position
zSensorsPos=zeros(numSensor,2);

zSensorsPosX=repmat((sensorRange:xDistance:xmax)',sqrt(numSensor),1);
```

```
zSensorsPosYtemp=repmat((ymax-sensorRange:-yDistance:sensorRange),sqrt(numSensor),1);
zSensorsPosYtemp=fliplr(zSensorsPosYtemp);

zSensorsPosY=zSensorsPosYtemp(:,1);
for i=2:sqrt(numSensor)
    zSensorsPosY=[zSensorsPosY;zSensorsPosYtemp(:,i)];
end
zSensorsPos=[zSensorsPosX,zSensorsPosY];

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ShipS_Plot(Ships,xMax,yMax,sMap,ShipsSensed,...
    sensorRange,predict,ghost,Err,FixSensorPos,zSensorsPos,...
    iter, NumDifShips,AverageTime)
%SHIPS_ADVANCETIME Update ship positions
%
% Syntax: M=ShipS_Pot(Ships,xMax,yMax)
% Ships - struct array with the number of each ship type, min and max
%         velocity, and the position matrix
% xMax  - maximum field x coordinate
% yMax  - maximum field y coordinate
%
% Author: Roberto Henriques, Victor Lobo
% Created: May 08-2008
% Version 1.0
% Copyright (c) by the Ga2 programming team. 2008

%% SHIPS
%subplot(2,1,1)
subplot(3,2,[1 3 5])

% All ships
ShipsPosition=[Ships.FishShipsPos(:,1:2); ...
               Ships.TransportShipsPos(:,1:2); ...
               Ships.SportsShipsPos(:,1:2)];

plot(Ships.FishShipsPos(:,1),Ships.FishShipsPos(:,2), 'ro');
hold on
%plot(Ships.SportsShipsPos(:,1),Ships.SportsShipsPos(:,2), 'ro');
plot(Ships.TransportShipsPos(:,1),Ships.TransportShipsPos(:,2), 'bo');
axis equal
axis ([0 xMax 0 yMax])

% plot sensed ships
plot(ShipsSensed(:,1),ShipsSensed(:,2),...
    'o','MarkerFaceColor',[1 0 0]);

% plot predict movement
if isempty(predict)~=1
    plot(predict(:,1),predict(:,2),'g.');
end

% plot ghost
if isempty(ghost)~=1
    plot(ghost(:,1),ghost(:,2),'m.');
end

%% UAVs
%plot(sMap.codebook(:,1),sMap.codebook(:,2),'b>','MarkerFaceColor',[0 0 1]);
% network
som_grid(sMap,'Coord',sMap.codebook,'Marker','>',...
    'markercolor',[0 0 1],'LineColor',[0 0 1]);

% plot sensor range
N=10; % number of points
t=(0:N)*2*pi/N;
for i=1:prod(sMap.topol.msize)
```

```
    x=sensorRange*cos(t)'+ sMap.codebook(i,1);
    y=sensorRange*sin(t)'+sMap.codebook(i,2);
    plot(x,y,'b-.');
end

%% Fixed sensors
plot(FixSensorPos(:,1),FixSensorPos(:,2),'g>');
% plot fix sensor range
N=10; % number of points
t=(0:N)*2*pi/N;
for i=1:length(FixSensorPos)
    x=sensorRange*cos(t)'+ FixSensorPos(i,1);
    y=sensorRange*sin(t)'+ FixSensorPos(i,2);
    plot(x,y,'g-.');
end

%% zigzag sensors
plot(zSensorsPos(:,1),zSensorsPos(:,2),'c>');
% plot zigzag sensor range
N=10; % number of points
t=(0:N)*2*pi/N;
for i=1:prod(sMap.topol.msize)
    x=sensorRange*cos(t)'+ zSensorsPos(i,1);
    y=sensorRange*sin(t)'+ zSensorsPos(i,2);
    plot(x,y,'c-.');
end

%% plot Delaunay
%TRI = delaunay(sMap.codebook(:,1),sMap.codebook(:,2),{'Qz'});
%triplot(TRI,sMap.codebook(:,1),sMap.codebook(:,2),'b-.')

% aqui altero a vizinhança do sMap com base no Delaunay
% ou seja, altero as ligacoes do neuronios com base na triangulacao de
% Delaunay
%sMap.codebook
hold off

%% plot ERR
subplot(3,2,2)
plot (Err(1:iter,1),'b-');hold on;
plot (Err(1:iter,2),'g-');
plot (Err(1:iter,3),'c-');hold off;
title('Percentage of sensed Ships')

%% plot NumDifShips
subplot(3,2,4)
plot (NumDifShips(1:iter,1),'b-');hold on;
plot (NumDifShips(1:iter,2),'g-');
plot (NumDifShips(1:iter,3),'c-');hold off;
title('Number of diferent ships sensed/Total ships')

%% plot Averagetime
subplot(3,2,6)
plot (AverageTime(1:iter,1),'b-');hold on;
plot (AverageTime(1:iter,2),'g-');
plot (AverageTime(1:iter,3),'c-');hold off;
title('Average Time on sensed ships')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [sMap ShipsSensed, predict,ShipsIds]= ShipS_SensorManagment(xmax,ymax, Ships,
sMap, sD,...
        numSensor,sensorRange,sensorVel,ghost, niterations_1,...
        radius_ini_1,alpha_ini_1)

% NAME: MWS_SensorManagment
%
% OBJECTIVE:
```

```
% INPUT PARAMETERS:
%     xmax          = Maximum length of Motorway
%     M             = cars matrix
%     sMap          = Som Map
%     xNeurons      = number of units in x
%     yNeurons      = number of units in y
%     niterations_1 = number of epochs in the first train
%     niterations_2 = number of epochs in the second train
%     radius_ini_1  = initial radius in the first train
%     radius_ini_2  = initial radius in the second train
%     alpha_ini_1   = alpha initial value in the first train
%     alpha_ini_2   = alpha initial value in the second train

% COMMENTS:
%
%
% V.0.1 2007/03/18 By R.Henriques,V.Lobo,F.Bacao

%% Sensors

%sMap.codebook gives the sensors' position
%check which ships are 'seen' by the sensors
indices=[];
ShipsPosition=[Ships.FishShipsPos; ...
               Ships.TransportShipsPos; ...
               Ships.SportsShipsPos];
NumberOfShips=length(ShipsPosition);

% calculate the distance between each sensor and each ship
indicesG=[];
for i=1:numSensor
    dist=(repmat(sMap.codebook(i,:),NumberOfShips,1)-ShipsPosition(:,1:2)).^2;
    dist=sqrt(dist(:,1)+dist(:,2));
    indices=find(dist<sensorRange);
    indicesG=[indicesG;indices];
end
indices=unique(indicesG);
ShipsSensed=ShipsPosition(indices,:);

%% add some predictable route points from the seen ships
% find ships detected that have an angle defined(col4)
predictInd=find(ShipsPosition(indices,4)>0);
predictPoints=[];
nIterations=10;
for i=indices(predictInd)'
    movement=50;
    for j=1:nIterations
        predictPoints=[predictPoints;...
                    ShipsPosition(i,1)+ (movement*j).*cos(ShipsPosition(i,4)),...
                    ShipsPosition(i,2)+ (movement*j).*sin(ShipsPosition(i,4))];
    end
end
% remove predictPoints out of bounds
if isempty(predictPoints)~=1
    removeInd=find(predictPoints(:,1)<0|predictPoints(:,2)<0|...
        predictPoints(:,1)>xmax|predictPoints(:,2)>ymax);
    predictPoints(removeInd,:)=[];
end
predict=predictPoints;

%% Data used in the train
Data=[];
weights=[];
% ships detected (weight=3)
Data=ShipsPosition(indices,1:2);
weights=ones(length(indices),1).*1;
% predict points (weight=2)
Data=[Data; predictPoints];
weights=[weights; ones(length(predictPoints),1).*3];
```

```
% ghost points (weight=1)
Data=[Data; ghost];
weights=[weights; ones(length(ghost),1).*1];

%% create Som Data based on the seen ships
sD=som_data_struct(Data);

%% train SOM
%disp('lst train');
sMap = ShipS_som_batchtrain(sMap,sD,sensorVel,sensorRange,...
        'radius_ini',radius_ini_1,...
        'radius_fin',0,...
        'alpha_ini',alpha_ini_1,...
        'trainlen', niterations_1, ...
        'weights', weights, ...
        'epochs');
Units=sMap.codebook;

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```