Geometric Hashing: An Overview

HAIM J. WOLFSON
Tel Aviv University
ISIDORE RIGOUTSOS
IBM T.J. Watson Research Center

Geometric hashing, a technique originally developed in computer vision for matching geometric features against a database of such features, finds use in a number of other areas. Matching is possible even when the recognizable database objects have undergone transformations or when only partial information is present. The technique is highly efficient and of low polynomial complexity.

Object recognition is the ultimate goal of most computer vision research. An ideal object recognition system should be able to recognize objects in an image that are partially occluded or have undergone geometric transformations. Most systems will use a large database of models and apply model-based recognition.

Say you want to give a robot the ability to recognize all objects and tools on a factory floor. If there are only a few hundred objects, you could design a database of these objects and store it in the robot's memory. When the robot receives a sensory image of its environment from a video camera or a range sensor, it should be able to quickly retrieve from memory objects that appear in the image. Although quite natural in human vision, this task in a robot requires the solution of several complicated problems:

1. The objects in the acquired scene appear rotated and translated relative to their initial database position, and the whole scene undergoes a sensor-dependent transformation, such as the projective transformation of a video camera.
2. The objects in the scene may partially occlude each other, and the scene may include additional objects not included in the database.
3. It is computationally inefficient to retrieve each individual object from the database and compare it against the observed scene in search of a match. For example, if the scene contains only round objects, it does not make sense to retrieve rectangular objects to match against it.

We need a method that allows direct access to only the relevant information—such as an indexing-based approach. For example, if you are looking for words in long strings of text, you could use a table accessed by indices that are functions of individual words. The table contains the strings where the word appears and the location of the word in the strings. It would be easy then to locate a word by retrieving all of its appearances from the table.

This kind of approach was originally proposed for geometric object recognition, making use of indices based on local geometric features that remained invariant to the object transformation. The features were local to handle partial occlusion, and their indexing function was invariant to the relevant transformation, because unlike words in text, geometric features have both location and orientation. For over a decade now, indexing-based approaches have been gaining ground as the method of choice for building working recognition systems that can

1070-9924/97/$10.00 © 1997 IEEE

operate with large model databases.

In its modern incarnation, geometric hashing, a method based on the indexing approach, originated in the work of Schwartz and Sharir.[1,2] These first efforts concentrated on the recognition of rotated, translated, and partially occluded two-dimensional objects from their silhouettes using boundary-curve matching techniques. As opposed to the simplified text analogy, the technique in reality is more complicated, requiring shape information rather than just the location of local features. Two shapes may have the same local features yet be entirely different in appearance. If the shape's rigidity is conserved, then not only the local features but also their relative spatial configuration are important.

In order to exploit geometric consistency and to tackle model-based object recognition in two-dimensional and three-dimensional settings, Schwartz, Wolfson, and Lamdan developed a new geometric hashing technique applicable to arbitrary point sets, or constellations, under various geometric transformations.[3-5] They developed efficient algorithms for recognizing flat rigid objects represented either by point sets or by curves under the affine approximation of the perspective transformation, and they extended the technique to recognize point sets under arbitrary transformations and to distinguish rigid 3D objects from single 2D images.[6]

Since this early work, many research groups have built and used geometric hashing systems. Most implementations have worked as well as classical model-based vision systems, on the whole delivering on the promise of greater efficiency. One of the advantages is that geometric hashing is inherently parallel. In fact, with minimal communication and maintenance costs, the underlying data structure can be easily decomposed and shared among a number of cooperating processors, and the technique has been implemented on the Connection Machine.[7-8]

Researchers also soon discovered that the distribution of indices over the space of invariants is nonuniform.[9-13] (This nonuniformity, however, is not specific to geometric hashing, but appears to be endemic to all indexing schemes.[14-15]) From a practical angle, nonuniform distribution results in different lengths for the hash entry lists. Since the length of the longest list dominates the time needed to carry out the histogram phase of the algorithm, a nonuniform distribution will adversely affect the method's performance. A uniform distribution, however, not only reduces execution time but can also result in much more efficient storage of the hash table data structure. Additionally, in parallel implementation, a more or less constant co-occupancy of all the hash bins results in an improved load balancing among the processors.[8] Knowledge of the expressions for the index distributions over the space of invariants greatly facilitates the equalization of the hash bin occupancy.

One of the advantages is that geometric hashing is inherently parallel.

In addition, for the case where models undergo similarity or affine transformations, you can incorporate a noise model into the geometric hashing framework and analytically determine its effect on invariants. This analysis provides a detailed description of the method's behavior in the presence of noise.

With this augmented framework, you can develop a Bayesian approach to object recognition with geometric hashing.[16] Augmentation of the traditional geometric hashing algorithm with an error model layer and a Bayesian layer allows the creation of working systems that can operate with real-world photographs and large model databases.[17]

Underlying ideas

In recognizing objects, geometric hashing and indexing methods are efficient and can easily be made parallel. These methods are especially attractive in model-based schemes, but they also hold significant advantages in pairwise object-scene comparisons because of their ability to also handle partially occluded objects. However, this is difficult because it is not known which database objects will appear and what their pose will be. The model information is encoded in a preprocessing step and stored in a large memory, in this case a hash table. The contents of the hash table are independent of the scene and can thus be computed offline, not affecting the recognition time. Access to the memory is based on geometric information that is invariant of the object's pose and computed directly from the scene.

During the recognition phase, and when presented with a scene, the method accesses the previously constructed hash table, indexing geometric properties of features extracted from the scene for matching with candidate models. A search of all scene features is still required, but geometric hashing obviates a search of all models and their features.

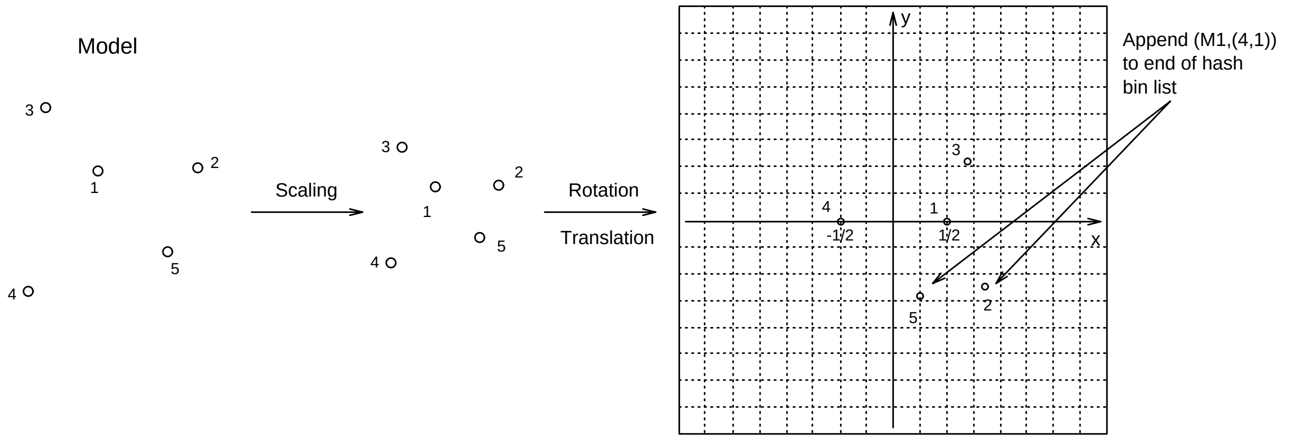


Figure 1. Determining the hash table entries when points 4 and 1 are used to define a basis. The models are allowed to undergo rotation, translation, and scaling. On the left of the figure, model M_1 comprises five points.

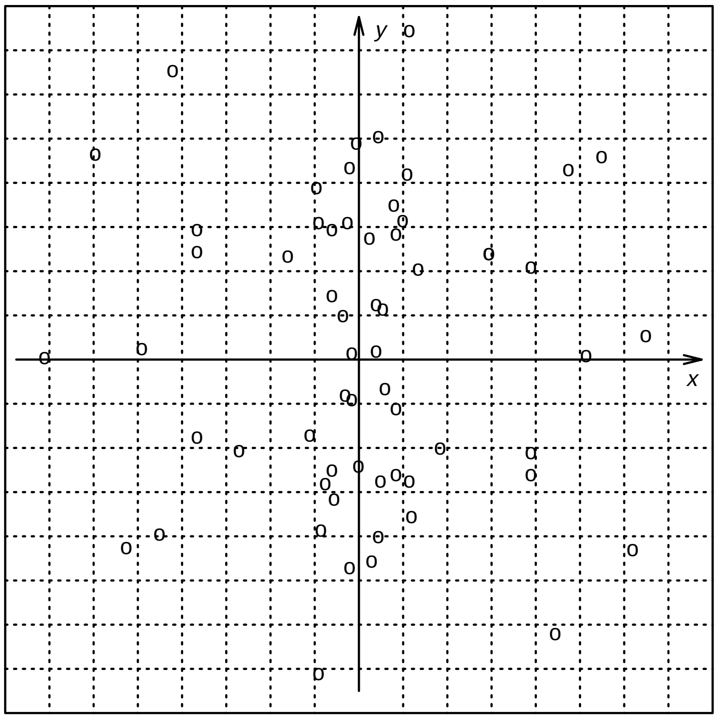


Figure 2. The locations of the hash table entries for model M_1. Each entry is labeled with the information "model M_1" and the basis pair (i, j) used to generate the entry. The models are allowed to undergo rotation, translation, and scaling.

Scene features—including such things as points, linear and curvilinear segments, and corners—are accumulated during the feature extraction stage. The collection of features is represented by a set of dots, each dot representing a feature's location. Associated with each dot is a list of one or more attributes, depending on the feature's type.

Suppose we wish to perform recognition of patterns of point features in the presence of similarity transformations—that is, the point features may be translated, rotated, or scaled. (Geometric hashing can tackle other transformations, such as rigid and affine transformations, but similarity transformations are of moderate difficulty and effectively showcase the methodology.) The left side of Figure 1 shows model M_1, which consists of five dots with position vectors $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$, and $\mathbf{p}_5$. We want to encode this dot information appropriately and store it into a table. This way, if the system detects this collection of dots in a scene, it could conclude that they belong to the model M_1.

If we assume for the moment that each dot has a unique, distinctive color, a potential albeit simplistic indexing scheme would use the color as the dot's index: an entry in the hash bin would include the identity of the model to which the dot belongs. In the recognition stage, the system would simply scan the dots, access the hash table using each dot's color, and increase the count of the models appearing in the accessed table bins. Models accumulating high counts have high probability to be present in the scene. The computational complexity of such a scheme would be linear in the number of the scene dots.

However, what happens in the least informative case, where dots belonging to a model have no attributes except for their geometric configuration? Is there a distinctive geometric "color"? Yes, the natural geometric "color" of a dot is the set of its coordinates, but coordinates depend on a reference frame. The question then becomes one of whether there is a natural reference frame for a model that will remain present under partial occlusion. One straightforward such choice

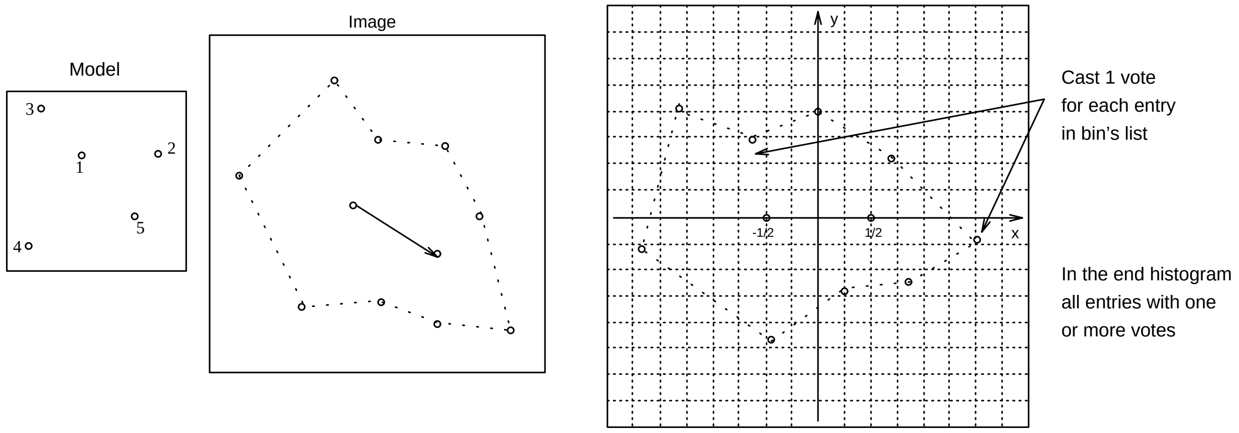


Figure 3. Determining the hash table bins that are to be notified when two arbitrary image points are selected as a basis. Similarity transformation is allowed.

is a pair of dots belonging to the model and defining an unambiguous reference frame that remains unchanged if the model undergoes rotation, translation, and/or scaling.

Let's take the pair of dots $\mathbf{p}_4$, $\mathbf{p}_1$ as an ordered basis to such a reference frame. As shown in Figure 1, we scale the model M_1 so that the magnitude of $\overrightarrow{\mathbf{p}_4\mathbf{p}_1}$ in the *Oxy* system equals 1. Suppose now that we place the midpoint between dots 4 and 1 at the origin of a coordinate system *Oxy* in such a way that the vector $\overrightarrow{\mathbf{p}_4\mathbf{p}_1}$ has the direction of the positive x axis. The remaining three points of M_1 will land in three locations. Let's record in a *quantized* hash table—in each of the three bins where the remaining points land—the fact that model M_1 with basis "(4, 1)" yields an entry in this bin.

Since our goal is to perform recognition under partial occlusion, we are not guaranteed that both basis points $\mathbf{p}_1$ and $\mathbf{p}_4$ will appear in each scene where model M_1 will be present. Consequently, we encode the model dot's information in all possible ordered basis pairs. Namely, the hash table contains three entries of the form (M_1, (4, 2)), three entries of the form (M_1, (4, 3)), and so on. Each triplet of entries is generated by first scaling the model M_1 so that the corresponding basis has unit length in the *Oxy* coordinate system, and then by placing the midpoint of the basis at the origin of the hash table in such a way that the basis vector has the direction of the positive x axis. The same process is repeated for each model in the database. Of course, some hash table bins may receive more than one entry. As a result, the final hash table data structure will contain a list of entries of the form (*model*, *basis*) in each hash table bin. Figure 2 shows the locations of all the hash table entries for model M_1.

In essence what we have done is define in turn orthonormal bases for the coordinate system *Oxy* using a pair of vectors $\mathbf{p}_x^s \triangleq (\mathbf{p}_{\mu_2} - \mathbf{p}_{\mu_1})$ and $\mathbf{p}_y^s \triangleq \mathrm{Rot}_{90}(\mathbf{p}_y^s)$; μ_1 and μ_2 denote distinct points taken from the model M_1. For each choice of a basis, the remaining points $\mathbf{p}$ of M_1 are represented in this basis using the equation

$$\mathbf{p} - \mathbf{p}_0^s = u\mathbf{p}_x^s + v\mathbf{p}_y^s \qquad (1)$$

where $\mathbf{p}_0^s = (\mathbf{p}_{\mu_1} + \mathbf{p}_{\mu_2})/2$ is the midpoint between $\mathbf{p}_{\mu_1}$ and $\mathbf{p}_{\mu_2}$. The scalar quantities u and v remain invariant under similarity transformation of M_1, and their quantization allows us to determine an index (u_q, v_q) that will take us into a location of a 2D hash table data structure. In the hash bin that is accessed via (u_q, v_q) we enter the information (m, ($\mathbf{p}_{\mu_1}$, $\mathbf{p}_{\mu_2}$)).

In the recognition phase, a pair of points ($\mathbf{p}_{\mu_1}$, $\mathbf{p}_{\mu_2}$) from the image is chosen as a candidate basis. As before, this ordered basis defines a coordinate system *Oxy* whose center coincides with the midpoint of the pair; the direction of the basis vector $\mathbf{p}_{\mu_2} - \mathbf{p}_{\mu_1}$ coincides with that of the positive x axis. The magnitude of the basis vector defines the "unit" length for *Oxy*. The coordinates of all other points are then calculated in the coordinate system defined by the chosen basis. Each of the remaining image points is mapped to the hash table, and all entries in the corresponding hash table bin receive a vote. In essence, for the selected basis and for each remaining point in the scene, Equation 1 is used to determine the index (u_q, v_q) of a hash bin to access. As shown in Figure 3, each pair of *(model,*

Preprocessing phase

For each model *m* do the following:

1. Extract the model's point features. Assume that *n* such features are found.
2. For each ordered pair, or basis, of point features do the following:

(a) Compute the coordinates (*u*, *v*) of the remaining features in the coordinate frame defined by the basis.

(b) After proper quantization, use the tuple (u_q, v_q) as an index into a 2D hash table data structure and insert in the corresponding hash table bin the information (*m*, (*basis*)), namely the model number and the basis tuple used to determine (u_q, v_q).

Recognition phase

When presented with an input image, do the following:

1. Extract the various points of interest. Assume that S is the set of the interest points found; let *S* be the cardinality of S.
2. Choose an arbitrary ordered pair, or basis, of interest points in the image.
3. Compute the coordinates of the remaining interest points in the coordinate system *Oxy* that the selected basis defines.
4. Appropriately quantize each such coordinate and access the appropriate hash table bin; for every entry found there, cast a vote for the model and the basis.
5. Histogram *all* hash table entries that received one or more votes during step 4. Proceed to determine those entries that received more than a certain number, or threshold, of votes: Each such entry corresponds to a potential match.
6. For each potential match discovered in step 5, recover the transformation T that results in the best least-squares match between all corresponding feature pairs.
7. Transform the features of the model according to the recovered transformation T and verify them against the input image features. If the verification fails, go back to step 2 and repeat the procedure using a different image basis pair.

Figure 4. The two stages of the geometric hashing system.

basis) found in the accessed bin gets a vote.

If we select a pair of scene points that corresponds to a basis on one of the models, we expect it to accumulate as its score the votes from all other unoccluded points belonging to this model. If there are sufficient votes for one or more (model, basis) combinations, then a subsequent stage attempts to verify the presence of a model with the designated basis matching the chosen basis points. In the case where model points are missing from the image because they are obscured, recognition is still possible as long as a sufficient number of points hash to the correct bins. The list of entries in each bin may be large, but because there are many possible models and basis sets, the likelihood that a single model and single basis set will receive multiple votes is quite small, unless a configuration of transformed points coincides with a model or part of it.

In general, we do not expect the voting scheme to give only one candidate solution. The goal of the voting scheme is to act as a sieve and reduce significantly the number of candidate hypotheses for the verification step. *For the algorithm to be successful it is sufficient to select as a basis tuple any set of image points belonging to some model.* It is not necessary to hypothesize a correspondence between specific model points and specific scene points, since all models and basis pairs are stored redundantly within the hash table. Classification or perceptual grouping of features can be used to make the search over scene features more efficient—for example, by making use of only special basis tuples.

The two stages making up the core of the geometric hashing system are outlined in Figure 4.

We have seen that two points suffice to define a basis when the models are allowed to undergo a 2D similarity transformation. However, geometric hashing represents a unified approach that applies also to many other useful transformations encountered in object recognition problems. The only difference from one application to another is the number of features that have to be used to form a basis for the reference frame. This, of course, affects the computational complexity of the algorithm in the different cases, which still remains polynomial.

The following lists give a number of examples where this general paradigm applies. Almost all cases involve point matching. Use of other features, such as lines, can be understood by analogy. The first list covers recognition of 2D objects from 2D data:

1. *Translation in 2D:* The technique is applicable using a one-point basis, the point being viewed as the origin of the coordinate frame.
2. *Translation and rotation in 2D:* A two-point basis can be used, but one point with a direction (obtained, say, from an edge segment) provides enough information for a unique definition of a basis.
3. *Translation, rotation, and scaling in 2D:* Discussed earlier.
4. *Affine transformation in 2D:* A three-point basis defines an unambiguous reference frame.[3-5]
5. *Projective transformation in 2D:* A four-point basis is needed to recover a projective transformation between two planes.

When 3D data—such as range or stereo data of the objects—is available, the recognition of 3D objects from 3D images must be considered. Development of techniques for this case is especially important in an industrial environment, where 3D data can be readily obtained and used. Geometric hashing for 3D rigid transformation (translation and rotation) has been applied in CAD/CAM, medical imaging, and protein comparison and docking in molecular biology (as described in other articles in this issue). Here are examples of 3D transformations:

1. *Translation in 3D:* As in the 2D case, a one-point basis will suffice.

2. *Translation and rotation in 3D:* This is the interesting case corresponding to rigid motion. A basis consisting of two noncollinear lines suffices. Alternatively, three points, with additional triangle-side length information, can be used to define a basis.

3. *Translation, rotation, and scaling in 3D:* A basis comprising two noncollinear and nonplanar lines will suffice. Alternatively, a point and a line can be used.

In general, if

♦ the database contains M known models, each comprising n features,

♦ the scene during recognition contains S features, and

♦ c features are needed to form a basis,

then the time complexity of the preprocessing phase is $O(Mn^{c+1})$. The complexity of the recognition phase is $O(HS^{c+1})$, where H represents the complexity of processing a hash table bin. As in all hashing techniques, H depends on the hash table occupancy and bin distribution. If the size of the table is of the order of the elements it contains and the distribution is uniform, the access complexity H will be equal to $O(1)$. If, on the other hand, the table is small or all of the features hash into only a few bins, the access time may be dominated by the number of elements in the table, which is Mn^{c+1}.

This is, however, an unlikely situation. It is important to note that the hash table and its structure are known in advance and before the recognition phase. Thus, you can evaluate this structure and decide whether it requires the application of rehashing procedures, the splitting of the table into several tables, or the changing of the index structure to a higher-dimensional one. It is also important to mention the bins with high occupancy, which cause significant computational effort, can simply be ignored—their information content is not salient enough to assist recognition. Thus, you can decide a priori on an upper bound to the size of the hash table bins that will be processed. An extension of this idea leads to weighted voting, where the bin information is inversely proportional to the bin's size.

In the recognition of 3D objects from single 2D images we have the additional problem of different dimensions. A number of methods have been suggested to tackle this problem using geometric hashing; see elsewhere.[6]

Index distributions

One issue of particular importance is index distribution over the space of invariants when the allowed transformation is known and fixed. The assumption is that all point features are identically and independently distributed following a random process with a known probability density function $f()$. Recall that the indices used to access the hash table are the quantized solution (u, v) to Equation 1. Since the properties of the random process generating the point features are known, the joint probability density function $f(u, v)$ of u and v can be computed using the expression

$$\int_{\mathbb{R}^4} f\big(x(u,v), y(u,v)\big) f\big(x_{\mu_1}, y_{\mu_1}\big) f\big(x_{\mu_2}, y_{\mu_2}\big) \qquad | \mathcal{J} |^{-1}\, dx_{\mu_1} dx_{\mu_2} dy_{\mu_1} dy_{\mu_2}, \tag{2}$$

where $\mathcal{J}$ is the Jacobian of the transformation (as in Equation 1). Evaluation of this integral yields the distribution of indices over the space of invariants for the transformation under consideration.

For example, in the case of similarity transformations and feature points generated by the Gaussian random process

$$N\left(0, \begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix}\right),$$

evaluation of the integral 2 yields

$$f(u,v) = \frac{12}{\pi} \cdot \frac{1}{(4(u^2+v^2)+3)^2}.$$

The distribution over the space of invariants for synthetically generated indices, as well as several of its contours, are shown in Figure 5, with the occupancies of the hash table encoded as heights.

When the random process giving rise to the feature points comprising the various database models is not known, it is typically possible to obtain an approximation $f^*()$ of the probability density function using numerical techniques: you can use $f^*()$ instead of $f()$ in Equation 2. Alternatively, you can use a numerical approximation of $f^*(u, v)$.

From hashing to rehashing

If the probability density function for the distribution of indices over the space of invariants is available, you can effectively use it to provide

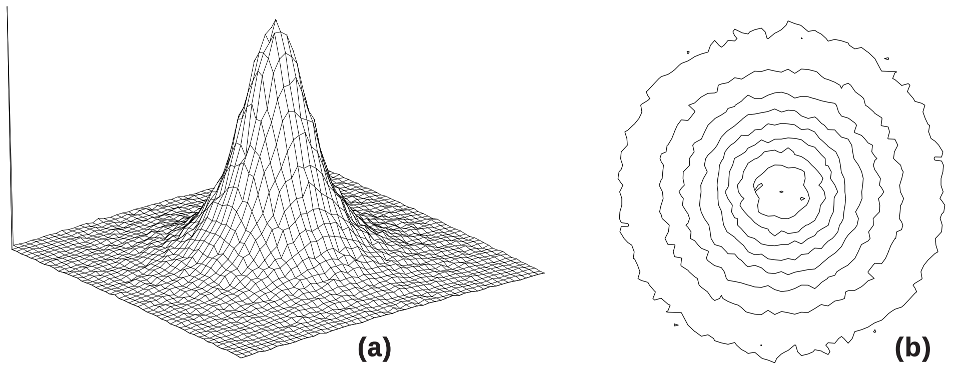


Figure 5. The distribution over the space of invariants, and several of its contours, for the case of Gaussian-distributed point features. Similarity transformation is allowed.

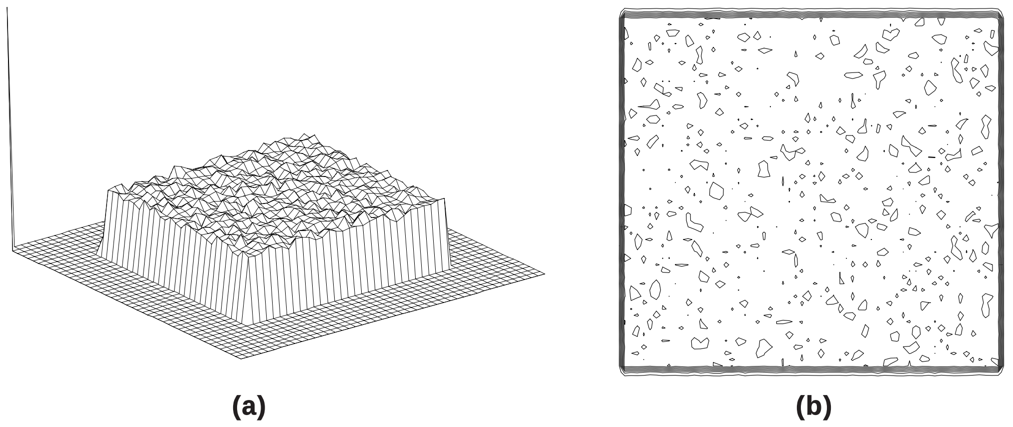


Figure 6. Hash table equalization for the case of similarity transformations and point features generated by the Gaussian process $N\left(0,\begin{pmatrix}\sigma & 0\\ 0 & \sigma\end{pmatrix}\right)$: (a) the expected distribution of remapped invariants; (b) several of the distribution's contours.

substantial improvements in storage requirements and performance. The outlined methodology is generally applicable to all indexing-based object identification methods.

The nonuniform occupancy of the hash bins results in bins that contain a large number of entries. Since the longest such list dominates the time spent in the histogram phase, a uniform distribution of the entries is desirable. A uniform distribution can be achieved by transforming the coordinates of point locations so that the equispaced quantizer in the space of invariants yields an expected uniform distribution. To this end, knowledge of the expected joint probability density function $f(u, v)$ (or an approximation of it) for the distribution of the untransformed coordinates (that is, the tuple of invariants) is required.

In essence we seek a mapping, $\mathbf{h}\colon\ \mathbb{R}^2 \to \mathbb{R}^2$, that evenly distributes the hash bin entries over a rectangular hash table. Notice that the range of the function $\boldsymbol{h}$ is the space of transformed invariants and not the space of features extracted from the input.

For the similarity transformation example above, one such mapping is as follows:

$$h(u,v) = \left(1 - \frac{3}{4(u^2+v^2)+3},\ \operatorname{atan}2(v,u)\right) \tag{3}$$

In this expression, atan2(·,·) returns the phase in the interval $[-\pi, \pi]$. It is interesting to note that the computed rehashing function does not include the standard deviation s of the feature-generating process as a parameter. Figure 6 shows the result of hash table equalization for synthetic data. The reference spike at the upper left corner of the hash table is reproduced from Figure 5 and provides a measure of the benefits incurred by the rehashing operation. Clearly, the remapping is very efficient. This table has the same number of bins as the one in Figure 5.[17]

Exploiting existing symmetries

In addition to savings from the use of rehashing functions, further computational and storage savings are possible. Symmetries typically exist in the storage pattern of entries in the hash table; these symmetries are independent of the use of rehashing functions and thus can be used in conjunction with the rehashing functions.

For rigid and similarity transformations, the symmetries are with respect to a *point:* For every entry of the form $(\mu, (\mu_1, \mu_2))$ at location (u, v) of the hash table, there will be an entry $(\mu, (\mu_2, \mu_1))$ at location $(-u, -v)$. For the affine transformation, the symmetries are with respect to an *axis:* Every entry of the form $(\mu, (\mu_1, \mu_2))$ at location (u, v) of the hash table will have a counterpart $(\mu, (\mu_2, \mu_1))$ at location (v, u). The practical importance of this is that we can dispose of half of the hash table—at the expense of minimal additional bookkeeping. This will result in entry lists that, on average, will be half as long when spread among the existing set of processors, leading to an expected speedup by a factor of two.

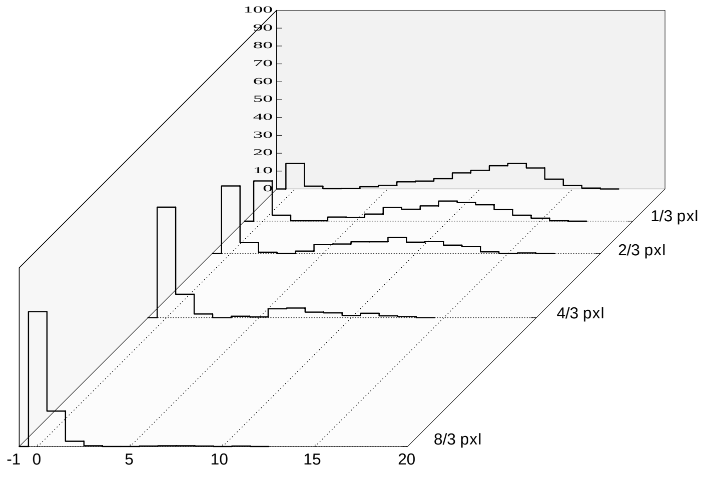
Noise modeling

We have so far implicitly assumed that the feature points in both the preprocessing and recognition phases are noise-free, an assumption that does not hold in practice.[17] Figure 7 shows the method's performance as a function of the amount of noise. Noise at input leads to positional errors, which in turn translate to errors in the computed invariants. "Small" input errors will give rise to the same computed invariant and thus the same index as the noise-free input. The semantics of *small* directly depends on the coarseness of quantization of the space of invariants. Once this coarseness is decided, an associated degree of tolerance is implicitly built into the hash table.

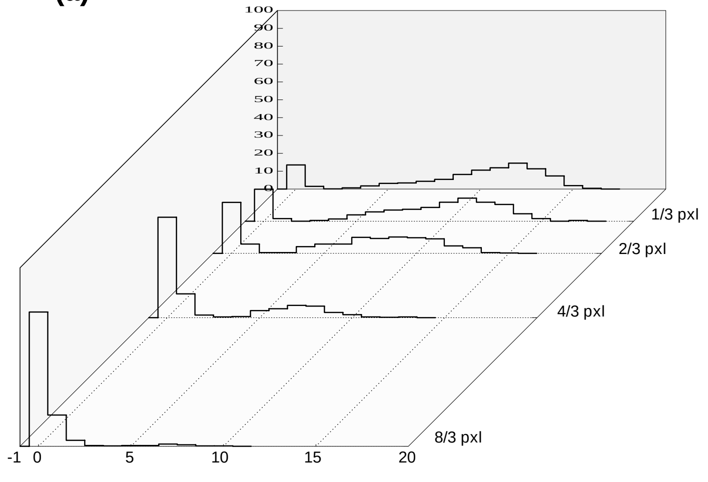
Positional errors typically translate into the computation of "wrong" hash bin indices. But because of the nature of the employed hashing functions, the respective "wrong" bins are in the neighborhood (in a Euclidean sense) of the "correct" bins that would have been accessed had the input been noise-free. Other solutions to this problem involve accessing a rectangular region of the table instead of a single bin,[18] or weighted voting.[9]

The exact shape of the neighborhood to be accessed during the recognition phase is generally complicated.[17] In particular, the size, shape, and orientation of the regions that need to be accessed depend directly on the selected basis tuple, as well as on the computed hash locations. Figure 8 shows this dependence for certain point arrangements and the similarity transformation. The variations of the region's shape are much more pronounced when an affine transformation is allowed.

Since the ultimate goal is the creation of working systems that can perform satisfactorily in the presence of noise, the method was enhanced by incorporating a noise model. The derived formulas, in addition to being useful in quantifying the observed behavior, turned out to also be compatible with a Bayesian interpretation of geometric hashing.



(a)



(b)

Figure 7. Percentage of the embedded model's bases receiving *k* votes, when used as probes, for different amounts of Gaussian noise. The models can only undergo similarity transformations. (a) The model points are distributed according to a Gaussian of $\sigma = 1$. (b) The model points are distributed uniformly over the unit disc. In both cases, the database contained 512 models, each consisting of 16 points.

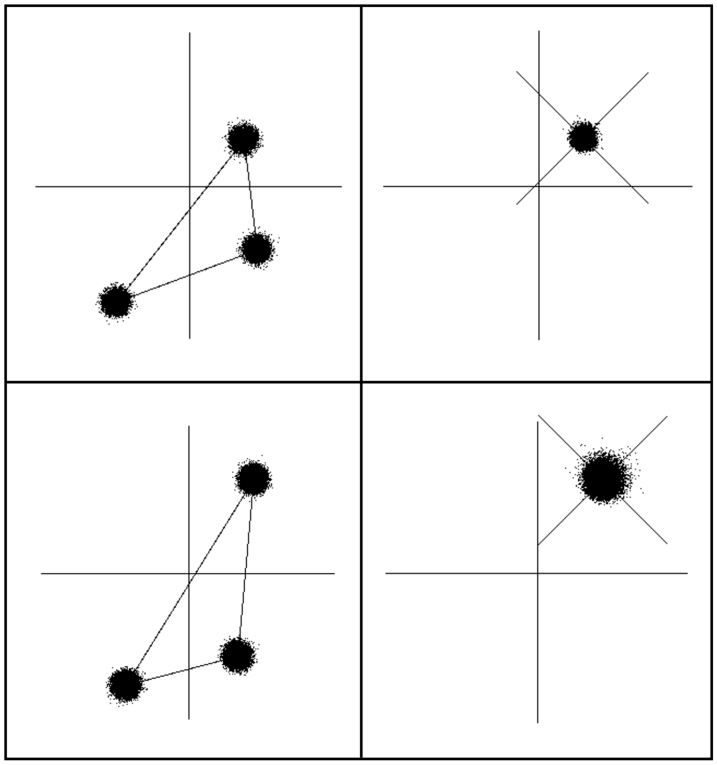


Figure 8. Regions of the hash table that need to be accessed when there is Gaussian error in the positions of the point features. Similarity transformation is allowed. The left graph of each pair shows the feature space domain, whereas the right shows the space of invariants. For presentation purposes, the amount of Gaussian error was deliberately made large.

First, the sensor noise was modeled by treating it as an additive Gaussian perturbation. The perturbations of the various feature points were assumed to be statistically independent and distributed according to a Gaussian distribution of standard deviation σ, centered at the "true" value of the variable. These errors were propagated through the hashing function of choice, and second and higher-order error terms were dropped.[17]

The derived expressions allowed us to draw the following qualitative conclusions:

1. The larger the separation of the two basis points, the smaller the spread in the space of invariants in the presence of error.
2. For a given basis separation, the distance of the point whose coordinates we compute in the coordinate frame of the basis also affects the spread: The smaller this point's distance from the center of the coordinate frame, the smaller the spread in the space of invariants.
3. A trade-off exists between the indexing power of an invariant tuple and its sensitivity to noise: Index values corresponding to relatively unpopulated regions of the space of invariants carry more information but are very sensitive to noise—the opposite also holds true.

Bayesian formulation

Recall that given a scene S containing S points, $S = \{\mathbf{p}_l\}_{l=1}^{S}$, the geometric hashing method selects two points as a basis pair, say B = $\{\mathbf{p}_\mu, \mathbf{p}_\nu\}$, and attempts to determine if a model is present in the scene. The knowledge base consists of a database of M models $\{M_k\}$, for $k = 1,\ldots, M$. Occasionally the verification step will fail to find a model that obtains sufficient support, at which point another basis pair is selected and the entire analysis is repeated. The only source of evidence during each analysis is the set $S' = S - B$ of scene points, with the exception of those forming the basis.

We wish to compute the probability $\Pr((M_k, i, j, B) \mid S')$ that model M_k is present, with points i and j of the model respectively matching $\mathbf{p}_\mu$ and $\mathbf{p}_\nu$ of the basis set B, based on the information from the hash locations as computed by the scene points S′ relative to the basis set. In particular, we wish to find the maximum of this expression over all possible M_k, i, j, and B—a *maximum likelihood approach* to object recognition. The model/basis combination maximizing this expression is the most likely match given the collection of hash values generated from S′. A reasonable assumption made is that, in the absence of any match, the probability value of even the maximum winner will not be large. On the other hand, if there is a match, then for some choice (or, most likely, multiple choices) of B there will be a large probability value for some (M_k, i, j, B). If there are multiple models present, then several model/basis combinations will share a large probability. It is sufficient to determine those few combinations that lead to the largest probabilities. Indeed, one can always subject these "winners" to a verification phase. It suffices to determine the relative probabilities and not the actual values.

Additional conditional independence assumptions are needed: It is assumed that a large number of hash values is expected near the points of the table where (M_k, i, j) hash entries occur and a uniform density (or some fixed density) is expected elsewhere, regardless of what other hash values are known to occur. The assumptions are reasonable if the features are chosen judiciously.

Using Bayes's theorem, the above formulation can be shown equivalent to maximizing

$$\log(\Pr((M_k,i,j,B))) + \sum_{p_\xi \in S'} \log\left(\frac{\Pr(p_\xi \mid (M_k,i,j,B))}{\Pr(p_\xi)}\right) \tag{4}$$

over all possible model/basis combinations and basis selections.

This maximization captures the essence of the geometric hashing approach within a Bayesian framework: After having defined a basis B using points from a scene, votes are tallied for all model/basis combinations using the information carried by the individual points in the scene, and the hash locations are computed relative to the basis B. The model/basis combinations that have accumulated a lot of votes (or a large weighted vote) are probable instances of a model: The basis combination from that model will match the chosen basis B. The redundant representation of the known models obviates the need for exhaustive consideration of all model/basis combinations and basis selections before an answer can be reached. Also, the denominators in each term of the second sum are the expected probability density values (whose calculation was outlined earlier) attached to a given location in hash space.

We can quantify the contribution from a particular hash value generated by a scene point $\mathbf{p}_x$ to a model/basis combination, thus allowing us to extend the geometric hashing method to a Bayesian maximum-likelihood model-matching system as follows:

During preprocessing, every model M_k, and every basis pair comprising points from M_k, computes the hash locations of every other point within M. At each such hash location (x, y) we make an entry that contains information about the model M_k, the basis pair (i,j), the model point ℓ (other than the basis points), and a predicted normalized covariance radius, which for example in the similarity transformation case is $\tau = (4(x^2 + y^2) + 3)\,\sigma^2$. The entries containing this information are organized in such a way that given a location (u, v), all such records having an (x, y) value lying nearby can be easily accessed.

During recognition, feature points are extracted from the scene and a trial basis formed using a pair of these points—for example, $\mathbf{p}_\mu$ and $\mathbf{p}_\nu$. The coordinates of the remaining points in S are then computed relative to the basis set, and a hash access is made to a location (u, v) in the hash domain. All nearby records of the form $(x, y, M_k, i, j, \ell, t)$ are accessed. For each such nearby record, we record a weighted vote for the model/basis combination (M_k, i, j). For the similarity transformation, the amount of the vote is determined as follows:

$$z = \log\left(1 + \frac{(4(u^2+v^2)+3)^2 \left\|p_\mu - p_v\right\|^2}{12S\tau} \cdot \exp\left(\frac{-\left\|(u,v)-(x,y)\right\|^2}{\tau / \left\|p_\mu - p_v\right\|^2}\right)\right) \tag{5}$$

The neighborhood is defined as an expression involving z: Something is considered to be in the neighborhood if the value of the respective z is greater than some threshold. Note that the above expression incorporates the value of σ, the expected error in positioning of the scene points, the number of scene points, S, and the basis-pair separation distance. The value z is only an approximation of the expression (Equation 4), which is the total contribution of the hash (u, v) to the model/basis (M_k, i, j), obtained by neglecting all terms in f except for the entry at (x, y).

The geometric hashing method allows systems to recognize objects even when the objects have undergone an arbitrary transformation and when parts of them might be occluded. The strength of the technique is in its efficiency, in its ability to operate in the presence of only partial information, and in its applicability to almost any domain where geometric matching is required; it does not require any domain-specific knowledge—only the location of certain geometric interest features. The method has been successfully applied to pattern-matching problems in computer vision, CAD/CAM, and medical imaging—some covered in this special issue. A somewhat unexpected application, introduced by Nussinov and Wolfson,[19] was to problems in structural molecular biology[20-21] and medicinal chemistry.

We have demonstrated the implementation of the method for the least informative feature: a point. Features carrying more information—such as line segments, arcs, and corners of specified angles—significantly speed up the algorithm by reducing the number of features required for an unambiguous definition of a basis. Stein and Medioni have used gray encodings of groups of consecutive edge segments (supersegments) of varying cardinalities as informative features.[11] Their TOSS system for 3D object recognition from range sensor data uses characteristic curves and local differential patches.[22] Forsyth et al.[23] use descriptors based on *pairs* of planar curves; the descriptors are invariant under perspective transformations and thus generalize the affine invariant curve-matching approach of Lamdan et al.[4]

Geometric hashing gains its efficiency from indexing, or hashing, of geometric invariants in a large memory. The matching process can be separated into two stages, with complexities adding one to another—and not multiplying each other; the preprocessing stage can occur off-line. In this stage, the system stores the relevant information in the hash table using transformation-invariant access keys. In the recognition stage, the system directly accesses the relevant locations of the memory, without retrieving superfluous information. An advantage of the technique is that the various successful retrievals, or hits, are scored in a way that preserves the overall rigidity constraints of the objects, thus sharpening the "correct" hypothesis.

The geometric hashing method was recently extended to efficiently handle the matching of objects with internal degrees of freedom, such as rotational or sliding (prismatic) joints.[24] This is an important extension, since robots, humans, many manufactured objects, and biological molecules can be modeled as assemblies of rigid subparts connected by such joints. This technique and its derivatives have been applied to the recognition of articulated objects in computer vision,[25] to the docking of flexible receptor-drug molecules,[26] and to the detection of partially similar molecules in databases of drugs.[15] An interesting point about this extension is that both geometric hashing and the generalized Hough transform techniques applied to rigid object matching can be viewed as particular cases of this new flexible object-matching technique. ♦

Acknowledgments

Haim Wolfson's research has been supported in part by the Hermann Minkowsk-Minerva Center for Geometry at Tel Aviv University, by grant no. 95-0028 from the U.S.-Israel Binational Science Foundation, by a grant of the Israeli Ministry of Science, by a basic research grant from Tel Aviv University, and by the Daat consortium grant, administered by the Israeli Ministry of Industry and Trade.

References

1. J. Schwartz and M. Sharir, "Indentification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves," *Int'l J. Robotics Research*, Vol. 6, No. 2, 1986, pp. 29–44.
2. A. Kalvin et al., "Two-Dimensional Model-Based Boundary Matching Using Footprints," *Int'l J. Robotics Research*, Vol. 5, No. 4, 1986, pp. 38–55.
3. Y. Lamdan, J. Schwartz, and H. Wolfson, "On Recognition of 3D Objects from 2D Images," *Proc. IEEE Int'l Conf. Robotics and Automation*, IEEE Computer Soc., Los Alamitos, Calif., 1988, pp. 1407–1413.
4. Y. Lamdan, J. Schwartz, and H. Wolfson, "Object Recognition by Affine Invariant Matching," *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, IEEE Computer Society, 1988, pp. 335–344.
5. Y. Lamdan, J. Schwartz, and H. Wolfson, "Affine Invariant Model-Based Object Recognition," *IEEE Trans. Robotics and Automation*, Vol. 6, No. 5, 1990, pp. 578–589.
6. Y. Lamdan and H. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Int'l Conf. Computer Vision*, IEEE Computer Society, 1988, pp. 238–249.
7. O. Bourdon and G. Medioni, "Object Recognition Using Geometric Hashing on the Connection Machine," *Proc. Int'l Conf. Pattern Recognition*, IEEE Computer Society, 1990, pp. 596–600.
8. I. Rigoutsos and R. Hummel, "Massively Parallel Model Matching: Geometric Hashing on the Connection Machine," *Computer*, Vol. 25, No. 2, Feb. 1992, pp. 33–42.
9. M. Costa, R. Haralick, and L. Shapiro, "Optimal Affine Matching," *Proc. Sixth Israeli Conf. Artificial Intelligence and Computer Vision*, Israeli Information Soc. Press, Ramat Gan, Israel, 1989, pp. 35–61.
10. M. Costa, R. Haralick, and L. Shapiro, "Optimal Affine-Invariant Matching: Performance Characterization," *Proc. SPIE Conf. Image Storage and Retrieval*, SPIE, Bellingham, Wash., 1992.
11. F. Stein and G. Medioni, "Efficient Two-Dimensional Object Recognition," *Proc. Int'l Conf. Pattern Recognition*, IEEE Computer Society, 1990, pp. 596–600.
12. P. Flynn and A. Jain, "3D Object Recognition Using Invariant Feature Indexing of Interpretation Tables," *Computer Vision, Graphics, and Image Processing: Image Understanding*, Vol. 55, No. 2, 1992, pp. 119–129.
13. R. Mohan, D. Weinshall, and R. Sarukkai, "3D Object Recognition by Indexing Structural Invariants from Multiple Views," *Proc. Int'l Conf. Computer Vision*, IEEE Computer Society, 1993, pp. 264–268.
14. I. Rigoutsos and A. Califano, "Searching in Parallel for Similar Strings," *IEEE Computational Science & Engineering*, Vol. 1, No. 2, Summer 1994, pp. 60–75.
15. I. Rigoutsos, D. Platt, and A. Califano, *Flexible 3D-Substructure Matching and Novel Conformer Derivation in Very Large Databases of Molecular Information,* Tech. Report RC20497, IBM T.J. Watson Research Center, Yorktown Heights, N.Y., 1996.
16. I. Rigoutsos and R. Hummel, "A Bayesian Approach to Model Matching with Geometric Hash-

ing," *Computer Vision and Image Understanding*, Vol. 61, No. 7, July 1995, pp. 11–26.

17. I. Rigoutsos, *Massively Parallel Bayesian Object Recognition*, doctoral dissertation, New York Univ., Computer Science Dept., New York, 1992.
18. Y. Lamdan and H. Wolfson, "On the Error Analysis of Geometric Hashing," *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, IEEE Computer Society, 1991, pp. 22–27.
19. R. Nussinov and H.J. Wolfson, "Efficient Detection of Three-Dimensional Motifs in Biological Macromolecules by Computer Vision Techniques," *Proc. Nat'l Academy of Sciences USA*, Vol. 88, Nat'l Academy of Science, Washington, D.C., 1991, pp. 10495–10499.
20. D. Fischer, R. Nussinov, and H. Wolfson, "3D Substructure Matching in Protein Molecules," *Proc. 3rd Int'l Symp. Combinatorial Pattern Matching, Lecture Notes in Computer Science,* No. 644, Springer-Verlag, New York, 1992, pp. 136–150.
21. D. Fischer et al., "A Geometry-Based Suite of Molecular Docking Processes," *J. Molecular Biology*, 1995, pp. 459–477.
22. F. Stein and G. Medioni, "Structural Hashing: Efficient Three-Dimensional Object Recognition," *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, IEEE Computer Society, 1991, pp. 244–250.
23. D. Forsyth et al., "Invariant Descriptors for 3D Object Recognition and Pose," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 13, No. 10, 1991, pp. 971–991.
24. H.J. Wolfson. "Generalizing the Generalized Hough Transform," *Pattern Recognition Letters*, Vol. 12, No. 9, 1991, pp. 565–573.
25. A. Beinglass and H. J. Wolfson, "Articulated Object Recognition, or How to Generalize the Generalized Hough Transform," *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, IEEE Computer Society, 1991, pp. 461–466.
26. B. Sandak, R. Nussinov, and H.J. Wolfson, "An Automated Robotics-Based Technique for Biomolecular Docking and Matching Allowing Hinge-Bending Motion," *Computer Applications in the Biosciences*, Vol. 11, 1995, pp. 87–99.

Haim J. Wolfson received his BSc, MSc, and PhD in mathematics from Tel Aviv University. From 1985 to 1989 he was with the Robotics Laboratory of the Courant Institute of Mathematical Sciences, New York University, and in 1989 he returned to Tel Aviv University, where he is currently an associate professor of computer science. His main research interests focus on efficient algorithms for object recognition in computer vision and for protein-structure comparison and biomolecular recognition in structural molecular biology. Together with Ruth Nussinov from Tel Aviv University's Faculty of Medicine, he is leading an interdisciplinary research team in computational structural biology.

Isidore Rigoutsos received his BSc in physics from the University of Athens, Greece, and his PhD in computer science from the Courant Institute of Mathematical Sciences of New York University. Since 1992, he has been with IBM's T.J. Watson Research Center, where he is currently leading the effort in bioinformatics and pattern discovery in biological sequences. His research activities focus on combinatorial approaches to computational biology and bioinformatics, the design of efficient invariant schemes for knowledge representation, and applied mathematics. Rigoutsos is a member of the AAAS, the IEEE, and the IEEE Computer Society.

The authors can be reached in care of Rigoutsos at IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598; e-mail, rigoutso@watson.ibm.com.