Model-Based Object Recognition by Geometric Hashing

Haim J. Wolfson

Computer Science Dept., School of Math., Tel Aviv University, wolfson@math.tau.ac.il
and
Robotics Lab, Courant Inst. of Math., New York University, wolfson@acf8.nyu.edu

Abstract

The *Geometric Hashing* paradigm for model-based recognition of objects in cluttered scenes is discussed. This paradigm enables a unified approach to rigid object recognition under different viewing transformation assumptions both for 2-D and 3-D objects obtained by different sensors, e.g. vision, range, tactile. It is based on an intensive off-line model preprocessing (learning) stage, where model information is indexed into a hash-table using minimal, transformation invariant features. This enables the on-line recognition algorithm to be particularly efficient. The algorithm is straightforwardly parallelizable. Initial experimentation of the technique has led to successful recognition of both 2-D and 3-D objects in cluttered scenes from an arbitrary viewpoint. We, also, compare the *Geometric Hashing* with the *Hough Transform* and the *alignment* techniques. Extensions of the basic paradigm which reduce its worst case recognition complexity are discussed.

1 Introduction

Object recognition is a major task in computer vision. The ability to recognize objects in cluttered scenes is essential to the functionality of a flexible robotic system. In most practical industrial applications we may assume that the objects to be recognized are known in advance. This model-based approach proved to be promising, and the best currently known object recognition systems are model-based (for comprehensive surveys see [4, 6]). Ideally, such systems should be able to deal with substantial occlusion, and recognize rigid, articulated and even deformable objects under different viewing transformations. The recognition algorithms should be efficient enough to enable on-line recognition of objects belonging to large libraries.

In order to discuss object recognition algorithms one has to give some mathematical description of an object. Let us assume that some *interest features* can be extracted from the images, so that both the model-objects and the observed (cluttered) scene can be represented by sets of these *interest features*. In the most general (and least informative) case the *interest features* will be just points or lines. In such a situation, the recognition of a partially occluded object in a scene amounts to the discovery of a match between a subset of the scene *interest features* and a subset of the *interest features* of some model object. The discovered match should be consistent with some (viewing) transformation. Given m *interest features* on a model and n *interest features* in the scene, there are $O(n^m)$ ways to match the model features to the scene features. Since such an exponential complexity is unacceptable for object recognition algorithms, various approaches were suggested to prune the space of possible matches. Some of them employ efficient tree search techniques, where the pruning is based on geometric constraints ([8]). However, for recognition of partially occluded objects, the search still remains exponential in the number of scene features (see [9]).

To overcome this exponential complexity, one may observe, that a transformation of a **rigid** object is usually defined by the transformation of a small number of the object's features (points). This geometric observation is at the core of the, so called, Hough Transform or *pose clustering* ([2, 27, 22]), the *alignment* ([13, 14]), and the *Geometric Hashing* ([19, 18, 20]) techniques.

In this paper we address the recognition of objects in cluttered scenes using the *Geometric Hashing* technique. In its present form, the *Geometric Hashing* paradigm enables a unified approach to rigid object recognition under different viewing transformation assumptions both for 2-D and 3-D objects, obtained by different sensors, e.g. vision, range, tactile ([20]). This technique is especially suitable for dealing with

partially occluded objects in cluttered scenes due to its representation scheme which encodes geometric constraints between local features. The *Geometric Hashing* paradigm is based on an intensive off-line model preprocessing (learning) stage, where model information is indexed into a hash-table using minimal, transformation invariant features. This enables the on-line recognition algorithm to be particularly efficient. The recognition time depends directly on the complexity of the scene to be recognized, and increases only sub-linearly with the number of model-objects in the data base. It is also straightforwardly parallelizable, and can be quite easily implemented on a fast special purpose hardware. Initial experimentation of the technique has led to successful recognition of both 2-D and 3-D objects in cluttered scenes from an arbitrary viewpoint ([19, 18, 20]).

In this paper we first survey the basic ideas of the *Geometric Hashing* technique. Then, we compare it with two other efficient model based object recognition techniques, the, Hough Transform paradigm ([2, 27, 22]), and the *alignment* technique ([13, 14]). We discuss the advantages and deficiences of these different techniques compared to each other. Finally, we present some extensions of the basic technique, which can further reduce its time complexity.

2 The *Geometric Hashing* Paradigm

Recently, the, so called, *Geometric Hashing* technique for model based object recognition was introduced by Lamdan, Schwartz and Wolfson. Efficient algorithms were developed for recognition of flat rigid objects assuming the affine approximation of the perspective transformation ([19, 18]), and the technique was also extended to the recognition of arbitrary rigid 3-D objects from single 2-D images ([20]).

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.

The *Geometric Hashing* paradigm presents 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.

In the sequel we present the *Geometric Hashing* method mainly for point matching. This is done for two reasons. First, the 'point matching' case allows a succinct mathematical representation of the basic method, without getting overburdened with extraneous details. Second, since a single point is a 'least informative feature', we present the method in the most difficult case. Clearly, better results can be achieved by using more stable and more informative features such as lines and various groupings (see Section 4.1).

2.1 Recognition under the Affine Transformation

To illustrate the method we first discuss flat object recognition under the affine transformation (rotation, translation, scale, and shear). It is well known that the perspective projection is well approximated by a parallel projection with a scale factor (see [16, 12, 23, 28]). This approximation is especially suitable for flat bodies, which are relatively far from the camera. Hence, we may assume, that two different images of the same flat object are in an affine 2-D correspondence, namely, there is a **non singular** 2×2 matrix $\mathbf{A}$ and a 2-D (translation) vector $\mathbf{b}$, such that each point $\mathbf{x}$ in the first image is translated to the corresponding point $\mathbf{Ax} + \mathbf{b}$ in the second image.

Let us assume that the model objects and the scene are described by sets of *interest points*, which are invariant under the affine transformation. The choice of the interest operator is not essential to our method. Corners, points of sharp concavities and convexities, or points of inflection are suitable candidates (see [19]). 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 the following subset isometry problem :

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

As was mentioned before we have to address two major interrelated problems: *representation* and efficient *matching.*

2.1.1 Representation of geometric constraints

Our goal is to represent a set of planar points belonging to a rigid body by few intrinsic parameters. This representation should efficiently encode the geometric constraints of a rigid body, be affine invariant, and enable handling of occlusion.

Assume that we have an arbitrary set of m points belonging to a rigid body. An affine transformation of a planar rigid body is uniquely defined by the transformation of three ordered non-collinear points ([16, 29]). Moreover, one can pick any triplet of non-collinear points in the set and represent all the other points using this triplet. Specifically, let $\mathbf{e}_{00}, \mathbf{e}_{10}, \mathbf{e}_{01}$ be an ordered triplet of such points. Then, any point $\mathbf{v}$ in the plane can be represented in this **affine basis**, namely, there is a pair of scalars (α, β), such that $\mathbf{v} = \alpha(\mathbf{e}_{10} - \mathbf{e}_{00}) + \beta(\mathbf{e}_{01} - \mathbf{e}_{00}) + \mathbf{e}_{00}$.

The above coordinates are invariant under an affine transformation. Accordingly, we will represent the m points of our set by their coordinates in the affine basis triplet $(\mathbf{e}_{00}, \mathbf{e}_{10}, \mathbf{e}_{01})$. Naturally, the coordinates of the basis points are $(0,0)$, $(1,0)$ and $(0,1)$, respectively.

This representation allows comparison of occluded objects, since the point coordinates of an occluded object in the scene will have a partial overlap with the coordinates of the stored model , if both are represented in a coordinate frame which is based on the same triplet of points. This dependence of the representation on a specific basis triplet may, however, preclude recognition when at least one of the basis points is occluded. Hence we represent the object points by their coordinates in **all** possible affine basis triplets. More specifically, given a model object, the following preprocessing is applied to each model object:

a) Extract the objects *interest points.* (Assume that m *interest points* were found.)

b) For each ordered non-collinear triplet of model points (affine basis) compute the coordinates of all other $m-3$ model points in the affine coordinate frame defined by the basis triplet. Use each such coordinate (after a proper quantization) as an address (index) to a hash-table, and record in the table the pair *(model, basis)*, namely, the model and the affine basis at which the coordinate was obtained.

The complexity of this preprocessing step is of order m^4 per model. Each of the m points is represented in m^3 different bases. The major advantage of this somewhat redundant representation is its ability to allow efficient recognition of objects in an occluded scene.

Note, that the preprocessing step is done without any knowledge of the scene to be recognized. Hence, it can be executed off-line, so that its execution time does not add to the actual recognition time. New models added to the data-base can be processed independently without recomputing the hash-table.

The hash table preparation stage may be viewed as a *learning* stage of the algorithm. In this stage relevant information of the models is memorized in its various representations. The triplet of points, which serves as an (affine) basis, for a given representation may be viewed as a (geometric) *focus of attention.* The hash table serves as a memory of the model-objects under different foci of attention.

2.1.2 Matching

The matching stage of the algorithm uses the hash table, prepared in the representation (learning) stage. Given a scene of *interest points* one chooses an affine basis (focus of attention) in the scene, and tries to match the coordinates of the scene points to those memorized in the hash table. Specifically, given an image of a scene with partially occluded objects one does the following :

a) Extract its *interest points.* (Assume we have n such points.)

b) Choose an arbitrary ordered triplet of non-collinear points in the scene and compute the coordinates of the scene points referring to this triplet as an affine basis. (If appropriate, one might, of course try some 'intelligent' choice of the affine basis, rather then chosing it at random.)

c) For each such coordinate check the appropriate entry in the hash-table, and for every pair *(model, affine basis)*, which appears there, tally a vote for the model and the affine basis as corresponding to the pair which was chosen in the scene. (The accumulator of the votes will have $\sum_{i=1}^{N} m_i^3$ entries, where N is the number of models and m_i is the number of *interest points* on the i'th model.)

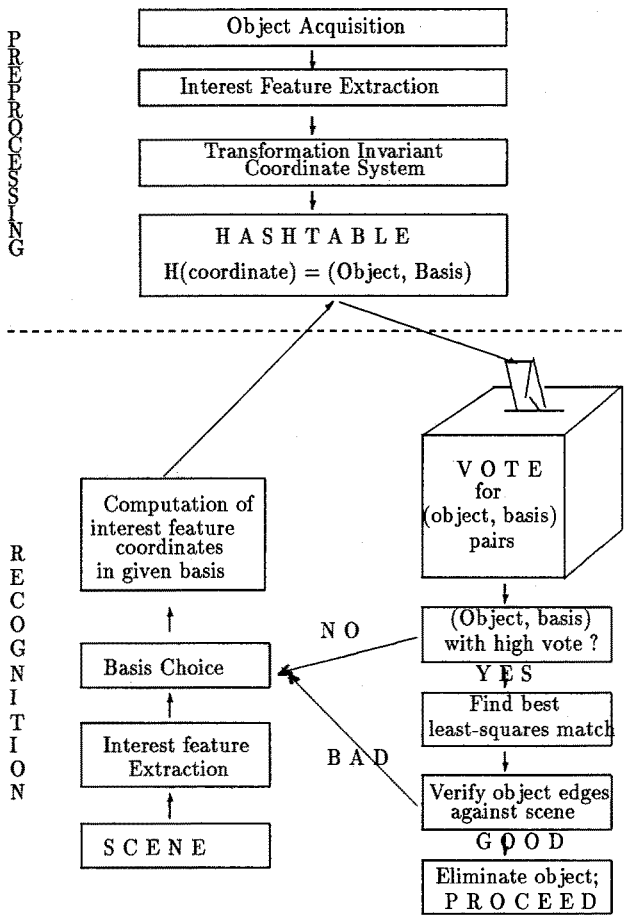

Figure 1: The General Scheme of the Algorithm

d) If a certain pair *(model, affine basis)* scores a large number of votes, conjecture that this pair corresponds to the one chosen in the scene. The affine transformation between the coordinate frames based on these triplets is assumed to be the transformation between the appropriate model and the scene.

e) Consider all the model-image point pairs which voted for the affine transformation of Step (d), and find the affine transformation giving the best least-squares match between all these corresponding point pairs (see [19]). Since the computation of this transformation is based on more than 3 point pairs, it will, hopefully, be more accurate.

f) Transform all the edges of the model according to the affine transformation of Step (e) and **verify** them versus the scene edges. If the verification fails, go back to Step (b) and begin the procedure for a different image affine basis. This final verification is done not only for the *interest points* but for all the relevant model edges. (See Fig. 1 for a general scheme of our procedure.)

It is important to mention that in general we do not expect the voting scheme to give only one candidate solution (see [21]). Its goal is to reduce significantly the number of candidates for the verification step (f), which might be quite tedious and time consuming (see [10]).

Since the voting is done simultaneously for all models and all possible bases on a model, **for the algorithm to be successful it is enough to pick three points in the scene, belonging to some model.** In such a case the model with the appropriate affine basis gets a high score in the voting procedure.

The voting process, per basis-triplet, 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.

The presented method assumes no *a-priori* classification of the model and scene points to reduce the number of candidates for matching basis-pairs. In this difficult case its worst case complexity is $O(n^4)$. If some classification or perceptual grouping of features is available, it can be incorporated into this method by concentrating only on some special basis-triplets ([18]). We further discuss this issue in Section 4.

The method is parallel in a straightforward manner. It has few serial steps as can be seen from the diagram of Fig. 1. A prototype of the affine matching algorithm has been implemented on the highly parallel Connection Machine by Bourdon and Medioni ([5]).

Because of the duality between lines and points in the affine plane, all the algorithms, that were developed for point matching, apply directly to line matching. Namely, given a line $a\mathbf{x} + b\mathbf{y} = 1$, one may view it as a point $(\mathbf{a}, \mathbf{b})$ in the dual space. Since line extraction is, usually, more reliable than point extraction, one may benefit from this dual approach.

2.2 Error Analysis

So far, we have discussed our algorithm in its 'noiseless' version. However, since real scene measurements are noisy, this noise will affect both the implementation of the algorithm and its performance. We give here a short description of the issues raised by noise analysis, and their handling. A more extensive noise analysis, including simulation results, is given in ([21]).

We assume that the models can be acquired under 'ideal' circumstances (from a CAD model, for example), hence the preprocessing step is noiseless. In the recognition step, image coordinates of *interest points* are measured. These coordinates are represented by $2-D$ vectors. One may define a norm on this $2-D$ vector space. We will usually use either the Euclidean (L_2), or the maximum coordinate (L_∞) norm. Assume that image point measurements introduce an error of at most ϵ in the given norm.

The computation of the coordinates of an *interest point* $\mathbf{v} = (v_1, v_2)$ in the affine basis $\mathbf{e}_{00}, \mathbf{e}_{10}, \mathbf{e}_{01}$ can be formulated as a solution of the linear system of 2 equations in 2 unknowns $\mathbf{Ax} = \mathbf{b}$. The two columns of the matrix $\mathbf{A}$ are the difference vectors of the basis *interest points* $\mathbf{e}_x = \mathbf{e}_{10} - \mathbf{e}_{00}$ and $\mathbf{e}_y = \mathbf{e}_{01} - \mathbf{e}_{00}$ respectively, and the free vector is $\mathbf{b} = \mathbf{v} - \mathbf{e}_{00}$. These vectors are represented in image coordinates, while the solution vector $\mathbf{x} = (\alpha, \beta)$ gives the representation of the point $\mathbf{v}$ in the affine basis $\mathbf{e}_{00}, \mathbf{e}_{10}, \mathbf{e}_{01}$ coordinates .

Taking the errors into account, one has to solve the following linear system :

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$$

where $\delta\mathbf{A}$, $\delta\mathbf{x}$, and $\delta\mathbf{b}$ are the errors of the matrix $\mathbf{A}$ and the vectors $\mathbf{x}$ and $\mathbf{b}$ respectively. By the nature of our point measurements, we may assume that the absolute values of entries of the matrix $\delta\mathbf{A}$ and the vector $\delta\mathbf{b}$ are less than some given measurement error ϵ. The stability and accuracy of solutions to systems of linear equations is a well investigated problem of Numerical Analysis. An extensive treatment of such 'approximate' linear systems is given, for example, in [17]. A good estimate of the maximal error in the k'th coordinate of $\mathbf{x}$ ([17], p. 45) is $Delta\mathbf{x}_k \leq (1 + x)c_k\epsilon$, $(k = 1, 2)$, where $x = |\mathbf{x}_1| + |\mathbf{x}_2|$, is the sum of the absolute values of the unknowns, and $c_k = |\mathbf{c}_{k1}| + |\mathbf{c}_{k2}|$, is the sum of the absolute values of the k'th row in the inverse matrix $\mathbf{C} = \mathbf{A}^{-1}$. This inequality gives an estimate of the maximal error which can be introduced by the image measurement noise into the coordinates of the hash-table address $\mathbf{x}$. Hence, the voting procedure reflects this noise. Namely, for an address $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, all the bins with addresses $(\mathbf{x}_1 \pm \Delta\mathbf{x}_1, \mathbf{x}_2 \pm \Delta\mathbf{x}_2)$ participate in the voting. This ensures that votes for a correct model basis are not missed due to noise. In practice, tighter bounds, usually, apply ([17]). Note, that the amount of error ϵ is defined by the known imaging process, hence the worst case $\delta\mathbf{A}$ and $\delta\mathbf{b}$ can be computed in advance. Since for a given basis triplet and an additional image point both $\mathbf{C}$ and x can be computed, we can evaluate the relative merit of voting for a given coordinate, and eliminate those (unstable) coordinates which are going to introduce excessive noise. It should be noted that if a certain basis-triplet belonging to some model did not get enough votes, we still have a chance to recover this model from another basis-triplet.

Since, the appropriate voting bins for each address can be evaluated in advance, we do not expect a correct basis triplet to achieve less votes than the corresponding number of unoccluded model points (except the eliminated 'unstable' votes). There still remains the possibility of a 'random' basis-triplet achieving a large number of votes. Such a 'wrong' candidate will be discovered by two verification procedures that are incorporated in the algorithm (see (e) and (f) of Section 2.1.2). Although 'wrong' candidates will be

discovered in the verification steps and discarded, we would still like to show that the probability of a 'random' configuration to get a high vote is small. Such a discussion and simulation results are presented in [21].

2.3 General Framework

The algorithm that was illustrated in the previous section is suitable not only for the affine 2-D case. It represents a unified approach which 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 (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.

The following list gives a number of examples in which this general paradigm applies. Almost in all the cases we will discuss point matching. Use of other features, such as lines, can be understood by analogy. We discuss first recognition of 2-D objects from 2-D data.

1) *Translation in 2-D* - the technique applies with a *one point* basis. This point may be viewed as the origin of the coordinate frame. Hence, the worst case complexity of the recognition step is $O(n^2)$.

2) *Translation and rotation in 2-D* - a two point basis can be used, however, one point with a direction (obtained, say, from an edge segment) has enough information for a unique definition of a basis.

3) *Translation, rotation and scale in 2-D* - the key observation here is that since the similarity transformation is orthogonal, two points (or, a line segment) are enough to form a basis which spans the 2-D plane. (The first point is assigned coordinates $(0,0)$ and the second $(1,0)$. The third basis point $(0,1)$ is uniquely defined by these two points.) Hence, the *Geometric Hashing* technique can be applied with a two point basis, resulting in a recognition stage of complexity $O(n^3)$.

4) *Affine transformation in 2-D* - was discussed in Section 2.1. See [19, 18] for some recognition results.

5) *Projective transformation in 2-D* - a 4 point basis is needed to recover a projective transformation between two planes (see, for example, [7]).

In case 3-D data (such as range or stereo) of the objects is available the recognition of 3-D objects from 3-D images has to be considered. Development of techniques for this case is especially important in an industrial environment, where 3-D data can be readily obtained and used.

6) *Translation in 3-D* - exactly as the 2-D case. One point basis will suffice.

7) *Translation and rotation in 3-D* - this is the *interesting* case representing a rigid motion. A two non-collinear line basis is enough.

8) *Translation, rotation and scale in 3-D* - a two non-collinear and non-planar line basis (or a point and a line) gives a minimal set.

In the previous discussion we considered cases 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. However, in the *recognition of 3-D objects from single 2-D images* we have the additional problem of the reduced dimension in the image space compared with the model space. A number of methods have been developed to tackle this problem by the *Geometric Hashing* technique (see [20]). One of this methods, based on the recovery of the correct viewing angle and the appropriate similarity transformation, has been implemented and successfully tested (see [20]).

Specifically, in the model learning stage the viewing sphere is tesselated into (few hundred) discrete viewing angles. For a fixed angle, the viewing transformation is well approximated by the orthographic projection. Hence, two different images of the same object taken from the same viewing angle are in a 2-D similarity correspondence (case (3)). The usual model representation scheme is applied here, except for the additional information of the viewing angle, which is also memorized in the hash-table. Namely, given a two point basis, the coordinates of all other *interest points* are computed and the information *(model, basis, viewing angle)* is stored in the hash-table at the addresses defined by the appropriate coordinates. In the recognition stage, the usual procedure for the 2-D similarity case is applied, although the votes this time are accumulated for the triplets *(model, basis, viewing angle)* (see [20] for details). Since the number of possible viewing angles is fixed, the method retains the recognition complexity of $O(n^3)$ (2-D similarity).

3 Comparison with Other Methods

In this section we compare the *geometric hashing* with the *alignment* technique and with the *generalized Hough Transform*. Since all these techniques share some common ingredients it is important to understand their differences and relative advantages.

3.1 Comparison with the Hough Transform

Since the *Geometric Hashing* method involves a voting procedure it immediately invokes association with the *generalized Hough Transform* (see [2] or [3] pp. 128-131) or, so called, *pose clustering* techniques ([27, 22]). We consider in detail the differences between the *Geometric Hashing* and *pose clustering* techniques.

3.1.1 The Pose Clustering (Hough) Method

In the *pose clustering (Hough Transform)* approach, recognition of an object in a scene is achieved by finding a transformation between a model-object and the scene, which maps a large enough number of the model *interest features* into scene *interest features*. The transformation is discovered by voting for its parameters, which are consistent with hypothetical pairings of subsets of object and image *interest features*.

Let us consider the example of object recognition under the affine (rotation, translation, scaling, and shearing) transformation using the Hough paradigm. Assume as in Section 2.1 that both the models and the scene are represented by *interest points*. An affine transformation can be represented by six independent parameters. Hence, the *Hough* technique regards an affine transformation as a point in the 6-D space, each parameter describing a coordinate of this 6-D point. Although the parameters may achieve a continuum of values, each coordinate (parameter) axis is quantized into a finite number of values, thus creating 6-D bins (volumes) of admissible transformations.

Since an affine transformation is uniquely defined by the correspondence of triplets of points, each correspondence of a triplet of model points with a triplet of scene points provides a *candidate* affine transformation. Hence, in the *Hough* technique each triplet of model points is compared with each triplet of scene points, an affine transformation is computed for each such comparison, and votes are cast for the appropriate bins (according to some noise model) in the 6-D transformation table. For a model of m points and a scene of n points, m^3n^3 transformations are computed. If both m and n are of order n, the voting complexity is $O(n^6)$ (assuming constant time for each table access).

A well known problem of the *Hough* technique is the large size of the transformation table (6-D in the affine case). Hence, the usual practice (see, for example, [22]) is to recover the transformation parameters sequentially by first projecting the votes of the transformation table onto a small number (usually, one) of parameters, recovering this parameter, and proceeding in a similar manner for the remaining parameters. This, in turn, introduces the problem of false peaks in the collapsed parameter space.

3.1.2 Differences between the Hough technique and Geometric Hashing

At first glance there are noticeable similarities between the *Geometric Hashing* technique as described in Section 2.1 and the *pose clustering (Hough)* technique as described in the previous subsection. Both use large tables, and both use voting procedures to recover the correct transformation. In this subsection we will state the major differences between both techniques, resulting in a different complexity, and different approaches in their statistical evaluation.

One of the main differences is the independent processing of the model and scene information by *Geometric Hashing*. The model hash-table is prepared without any knowledge of the scene, due to the transformation invariant shape representation (see subsection 2.1.1). This allows the recognition to be performed on all the models and all the different encodings of models (by different bases) simultaneously, while in the *pose clustering* technique one has to check sequentially each model against the scene, and each model basis against the considered image basis, making the recognition time linearly dependent on the number of models in the data base, and cubically dependent on the number of model points. Given N models, order of m points on a model, and order of n points in the scene, the recognition time of the *Hough* technique is $O(N \times m^3 \times n^3)$, while the worst case time complexity of *Geometric Hashing* is $O(n^4)$[1].

[1] Assuming constant time for each hash-table bin processing. This is a reasonable assumption, as can be seen from the remark on *weighted* voting at the end of this section.

Another major difference is in the voting procedure. In the *pose clustering* one is voting for certain *a-priori* unknown transformation parameters, hence a large table (6-D for affine transformation) is kept. In *Geometric Hashing* one is voting only for *a-priori* known model object representations (by bases). Thus, for example, an m-point model has only m^3 admissible representations (and thus possible transformations) under the affine transformation. It should be noted that in *Geometric Hashing* the equivalent of the transformation parameter table of the Hough technique is not the hash-table, but the discrete accumulator of the votes for different bases. The size of this accumulator is relatively small and well defined in advance (see (c) of section 2.1.2). Also, the voting in this accumulator is done **only for one image basis** at a time. This voting procedure compares one given image basis versus all possible model bases, looking for peaks in the compatibility score. On the other hand, in the Hough procedure one compares **all** image bases against **all** model bases, each of such pairings voting for one possible transformation. This introduces more background noise and makes it more difficult to discover the peaks.

There is no equivalent for the hash table of the *Geometric Hashing* in the *pose clustering* technique. However, one should note, a number of properties. First, the dimensionality of this table is a function of the dimensionality of the image. It is, usually, 2-dimensional for intensity images, and 3-dimensional for range data. Second, this table is prepared in advance for a given database. Hence, its bin size can be controlled as a function of the specific database. In particular, one may introduce a *weighted* voting scheme (see [15, 11]). In such a scheme bins which are not informative, namely, bins, representing coordinates with a big number of *(model, basis)* candidates, will achieve a small weight, or, even, zero weight to preclude their time consuming processing, while votes of 'small' bins will have bigger weights.

3.2 Comparison with Alignment

Recently considerable work was done using the, so called, *alignment* method ([13, 14]). As the two other methods mentioned, this method utilizes minimal sets of features which suffice to establish a unique transformation between a model and its alleged instance in a scene.

3.2.1 The Alignment

In the *alignment* technique a transformation is hypothesized based on the correspondence of minimal sets of features, and then the *candidate* correspondence is verified in regard to the other features.

Let us, again, consider the case of affine 2-D matching ([13]), and assume that both the model and the scene are represented by *interest points*. As in the *pose clustering* technique one picks a triplet of points is the scene and a triplet on the model, and computes the affine transformation between these two triplets. Given the *candidate* affine transformation, all the other model points are transformed and matched with the image points. This match should be done according to an error model analogous to the one given in Section 2.2 (see [25]). If the number of matching *interest points* is above a certain threshold, additional edge verification is invoked.

The time complexity of alignment for the affine case is $O(N \times n^3 \times m^4)$, where N, m, n are as in the previous subsection.

3.2.2 Differences between the Alignment and the Geometric Hashing

The *alignment* technique and the *geometric hashing* are based on the same geometric principles. If they employ the same error model and *candidate* transformation acceptance thresholds, both techniques should eventually accept (or reject) the same *candidate* transformations. 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, 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 the voting procedure of *geometric hashing* discovers 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 and triplet bases 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, and scene points are of order n and the number of models in the data-base is N.

A frequently expressed concern about hash-table voting is the possibility of having a large number of *(model, basis)* candidates for a certain index (coordinate). Note, that if the same error model is used, the *alignment* procedure will have to deal with exactly the same number of candidate transformations.

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 memorizing 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 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 first try interpretations which scored high but are uncorrect. Since these interpretations will be rejected by the verification step, in its fast version the algorithm will fail to recognize the model object. 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.

The above analysis holds for the techniques in their 'pure' form. Obviously, various 'intelligent' basis choice methods, can speed up both the *alignment* and the *geometric hashing.* For example, in the HYPER alignment method ([1]) the bases for the similarity transformation are defined by 'privileged' physical segments in the image. A version of *Geometric Hashing* (see [26]) exploits, so called, super-segments. The basic observation is that all these improvements can be equally well incorporated in both techniques.

4 Reduction of Complexity

In the previous sections we have represented the *Geometric Hashing* technique in the most general case. We have used the technique for 'minimal information' features, such as points or lines. Obviously, if more informative features are available, the complexity of the algorithms presented can be further improved. In this section we examine some of the complexity reductions for the 2-D affine matching case. Obvious analogues exist for other cases as well.

4.1 Informative Features

The discussion of the previous sections assumed existence of *interest points* in the image. One needs three such features to define uniquely an affine 2-D transformation. However, more informative features may be available. For example, if one can reliably extract a segment, its endpoints supply two basis points. If one can extract a triangle, the basis is determined completely. One possibility of reliably extracting triangles, based on concavities is discussed in the next subsection.

4.1.1 Concavities

In the case of an affine transformation each concavity supplies us with a stable feature from which an affine invariant basis can be recovered. A concavity is, usually, bounded by a single segment of the convex hull which we call (following [24]) the *concavity entrance.* It is a simple geometric observation that the concavity entrance segment is **invariant** under affine transformations (see [18]). Moreover, if the concavity is unoccluded, it uniquely defines an affine basis triplet, since the point of the concavity, which is on the most distant tangent line, parallel to the concavity entrance, is also affinely invariant (if this point is not unique one may choose the leftmost such point). The computation of a concavity based affine triplet is stable and computationally efficient, since it is based on convex hull extraction.

Now, if one chooses affine basis triplets defined by concavities, the worst case computational burden of recognition becomes linearly dependent on the number of concavities in the scene, instead of having a cubic dependence on the number of *interest points.* This computational burden can be even further reduced by

introducing affine invariant *shape signatures* of concavities, and comparing only those concavities having similar signatures (see [18]).

Even if a concavity is partially occluded, we can still extract relevant information from it. For example, if only the *entrance* segment is visible, its endpoints serve as two points of the affine basis (they suffice, for example, for a similarity invariant basis). Hence, one may choose affine triplets based on a *concavity entrance* segment plus one additional *interest point*, thus still significantly reducing the worst case recognition time complexity, compared to the general case of three *interest point* affine bases.

4.2 Shape Signatures

In the previous section we mentioned that one may distinguish different concavities by using affine invariant *shape signatures* so that only affine bases derived from similar concavities have to be matched. This *shape signature* technique applies also to other cases, where the examined feature has more than minimal information. To illustrate this point, let us consider the *geometric hashing* procedure for the case of planar rigid motion (rotation and translation) using two point bases.

In many industrial applications one is faced with the restricted problem of object recognition under translation and rotation only (case (2) of section 2.3). This might be the case in recognition of flat objects moving on a conveyor belt under a stationary camera. One may apply the *geometric hashing* technique for this case, using two point bases. However, in this case two points have additional information which remains invariant under rigid motion. This information is the distance between the points. Thus, different 2-point bases can be initially distinguished by the distance between the basis points. This is the *shape signature* of such a basis. Hence, one may introduce an additional coordinate into the hash-table address (index), which specifies the *shape signature* (or, one may call it, geometric color) of the basis for which the coordinate was computed. This allows to reduce the voting procedure to relevant bases only.

Analogous *shape signature* coordinates can be introduced also in other cases when the basis supplies more than the minimal information needed. Of special interest is the case of 3-D object recognition from range data. There, a 3 point basis can be used. However, since the possible transformation is rotation and translation only, each 3 point basis has a unique signature, which is defined by the measures of the basis triangle. Only identical triangles have to be matched. This case has an exciting application in *Molecular Biology* for the problem of structural comparison between protein molecules ([?]).

4.3 Projection to a Subspace

When the number of *interest points* on the models is large, one may exploit invariant properties of subspaces, assuming that enough *interest points* are located on such subspaces. For example, in the 2-D affine case, one may use affine invariant properties of lines such as the one given in the following

Lemma (see p.73 in [16]): Two straight lines which correspond in an affine transformation are 'similar', i.e. corresponding segments on the two lines have the same length ratio.

Moreover, the same statement holds for sets of parallel lines. Hence, if we have a set of points, which are located on parallel lines in a model, and another set of points on parallel lines in the scene, we can efficiently check the conjecture, that some of these points correspond.

In this case the information stored in the hash-table for each coordinate will be *(model, line, basis-pair)*, and the voting will be done accordingly. Note that a pair of points define an affine invariant line basis, hence the complexity of recognition is reduced by a factor of n. The price for this reduction, is the restriction to points on parallel lines, which may be practical, when a large number of *interest points* is involved. This is exactly the case, when complexity reduction is most desirable.

5 Conclusions and Future Research

We have surveyed the *Geometric Hashing* paradigm, and compared it with the *Hough Transform* and *Alignment* techniques. Certain extensions of the *Geometric Hashing* allowing further reduction of recognition complexity have been discussed. Future research should include additional extensions of the technique. In particular, recognition of articulated objects composed of rigid parts with internal degrees of freedom is under current investigation.

References

[1] N. Ayache and O. D. Faugeras. HYPER: A New Approach for the Recognition and Positionning of Two-Dimensional Objects. *IEEE TPAMI*, 8(1):44–54, 1986.

[2] D. H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[3] D. H. Ballard and B. C. M. *Computer Vision*. Prentice-Hall, 1982.

[4] P. J. Besl and R. C. Jain. Three-Dimensional Object Recognition. *ACM Computing Surveys*, 17(1):75–154, 1985.

[5] O. Bourdon and G. Medioni. Object Recognition using Geometric Hashing on the Connection Machine. Technical report, Inst. for Robotics and Intell. Systems, USC, 1989.

[6] R. T. Chin and C. R. Dyer. Model-Based Recognition in Robot Vision. *ACM Computing Surveys*, 18(1):67–108, 1986.

[7] B. N. Delone and D. A. Raikov. *Analytic Geometry*, volume 2. Moscow, 1949.

[8] W. E. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE TPAMI*, 9(4):469–482, 1987.

[9] W. E. L. Grimson. The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search. In *Proc. of ICCV*, pages 218–227, Tampa, Florida, Dec. 1988.

[10] A. Heller and J. Stenstrom. Verification of Recognition and Alignment Hypothesis by Means of Edge Verification Statistics. In *Proc. of the DARPA IU Workshop*, pages 957–966, Palo Alto, Ca., 1989.

[11] J. Hong and H. J. Wolfson. An Improved Model-Based Matching Method Using Footprints. In *Proc. of ICPR*, pages 72–78, Rome, Italy, Nov. 1988.

[12] B. K. P. Horn. *Robot Vision*. MIT Press, 1986.

[13] D. P. Huttenlocher and S. Ullman. Object Recognition using Alignment. In *Proc. of ICCV*, pages 102–111, London, 1987.

[14] D. P. Huttenlocher and S. Ullman. Recognizing Solid Objects by Alignment. In *Proc. of the DARPA IU Workshop*, pages 1114–1122, Cambridge, Massachusetts, Apr. 1988.

[15] E. Kishon and H. Wolfson. 3-D Curve Matching. In *Proc. of AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, pages 250–261, St. Charles, Illinois, 1987.

[16] F. Klein. *Elementary Mathematics from an Advanced Standpoint ; Geometry*. Macmillan, New York, 1925 (Third edition).

[17] I. B. Kuperman. *Approximate Linear Algebraic Equations*. Van Nostrand, 1971.

[18] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Object Recognition by Affine Invariant Matching. In *Proc. of CVPR Conf.*, pages 335–344, Ann Arbor, Michigan, June 1988.

[19] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. On Recognition of 3-D Objects from 2-D Images. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 1407–1413, Philadelphia, Pa., Apr. 1988.

[20] Y. Lamdan and H. J. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proc. of ICCV*, pages 238–249, Tampa, Florida, Dec. 1988.

[21] Y. Lamdan and H. J. Wolfson. On the Error Analysis of 'Geometric Hashing'. Technical report, Robotics Lab, Courant Inst. of Math., NYU, 1989.

[22] S. Linnainmaa, D. Harwood, and L. Davis. Pose Determination of a Three-Dimensional Object Using Triangle Pairs. *IEEE TPAMI*, 10(5):634–647, 1988.

[23] Y. Ohta, K. Maenobu, and T. Sakai. Obtaining Surface Orientation from Texels under Perspective Projection. In *Proc. of IJCAI*, pages 746–751, Vancouver, B.C., Canada, Aug. 1981.

[24] J. Schwartz and M. Sharir. Some Remarks on Robot Vision. In *Trans. of 3'rd Army Conf. on Applied Math. and Computing*, pages 1–36, Atlanta, Ga., May 1985.

[25] D. Shoham and S. Ullman. Aligning a Model to an Image using Minimal Information. In *Proc. of ICCV*, pages 259–263, Tampa, Florida, Dec. 1988..

[26] F. Stein and G. Medioni. Graycode Representation and Indexing: Fast Two Dimensional Object Recognition. Technical report, Inst. for Robotics and Intell. Systems, USC, 1989.

[27] G. Stockman. Object Recognition and Localization via Pose Clustering. *J. of Computer Vision, Graphics, and Image Processing*, 40(3):361–387, 1987.

[28] D. Thompson and J. Mundy. Three-Dimensional Model Matching from an Unconstrained Viewpoints. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 208–220, Raleigh, N. Carolina, 1987.

[29] H. J. Wolfson and R. Nussinov. Efficient Detection of Motifs in Biological Macromolecules by Computer Vision Techniques. Technical report, Tel Aviv University, 1990. *in preparation.*

[30] I. Yaglom and V. Ashkinuze. *Ideas and Methods of Affine Projective Geometry*. Moscow, 1962.