GeoDec: A Framework to Effectively Visualize and Query Geospatial Data for Decision-Making

Cyrus Shahabi, Farnoush Banaei Kashani, Ali Khoshgozaran, Luciano Nocera and Songhua Xing
Information Laboratory (InfoLab)
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
http://infolab.usc.edu
{shahabi,banaeika,jafkhosh,nocera,sxing}@usc.edu

***Abstract*— In this paper, we discuss *GeoDec*, our end-to-end system that enables geospatial decision-making by virtualizing the real-world geolocations. With GeoDec, first the geolocation of interest is rapidly and realistically simulated and all relevant geospatial data are accurately fused and embedded in the virtualized model. Subsequently, users can interactively formulate abstract decision-making queries in terms of a wide range of fundamental spatiotemporal queries supported by GeoDec and evaluate the queries in order to verify decisions in the virtual world prior to executing the decisions in real world. GeoDec blends a variety of techniques developed in the fields of databases, artificial intelligence, computer graphics and computer vision into an integrated three-tier architecture. We elaborate on various components of this architecture, which includes an extensive, multimodal and dynamic data tier, an efficient, expressive and extensible query-interface tier and an immersive, flexible and effective presentation tier.**

I. Introduction

Geospatial decision-making is a critical requirement and enabling component of numerous geospatial applications such as urban planning, emergency response, military intelligence, simulator training and serious gaming. With the abundance of the available geospatial data today (e.g., satellite and aerial imagery, maps, vector data, point data), arguably the most effective approach for geospatial decision-making at/about a geolocation is by virtualization of the real-world geolocation. Therefore, an ideal application for a decision maker to immerse in and navigate the geolocation and intuitively formulate and evaluate various abstract decision-making questions should contain the following components: 1) realistic modeling and simulation of a geolocation of interest; 2) seamless embedding and fusion of a host of geotagged and timestamped real datasets associated with the geolocation; 3) efficient execution and presentation of a basic (but expressive) set of spatiotemporal queries on top of the information-rich virtual geolocation. For example, through this three-step virtualization process (i.e., simulation/modeling, data embedding/fusion and querying and query presentation), one can simulate a city such as New York City, embed the road network data, building point data, elevation data and microclimate data in the virtual city and use a combination of range queries, nearest neighbor queries, visibility queries, etc., to evaluate decision-making queries such as "*if the Freedom Tower is built at Ground Zero, how will it affect the wind-vector field of the South Manhattan?*". Using such an approach, decision-makers can effectively investigate and verify their decisions in an equivalent virtual geolocation prior to executing the decisions in real world.

We assume and argue that such a virtualization-based decision-making technology must satisfy the "R-A-I-S-E" requirements in order to be effective: *Realistic* simulation, *Accurate* information fusion, *Interactive* query and access, *Scalable* infrastructure and *Efficient* time-to-build. Considering R-A-I-S-E as our design objectives, we have designed and developed an end-to-end virtualization-based geospatial decision-making system, dubbed *GeoDec* (*Geospatial Decision-making*). GeoDec blends a variety of techniques developed independently in the fields of databases, artificial intelligence, computer graphics and computer vision into a three-tier architecture to facilitate the process of decision-making. As we elaborate in Section IV, GeoDec enables users to interact with a virtualized geolocation through a set of spatiotemporal queries to access various layers of GIS data seamlessly integrated into GeoDec's database server.

Current geospatial visualization systems fall in three main categories: i) Earth visualization (EV) platforms, as first envisioned by Al Gore [9], [10], like Microsoft Virtual Earth™or Google Earth™, display georealistic 3D worlds created from aerial imagery, ii) Game-based systems, such as Half-Life 2™ [11], have excelled at rendering simulated urban environments that seem and act realistically, SimCity [17] an entertaining city planning simulation and Spore™ [18], engage users in interactive worlds where "fictional data" analysis is used to support the game-play and iii) Geographic Information Systems (GIS), such as the ESRI family of products [7], offer query and analysis of static worlds. There are also numerous research projects that use geospatial visualization; for example, UrbanSim [3] focuses on socio-economic modeling for city planning, the GeoVISTA Center [8] develops geo-visualization tools for decision making, and UCI RESCUE [14], [20] targets crisis-response management. Yet no one system has studied the fundamental ways of asking questions about the real world in space and time, and visualize the responses, in an effective and comprehensive way. In particular, the GUI of EV systems are effective for simplistic browsing and keyword-based searches,

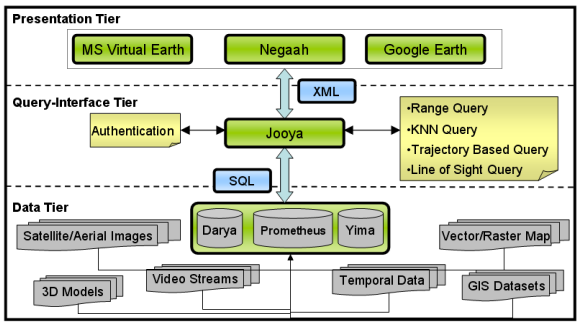

Fig. 1. The GeoDec architecture

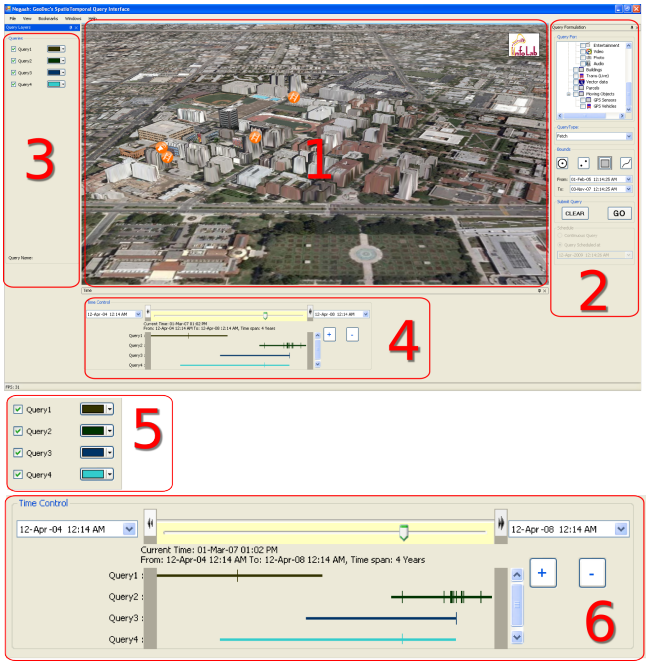

Fig. 2. The user interface of GeoDec detailing its important elements: 1) rendering area, 2) query creation panel, 3) query results panel and 4) temporal navigation. 5) and 6) show magnified renderings of 3) and 4) respectively. A magnified version of 2) is presented in Figure 3

mostly for static information, but lack any means for in-depth spatio-temporal querying and analysis. The GIS systems deploy mostly "pre-defined" (i.e., with no interactive geospatial query creation capabilities), sophisticated but domain-specific queries and visualizations that generally rely on 2D traditional interfaces that are more familiar to professional users. Hence, each and every application requires its own training time to get familiarized with all the customized gadgets of the new GUI. Games EV provide natural interactive interfaces; however these interfaces are specifically crafted to support the game play and are therefore not generalizable to all sorts of queries in time and space. Finally, with the exception of games, the collaborative abilities of the existing systems is extremely limited due to the lack of adequate infrastructure and appropriate interfaces, but most importantly and fundamentally, the lack of an adequate geospatial query creation prevents this dialogue between users from occurring.

The remainder of this paper is organized as follows. Section II describes the three-tier architecture and the components of GeoDec in detail. In Sections III and IV, we elaborate on the techniques we used to implement the visualization and querying process. Finally, Section V concludes the paper.

II. GeoDec Architecture

GeoDec adopts a three-tier architecture depicted in Figure 1. This design makes the interface independent of the inherent data model and facilitates scalability by allowing several visualization components to specify queries and receive the results back in a uniform language hiding the source of information. The three-tier architecture has been extensively implemented in many database driven applications [5], [6]. Compared to the simpler two-tier client/server architecture (where the server usually refers to the DBMS software responsible for handling data storage on disk pages and the client handles the user interface), the three-tier architecture adds a middle layer sometimes called the application layer between the client and server.

In this section, we study each layer and show how the interaction between these layers makes GeoDec a scalable tool for geospatial applications.

A. Negaah-the Querying and Visualization Interface

One of the main design goals of GeoDec is the provision of an immersive environment that enables users to interact with GeoDec and perform a wide range of spatiotemporal queries intuitively, in order to facilitate the process of decision-making. The key advantage of having a three-tier architecture is to allow several interfaces interact with our spatiotemporal database server (Section II-C) thus expanding user's experience working with a wide range of interfaces customized for users' specific data and query needs. We have integrated several well-known geospatial interfaces such as Google Earth, Google Maps and Microsoft Virtual Earth into GeoDec and have also developed Negaah, our proprietary interface that provides a wide range of spatiotemporal queries not supported by typical web-based geospatial and mapping applications.

Integrating multiple interfaces into the GeoDec's top layer allows us to overlay users' query results on top of satellite or raster maps provided by Google or Microsoft servers and provide users with a well-known and easy to use environment to view their query results. Furthermore, users enjoy other features of such applications while interacting with query results generated by the GeoDec's engine. Although these interfaces are powered by highly scalable servers and a huge variety of geospatial data sources, they do not offer enough flexibility to implement the wide range of spatial and temporal queries desirable for a geospatial decision-making system like GeoDec. For instance, Google Maps or Microsoft Virtual Earth do not allow users to specify a bounding box for their spatial queries or use a time slider to move back and forth in time while viewing the results of a temporal query. In order to fill this gap we developed our own interface Negaah using Microsoft .NET C# framework, which offers significantly more customized queries and more intuitive interaction to users.

Negaah is a visualization interface for GeoDec that allows users to navigate and interactively query the 3D environment in real-time. The main elements of the GUI are as follows (numbered similarly in Figure 2: 1) 3D rendering of the geolocation with superimposed interactive visualizations of query results; 2) query creation panel with available data types, query types and spatial and temporal bounds; 3) query result layers panel where issued queries are presented and 4) an advanced time-line that enables users to pan and zoom in time and define time ranges. Sections IV offers more in-depth details regarding GeoDec's querying capabilities. As Negaah is GeoDec's main visualization interface, we have integrated all querying and visualization functionalities GeoDec can offer in Negaah. A subset of these features are available on other interfaces based on their level of support for the necessary interactions and querying.

As a sophisticated querying and visualization tool, Negaah is capable of modeling 3D buildings, terrains and more complex objects such as trees.

Modeling a virtual space in Negaah consists of the following four phases.

- **Background image construction**: during the first phase, a large satellite image is mapped on the ground as the background. This calibrated image serves as the building block for the second phase of the modeling process.
- **3D object construction and indexing**: in this phase, 3D building components of the virtual space are constructed from the background image using a semi-manual technique [13] and are added to the model.
- **Terrain modeling**: during this phase, the flat model constructed is enhanced with terrain information to create a more realistic model as well as enabling certain queries in GeoDec to use the terrain elevation data.
- **Texture mapping**: during the last modeling phase, high resolution texture images are added to the model in order to create realistic objects in space.

Currently, We have constructed detailed virtual representations of the University of Southern California main campus, an area in downtown Los Angeles and Shanghai JiaoTong University, Minhang Campus, China. The USC park campus, covering an area of 226-acres, was modeled by 256 structures components in about two hours.

B. Jooya-the Middleware Layer

As discussed in Section II-A, one of the key features of GeoDec is allowing several independently developed graphical user interfaces to query our spatio-temporal database layer. In order to create a conceptually simple yet effective way of allowing such interchangeability, we developed *Jooya*, our middleware layer to sit between the client and the database tier of *GeoDec*. The benefits of our three tier architecture are threefold. First, Jooya abstracts the details of indexing, storage and querying of spatio-temporal data for the client tier. Second, Jooya includes a customized query-interface blade for each GUI that exists in the client tier. This enables Jooya to act as an interface for web-based geo-spatial GUIs such as Google Earth as well. Third, Jooya offers a universal way of specifying spatio-temporal queries created in the client side. More specifically, each request received by Jooya conforms to the following generic form:

*<**Data Type, Query Type, Spatial Bounds, Temporal Bounds**>*

The data type refers to any of the data sources supported by GeoDec.The query types currently supported by GeoDec are range query (i.e., return all objects within the specified spatial bound), shortest path, nearest neighbor and visibility. Finally, each request includes the spatio-temporal bounds that are specified by the user (see SectionIV). We implemented Jooya in C# .NET framework to seamlessly integrate our middle tier layer with our proprietary client tier which is also written in C#. Jooya publishes a set of Web Services using WSDL (Web Service Description Language) that can be used by clients to query geo-spatial data. For better interoperability we will make our Jooya web services comply with the Open Geospatial Consortium's OpenGIS standards [1].

More specifically, to interact with Jooya, a client uses the query formulation discussed above as a Remote Procedure Call (RPC) for its data needs. Depending on the user's query type, Jooya sends a query to our database tier through low-level SQL commands via ODBC connection, or contacts the miscellaneous data sources that contain highly dynamic data (such as traffic data) not stored in the database layer. Examples of queries sent to the database tier are queries for infrequently updated or bulky data such as road network data, GPS traces, and 3D building models. On the other hand, live traffic information or live location of moving objects are examples of data sets obtained via a variety of web sources.

It is important to note we have made a key design decision to make the middle tier and the database *aware* of the elements presented to the user on the GUI. Therefore, each such object can itself act as a spatial bound for a query. For instance, the user can select (the centroid of) two buildings to act as the geo-bounds for querying the vector data instead of specifying the latitude and longitude coordinates. Such knowledge of objects at different tiers can fundamentally change the user experience while interacting with a spatio-temporal querying interface like Negaah. We finally note that the query-centric design of the three-tiers enables a smooth and efficient addition of new data sources and queries. This is due to the fact that (say) adding a new data type to GeoDec does not change how clients interact with Jooya and simply gives them a new binding for the "data type" value in their requests.

C. Darya-the Spatiotemporal Database Server

The back-end of GeoDec termed *Darya* is the system's spatiotemporal database engine running Oracle 10g. This module is in charge of managing spatiotemporal data stored in a database, a task that includes data modeling, storage and retrieval (querying). Darya stores a variety of data sources such as road network information (i.e., vector data), 3D building models, terrain data, gazetteer data, satellite and aerial imagery and raster maps. The main advantage of Darya is enabling fast and efficient access to multi-dimensional datasets that change infrequently. One of the key features of Darya is the fact that

the majority of its data is tagged by both time and space thus allowing a wide range of spatial, temporal and spatiotemporal queries to be efficiently supported by GeoDec. For instance, all buildings are tagged by their valid time interval (i.e., date of creation and date of demolition, if any). Therefore, users can select an area of interest as well as a time duration and use Negaah's time slider to move back and forth in time and see the changes in the querying region and within the given time frame.

In the database layer, our main focus is on storage and querying spatial data such as bulky vector data (e.g., road networks). Mainly, Darya embeds a multi-resolution vector data compression technique [12] which effectively compresses the result of query windows, taking into account the client's display resolution.

III. Geospatial Data Fusion

Creating a realistic model of a geographic space allows users to be *immersed* in a realistic environment. However, enabling decision-making requires enriching the model with various layers of geospatial data such as vector data (e.g., road network information, parcel data) and raster imagery. In this section, we briefly study our approaches for integrating these heterogeneous (and sometimes inconsistent) data sources into our models.

A. Road network and map fusion

In order to generate a useful visualization of integrated geospatial data (i.e., vector data, satellite imagery, parcel data and raster maps), a simple superimposition of various geospatial sources is not sufficient to align the sources with each other for the following reasons: **(i)** For many raster maps, the geocoordinates of the maps are unspecified **(ii)** For other geospatial data with specified geocoordinates, the projections and transformations used to produce the maps, orthorectified imagery or vector data are unknown.

The current commercial tools to solve this problem require heavy user interventions. It is simply too slow and tedious to fully exploit the rich sources of information available in geospatial datasets. Towards this end, we have developed a set of techniques for automatically aligning maps and road vector data on orthorectified imagery [4]. In order to allow integration of the aligned data with the 3D model, the maps and road vector data are aligned to one of the satellite images used to construct the 3D model. Our vector-to-imagery conflation technique exploits a combination of the knowledge of the road network with image processing techniques. We first find road intersection points from the road vector dataset. For each intersection point, we then perform image processing in a localized area to find the corresponding intersection point in the satellite image. Finally, we compute the transformation matrix from the two sets of intersection points to align the vector data with the imagery. The running time is dramatically lower than traditional image processing techniques due to the small areas on which image processing is applied. To integrate maps with satellite imagery, we utilize common vector datasets as "glue". We first identify road intersections on imagery by the technique described above, and then we also detect the road intersections on maps [4]. Finally, we apply a geospatial point pattern matching algorithm to find matches between the two point sets. Now that we have a set of matched control point pairs, we partition the map into small triangles, and utilize the local transformation matrix to transform the map piece by piece. By doing so, we find the location (i.e., the geocoordinate) of the map and align the map with satellite imagery. Our proposed approach facilitates the close integration of vector datasets, imagery and maps, thus allowing the creation of intelligent images that combine the visual appeal and accuracy of imagery with the detailed attribution information often contained in diverse maps.

Experiments show that our approach can conflate various maps with imagery, such that in our experiments on TIGER-maps covering part of St. Louis county, MO, 85.2% of the conflated map roads are within 10.8 m from the actual roads compared to 51.7% for the original and georeferenced TIGER-map roads. The assessment of the impact of conflation adjustments on a decision making tool depends on the accuracy of the original data-sets, but more crucially it will depend on how relative discrepancies and the geographical layout will impact query results and their outcomes (impact of the accuracy of query results in the decision that are made). For instance, querying what buildings are withing a given road network distance from a query point, might not require the same spatial accuracy than querying which building are visible from that query point, and the impact of the resulting errors on the decision process may vary depending on context and human factors.

B. Data Integration and Efficient Geospatial Querying

In order to efficiently and accurately integrate a wide variety of information about a geographic location, GeoDec utilizes a geographic data integration system called *Prometheus* [19]. Prometheus organizes the available information sources in a domain hierarchy containing well-known domain concepts, such as, satellite image, map, or vector data. Moreover, Prometheus also models different types of integration operations and their effects. Examples of the integration operations include *Overlay* and *Align*. The Overlay operation may result in two information layers not aligning with each other while the Align operation described in Section III-A improves the alignment between two data layers. When GeoDec receives a request to retrieve the data for a geographic location, Jooya sends a request to the data integration system to obtain different types of information (i.e. point data, vector data, raster maps, and satellite imagery). Prometheus determines the relevant sources for the requested data, retrieves relevant data from the sources, performs necessary alignment or integration operations, and returns the integrated information to Jooya which is then sent back to Negaah to be displayed on the 3D model.

IV. Querying with GeoDec

One of the most important features of GeoDec is its querying capability, which differentiates it from many other

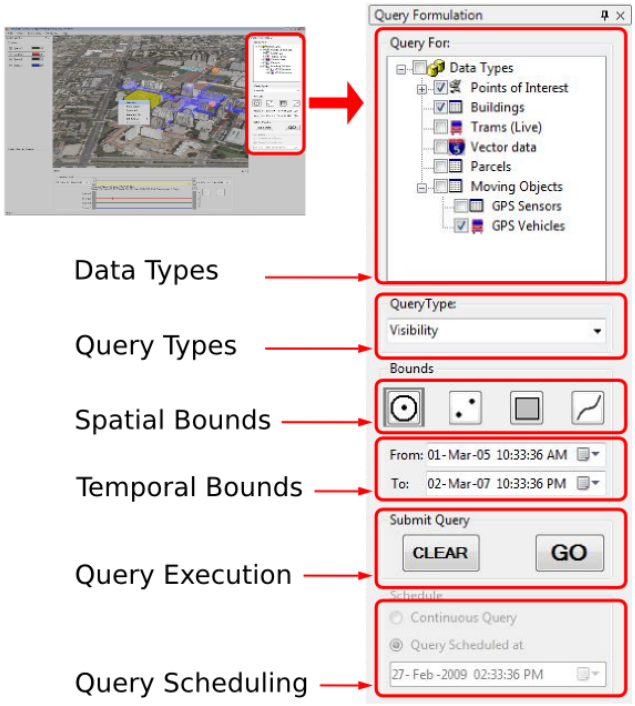

Fig. 3. GeoSpatial query creation building blocks.

geospatial visualization applications. In this section, we first discuss how to create typical spatial-temporal queries followed by mechanism called *Query Encapsulation* when the queries are executed and presented through the three layers. Finally we discuss two cases to see how these queries can help in decision making.

A. Visual Query Creation

To define the query interface we relied heavily on the universal space and time coordinates to derive relevance between our objects. Therefore, our building blocks allow a user to ask a question about (one or more) geospatial bounds and/or temporal intervals in a natural manner. The result is an intelligent and transformative display of the relevant objects to the question. Subsequently, we allow the user to interact further with the query result and formulate more queries. Figure 3 presents a diagram of the building blocks featured in *GeoDec*'s query creation.

To define our query creation interface, we have considered the extensive set of queries supported in our original GeoDec prototype; in the process we have further ensured that our devised set of building blocks was able to support all these queries while being simple and intuitive to most users. Ultimately, the formulation of a query involves determining five building blocks, namely Data Types, Query Type, Spatial Bounds, Temporal Bounds, and optionally Query Scheduling. When creating a query, users select each one of the building blocks in turn. There is no specific order in this process as the user interface maintains consistency. For sake of simplicity we detail how a user would go sequentially over each building block hereafter.

First, the *Data Types* is determined, which can be done (or any combinations of) object(s) or information the user is interested in (e.g., road networks, buildings). Next, *Query Types* will be specified, which can be a Fetch (Range) Query, Nearest Neighbor Query, Shortest Path or Visibility Query. Third, the user will formulate the *Spatial Bound*, as depicted in Figure 3, four modes of spatial bound are supported: 1) *Circular bound*; 2) *Two Points bound*; 3) *Rectangular bound* and 4) *Trajectory bound*. Currently, *Circular and Rectangular bounds* are compatible with Fetch Query and Visibility Query and *two Points bound* is only used to support Shortest Path Query while *Trajectory bound* works with both Nearest Neighbor Query and Fetch Query. After selecting the desired mode, the user can draw the corresponding geometry shape in the rendering area by dragging the mouse as the spatial bound. Fourth, the *Temporal Bound* of the query is set as the *From Time* and *To Time*, which means all the returning objects should have a valid lifetime overlapping this temporal bound. Then, after these steps, in the *Query Execution* Step, the user can click "Go" button to submit the query or "Clear" button to start over. Sometimes, rather than simply clicking the "Go" button, a user need to manipulate the execution time of this query during the "Query Scheduling" step: the query can be either an one time query scheduled at a certain time or an continuous query at a fixed time interval.

After a query has been submitted, the query itself will be recorded into the database and is appended in the client GUI to the historical query results in the *query result panel* (Figure 2). The user can show/hide the results of issued queries by checking/unchecking the checkbox left to the queries. In addition, the user can also customize the color that is blended on the graphical objects for each query result by selecting the desired color from a drop down box accessible on the right of each query. These mechanisms provides fine grained filtering and customization of the query result and is significantly more flexible than traditional query layers: in some regard this is similar to the visual layers that are found in image manipulation software such as Adobe Photoshop. Finally, query results are also appended in the *Temporal Navigation Panel* (Figure 2) and shown with period of validity and events if applicable.

One important way in which we maximize *GeoDec*'s querying capabilities is by supporting the interactive reusability of queries: any object returned as a result from a previous query can in fact be selected either for information retrieval, or as a building block for the next queries. Ideally, every rendering object that is visualized in the system is pickable. In our interface, a user can freely select one or more objects by clicking and retrieve its/their related information (e.g., name, Object Id, valid life time and the spatial coordinates) through a standard right click menu. Some objects may contain addition information: for example, buildings feature a view of their blueprints, video/audio clips contain multimedia data can be played, etc. Furthermore, new queries can be formulated by selecting any existing query result object and using the selection to define the new query spatial bound. Note that the selected objects do not necessarily share the same Data Type. For example, a user can select five buildings and four video

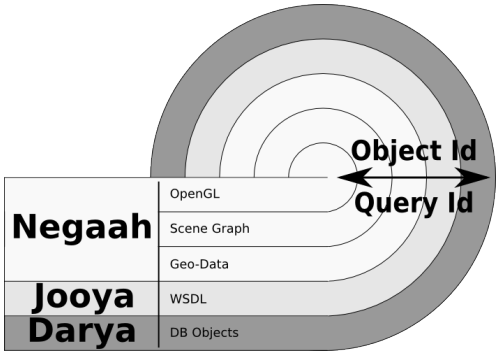

Fig. 4. Query Encapsulation.

icons to build a bound of type trajectory and use that bound in a new query to retrieve photos along the trajectory.

B. Query Encapsulation

Similar to the OSI (Open System Interconnection) [2] model which deploys a seven layer network architecture, the communication among the three layers in *GeoDec* is done through *Query Encapsulation* as depicted in Figure 4. After the raw data is retrieved from *Darya* through low level SQL queries, it is formulated into an XML SOAP response through the Web Services provided by *Jooya* using WSDL. Then, in *Negaah*, the query result is finally converted into a graphical representations and rendered in OpenGL. During this process, There are two intermediate data structures called *Geo-Data* and *Scene Graph* that are used to encapsulate query results and the query visualizations; these data structures are essential in support of query interactivity as they allow to consistently map queried data to visualizations users can interact with. More specifically we link queries and graphical objects in *Negaah* by consistently referencing queries and objects using a unique *Object Id* and *Query Id*.

The *Geo-Data* is the logical representation of a result object. It contains the query result object's spatial coordinates, valid lifetime, object ID and other properties. From a software architecture perspective we rely on the natural hierarchy of geospatial data (e.g., GPS measures combine in GPS traces, groups of buildings form neighbourhoods, groups of neighbourhoods from cities...) to define our *GeoData* results. Consequently, we use a hierarchy of objects derived from simple "primitives" to represent more complex objects. More specially we define *DataPrimitives* for scalar values, text, documents, 3d models, and *GeoPrimitives* for spatially located entities such as points, lines, poly-lines. The temporal dimension is then added to *GeoPrimitives* by composition (e.g., we obtain a *GeoTemporalPoint* by composing *GeoPoint* and *Time* objects). To give a concrete example, a picture taken at a specific location and time is a *GeoData* object that is composed of a *GeoTemporalPoint* and an *Image* "DataPrimitive"); complex geometries such as buildings are similarly built using *GeoPolyLine* objects.

The *Scene Graph* is a graph structure where each node encapsulates the graphical representation of query results in the form of 3D shapes or 2D raster graphics (e.g., icon or text). The use of a dedicated *Scene Graph* also provides the following benefits: i) it abstract the query visualization allowing for customized rendering adapted to the client hardware (i.e. a *bus* can be represented as either a 2D icon or a 3D model and the representation is determined by the user and the hardware rather than beeing pre-defined by the system) and promotes the reuse of graphic resources at the hardware level ii) facilitates rendering and extensibility by abstracting the query representation from the actual rendering implementation (i.e. in OpenGL or Direct3D), iii) provides a natural hierarchical representation of the query results that further facilitates interaction by providing selective access at each level of the hierarchy to whole or part of the query results.

C. Case Studies

To further explain how our *GeoDec* GUI supports generic complex querying tasks, we present here two specific example cases.

Case Study 1 Suppose we have a security situation and want to find information that can help identify a suspect. With our GUI users would first use the query formulation panel to define appropriate bounds (space and time) to look for multimedia data (say 911 reverse type calls and text messages for tips, video surveillance, or participatory data collected by law enforcement personnel) in relation to the event. Submitting this query would generate a layered visualization of these elements and provide subsequent access to the media themselves and related information through the context menu. If a message points to a suspect that say was seen coming out of a particular bus, a subsequent query specified with time and space dimensions set in accordance with those of the message, will allow to find and visualize the relevant bus trajectory in time (historical records of bus trajectories can be readily available for most public buses from on-board GPS sensors). The user can further focus on the important data by turning on or off and customizing the query result layers in panel 3 of Figure 2; for instance by hiding irrelevant objects or coloring objects differently. To the same effect, users can further narrow their view of the data of interest (say to visualize the multimedia data close to the bus trajectory) either by combining the previous query results using the panel 3, or by issuing a new query in panel 1, where the bounds are specified by direct selection of the query result objects (typically the bus path). Finally using the time-line interface 4 one can zoom in time and find surveillance video streams along the bus path to check for visual evidence that could identify the suspect.

Case Study 2 As another example, let's consider an evacuation scenario where the building in which the situation is occurring has been identified. Using our proposed GUI interface, one can specify the region and time of interest by direct interaction with the rendering area of panel 1 and the temporal navigation of panel 4 of Figure 2. Then one could obtain information on public places, ..., traffic, buses going through the areas with their drivers' contact information all displayed in a geospatial context as separated layers. To actually evacuate the area specific complex queries can be specified

in the query panel 1 resulting in maps of visibility from the building of interest; this information visually superimposed with the results of a query showing the road network and traffic information (by layer manipulations in panel 3) is of prime importance to first responders in order to secure the area by minimizing direct exposure to potential harm from the building of interest.

These simple examples show how *GeoDec*'s set of fundamental query building blocks can be effective at supporting sophisticated spatio-temporal analysis and can be actually able to uncover critical information.

V. Conclusion and Future Work

In this paper, we discussed GeoDec, a geospatial decision-making tool that enables rapid, realistic, accurate and scalable virtualization of geolocations, while allowing for interactive and extensive querying of the data embedded in the virtual geolocation for real-time decision-making. Particularly, we explained the benefits of adopting a three-tier architecture for such a geospatial decision making system and described the design and implementation of various components of the GeoDec architecture. We also elaborated on the challenges with the implementation of the virtualization process in details and presented our solution for each case. In sum, through GeoDec, we study the fundamental ways to ask questions about the real world in space and time, and visualize the responses.

We intend to pursue this work in three directions. First, in order to be able to support real applications with extremely dynamic data (e.g., microclimate, pollution), we plan to extend the spatiotemporal data indexing capabilities of GeoDec at the data tier to support efficient updates. Second, we plan to extend GeoDec to formulate and process more complex geospatial queries, such as spatial skyline queries [16] and kNN queries on land surface [15]. We also intend to extend GeoDec to support more advanced interfaces at the presentation tier, such as the interfaces running on mobile phones.

VI. Acknowledgements

This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0534761 and CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Open geospatial consortium. http://www.opengeospatial.org.

[2] X.200 : Information technology - open systems interconnection - basic reference model: The basic model.

[3] A. Borning and P. Waddell. UrbanSim: interaction and participation in integrated urban land use, transportation, and environmental modeling. In *Proceedings of the 2006 international conference on Digital government research*, pages 133–134, San Diego, California, 2006. ACM.

[4] C.-C. Chen, C. A. Knoblock, and C. Shahabi. Automatically and accurately conflating raster maps with orthoimagery. *GeoInformatica*, 12(3):377–410, 2008.

[5] W. W. Eckerson. Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications. *Open Information Systems 10*, 3(20), 1995.

[6] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems, 2nd Edition*. Benjamin/Cummings, 1994.

[7] Environmental Systems Research Institute, Inc. (ESRI GIS and Mapping Software. http://www.esri.com.

[8] GeoVISTA Web Site. http://www.geovista.psu.edu.

[9] A. Gore. *Earth in the balance: Ecology and the human spirit.* Boston: Houghton Mifflin., 1992.

[10] A. Gore. The Digital Earth: Understanding our planet in the 21st Century. http://www.isde5.org/al_gore_speech.htm, January 1998.

[11] Half Life 2 Web Site. http://orange.half-life2.com/hl2.html.

[12] A. Khoshgozaran, A. Khodaei, M. Sharifzadeh, and C. Shahabi. A hybrid aggregation and compression technique for road network databases. *Knowledge and Information Systems*, 2008.

[13] S.-C. Lee, A. Huertas, and R. Nevatia. Modeling 3-d complex buildings with user assistance, WACV 2000, Palm Springs, pp. 170-177.

[14] S. Mehrotra, C. T. Butts, D. Kalashnikov, N. Venkatasubramanian, R. R. Rao, G. Chockalingam, R. Eguchi, B. J. Adams, C. Huyck, S. Santini, and R. Schettini. Project rescue: challenges in responding to the unexpected. In *Internet Imaging V*, volume 5304, pages 179–192, San Jose, CA, USA, Dec. 2003. SPIE.

[15] C. Shahabi, L.-A. Tang, and S. Xing. Indexing land surface for efficient knn query. *PVLDB*, 1(1):1020–1031, 2008.

[16] M. Sharifzadeh and C. Shahabi. The spatial skyline queries. In *Proceedings of the 32nd international conference on Very large data bases*, pages 751–762, Seoul, Korea, 2006. VLDB Endowment.

[17] SimCity Web Site. http://en.wikipedia.org/wiki/SimCity, and http://simcitysocieties.ea.com.

[18] Spore Web Site. http://www.spore.com.

[19] S. Thakkar, J. L. Ambite, and C. A. Knoblock. Composing, optimizing, and executing plans for bioinformatics web services. *The VLDB Journal, Special Issue on Data Management, Analysis, and Mining for the Life Sciences*, 14(3):330–353, 2005.

[20] UCIRescue Web Site. http://www.itr-rescue.org, http://www.responsphere.org, and http://sami.ics.uci.edu.