Learning Geometric Hashing Functions for Model-Based Object Recognition

George Bebis[†], Michael Georgiopoulos[†], and Niels da Vitoria Lobo[‡]

[†]Department of Electrical & Computer Engineering, University of Central Florida, Orlando, FL 32816
[‡]Department of Computer Science, University of Central Florida, Orlando, FL 32816

Abstract

A major problem associated with geometric hashing and methods which have emerged from it is the non-uniform distribution of invariants over the hash space. This problem can affect the performance of the method significantly. Finding a "good" geometric hash function which redistributes the invariants uniformly over the hash space is not easy. In this paper, a new approach is proposed for alleviating the above problem. It is based on the use of an "elastic hash table" which is implemented as a Self-Organizing Feature Map Neural Network (SOFM-NN). In contrast to existing approaches which try to redistribute the invariants over the hash bins, we proceed oppositely, spreading the hash bins over the invariants. During training, the SOFM-NN resembles an elastic net which deformes over the hash space. The objective of the deformation process is to spread more hash bins in hash space areas which are heavily occupied and less hash bins in lower density areas. The advantage of the proposed approach is that it is a process that adapts to the invariants through learning. Hence, it makes absolutely no assumptions about the statistical characteristics of the invariants and the geometric hash function is actually computed through learning. Furthermore, the well known "topology preserving" property of the SOFM-NN guarantees that the computed geometric hash function should be well behaved. Finally, the proposed approach is inherently parallelizable.

1. Introduction

Indexing-based approaches for model-based object recognition have received a lot of attention in the last few years. The basic idea is to speedup searching by sacrificing space. Initially, features which remain unchanged under certain geometric transformations (invariants) are extracted from each model. Then, a model database is built by establishing proper associations between features and models. During recognition, scene features are used to retrieve the most feasible associations stored in the model database. Efficient indexing schemes are used for both organizing and searching the model database.

Geometric hashing [1] is a well known indexing-based object recognition technique. Like similar indexing-based object recognition approaches, geometric hashing suffers from a major problem: the non-uniform distribution of invariants over the hash space. This can result in an inefficient storage of the data entries over the hash table which can slow down the recognition time of the algorithm significantly. Also, taking into consideration that geometric hashing is highly amenable to parallel implementation, a uniform distribution of data is highly desirable for efficiently solving the load-balancing problem. The key solution to the problem is the selection of a "good" geometric hash function which should be able to uniformly redistribute the data over the hash table.

Costa, Haralick, and Shapiro [2] have tried a number of different hash functions in order to choose the one which performs best. However, the selection of a good hash function seems to be problem dependent. Rehashing has been suggested by Rigoutsos and Hummel [3],[4]. The basic idea behind this approach is to find a transformation (rehashing) which maps the distribution of invariants to a uniform distribution, using concepts from probability theory. This approach suffers from two drawbacks. First, it is based on the assumption that the pdf of model point features is known *apriori*. However, such an assumption can be very unreliable, especially in cases where the number of model objects is not large. Second, the derivation of the rehashing transformation involves complex computations and, in certain cases, is even intractable [5].

In this paper, a new approach for dealing with the problem of the non-uniformity of invariants is presented. This approach, based on the use of an **elastic hash table**, makes no assumption about the pdf of the model point features and is notable for its simplicity. More-

0-8186-7042-8/95 $4.00

over, it does not require the estimation of the pdf of invariants and it is applicable even in cases where a closed-form analytical expression for the pdf of invariants cannot be derived. The "elastic hash table" is implemented as a Self Organizing Feature Map Neural Network (SOFM-NN) [6], trained with a variation of the Kohonen algorithm that we have developed, motivated by the conscientious competitive learning algorithm [7].

The organization of the paper is as follows: Section 2 reviews the geometric hashing technique and the fundamental ideas of rehashing. Section 3 outlines our approach. Problems associated with the Kohonen learning algorithm and the Kohonen algorithm with conscience are presented in Section 4. Experimental results and comparisons are given in Section 5. Finally, Section 6 presents our conclusions.

2. Geometric hashing and rehashing

Geometric hashing consists of two phases: a preprocessing phase which is performed off-line and a recognition phase. During the preprocessing phase, the models are represented in an affine invariant way. For each ordered non-collinear triplet of model points (basis-triplet), a coordinate system is defined and the coordinates of all the other model points are recomputed in terms of this coordinate system. The recomputed coordinates (invariants) are used then as an index into a hash table where the entry *(basis-triplet, model)* is recorded.

During the recognition phase, an arbitrary ordered triplet of non-collinear points is chosen from the scene. Then, the coordinates of the remaining scene points are recomputed in terms of the coordinate frame defined by this triplet. The recomputed coordinates of each point are used as an index into the hash table and for each entry *(basis-triplet, model)* recorded there, a vote is cast. If a certain entry *(basis-triplet, model)* scores a large number of votes, then it is concluded that this triplet corresponds to the one chosen from the scene. The unique transformation which maps the model triplet to the scene triplet is assumed to be the transformation between the model and the scene.

Geometric hashing employs a very simple hash function which consists of a linear scaling of the invariants followed by some kind of quantization in order to yield an integer index which fits the dimensions of the hash table. In this case, hashing merely implies a quantization of the space of invariants. In order for the distribution of entries over the hash table to be uniform, hashing should be able to divide the space of invariants into equiprobable regions. In general, the invariants are heavily non-uniformly distributed over the hash space. To illustrate this, we have considered a small database of objects (see Figure 2a). Applying the preprocessing step yields the distribution of invariants shown in Figure 2c. The non-uniformity of invariants has as a result the non-uniform storage of the data entries over the hash table.

Rehashing has been proposed by Rigoutsos and Hummel [3],[4] for dealing with the non-uniformity of invariants. Specifically, rehashing is a transformation which maps the distribution of invariants into a uniform distribution. This is accomplished by first computing the expected pdf of the distribution of invariants, assuming that the model point features are generated by a known random process (Gaussian with zero mean or uniform over a convex domain). Three classes of model object transformations were considered: rigid, similarity, and affine. However, analytical rehashing transformations were not derived in every case because of the intractability of the computations involved [5]. In the cases where analytical formulas were derived, the pdf of invariants was shown to resemble a Gaussian with zero mean.

To show the effectiveness of the rehashing transformation, we have considered the invariants shown in Figure 2c. As can be observed, the distribution of invariants does resemble a Gaussian with zero mean. Applying the rehashing transformation (assuming similarity transformations), yields the redistributed invariants shown in Figure 2e. However, the distribution of invariants can be very different from a Gaussian as is illustrated in section 5. Rehashing does not perform satisfactorily in these cases.

3. Alleviating the non-uniformity of invariants using the SOFM-NN

In this paper, a new approach for dealing with the non-uniformity of invariants is proposed. The key idea is to visualize the hash table as an elastic, two-dimensional, lattice with the hash bins distributed over the nodes of the lattice. Initially, the lattice can have any structure. Our goal is to find a procedure capable of distributing the hash bins according to the density of invariants, assigning more hash bins in the vicinity of a large number of invariants and less hash bins in the vicinity of a small number of invariants. Thus, the purpose is to distribute the hash bins over the invariants instead of distributing the invariants over the hash bins.

The key idea is to represent the elastic hash table as a SOFM-NN. The SOFM-NN consists of an input layer and a single output layer of nodes which usually

form a two-dimensional array. Each input is fully connected to each output node and a weight is associated with every connection. It is the output layer of nodes that will play the role of the elastic hash table with each output node corresponding to a different hash bin. At the beginning of the training process, the weights associated with each node are chosen randomly over the space of invariants. This action can be seen as spreading the hash bins randomly over the invariants. During training, the network is exposed to sample invariants and the distances between them and the weights associated with the nodes of the SOFM-NN are computed. The node whose weights are closest to the input invariants is declared as the winner (*competitive learning*). Then, the weights associated with the winner node and nodes within a neigborhood of it change in a way such that they divide the space of invariants into a number of groups. Eventually, each group will access a specific node (hash bin).

The behavior of the SOFM-NN during training resembles an elastic net which deforms through learning trying to closely resemble the input data. Actually, the role of the training procedure is to distribute the weight vectors in the input space in such a way that they approximate the pdf of the inputs. Kohonen [6] has argued that the density of the weight vectors which have been assigned to an input region, approximates the density of the inputs occupying this region. In other words, the final structure of the weights, should reflect the statistical characteristics of the invariants (non-parametric model).

After training has been completed, the SOFM-NN has learned to implement a non-linear mapping from the input space to the "node" space (hash bins). The node to be accessed when random invariants are presented to the SOFM-NN, will be determined again through competition. A very useful property of the Kohonen algorithm is that weight vectors tend to be ordered according to their mutual similarity (*topology preserving property*). This property is a direct consequence of the use of topological neighborhoods during training. The importance of this property is that after learning has been completed, nearby nodes respond to similar inputs. Thus, the mapping from the input space to the node space will be well behaved.

The above properties are very important in the implementation of our approach. The first property implies that the space of invariants should be partioned into a number of equiprobable regions. As a result, each hash bin should be assigned an even number of entries. The second property implies that partial voting (a heuristic whose use has been shown to be almost imperative [3],[8]), can be efficiently implemented since the mapping from the space of invariants to the space of "nodes" will be proximity preserving.

We have considered some examples in order to demonstrate our approach. First, we have randomly chosen a number of two dimensional points from a uniform rectangular distribution (Figure 1a), and a 10 x 10 SOFM-NN. Figure 1b shows the initial structure of the SOFM while Figure 1c shows an intermediate step. The final structure of SOFM is shown in Figure 1d. If the distribution of invariants was non-uniform, then the region where the weight vectors are more crowded would also be the region where the invariants fall more densely. This is demonstrated in Figure 1f where a 10 x 10 SOFM-NN has been trained again using a set of two-dimensional points drawn from a Gaussian distribution (Figure 1e).

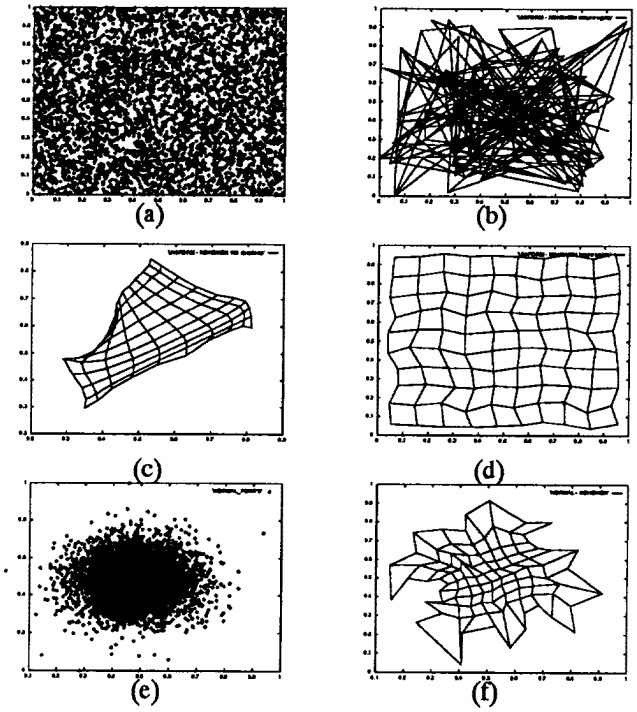

Figure 1. (a) 2,000 uniformly distributed points, (b) the initial structure of the SOFM, (c) the structure of the SOFM after 100 iterations, (d) the final structure the SOFM, (e) 2,000 gaussian distributed points, (f) the final structure of the SOFM.

4. Adding conscience to the Kohonen learning algorithm

Competitive learning algorithms often lead to solutions where several network nodes remain underutilized or completely unutilized. This can deteriorate the performance of the method significantly since specific nodes will be winning the competitions consistently. The Kohonen learning algorithm attempts to overcome these problems by using topological neighborhoods [6]. Although this approach is quite effective, it does not

alleviate all the problems completely. There are other approaches in the literature which try to attack these problems. Three of them which seem to be quite promising were considered in this study (the convex combination method [10], the competitive learning with conscience [7] and the competitive learning with attention [11]).

Experimental results demonstrate that the competitive learning with conscience approach is superior to the other two [9]. However, none of these approaches demonstrated the topology preserving property. This is a direct consequence of the fact that the above algorithms change the weights of the winning node only. Since the topology preserving property is very critical in the computation of a well behaved geometric hash function, we combined the Kohonen learning algorithm, which preserves the topology, with each one of the above heuristics, which improve node utilization. Our objective was to strengthen the performance of the Kohonen algorithm in terms of node utilization, while preserving the topology at the same time. The results obtained indicated that all the variations preserved the topology (slight distortions were observed), however, the Kohonen algorithm with conscience gave the best results in terms of node utilization [9].

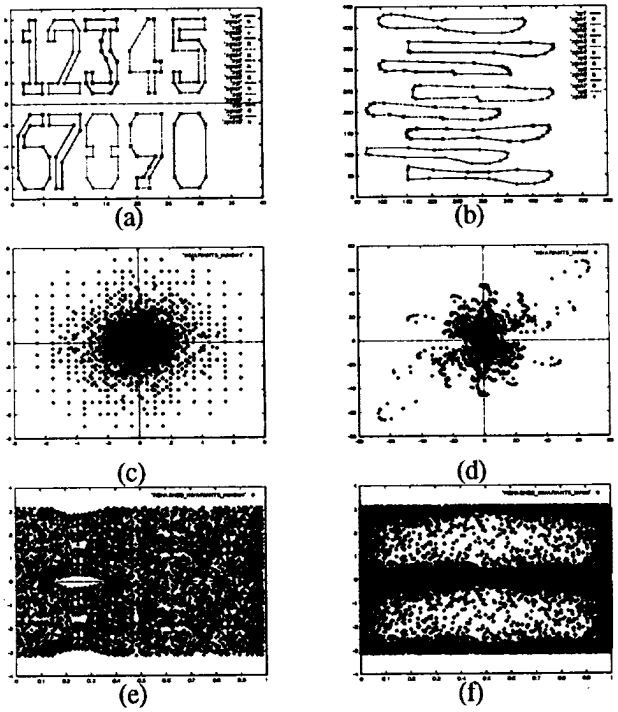

Figure 2. (a) The set of numbers, (b) the set of knives, (c) the invariants for the set of numbers, (d) the invariants for the set of knives, (e) the rehashed invariants for the set of numbers, (f) the rehashed invariants for the set of knives.

5. Simulations and experimental results

This section presents a number of experimental results and comparisons which demonstrate the effectiveness of the proposed approach. The implementation details of the SOFMC-NN can be found in [9].

5.1. Experiment 1

The set of objects used in the first experiment is shown in Figure 2b. Similarity transformations have been considered in this experiment. The distribution of invariants related to this data set is shown in Figure 2d. Applying the rehashing transformation derived for the case of similarity transformations, results in the non-uniform distribution shown in Figure 2f. The reason that rehashing has failed to perform well in this case, is that the computed distribution of invariants does not resemble a Gaussian. Figures 3a and 3c shows the distribution of hash entries over a 20 x 20 hash table for the case of the original invariants and the rehashed invariants correspondingly. Obviously, rehashing does not perform satisfactorily.

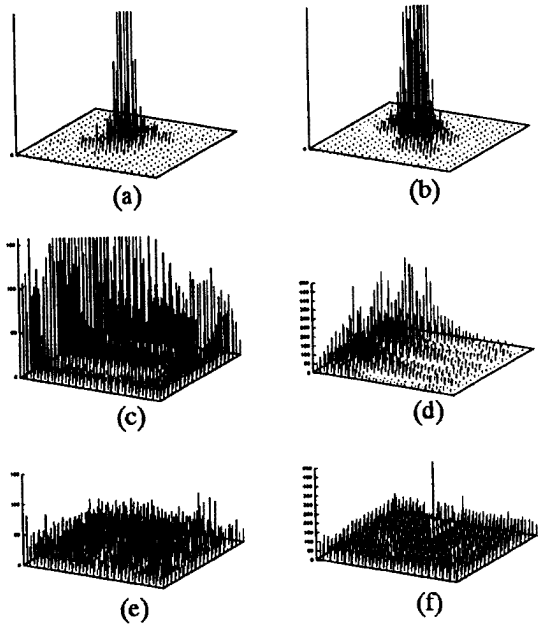

Figure 3. (a), (c) and (e) the distribution of hash entries under similarity transformations using the original approach, rehashing, and the SOFMC-NN correspondingly; (b), (d) and (f) the distribution of hash entries under affine transformations.

To demonstrate our approach, we have chosen a SOFMC-NN with two inputs and 400 output nodes, arranged on a 20 x 20 grid. The network was trained using the Kohonen learning algorithm with conscience for 500 epochs. The number of invariants used during training was 31,572 and these were normalized in the

interval [0, 1] x [0, 1]. The feature map to which the network converged is shown in Figure 4a. Figure 3e illustrates the distribution of entries over the elastic hash table, which was computed by voting using all the training patterns. To show the amount of hash table utilization, we have computed the standard deviation (SD) of the number of entries that each hash bin stores. The first row of Table (1) shows the results.

TABLE 1. Standard deviation of the number of hash entries.

	Standard deviation		
	Original	Rehashed	SOFMC-NN
Similarity transf.	62.31	35.33	13.40
Affine transf.	194.20	86.06	25.38

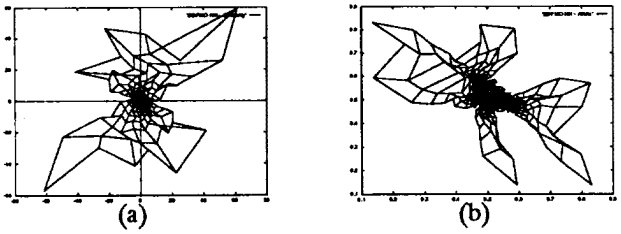

Figure 4. (a) the structure of the SOFMC for the case of similarity transformations, (b) the structure of the SOFMC for the case of affine transformations.

5.2. Experiment 2

This experiment deals with affine transformations. The database of the real objects used is shown in Figure 5. Invariants based on unstable basis triplets have been rejected using the area based criterion given in [8]. It should be mentioned that this approach for rejecting unstable triplets is very heuristic and it may reject basis triplets which are not truly unstable. However, we have used it in our experimentation because of its simplicity and because it does not affect the ideas demonstrated here.

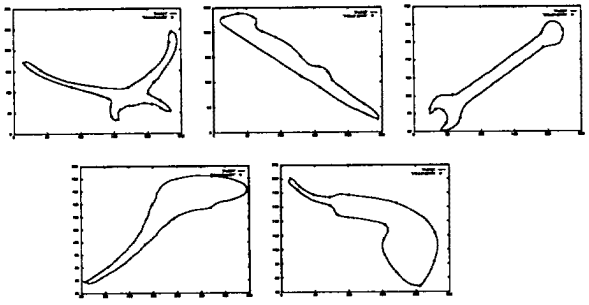

Figure 5. The set of model objects

Figure 6a shows the computed invariants. Observing the distribution of invariants in Figure 6a more carefully, we can see that the third quadrant is less crowded than the other quadrants. This seems to be in agreement with the qualitative results obtained in the case of affine transformations and under the assumption that the distribution of model points is uniform over a convex domain [4],[5]. However, no rehashing transformation was derived under this assumption because of the intractability of the computations involved. The rehashing transformation we used here is the only one derived in the case of affine transformations and it is based on the assumption that the distribution of model features is Gaussian over a convex domain. Obviously, the rehashing transformation does not perform satisfactorily, as is illustrated in Figure 6b.

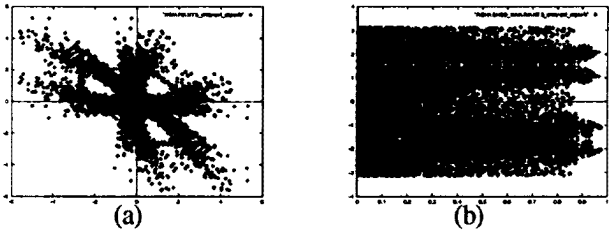

Figure 6. (a) The distribution of (affine) invariants for the set of different objects, (b) the rehashed distribution of invariants.

Figures 3b and 3d show the distribution of hash entries over a 20 x 20 hash table, for the case of the original invariants and the rehashed invariants correspondingly. A SOFMC-NN with the same architecture as in the previous experiment was utilized. The same network parameters were chosen as before except for the number of epochs which was chosen to be 100. The number of invariants used to train the SOFMC-NN was 41,292, normalized in the interval [0, 1] x [0, 1]. The feature map to which the network converged is shown in Figure 4b. Figure 3f illustrates the distribution of hash entries over the elastic hash table and the second row of Table (1) shows the computed SD's (hash table occupancy) for all the approaches.

5.3. Experiment 3

In this section, we report two recognition experiments. In the first experiment, the scene shown in Figure 7a was considered. The recognition results are shown in Figure 7e. The first two rows of Table (2) present the number of hypotheses tried by each approach until both models were recognized correctly (60% or more of the model points were required to match with the scene). In the hypothesis verification step, a candidate match was verified if the corresponding scene basis triplet had received at least one fourth of the maximum number of votes. Next, we considered the scene of Figure 7b. The recognition results are shown in Figure 7f and the number of hypotheses tried by each method is shown in the last three rows of Table (2). Overall, the proposed

approach verified about 35% - 50% less hypotheses than the hypotheses verified by geometric hashing without rehashing and 20% - 30% less hypotheses than the hypotheses verified by geometric hashing with rehashing.

TABLE 2. Number of hypotheses tried during verification.

	Number of hypotheses		
	Original	Rehashed	SOFMC-NN
Scene1 (Model3)	140998	99329	69835
Scene1 (Model4)	11335	8550	6861
Scene2 (Model1)	320512	292046	204997
Scene2 (Model2)	220435	178334	144546
Scene2 (Model3)	34295	26947	19784

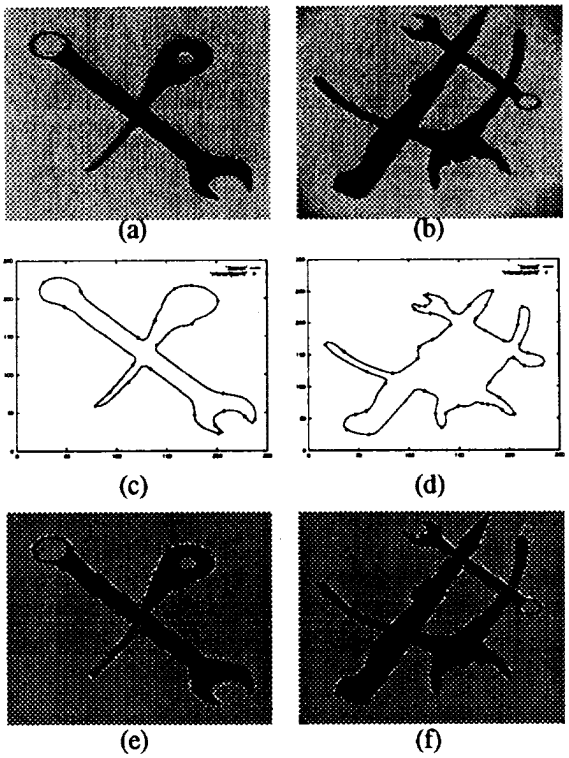

Figure 7. (a) and (b) two real scenes with overlapped models, (c) and (d) the boundary contours with the interest points (curvature maxima and zero-crossings) marked, (e) and (f) the recognition results.

6. Conclusions

The advantages of the proposed approach are very important, given that the choice of a good hash function is problem dependent. The availability of a learning scheme which can find a geometric hash function with stable behavior, that is independent of the problem at hand, is particularly attractive. Furthermore, the independence of the proposed approach from any assumption suggests that the proposed approach is a very useful tool in helping us to derive approximate analytical rehashing functions in cases where a closed form solution cannot be found using traditional approaches. Also, extending the SOFMC-NN to handle 3-D hash spaces is quite possible. One problem associated with the proposed approach is that the training of the SOFMC-NN is rather time consuming. However, this is not a serious disadvantage since the SOFMC-NN can be parallelized.

Acknowledgements

The first author is particularly obliged to Dr. Isidore Rigoutsos for his valuable comments and help. This work was supported by a grant from the FSGC (Florida Space Grant Consortium) and TRDA (Technological Research and Development Authority).

References

[1] Y. Lamdan, J. Schwartz and H. Wolfson, "Affine invariant model-based object recognition", *IEEE Trans. on Robotics and Automation*, vol. 6, no. 5, pp. 578-589, October 1990.

[2] M. Costa, R. Haralick, and L. Shapiro, "Optimal affine matching", *In Proceedings of the 6th Israeli Conference on Artificial Intelligence and Computer Vision*, December 1989.

[3] I. Rigoutsos and R. Hummel, "Robust similarity invariant matching in the presence of noise", *In Proceedings of the 8th Israeli Conference on Artificial Intelligence and Computer Vision*, December 1991.

[4] I. Rigoutsos and R. Hummel, "Implementation of geometric hashing on the connection machine", *In Workshop on Directions in Automated Cad-based Vision*, pp. 76-84, June 1991.

[5] I. Rigoutsos, *Massively parallel bayesian object recognition*, Ph.D. thesis, Computer Science Department and Courant Institute of Mathematical Sciences, New York University, August 1992.

[6] T. Kohonen, *Self-organization and associative memory*, Springer-Verlag, 3rd edition, 1989.

[7] D. DeSieno, "Adding a conscience to competitive learning", *IEEE International Conference on Neural Networks*, vol. I, pp. 117-124, 1988.

[8] M. Costa, R. Haralick, T. Phillips, and L. Shapiro, "Optimal affine-invariant point matching", *SPIE Vol. 1095 Applications of Artificial Intelligence*, VII (1989), pp. 515-530.

[9] G. Bebis, M. Georgiopoulos and N. da Vitoria Lobo, "Increasing the efficiency of geometric hashing using self-organizing maps with conscience", submitted for publication.

[10] R. Hecht-Nielsen, "Counterpropagation networks", *Applied Optics*, vol. 26, pp. 4979-4984, 1987.

[11] G. Huerter, "Solution of the traveling salesman problem with an adaptive ring", *IEEE International Conference on Neural Networks*, vol. I, pp. 85-92, 1988.