Geometric Hashing : A General and Efficient Model-Based Recognition Scheme

Yehezkel Lamdan and Haim J.Wolfson

Robotics Research Laboratory
Courant Inst. of Math., NYU
715 Broadway, 12'th floor,
New York, N.Y. 10003.

Abstract: A general method for model-based object recognition in occluded scenes is presented. It is based on *geometric hashing*. The method stands out for its efficiency. We describe the general framework of the method and illustrate its applications for various recognition problems both in 3-D and 2-D. Special attention is given to the recognition of 3-D objects in occluded scenes from 2-D gray scale images. New experimental results are included for this important case.

1. Introduction.

In this paper we present a general technique for model-based object recognition in occluded scenes. In model-based recognition one is familiar with a certain set of objects, and the task is to find instances of these objects in a given scene. This task can be further complicated by partial overlap of the objects in the scene and possible existence of other occluding and unfamiliar objects. The model-based approach is suitable for restricted industrial environments and most of the practical object recognition systems are model-based (for comprehensive surveys see [1,2] , for some recent results see [3-7]).

In a model based object recognition system one has to address two major interrelated problems, namely, *object representation* and *matching*. The *representation* should be rich enough to allow reliable distinction between the different objects in the data-base, yet terse to enable efficient *matching*. A major factor in a reliable representation scheme is its ability to deal with partial occlusion.

Work on this paper was supported by Office of Naval Research Grants N00014-82-K-0381 and N00014-85-K-0077 Work Unit NR 4007006, and National Science Foundation Grant No. NSF-DCR-83-20085.

We present a unified approach to the *representation* and *matching* problems which applies to object recognition under various geometric transformations both in 2-D and 3-D. The objects are represented as sets of geometric features, such as points or lines, and their geometric relations are encoded using minimal sets of such features under the allowed object transformations. This is achieved by standard methods of *Analytic Geometry* invoking *coordinate frames* based on a minimal number of features, and representing other features by their coordinates in the appropriate frame. Our *matching* procedure is based on the *geometric hashing* technique (see [4,7,8]). It is divided into two main steps. The first one precompiles the *representations* of the data-base objects preparing a hash-table based on these *representations*. This step is executed off-line on the data-base objects and is independent of the image scene. The second step, recognition proper, is executed on the image scene using the previously prepared hash-table for fast on-line recognition. We will analyze the complexity of this method in comparison with the *alignment* technique ([6,9]) and the *generalized Hough Transform* technique ([10]).

In previous papers ([7,11]) we demonstrated our technique for recognition of flat objects observed from an arbitrary viewpoint. We used the affine approximation to the viewing transformation and presented point, line and curve matching techniques for this case. In this paper the focus is on the general applicability and efficiency of our technique for diverse recognition tasks in 2-D and 3-D using data which is obtained either from gray scale images, range images or by tactile sensors. Special attention is given to the recognition of 3-D objects from 2-D gray scale images, and new

experimental results are presented for this important case.

The paper is organized as follows. In Section 2 we explain our technique for recognition of 2-D objects which have undergone a similarity transformation (rotation, translation, and scale). This case is presented first, since it is relatively simple, yet encompasses our basic approach facilitating the understanding of the general framework which is given in Section 3. Section 3 also reviews various instances in which our method can be applied, from the simplest case of 2-D translation to the most general case of projective transformation. Section 4 focuses on recognition of 3-D objects from 2-D gray scale images. It also includes **new experimental results** for this case. Section 5 describes application of the method for recognition of polyhedral scenes, which applies both to range data and tactile sensing data. Section 6 compares our method with the *alignment* technique and the *generalized Hough Transform*. Finally, section 7 describes some future directions.

The algorithms which we describe have been actually tested in "real life situations" by recognition of 3-D objects in composite overlapping scenes (see Fig.'s 3-7 and the figures in [7, 11] for previous results).

2. Recognition under Similarity Transformation

To demonstrate our method we first discuss flat object recognition under a similarity transformation (rotation, translation and scale). This is the situation when the viewing transformation is approximated by a parallel projection, and the images of both the model objects and the scene have been taken from the same viewpoint. We describe the model objects and the scene by sets of *interest points*, which are invariant under rotation, translation and scale. The choice of the interest operator is not essential to our method (see [11] for interest operators). From now on we rephrase the model-based recognition problem to the point-set matching task, where one is given a set of known (model) point-sets and an observed (scene) point-set. The recognition task can be stated as follows:

Is there a transformed (rotated, translated and scaled) subset of some model point-set which matches a subset of the scene point-set ?

Since a similarity transformation is uniquely defined by the correspondence of a pair of points on the model with a pair of points in the scene, one can try to match pairs of model points against pairs of scene points to obtain candidate similarity transformations. Each such transformation has to be verified by matching the transformed model against the scene ([6]).

The complexity of such a scheme is quite unfavorable. Given m points on the model and n points in the scene, the worst case complexity is $(m \times n)^2 \times t$, where t is the complexity of verifying the model against the scene. Assuming m and n are of the same magnitude, and t is at least of magnitude m, the worst case complexity is of order n^5 for recognition of a single model in the scene. In case there is a number of models the complexity of recognition is multiplied by their number.

Our aim is to reduce significantly the complexity of recognition for a single model, and also to allow simultaneous processing of all data-base models in the scene, without trying to match them sequentially with the scene data. This reduction of complexity is achieved by dividing the algorithm into two stages, so that the model preprocessing stage can be accomplished *without any knowledge of the scenes to be recognized.* As was mentioned in the introduction we have to address two major interrelated problems: *representation* and efficient *matching*.

2.1. Similarity Invariant Representation of Planar Point Sets

Our goal is to represent a set of planar points by few intrinsic parameters in a rotation, translation and scale invariant manner. This can be done by defining an orthogonal coordinate frame based on an ordered pair of points from the set, and representing all other points by their coordinates in this frame . Such a coordinate frame is defined uniquely by assigning the coordinates (0,0) and (1,0) to the first and second point respectively. (In the sequel we refer to such a pair of points as a *basis pair* although the vector joining them is a unit vector which does not span the 2-D plane. However, since it defines uniquely the second orthogonal vector, which completes it to a linear basis (frame), we hope that our terminology will not be confusing.) The unit vector of the $y-axis$ is the orthogonal vector of the same length pointing 90 degrees in the counter-clockwise direction.

Any similarity transformation applied to the points of our set preserves their coordinates,

if they are represented in the same two-point basis. Namely, if we define the images of our two basis points under a similarity transformation as (0,0) and (1,0) respectively, the coordinates of all the other points are automatically preserved.

Such representation allows comparison of occluded objects, since the point coordinates of an occluded object in the scene have a partial overlap with the coordinates of the stored model, presuming both are represented in a coordinate frame which is based on the same pair of points. This dependence on a basis pair of points may, however, preclude recognition when one or both of the basis points are occluded. Hence, we represent the object points by their coordinates in all possible orthogonal frames, based on a pair of object points. More specifically, the following preprocessing is applied to each model object.

Assume a model of an object defined by m *interest points*. For each ordered pair of model points the coordinates of all other $m-2$ model points are computed in the orthogonal coordinate frame defined by this basis pair. Each such coordinate (after a proper quantization) is used as an entry to a hash-table, where the basis-pair at which the coordinate was obtained and the model are recorded. The complexity of this preprocessing step is of order m^3 per model. New models added to the data-base can be processed independently without recomputing the hash-table. The major advantage of this somewhat redundant representation is that it will enable efficient recognition of objects in an occluded scene.

2.2. Matching

Given an image of a scene with partially occluded objects, extract its *interest points*. Assume n such points. Choose an arbitrary ordered pair in the scene and compute the coordinates of the scene points taking this pair as a basis. For each such coordinate check the appropriate entry in the hash-table, and for every record *(model, basis-pair)*, appearing there, tally a vote for the model and the basis-pair as corresponding to the ones in the scene.

If a certain record *(model, basis-pair)* scores a large number of votes, take it as a matching candidate. The uniquely defined similarity transformation between the coordinate frames of the appropriate scene and model basis pairs is assumed to be a candidate transformation between the model and the scene. Such a transformation which is based on correspondence of two basis pairs may not be accurate enough due to noise, however, it will induce correspondences of additional points. In the next step one may find the similarity transformation giving the best least-squares match between these additional points (see [7]). Finally, the model edges are verified against the scene edges. If the current pair does not score high enough, we pass to another basis-pair in the scene. (See Fig. 1 for a general scheme of our procedure.)

For the algorithm to be successful it is enough to pick *any* two points in the scene, belonging to some model. In such a case the model and the appropriate basis pair get a high score in the voting procedure. The voting process, per basis-pair, is linear in the number of points in the scene. Hence, the overall recognition time is dependent on the 'density' of model points in the scene. Although, in the worst case, it is $O(n^3)$, in most cases, especially when the number of models in a scene is small, the recognition will be much faster. To illustrate the point let us consider the extreme and simple case where all the n *interest points* in the scene belong to one object. If the object belongs to the model data-base, it will be recognized in the first trial, hence the overall recognition time will reduce to $O(n)$. If the object does not belong to the data-base, it will be rejected after all $O(n^2)$ trials have been completed resulting in the worst case $O(n^3)$ complexity.

The method we have presented assumes no *a-priori* classification of the model and scene points to reduce the number of candidates for matching basis-pairs. In this case its worst case complexity is $O(n^3)$. If some classification is available, it can be easily incorporated into this method by concentrating only on some special basis-pairs (see [11] for a method which utilizes concavity entrances of object boundary curves).

The method we have presented is parallel in a straightforward manner. It has few serial steps as can be seen from the diagram of Fig. 1. Its major advantage is the independence between the model processing stage (hash-table preparation) and the actual recognition stage.

3. General Framework

The algorithm that we have illustrated in the previous section for the case of the similarity transformation applies to many other useful transformations which are encountered in object recognition problems. We will state some of the results in this section. The only difference from one transformation to another is the number of features (points) that have to be taken as a basis for the coordinate frame. This, of course, affects the complexity of the algorithm in the different cases. Our algorithm can be summarized in the following steps (see Fig. 1):

Preprocessing of model objects

For each model object :

A) Extract a set of *interest features* (points, lines, or other suitable features) from the model object. Denote their number by m.

B) Determine the minimal number of features which can serve as a basis for a coordinate frame, allowing expression of all other features by *transformation invariant* coordinates. The number of these features depends on the dimension of the object space and on the specific transformation. Denote their number by k. Now, for each k-tuple of model features compute the coordinates of all the other model features according to this basis k-tuple and hash these coordinates into a table which stores all the pairs *(model, basis k-tuple)* for every coordinate.

Recognition of familiar objects in the scene

Given an image of a scene:

A) Extract its *interest features*. (Let their number be n.)

B) Choose a basis k-tuple and compute the coordinates of the other features in this basis.

C) For each such coordinate check the appropriate entrance in the hash-table and vote for the pairs *(model, basis k-tuple)*, appearing there.

D) If a certain pair scored a large number of votes, conjecture that its model and basis k-tuple correspond to the one chosen in the scene.

E) For a candidate k-tuple compute the transformation giving the correspondence between it and the scene k-tuple, compute correspondences of additional *interest features* that this transformation induces, and find the best transformation (say, in least squares sense) which produces all these correspondences.

F) Verify the transformed model edges against the scene edges. If the verification fails, go back to Step B and choose another basis k-tuple in the scene.

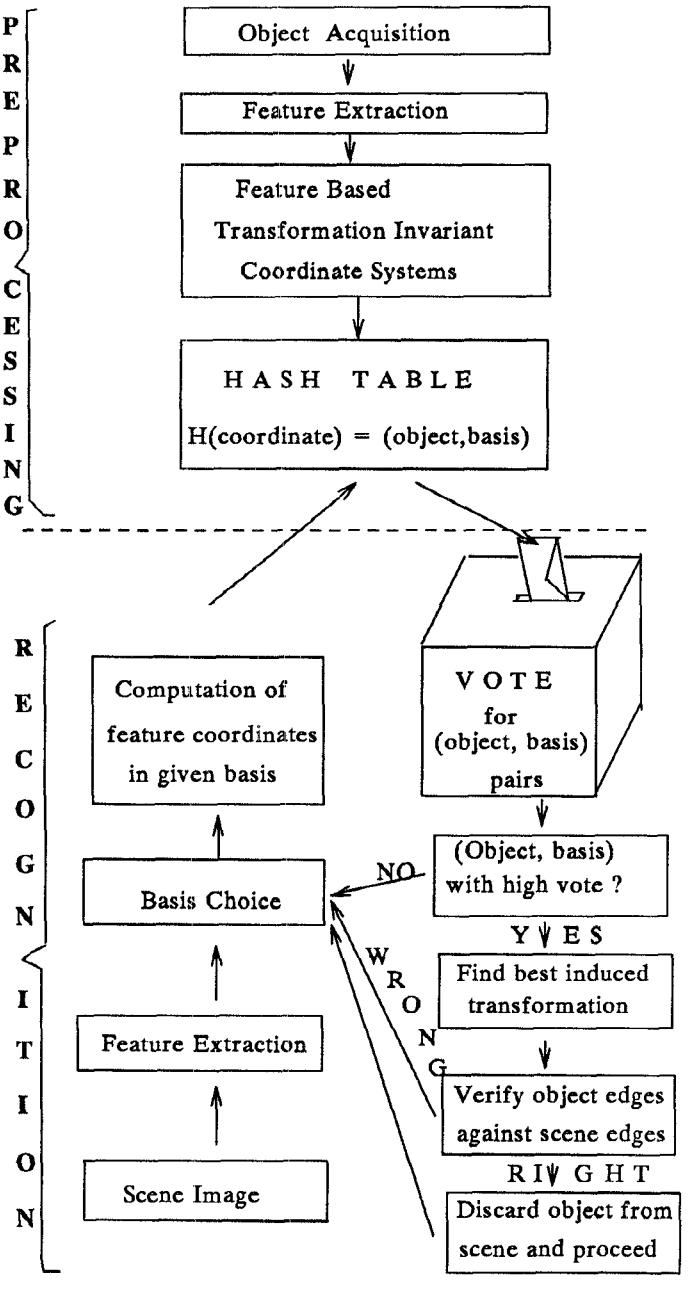


Figure 1 : The general scheme of the object recognition algorithm.

Assuming that m and n are of the same order of magnitude, the time complexity of the preprocessing stage is $O(n^{k+1})$, and the worst case time complexity of the recognition stage is

also $O(n^{k+1})$. Thus, $O(n^{k+1})$ is the worst case complexity of the algorithm. One should mention however, that in most cases the recognition stage will be much faster, and this is the actual time which concerns us, since the preprocessing stage has to be executed only once per data-base and can be done off-line.

In the following subsections we give a number of examples in which this general paradigm applies. Almost in all the cases we discuss point matching. Use of other features, such as lines, can be understood by analogy.

3.1. Translation in 2-D and 3-D

The following discussion applies equally to 2-D and 3-D. Given that the only transformation which a body can undergo is a translation, our technique applies with a *one point* basis. This point may be viewed as the origin of the coordinate frame. The worst case complexity of the algorithm in this case is $O(n^2)$. (In the 3-D case we assume knowledge of 3-D data of the scene.)

3.2. Translation and rotation in 2-D and 3-D, and similarity

To determine translation and rotation in 2-D we need a two point basis. Since this is also the case for the similarity transformation (see Sec. 2) we get the same complexity of performance. Hence, the more general similarity problem is solved at the same cost as the rigid motion problem.

Assuming range data the 3-D rigid motion problem can be solved by finding correspondence of coordinate frames, based on three point bases. The three points define the (x,y) plane and the unit length, and the normal to that plane defines the z-axis. The worst case complexity of our algorithm in this case is $O(n^4)$. This is also the solution for the similarity problem in 3-D.

3.3. Affine transformation

The case of the 2-D affine transformation has been extensively studied in [7] and [11]. This case is especially important, since the viewing transformation from 3-D to 2-D can be quite successfully approximated by an affine transformation with a vanishing determinant, and, constrained to a plane, it becomes a non-singular 2-D affine transformation. There a basis of three points is required. Consequently, the worst case complexity of the algorithm is $O(n^4)$. The efficiency of the algorithm can be significantly improved using special features of the shape such as concavities (see [11]). The reader is referred to the experimental results of [7, 11] for recognition examples of partially occluded flat bodies in cluttered scenes under the assumption of affine approximation to the perspective projection.

3.4. Projective transformation

Since the viewing transformation is a central projection, we are naturally interested in the application of our method to the projective case. We consider the projective transformation between two planes. It is well known (see, for example, [12]) that a coordinate frame for the projective plane can be defined by four points in general position. Hence, our method can be applied with a four point basis, resulting in an algorithm of $O(n^5)$ worst case complexity.

4. Recognition of 3-D Objects from 2-D Images

In the previous section we have discussed applications of our method to the case where both the data of the object and the data of the scene have been given in the same dimension, either 2-D or 3-D. In this section we consider the problem of recognition of 3-D objects from 2-D photographic images. The additional problem we have to tackle here is the reduced dimension of the image space compared with the model space. Thus, we do not have a one to one transformation between both spaces, which was exploited in the examples of the previous sections. We present a number of algorithms to tackle this problem together with experimental results for one of the algorithms (section 4.3), which achieves the best performance measured by asymptotical time complexity.

4.1. Correspondence of planes

One way to bypass the different dimension problem is to establish plane correspondence. Since we are trying to match a model of a 3-D rigid body with its translated and rotated version projected to the image plane, it is enough to establish correspondence between a set of points on some planar section of the object (it does not have to be a physical planar face) and the set of their projections in the image. Once the position of a planar section of a rigid body has been established, the position of the entire 3-D body is essentially solved (see also [9]). This brings us back to the type of problems discussed in the

previous section.

Hence, in the model preprocessing stage we may consider all the planar sections of the 3-D object which contain three or more *interest points* (for practical reasons, we will have to consider only those sections which have 'enough' points for recognition), constrain the bases sets only to given planes and for each coordinate hash the triplet *(model, plane, basis)*. Then we proceed exactly as described in Section 3.4 if we are dealing with the projective case, or as in Section 3.3 if we assume the affine approximation to perspectivity. The triplets (model, plane, basis) with high votes will be candidates for correspondence of a subset of image points with a subset of model points on a certain planar object section. Of course, in such a case, consistency of several strong candidates (corresponding to different planar sections) can be verified to extend the match.

4.2. Singular affine transformation

Assume the affine approximation to the perspective transformation. This transformation can be represented by a 2×3 rank 2 matrix $\mathbf{A}$ and a 2-D vector $\mathbf{b}$, so that a 3-D vector $\mathbf{x}$ is mapped into $\mathbf{Ax} + \mathbf{b}$ in the 2-D image (see [13]). We now extend the technique of Section 3.3 to this singular case.

A set of four non coplanar points in 3-D defines a 3-D affine basis by taking one of the points as the origin and the vectors between this origin and the other three points as the unit vectors of the (oblique) coordinate system. Any 3-D point is defined by its coordinate triplet in this basis. Hence we can preprocess the model points by their 3-D coordinates recording for each such coordinate the models and basis 4-tuples for which it appeared.

In the recognition stage, we pick an ordered set of four points in general position (without collinear triplets) in the 2-D image. The three vectors corresponding to this point set will be a spanning set for the image, but they will not be linearly independent. Let $\mathbf{e}_0$, $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ be the 4 points in the image and denote by $\mathbf{v}_i$ the vector from $\mathbf{e}_0$ to $\mathbf{e}_i$ $(i=1,2,3)$. Since the vectors $\mathbf{v}_i$ are linearly dependent, there are three scalars, not all zero, α, β, γ , so that

$$\alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3 = 0 .$$

Now, given a point $\mathbf{e}$ in the image, denote by $\mathbf{v}$ the vector from $\mathbf{e}_0$ to $\mathbf{e}$. Since $\mathbf{v}_1$ and $\mathbf{v}_2$ are a basis for the image plane, $\mathbf{v}$ has a unique pair of coordinates (ξ,η) in this basis. From all the above follows that for any scalar $t\neq 0$

$$\mathbf{v}=(\xi+t\alpha)\mathbf{v}_1 + (\eta+t\beta)\mathbf{v}_2 + t\gamma\mathbf{v}_3 .$$

Hence the point $\mathbf{e}$ in the image plane votes for all the pairs *(model, basis)* which appear in the hash-table for one of the coordinates $(\xi+t\alpha,\eta+t\beta,t\gamma)$ for all $t\neq 0$. The difference between this case and the one-to-one cases discussed in the previous section is, that instead of voting for one coordinate, representing one bin in the hash-table, we have to vote for all the bins lying on a given line in the three-dimensional hash-table. The time complexity of the recognition stage is $O(n^5\times m)$, where m is the complexity of voting along a line in the hash-table.

A similar technique can be applied to extend recognition under **perspective** transformation from the case of plane correspondence to finding correspondence between a 3-D object and its 2-D image. In the projective case 5 points define a basis for the 3-D space, and 4 points define a basis for the projective plane. The analogy to the previously discussed case is obvious.

Although the technique which is described here has an unfavorable complexity compared with our other techniques, it can be significantly sped up by choosing basis points belonging only to some prominent features in the image, such as corners etc.

4.3. Establishing a viewing angle with a similarity transformation

In this case we assume again the parallel projection approximation to the perspective transformation (orthographic projection). Consider a 3-D object being viewed from an arbitrary viewpoint. Two different images of the object taken from the same viewing angle will be in a similarity correspondence (see section 2). As discussed in Section 2, a two point basis is needed for recognition under a similarity transformation. However, since the viewing angle might be arbitrary, we have to incorporate this information into the preprocessing step. To allow enumeration of the viewing angle, which is continuous in nature, we tesselate the viewing sphere to obtain a discrete set of viewing angles (see, for example [14]).

In the preprocessing stage of the algorithm we hash the model point coordinates which are computed according to the different bases and viewing angles. Specifically, for any given viewing angle, we compute the coordinates of the model points in all possible two-point bases. For each such coordinate, we record in the hash table the triplet *(model, basis, angle)*. Since not all of the model *interest points* are visible from any given viewing angle, one has to record table entries only for the visible ones. (Note that for a given two-point basis, the coordinates of the other model points might vary as the viewing angle changes.)

In the recognition step we pick an ordered pair of scene points as a basis, compute the coordinates of all other scene points in this basis, and vote for the triplets *(model, basis, angle)* for which these coordinates appeared. The two point basis of the triplet *(model, basis, angle)* with the highest score is assumed to correspond to the basis pair which was chosen in the scene under the proposed viewing angle. This match will then be verified against the scene (see Fig. 1).

Assuming m model points and n scene points, the complexity of both the preprocessing and recognition steps is of the same order of magnitude as in the 2-D case (see Section 2), namely $O(m^3)$ and $O(n^3)$ respectively. This follows from the fact that the discretization of the viewing angle introduces only a **constant** factor since the tesselation of the viewing sphere is independent of the scene. Actually, this tesselation can be quite coarse. Hence, the complexity of this method is quite favorable compared with other schemes.

Implementation

We chose to implement this method because of its computational efficiency. In our experiments we used CAD-CAM object models in attempt to recognize their instances in composite occluded scenes (see Fig.'s 2-5). The models that we used ('cars' and a 'crane') are polyhedral. Our methods, however, do not require polyhedral model objects. Pairs of edge endpoints were chosen to be (natural) bases (each having two possible orientations). However, one can choose any other type of *interest points*, e.g. intersection points of edges. The viewing sphere was tesselated using square patches with sides of 10 degrees each (in spherical coordinates). We found this tesselation to be sufficiently accurate.

There is a dependence between the size of the solid angle bins and the hash-table coordinate quantization, since coarser tesselation introduces more noise into the coordinate measurement. If one enlarges the tesselation bins, the hash-table bins should be enlarged accordingly to match the granularity of the tesselation. Notice, that if one is interested only in correct recognition, there is usually no need for a fine tesselation. The viewing angle which scored high in the recognition stage can later be used as an initial estimate in an iterative procedure to obtain higher localization accuracy. This procedures typically rely on extending the initial matches, and minimizing the localization error in the least-squares sense (see, for example [3,7]). The above mentioned procedures enable us to use coarse tesselation of the viewing sphere resulting in reduction of the amount of storage needed for the preprocessing step.

The recognition stage of the algorithm starts with low-level processing of the scene image. Edges are extracted using the Laplacian of Gaussian edge detection operator. The zero-crossing points having higher magnitudes are linked into curves. Since our experimental images are of (almost) polyhedral objects, line segments were extracted from the curves. Relatively short segments were eliminated, since they usually correspond to image noise and carry little information (see Fig.'s 2b - 5b). This procedure was done using the software of the SCERPO system ([3]).

The actual recognition procedure has been executed on the processed images. We pick a pair of points in the scene. As in the model preprocessing stage, endpoints of straight line segments were taken. (To obtain higher recognition accuracy one can refine the position of the basis points by using endpoint proximity and edge collinearity.) For the triplets *(model, basis, angle)* that scored highest the appropriate transformation between the model and the scene was computed and verified. Usually, we got relatively high number of votes for the correct solutions. Because of the somewhat fine tesselation of the viewing sphere that was used in the preprocessing phase, we got clusters of high scoring candidates all of which had the same *(model, basis)*, but slightly different viewing angles. (The highest score is usually contributed to the candidate having the viewing angle which is the closet to the correct one.) This clustering can further assist in determining the correct

(model, basis) that matches the image basis together with the correct viewing angle.

Fig.'s 2c and 2d depict recognition examples of the car model of Fig. 2a in an unoccluded scene. In each case the transformation was computed using different image bases. One can observe slight variations in the positioning of the model due to this fact. Although in these examples we used very limited localization improvement procedures, one can see that the initial matches are already quite accurate. In Fig.'s 3-5 we show examples of recognition in occluded composite scenes. The scenes show different instances of the 'crane' and the 'short car' (observe that the 'cars' here are different from the one of Fig. 2). As one can see in Fig.'s 3c, 4c, and 5b, the localization of the image line segment endpoints is quite noisy. Nontheless, the recognition algorithm was still able to find the correct matches. In the future we plan to refine the scene's interest point extraction procedure to achieve more robustness and better accuracy. We also plan to incorporate the line equations of prominent edges as intrinsic features (points will be used as well).

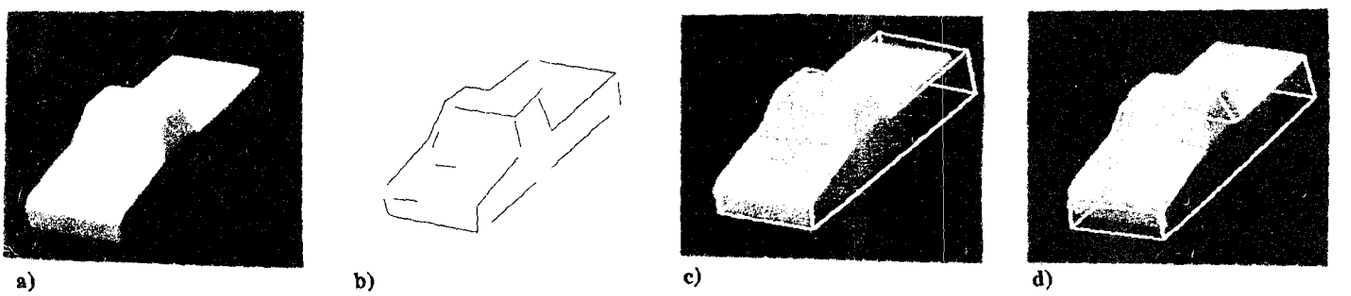


Figure 2 : a) a gray scale image of an unoccluded 'car'; b) the remaining image edges after the 'low-level' processing; c) recognition ; d) recognition using a different 'basis' in the image.

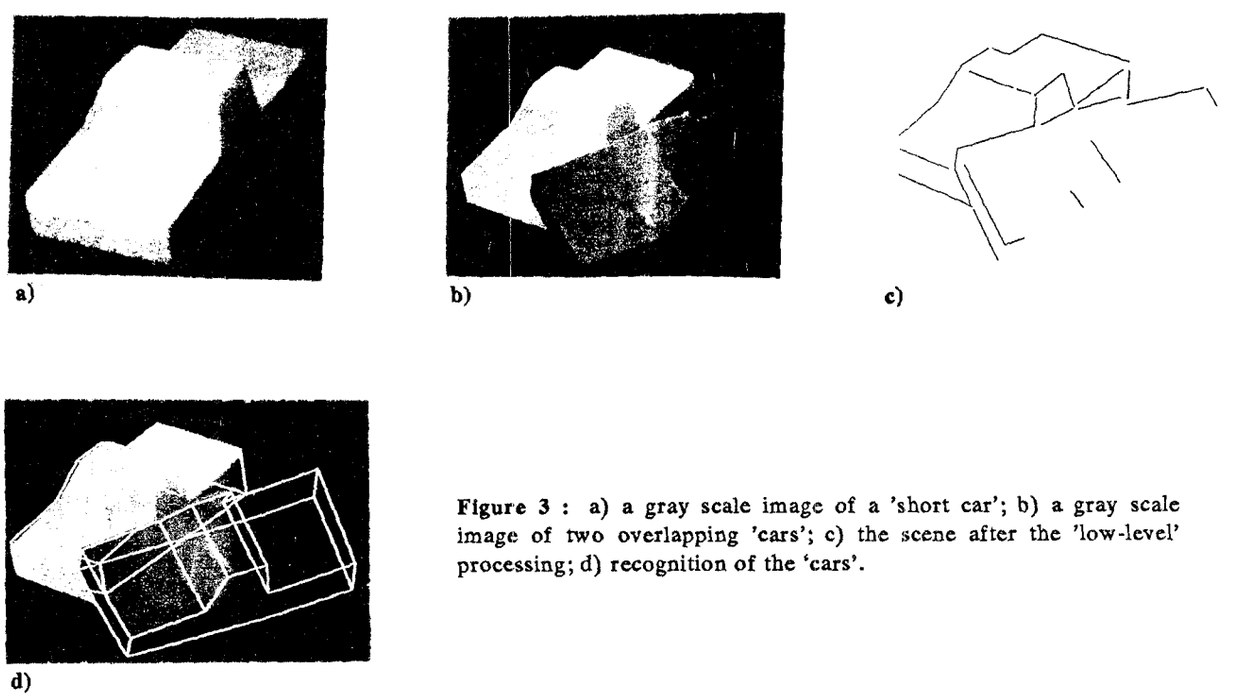
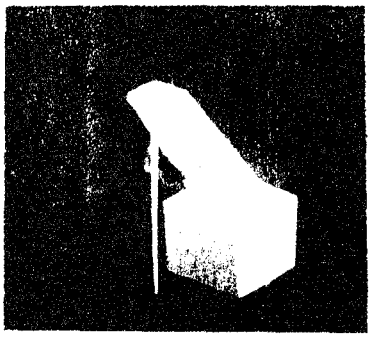
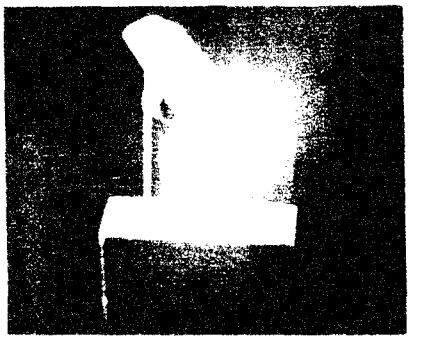


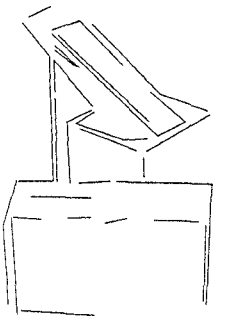
Figure 3 : a) a gray scale image of a 'short car'; b) a gray scale image of two overlapping 'cars'; c) the scene after the 'low-level' processing; d) recognition of the 'cars'.



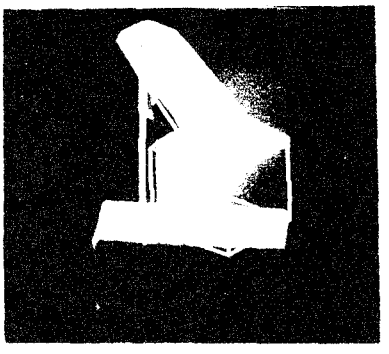
a)



b)

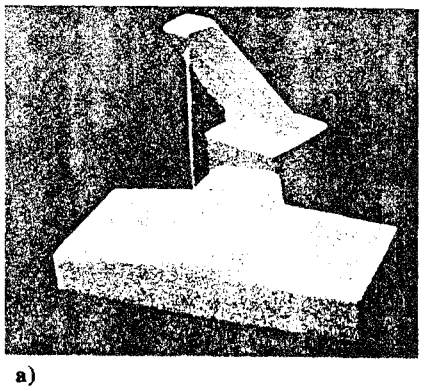


c)

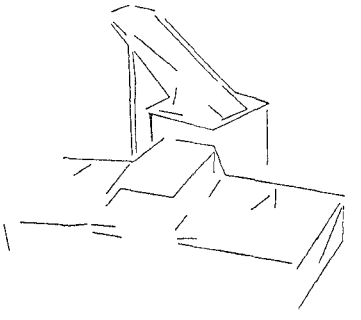


d)

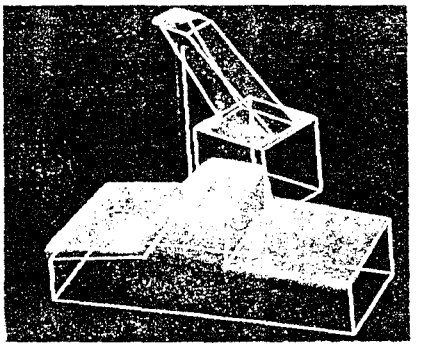
Figure 4 : a) a gray scale image of a 'crane'; b) a gray scale image of an occluded 'crane'; c) the scene after the 'low-level' processing; d) recognition of the 'crane'.



a)



b)



c)

Figure 5 : a) a gray scale image of a 'crane and car' scene; b) the scene after the 'low-level' processing; c) recognition of the 'car' and the 'crane'.

5. Recognition of Polyhedral Objects

In the previous sections we discussed the *Geometric Hashing* method for point sets. This was done mainly to facilitate the exposition. The method, however, applies in more general situations. In particular its extension to lines in 2-D and 3-D and to surfaces in 3-D is straightforward. To illustrate the application of the method to lines and surfaces, we discuss the recognition of polygonal objects in 2-D under the similarity transformation, and recognition of polyhedral 3-D scenes.

5.1. Polygonal Objects

Consider the problem of polygonal object recognition in 2-D scenes under orthographic projection. This problem attracted considerable attention and has been tackled by numerous methods. In [15] an iterative hypotheses generation and verification technique was applied, matching privileged segments in the model to segments in the scene. In [16] an interpretation tree of possible scene point and model face assignments is built and efficiently searched using local constraints. The worst case complexity of the method is, however, exponential. In [17] a Hough transform technique is applied.

Consider the application of the *Geometric Hashing* technique to the above mentioned problem. Since we are faced with a 2-D similarity transformation, the described procedure is analogous to the one described in Section 2. Here, however, by taking a single polygon edge as the unit vector of the $x-axis$, we uniquely define a 2-D orthogonal coordinate frame (see Fig. 6).

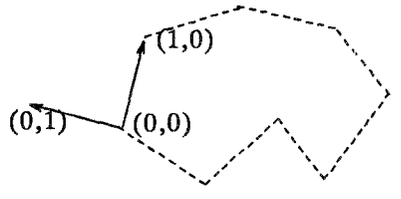


Figure 6 : A polygon with one edge chosen as the unit $x-axis$ vector, defining a unique orthogonal coordinate frame.

Hence, in the preprocessing of a polygon we choose an edge and express all other edges in the coordinate system based on this edge. In the hash-table the pair *(model, basis edge)* is recorded. The time complexity of this preprocessing step is $O(n^2)$ where n is the number of edges on a polygon. The recognition stage has worst case time complexity $O(n^2)$ as well, since it is enough to choose an edge in the scene, express all other edges in its coordinates and vote. So, our technique gives a quadratic worst case algorithm for the problem.

5.2. Polyhedral scenes

Now, consider the recognition of polyhedral objects from range or tactile data in 3-D using their surface plane information (see also [16,18]). Here a basis is defined by an edge of a polyhedron and the direction of a normal to one of the surfaces, having this edge as a boundary. As in the 2-D case a quadratic (in the number of edges) worst case algorithm solves a similarity transformation. If only surface normal information is used, two surface normals define a 3-D basis (together with their cross product). Although the representation of polyhedra by their surface normals is not unique, the algorithm can be applied to prune most of the wrong matching candidates, remaining with few candidates. Then, direct edge verification can be applied. Again the big advantage of our method is its efficiency and ability to deal with partial occlusion.

6. Comparison with Other Methods

In this section we compare the *geometric hashing* with the *alignment* technique and with the *generalized Hough Transform*. We have two reasons to compare with these two techniques. The first one is the relative success of these techniques in recognition of occluded objects. The second one is to understand the relative merits and disadvantages of the three techniques, since they share some common ingredients.

6.1. Alignment

Recently considerable work was done using the, so called, *alignment* method ([6,9]). This method utilizes minimal sets of features which suffice to establish a unique transformation between a model and its alleged instance in the scene. Such minimal sets on the model are matched against sets of features in the scene. For each match a corresponding transformation is computed, and the set of model edges is transformed to the scene to verify the 'candidate' transformation. The time complexity of this approach is $O(N\times n^k m^k \times t)$, where k is the minimal number of features needed to establish the unique transformation, $O(m)$ is the order of

magnitude of model features, n the total number of features in the scene, N the number of models to be checked, and $O(t)$ the verification time for one model, which can usually be assumed to be of order m.

The method presented in this paper and the *alignment* method usually apply the same geometrical techniques to compute the 'candidate' transformations between a model and a scene. For example the affine matching method of [6] uses alignments of triplets of points, while the method of [7] which is mentioned in Section 3.3 uses three point bases correspondence. The basic difference between the methods is in their algorithmic approach. While in the *alignment* method an exhaustive enumeration is applied over all the possible pairings of minimal sets of model and image features (unless there are some special groupings), in the *geometric hashing* the recognition stage is significantly sped up by using the previously prepared hash-table which encodes the relevant information about the model objects. Another major advantage of the *geometric hashing* algorithm is its ability to process all the model objects simultaneously. By picking a 'correct' basis in the scene we discover both the model it belongs to and the appropriate transformation between this model and the scene, while in the *alignment* method one has to process the 'candidate' models sequentially. For example, in the affine matching case, which was mentioned before, the worst case complexity of recognition is of order $N \times n^7$ for the *alignment* method, and $O(n^4)$ for *geometric hashing*. Here we assume that the number of model points, scene points and the verification time are of order n and the number of models in the data-base is N.

On the other hand when memory is a concern, it is better to use the *alignment* procedure which has almost no storage requirements, except the image and model edge data. The efficiency of the *geometric hashing* is achieved by precompilation of the model information into a hash-table using appropriate representations. It gives this method the ability to determine for a given scene minimal feature set (basis) a corresponding feature set on one of the models by considering only the other scene features which 'vote' for the correct interpretation. This 'voting' procedure requires, however, existence of few additional model features in the scene image except the basis. In the extreme case when such additional features do not exist, the algorithm will try first interpretations which scored high but are incorrect. Since these interpretations will be rejected by the verification step, in its fast version the algorithm will fail to recognize the model object. Of course, in such a case one may decide to backtrack and check candidate solutions with less votes. Eventually, the correct solution will be found after an exhaustive search resulting in the same worst case complexity as the *alignment* method.

To conclude, the *geometric hashing* is considerably more efficient than the *alignment*, when the scene contains 'enough' model features for efficient recognition by voting ('enough' usually means about 6-10). It is also efficient for multiple model processing. In case the number of model features is exceptionally small (for example, only one basis appears in the scene), both methods will have the same worst case complexity.

6.2. Generalized Hough Transform

Since the *geometric hashing* method involves a voting procedure it is sometimes confused with the *generalized Hough Transform* (see [10] or [19] pp. 128-131). Since not every voting procedure (c.f. presidential primaries) is a Hough Transform, we try to explain some of the differences between the two approaches.

In the *Hough Transform* approach one usually describes the transformation between a model and the scene by a set of transformation parameters. Each such parameter has a (continuous, infinite) range of possible values which is quantized into bins. The features of the scene vote for these parameter bins if the appropriate transformation data is consistent with the feature information.

One of the differences between our scheme and the *Hough Transform* is the fact that we do not vote for some 'unpredictable' transformation parameters, but for representations (by bases) of the *a-priori* known model objects. The hash-table, encoding these representations, is known in advance (before recognition), hence we can examine the size of the bins in the table, adjust the quantization to achieve better performance and even introduce a *weighted* voting scheme (see [8,20]) so that large bins which are not informative will achieve a small weight, or, even, zero weight to preclude their time consuming processing, while votes of small bins will have bigger weights. Also in our procedure we

do not vote for a continuous range of transformation parameters the way it is done by the *Hough Transform*, but for discrete values denoting models and their representations (by basis). In our case all the allowable transformations are implicitly precompiled in the hash-table.

7. Future Research

The method that we have presented naturally applies to numerous object recognition problems. In the near future we intend to focus on a number of issues, among them:

1) Implementation of matching procedures based on synthesis of point and line information.

2) Practical implementation of a matching procedure under the true perspective transformation.

3) Viewing transformation invariant shape representation by parts.

4) Recognition of objects using parametrized models.

Acknowledgements

We are grateful to David Lowe who kindly supplied us the low-level vision software of his SCERPO system.

References

1. P.J. Besl and R.C. Jain, "Three-Dimensional Object Recognition," *ACM Computing Surveys*, vol. 17 (1), pp. 75-154, 1985.
2. R.T. Chin and C.R. Dyer, "Model-Based Recognition in Robot Vision," *ACM Computing Surveys*, vol. 18 (1), pp. 67-108, 1986.
3. D.G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.
4. A. Kalvin, E. Schonberg, J.T. Schwartz, and M. Sharir, "Two Dimensional Model Based Boundary Matching Using Footprints," *The Int. J. of Robotics Research*, vol. 5(4), pp. 38-55, 1986.
5. D.W. Thompson and J.L. Mundy, "Three-Dimensional Model Matching from an Unconstrained Viewpoint," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 208-220, Raleigh, North Carolina, 1987.
6. D.P. Huttenlocher and S. Ullman, "Object Recognition Using Alignment," *Proc. of the 1'st Int. Conf. on Computer Vision*, pp. 102-111, London, 1987.
7. Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "On recognition of 3-D objects from 2-D images," *Proceedings of the IEEE Int. Conference on Robotics and Automation*, vol. 3, pp. 1407-1413, Philadelphia, April, 1988.
8. J. Hong and H. J. Wolfson, "An Improved Model-Based Matching Method using Footprints," *Proceedings of the Int. Conf. on Pattern Recognition*, Rome, Italy, November, 1988.
9. D.P. Huttenlocher and S. Ullman, "Recognizing Solid Objects by Alignment," *Proceedings of the DARPA Image Understanding Workshop*, vol. II, pp. 1114-1122, Cambridge, Massachusetts, April, 1988.
10. D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13(2), pp. 111-122, 1981.
11. Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Object Recognition by Affine Invariant Matching," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 335-344, Ann Arbor, Michigan, June, 1988.
12. B. N. Delone and D. A. Raikov, *Analytic Geometry*, 2, Moscow, 1949. *(In Russian.)*
13. F. Klein, *Elementary Mathematics from an advanced standpoint ; Geometry*, Macmillan, New York, 1925 (Third edition). *(English translation)*.
14. C. Goad, "Special Purpose Automatic Programming for 3-D Model -Based Vision," *Proc. Image Understanding Workshop*, pp. 94-104, Arlington, Virginia, June 1983.
15. N. Ayache and O.D. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8 (1), pp. 44-54, Jan., 1986.
16. W.E.L. Grimson and T. Lozano-Perez, "Model Based Recognition and Localization from Sparse Range or Tactile Data," *The Int. J. of Robotics Research*, vol. 3 (3), pp. 3-35, 1984.
17. D. H. Ballard and D. Sabbah, "Viewer Independent Shape Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5(6), pp. 653-660, November,1983.
18. J. Little, "Determining Object Attitude from Extended Gaussian Images," *Proc. 9'th Int. Joint Conf. on AI*, pp. 960-963, Los Angeles, California, 1985.
19. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, 1982.
20. E. Kishon and H. Wolfson, "3-D Curve Matching," *Proceedings of the AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, pp. 250-261, St. Charles, Illinois, October, 1987.