On Geometric Hashing and the Generalized Hough Transform

Yaron C. Hecker and Ruud M. Bolle, *Senior Member, IEEE*

Abstract— **The generalized Hough transform and geometric hashing are two contemporary paradigms for model-based object recognition. Both schemes simultaneously find instances of objects in a scene and determine the location and orientation of these instances. The methods encode the models for the objects in a similar fashion and object recognition is achieved by image features "voting" for object models. For both schemes, the object recognition time is largely independent of the number of objects that are encoded in the object-model database.**

This paper puts the two schemes in perspective and examines differences and similarities. We also study the object representation techniques and discuss how the object representations are used for object recognition and position estimation.

I. Introduction

ONE objective of scene analysis is to identify and estimate the position of objects in the world given some form of radiation-based measurements. This can be visible-light intensity images or depth data acquired with laser light. Of course, the objects in the scene may be occluded to some degree. The object models can be geometrically defined, for example, with points, circles, cylinders, etcetera. (Three comprehensive surveys of research papers focusing on this aspect of scene analysis are presented in [1], [2], and [3].) One set of techniques included in the class of so-called model-based approaches, applies to situations where the objects to be identified, are available *a priori* for rather precise measurements, before recognition is attempted. One can preprocess these measurements to develop representations that support a fast recognition process. Note that representation and recognition are very much tied together.

Recognition, the process in which every scene object is identified, is difficult partly because of the viewpoint correspondence problem: the model object can be measured with the camera located in one position, while an instance of the same model, that warrants identification, is typically viewed by a camera positioned somewhere else; or it could be that the camera is stationary but models have moved. In either case, the measurements will be distorted by the relative movements. But, many of these distortions are well approximated with simple, usually linear, transformations and we can fix reference frames to the two measurements, and describe the apparent distortions in terms of reference frame transformations, such as similarity or affine transformations. In this framework, the goal of an object recognition system is two fold: identify those models that have instances in the scene *and* compute the coordinate transformation that maps each such model onto its scene instance. These transformations identify the locations of the objects in the scene.

The set of algorithms commonly called *geometric hashing*, summarized in Lamdan's Ph.D. thesis [4], and in articles [5]–[11], exhibit an interesting approach to speeding up the above mentioned goals of model-based recognition. These algorithms precompute mostly invariant geometric relations of the models and store these relations in a look up table to be accessed during recognition. In this paper, we shed new light on geometric hashing by giving it a new and unconventional formulation, typically used to describe the *generalized Hough transform.*

The generalized Hough transform (see the original description by Ballard [12], [13] and extensions in [14]) is motivated by the standard Hough transform and formulated so that arbitrary shapes can be detected. The standard Hough transform is used to detect parameterized geometric objects such as lines [15]–[17], or circles [18] (see [19] for a review of the Hough transform); the arbitrary shapes that the generalized Hough transform detects may *not* easily be parameterized. Like geometric hashing, the generalized Hough transform precomputes geometric relations in the models and stores these relations in a look up table. By using the easily accessible information in these tables at recognition time, candidate models and transformations are hypothesized. First the generalized Hough transforms accumulate votes for image-model transformations in transformation histograms by processing the entire scene. Then these histograms have to be examined to accomplish recognition. Geometric hashing alleviates the need for expensive histograms and histogram analysis. The invariant relations, computed off-line and stored in lookup tables, are highly redundant and are exploited during recognition. (We note here that several other, somewhat different algorithms, such as those described in [20] and [21], are also called the generalized Hough transform; since these transforms do not use lookup tables, we do not consider these algorithms in this paper.)

Other important studies of these two sets of algorithms can be found in [22]–[24] where a quantitative analysis of the effect of noise on the Hough transform and geometric hashing is given. Though the general conclusion of these studies is that the performance of both of these techniques degrades substantially even with "moderate" amounts of noise, no direct comparison between the techniques is made.

Manuscript received May 4, 1991; revised January 12, 1993 and October 25, 1993.

Y. C. Hecker is with the Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York City, NY 10012 USA.

R. Bolle is with the Exploratory Computer Vision Group, IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 USA.

IEEE Log Number 9403007.

0018-9472/94$04.00 © 1994 IEEE

The major conclusions of our study are as follows:

- The generalized Hough transform and geometric hashing use a similar model representation scheme. The representation is created off-line, before any recognition is attempted. Although it requires time and space to create and maintain, it supports faster recognition.
- Some of the early curve matching algorithms of geometric hashing use the generalized Hough approach for object recognition. Evidence for matches among model and scene features is accumulated in histograms. High peaks identify models that match scene objects and their associated transformations.

  The geometric hashing algorithms in this group are the 2-D and 3-D curve matching algorithms for the case of rotations and translations [5]–[7]. We denote this group by *Curve geometric hashing*.
- Most of the geometric hashing algorithms have an approach to recognition which is very different from the generalized Hough approach. The geometric hashing algorithms in this category consist of an iterative scheme, in which *specially chosen subsets* of scene features are matched at each iteration step. These matches produce a few candidate models that are either refined or rejected in a verifying step of the iteration. The most important issue turns out to be the method for choosing the subsets of features. If done "correctly" the need for the costly histograms and histogram analysis is totally eliminated. This group of geometric hashing algorithms includes all the "basis" algorithms, namely, point and line algorithms [8]–[10], convex and concave affine Curve matching [10]. We denote this group by the name *Basis geometric hashing*.

The rest of this paper is organized as follows: Sections II and III review geometric hashing and the generalized Hough transform. Section IV discusses the object representation scheme common to both paradigms. Approaches to recognition are presented in Section V. Section VI summarizes the results.

II. Review: Geometric Hashing

The geometric hashing schemes are intended for 2-D and 3-D object recognition in cluttered scenes under various viewing assumptions, and for different sensor types. Objects are modeled as sets of local features, some of which might be occluded in the scene image. Recognition is defined as a mapping between a subset of scene features and a subset of model features for some model in the database. These mappings are constrained to belong to a predefined set of geometric transformations $\mathcal{T}$. The precise class of possible transformations depends on knowledge and assumptions about the sensor and object types, the camera and object configuration, and possible geometric transformations between the two. In all cases, the objects can only undergo rigid transformations in world coordinates before sensing—hence, only rigid objects are considered.

We next describe the two categories, basis geometric hashing and curve geometric hashing. The adjectives, *curve* and *basis*, shall be omitted whenever the meaning is clear.

A. Curve Geometric Hashing

Geometric hashing evolved from an algorithm due to Schwartz and Sharir [25] for boundary curve matching. This algorithm solved the curve matching problem under the restrictive assumption that one curve to be matched is a *proper subcurve* of the other. More specifically, given two curves such that one is a proper subcurve of the other, the Schwartz-Sharir algorithm finds the subcurve location by finding the best fit of the subcurve to the longer curve in a least-squares sense. Although fast and robust, if this algorithm is to be applied for recognizing objects, one has to segment the boundary of a scene curve into subcurves belonging to different objects in the scene. One way to do the segmentation is to use the heuristic that overlapping parts create sharp concavities [26]. Sharp concavities are assumed to be the *breakpoints* between subcurves belonging to different objects. Recognition of objects is achieved by matching every scene subcurve with every stored model curve.

Kalvin *et al.* [26] introduce a screening step for the original algorithm to handle cases where the database of objects is large. They define local, rotation and translation invariant object characteristics, termed *footprints*. These are computed from *arc length versus turning-angle* graphs of curves, which are invariant curvature-based curve descriptions and used to index into the model database. The lookup table, containing model identifiers, is created in a preprocessing step. Recognition is performed in two steps. First, model candidates are generated (for verification and localization) by using the footprints of the scene subcurve to index the lookup table and extract model curves. Second, the candidates are verified and localized by using a least squares algorithm. This indexing scheme was called geometric hashing.

The indexing scheme did not solve the problem that scene curves had to be segmented prior to recognition. To overcome this difficulty, Hong *et al.* [7] propose to include in the lookup table, in addition to model identifiers, the location of the footprint along the curve. In the preprocessing step, curve sample points and model identifiers are recorded in the lookup table as follows: curve sample point i of model $\mathcal{M}$ whose footprint is f, is recorded in the cell whose address is f. (When the range of footprints is too large to give each element a unique address, as is usually the case, one maps a range of footprints into each table cell.)

Recognition using footprints is performed in two steps. In the first step, for each model in the curve database, a one-dimensional accumulator array of possible relative shifts is initialized. In these histograms, votes for relative shifts of scene curves with respect to model curves are accumulated. The footprint of each scene curve point is used to index the lookup table and extract the pairs (*model, sample point*). For each such pair, a vote is cast for a curve model and the relative shift of the scene curve with respect to the model curve. For example, if scene curve point i has the same footprint as sample point j on model $\mathcal{M}$, then accumulator cell $j - i$ for model $\mathcal{M}$ is incremented. In the second step of recognition, peaks in the histograms are detected. If histogram $\mathcal{M}$ contains a peak at location k, this indicates that a piece of scene curve

corresponds to model $\mathcal{M}$ and the scene curve starts at location k of the model curve. These histogram peaks are verified and localized.

In a similar approach, Wolfson [5] sorts the footprints for fast indexing instead of storing them in a lookup table. Kishon [6] extends these ideas to three-dimensional curves and experiments with a number of different footprints; he also defines the footprints in a multiresolution fashion. Theoretical results regarding different footprints can be found in [7].

B. Basis Geometric Hashing

In the basis geometric hashing paradigm objects are represented as sets of local features. Typical features include points and line segments. Geometric constraints between these local features in the models are encoded in a lookup table. This is done in a preprocessing stage. The geometric constraints are invariant under the set $\mathcal{T}$ and are therefore useful for fast indexing during recognition. Coupled with each invariant constraint is a minimal set of local features, called a *basis set*, which serves to define a unique mapping between scene features and model features. These bases are stored in a lookup table whose addresses are the invariant constraints.

We first describe how the technique applies to 2-D affine transformations, then discuss how it applies to other transformations. In introducing the paradigm it is easiest to describe objects as a set of points represented in some coordinate system. It is an unspecified issue as to how to extract these points. Some possibilities for extending the paradigm to other features, such as lines or line segments, are discussed in [4] and in [27]; in [27] lines are treated as points in dual space.

Applied to Affine Transformations A 2-D affine transformation is defined by

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

where $\mathbf{A}$ is a nonsingular 2 by 2 matrix, and $\mathbf{b}$ is a 2-D vector. The affine transformation is a good approximation to the image plane transformation that occurs when a flat object, which is relatively far from the camera, is translated and rotated in 3-D space.

The six parameters of the affine transformation are uniquely defined by the correspondence of two ordered triplets of noncollinear points. In addition, an ordered triplet can be used to represent any fourth point in an affine invariant manner. Let $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ be three noncollinear points and $\mathbf{x}$ any fourth point in the plane. Since $\mathbf{x}_1 - \mathbf{x}_3$ and $\mathbf{x}_2 - \mathbf{x}_3$ are linearly independent, there exists unique ξ and η such that

$$\mathbf{x} - \mathbf{x}_3 = \xi(\mathbf{x}_1 - \mathbf{x}_3) + \eta(\mathbf{x}_2 - \mathbf{x}_3)$$

Applying any affine transformation to $\mathbf{x}$ yields

$$T(\mathbf{x}) = \xi(T(\mathbf{x}_1) - T(\mathbf{x}_3)) + \eta(T(\mathbf{x}_2) - T(\mathbf{x}_3)) + T(\mathbf{x}_3)$$

Thus $T(\mathbf{x})$ has the same coordinates (ξ, η) with respect to $T(\mathbf{x}_1)$, $T(\mathbf{x}_2)$, and $T(\mathbf{x}_3)$ as $\mathbf{x}$ has with respect to $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ (see Fig. 1). Hence, we call any noncollinear triplet an *affine basis*, and we call the coordinates (ξ, η) of $\mathbf{x}$ *affine coordinates*.

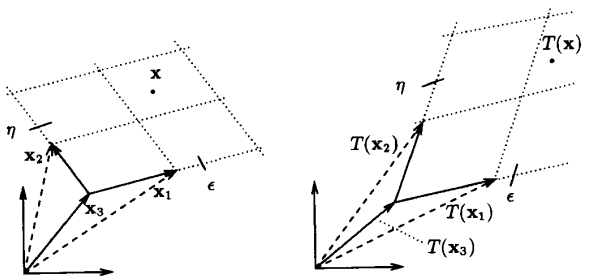

Fig. 1. Affine bases and coordinates: (a) $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ form a basis. (b) $T(\mathbf{x}_1)$, $T(\mathbf{x}_2)$, and $T(\mathbf{x}_3)$ form a basis in the perturbed image.

Picking the same basis in the model and scene, and representing all other points in their affine coordinates, provides a method for determining the number of point matches. Corresponding points will have the *same* coordinates with respect to both reference frames. In addition, the parameters of the transformation can be computed from the two bases. To accomplish fast recognition, a model point is represented by its affine coordinates in *every* affine model basis. In this way, to recognize an object, it is enough to pick *any* basis in the scene such that the three basis points all belong to one model, and compare the resulting coordinates to the stored coordinates.

The previous observations lead to the following algorithm:

Preprocessing For each model object $\mathcal{M}$ represented by m points, and for each triplet of noncollinear points b of $\mathcal{M}$,

1) Compute the affine coordinates of all other m–3 points in terms of basis b.
2) Store the basis b and model identifier $\mathcal{M}$ in a two-dimensional lookup table indexed by the affine coordinates. That is, for each pair of coordinates (ξ, η) computed in step 1, add $(b, \mathcal{M})$ to the lookup table cell whose quantized range of coordinates includes (ξ, η) This process is termed *hashing* and the table is called a *hash-table*.

Recognition Given a scene represented by n points do the following:

1) Choose a basis b^s in the scene, and compute the coordinates
$$\{(\xi_1, \eta_1), (\xi_2, \eta_2), \ldots, (\xi_{n-3}, \eta_{n-3})\}$$
of all other $n-3$ scene points with respect to this basis.
2) Retrieve from the hash-table all pairs $(b, \mathcal{M})$ of bases and models that are stored in cells whose coordinates are close to (ξ_i, η_i) for any i. The acceptable distance in coordinate values depends upon the noise level.
3) Rank the pairs $(b, \mathcal{M})$ based on the number of times they had been retrieved from the hash-table.
4) For all those passing a threshold do:
   (a) Compute the unique affine transformation that maps the model basis to b^s, the scene basis.
   (b) Verify the match between the model points mapped onto scene coordinates and the scene points, using all available points.
5) If not all scene point have been identified jump to step 1.

The worst case running time of the recognition stage is of order n^4 operations, assuming the verification step is of order n and m is also of order n. But, for a successful recognition of a model, it is enough to pick just one basis such that all three points belong to the model. Therefore, the expected running time is lower, and is largely dependent on the noise level in the image [11].

Alternative Basis Sets Geometric hashing applies to a wide variety of image plane or image space (in the case of 3-D scene data) sets of geometric transformations $\mathcal{T}$. For all classes of transformations the basic principles of preprocessing and recognition are the same as for the case of the affine transformations. The differences between the schemes for different classes $\mathcal{T}$ are in the different basis sets that has to be employed for each class of transformations. We list possible basis sets for different $\mathcal{T}$'s.

In the case of a world of 2-D objects the possible image plane transformations are:

- Translations—a one point basis set.
- Rotations—a one point basis set; or alternatively, a unit vector basis set.
- Translations and rotations—a two point basis set; or alternatively, one point and one unit vector.
- Similarity transformations—a two point basis set.
- Affine transformations—a three point basis set (shown above).

When objects are three-dimensional and both model data and scene data are available in 3-D, the following transformations are possible:

- Translations—a one point basis set.
- Rotations—a two point basis set; or alternatively, two unit vectors.
- Translations and rotations—a three point basis set; or alternatively, one point and two unit vectors.
- Similarity transformations—a three point basis set; or alternatively, two points and one unit vector.

In the case that objects are three-dimensional, and model data is available in 3-D form, yet scene data is only two-dimensional, there is no clear extension of the geometric hashing paradigm. This is due to the reduced dimension in the scene space compared to the model space. A number of approximations, including the singular 3-D affine transformation, correspondences of planes, and the tesselation of the viewing sphere, have been suggested [10]. The tesselation of the viewing sphere approach has also been tested with real data.

III. Review: The Generalized Hough Transform

A technique for recognition of arbitrary 2-D curves which resembles the standard Hough transform [15] was introduced by Ballard [12] and coined 'the generalized Hough transform.' The generalized Hough transform is intended for finding the position of a 2-D model's boundary in a noisy scene, where the models are mapped to the scene image by a transformation composed of a translation, rotation and change of scale. (A sample application would be searching for Lake Michigan in an aerial photograph.) Like geometric hashing, the generalized Hough transform is composed of an off-line model preprocessing stage, and an on-line recognition step.

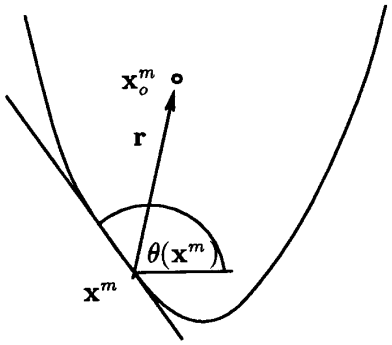

Fig. 2. Shown in this figure are the tangent $\theta(\mathbf{x}^m)$ to the curve at model point $\mathbf{x}^m$ and the reference vector $\mathbf{r}$ at the same point.

We first assume that the orientation and scale of the object are fixed, so that the model will be translated in the image scene relative to its position in the model image. In the following description we shall use $\mathbf{x}$ to denote scene points, to be distinguished from $\mathbf{x}^m$ when referring to model points.

A. Restricted to translations

The following preprocessing step is performed off-line:

1) For each point $\mathbf{x}^m$ on the model's boundary (see Fig. 2) compute $\theta(\mathbf{x}^m)$, the angle of the curve tangent at $\mathbf{x}^m$ (or alternatively the gradient direction at $\mathbf{x}^m$.)
2) Pick an arbitrary reference point $\mathbf{p}_0$ (with model coordinates $\mathbf{x}_0^m$; in the scene this point is denoted $\mathbf{x}_0$, the point to be looked for) and compute its relative position with respect to all boundary points; that is, for each model curve point $\mathbf{x}^m$ compute $\mathbf{r} = \mathbf{x}_0^m - \mathbf{x}^m$ (Fig. 2 illustrates these vectors). Note: given any scene boundary point $\mathbf{x}$ in the translated image and the reference vector $\mathbf{r}$ computed for this point, the location of the reference point $\mathbf{p}_0$ in the scene is given by $\mathbf{x}_0 = \mathbf{x} + \mathbf{r}$.
3) Create a lookup table containing all pairings of tangent angles with corresponding reference vectors,

$$\{(\theta(\mathbf{x}^m), (\mathbf{x}_0^m - \mathbf{x}^m))\}.$$

The table is indexed by discrete angles θ. Since θ is invariant under image plane translations, it serves as an index into the table during the recognition stage, for extracting reference vectors. This table is known as an *R-table* [12].

The recognition stage resembles the standard Hough technique: the coordinates of the reference point $\mathbf{p}_0$ in the translated image plane (the scene) are the unknown parameters. An accumulator array is created for a discretized set of possible x and y scene coordinates of $\mathbf{p}_0$. Now point $\mathbf{x}$ in the scene, which may or may not be part of the model boundary, can use its tangent angle $\theta(\mathbf{x})$ to index into the R-table, to extract those reference vectors that have been paired with $\theta(\mathbf{x})$ (hopefully only a few), to compute the coordinates of $\mathbf{p}_0$ in the scene,

and to vote for them in the array. Thus, for the case of a fixed orientation and scale of the model, the recognition step is as follows:

1) Initialize a 2-D accumulator array $\mathcal{A}(\mathbf{x}_0)$ of possible reference point locations to zero.
2) For each scene edge point $\mathbf{x}$, vote for possible locations of $\mathbf{p}_0$:
   For each table entry $\mathbf{r}$ at index $\theta(\mathbf{x})$, increment accumulator array cell
   $$\mathcal{A}(\mathbf{x}_0) \leftarrow \mathcal{A}(\mathbf{x}_0) + 1,$$
   where $\mathbf{x}_0 = \mathbf{x} + \mathbf{r}$.
3) Maxima in $\mathcal{A}$ correspond to possible reference point locations.

Having identified a few potential locations of model reference point in the scene, one can then use them to map all the curve points to the scene and verify the candidates one by one. When one is looking for a number of pre-modeled objects (these could be, for example Lake Michigan or Lake Superior or Lake Ontario) each of the possible model objects can be searched for independently, in a sequential manner, but this search could also be performed in parallel by creating an accumulator array for each of these known models and repeating the voting steps, once for every scene point, for each model.

B. Similarity Transformations

Rotation and scaling are incorporated into the generalized Hough transform by expanding the accumulator array and creating multiple R-tables for each model: one table for each possible quantized rotation angle and scaling factor. In fact, the additional tables need not be created explicitly; a simple computation during the recognition stage derives an entry in any rotated and scaled table from a corresponding entry in the single R-table which is actually stored (so that each stored value generates multiple votes). In describing this computation it is simpler to use the polar coordinates of reference vectors, i.e., $\mathbf{r} = (r, \alpha)$. Recognition in the case of similarity transformations takes the following form:

1) Initialize 4-D accumulator array $\mathcal{A}(\mathbf{x}_0, \beta, s)$ of possible scene locations $\mathbf{x}_0$ of the reference point $\mathbf{p}_0$ for any given rotation angle β and scaling factor s of the model.
2) For each scene edge point $\mathbf{x}$, and for each possible quantized scaling factor s and rotation angle β of the model, vote for possible locations of $\mathbf{p}_0$. In more detail, For each table entry $\mathbf{r} = (r, \alpha)$ at index $\theta(\mathbf{x}) - \beta$, increment accumulator array cell
   $$\mathcal{A}(\mathbf{x}_0, s, \beta) \leftarrow \mathcal{A}(\mathbf{x}_0, s, \beta) + 1,$$
   where $\mathbf{x}_0 = \mathbf{x} + \mathbf{q}$, and the rotated and scaled reference vector $\mathbf{q}$ is given in polar coordinates by,
   $$|\mathbf{q}| = sr \quad \text{and} \quad \text{angle}(\mathbf{q}) = \alpha + \beta\,,$$
   where *angle* is the function that returns the angle of its vector argument. Each (s, β) pair casts a set of votes for the model scaled by s and rotated by β
3) Maxima in array $\mathcal{A}$ correspond to possible reference point locations for a possible rotation and scale factor of the model.

We would like to make one comment on the name 'generalized Hough transform' chosen for this method that we have described, showing why the standard Hough transform is a special case of it. Analytical curves, of the type that the standard hough deals with, can be parametrized by coordinate system transformation parameters—translations in x_0 and y_0, rotation β, and scaling factors s. For example, a circle equation is given by

$$(x - x_0)^2 + (y - y_0)^2 = s^2,$$

and we can choose its center as the reference point. Given the gradient direction θ at point $\mathbf{x}$ on the circle and a scale factor s, the possible location of the reference point (x_0, y_0) is $\mathbf{x}+\mathbf{r}$ where $\mathbf{r}$ is given in polar form by

$$|\mathbf{r}| = s \quad \text{and} \quad \text{angle}(\mathbf{r}) = \theta.$$

The R-table is implicit (no need to create it) because the reference vector at index θ is just a unit vector in the direction of the gradient θ.

Objects composed of subparts can be recognized by combining the R-tables of each subpart into one global R-table for the whole object. Another extension to the basic generalized Hough transform is to use more complex strategies for incrementing the accumulator array, other than incrementing by unity. These two issues are discussed in [12]. An extension of the scheme to three-dimensional objects is described in [14].

IV. Object Representation

We examine in greater detail the object representations used by geometric hashing and the generalized Hough transform. The common central idea in the object representation schemes is to identify object features that contain both viewpoint independent parameters, which we term *indices*, and viewpoint dependent parameters, termed *references*. Both indices and references convey geometric properties of a feature. Indices contain viewpoint independent properties of a feature and are used to detect instances of a model in the scene; reference parameters, on the other hand, depend on the viewpoint and are used to compute the transformations of the scene object with respect to the models.

For example, consider recognizing polygons in line drawings and using the line segments that form the edges as features. If the scale of the objects is fixed, then the length of a line segment may serve as an index, and its two endpoints may serve as a reference. The internal representation of a polygon is a pairing of the length of each line segment with the two endpoints. This information is saved in a lookup table indexed by length. During recognition, line segments in the scene can be matched with the model line segments by using their lengths which are invariant under translations and rotations of the model. The model endpoints are aligned with the scene points to compute the pose of the object in the scene. By pose we mean both the orientation and location parameters.

A. Indices and References

Geometric hashing and the generalized Hough transform have a common object representation scheme. Let $\mathcal{T}$ denote the set of possible mappings of a model's coordinate frame to an arbitrary scene coordinate frame. A mapping in $\mathcal{T}$ is a coordinate frame transformation. With increasing generality, we have, as possible transformations: translations, rigid body transformations, similarity transformations, and affine transformations. Based on the allowed set of transformations $\mathcal{T}$ and on the set of world objects, features for object representation are selected that contain two types of parameters—*indexing parameters* and *referencing parameters*. These features are extracted both in the model and in the scene and are used to identify instances of models in the scene. In addition, they are used to compute the location of the model instances in the scene.

Given a feature ϕ, a reference $\mathcal{R}(\phi)$ is a viewpoint dependent parameter (vector) of a feature. If a feature is viewed from two different viewpoints, the reference parameters of the two instances of the feature can be used to compute the viewpoint transformation $\mathbf{T} \in \mathcal{T}$. This transformation is not necessarily unique; uniqueness depends on $\mathcal{T}$ and the reference parameters. In practice, reference parameters are chosen to provide a unique transformation. In that case, the reference parameter provides a basis for the transformation.

Indices $\mathcal{I}(\phi)$ are mostly viewpoint independent parameters of features ϕ. They are used to identify instances of a feature (and hence the object containing the feature) irrespective of the choice of coordinate system. Thus, scene feature $\phi_{\mathcal{S}}$ matches model feature $\phi_{\mathcal{M}}$, if these features are of the same type and have equal indices.

Let us look how this terminology applies to the generalized Hough transform for the case that $\mathcal{T}$ is the group of translations. To model an object, points with a small neighborhood on an object's smooth boundary are chosen as point features ϕ for recognition. Some translation-invariant property of these features, $\mathcal{I}(\phi)$, is used as index parameter. This can be, for example, the tangent $\mathbf{t}$ to the curve at point feature ϕ as the tangent does not change when the curve is translated. As reference parameters $\mathcal{R}(\phi)$ the coordinates $\mathbf{X}$ of the features [i.e., $\mathbf{X} = (x, y)$ or (x, y, z) for 2-D or 3-D cases, respectively] can be used. In the model, these coordinates are expressed in an object-centered coordinate system. A pairing between a model feature $\phi_{\mathcal{M}}$ and a scene feature $\phi_{\mathcal{S}}$ determines the translation $\mathbf{T}$ between the model and scene coordinate frames. This translation is directly computed from $\mathbf{X}_{\mathcal{M}}$ and $\mathbf{X}_{\mathcal{S}}$, the scene and model reference parameters, i.e., $\mathbf{T} = \mathbf{X}_{\mathcal{M}} - \mathbf{X}_{\mathcal{S}} \in \mathcal{T}$ The index $\mathbf{t}$ is used to select feature pairings.

In the above example, a unique translation $\mathbf{T}$ is computed, but, if $\mathcal{T}$ includes rotations as well as translations, infinitely many transformations between the two points are possible. Still, this set of transformations is useful since it constitutes only a subset of all possible transformations $\mathcal{T}$. This is meaningful if the space of transformations is finitely quantized.

Parameters which are not totally viewpoint independent may also be used as indices. For example, if $\mathcal{T}$ is the group of translations and rotations, the tangent $\mathbf{t}$ at model point-feature ϕ can be used once more as an index, albeit a less suitable one than where $\mathcal{T}$ is only the group of translations. When rotations are included in $\mathcal{T}$ the model has to be rotated to each possible quantized angle and each such model view treated as a separate model [12]. Therefore, such indices are used rarely, only when no viewpoint independent parameters have been found.

An object is represented by a set of pairs $\{(\textit{index}, \textit{reference})\}$; each pair represents a feature ϕ of the object. The indices are provided for fast identification of features, while the references serve to find the viewpoint transformation (Fig. 3). Typically, these lists are stored in a lookup table addressed by the indices. For example, let $\{\phi_i\}$ denote arbitrary points on the boundary of an object. Let $\mathbf{X}_i$ denote the coordinates of ϕ_i, $\mathbf{t}_i$ the tangent at ϕ_i, so

$$\mathbf{X}_i = \mathcal{R}(\phi_i) \quad \text{and} \quad \mathbf{t}_i = \mathcal{I}(\phi_i).$$

Then the set of pairs $\{(\mathbf{t}_i, \mathbf{X}_i)\}$ is stored in a table whose addresses are quantized tangent vectors $\mathbf{t}_i$ and whose contents are coordinates $\mathbf{X}_i$. This set of pairs constitutes the representation (model) of an object. If the database includes more than one model, a model identifier is stored with every $(\mathbf{t}_i, \mathbf{X}_i)$ pair. In this case, the object-model catalogue is a set of triplets

$$\{(\mathcal{I}(\phi_i), \mathcal{R}(\phi_i), \mathcal{M}_j)\},$$

where feature ϕ_i belongs to model $\mathcal{M}_j$.

B. Tuples of Features

At times, the available feature types in an object are not suitable for an index/reference representation. This happens when, for a given set of allowed transformations $\mathcal{T}$, the feature types do not contain viewpoint-invariant parameters and/or suitable reference parameters. For example, a line segment is not a suitable feature for the case where $\mathcal{T}$ is the group of similarity transformations since scaling is allowed. That is, a line segment does not contain any geometric information that is invariant under similarity transformations $\mathcal{T}$. Hence, no suitable index is available. On the other hand, in this particular example, suitable reference parameters exist—the coordinates of the two endpoints.

Geometric hashing solves this problem of "weak" features in a systematic fashion by forming *tuples* Φ of basic features ϕ. Each tuple is represented by index and reference parameters. To see why tuples provide the necessary information, consider the following example. Assume that the basic features are point features ϕ, and that $\mathcal{T}$ is the group of similarity transformations in two-space. New features Φ are defined by using triplets of point features. Geometrically, a triplet constitutes a triangle. The angles of a triangle are invariant under similarity transformations and hence can be used as indices. The coordinates of the vertices serve as references. In fact, it is easy to see that two angles and the corresponding vertices suffice as index and reference parameters, respectively.

In a similar manner, basic point-features can be grouped into tuples of size two, four, and five points, depending on the group of transformations $\mathcal{T}$ that is allowed. For translations in two-space, tuples of size two (pairs) are used. These pairs define features which contain a natural index parameter–the

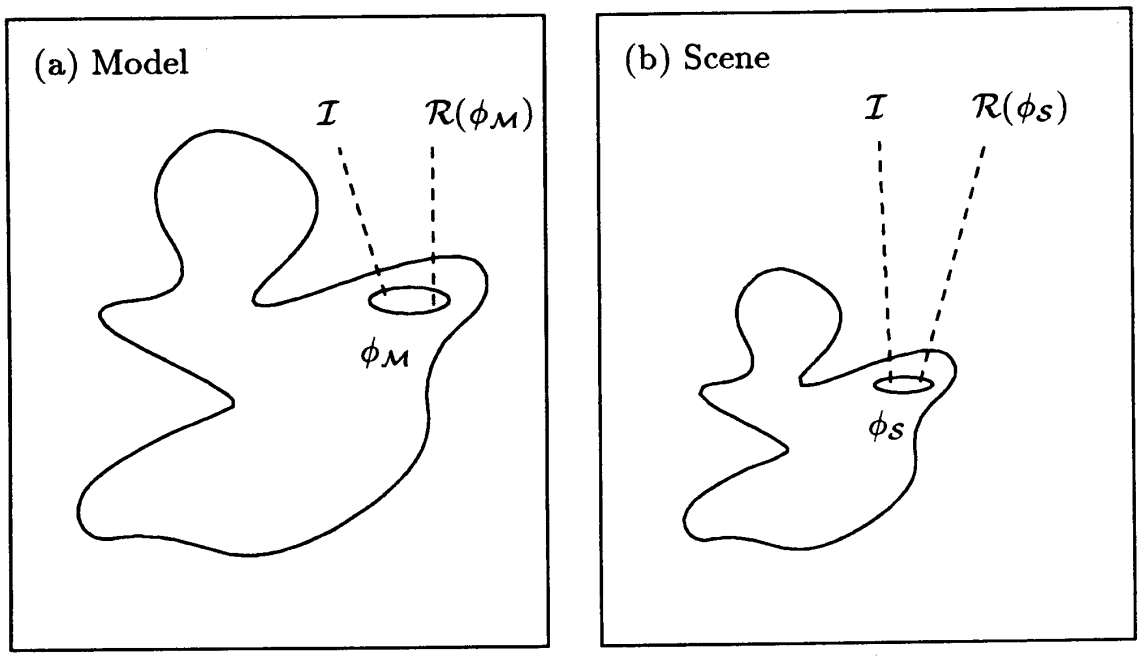

Fig. 3. Index and reference parameters of feature ϕ both in the model and in the scene.

distance between the point features; the coordinates of one of the points can be used as the reference parameter. For affine transformations quadruples are used. An algebraic approach [8]–[10] (rather than the geometric approach described here) yields natural index and reference parameter for any size tuple feature Φ.

This grouping technique applies not only to points, but also to lines, line segments, circles, and in fact to arbitrary features as basic features ϕ. The complex geometric structures that are formed by such tuples contain more information for invariant indexing and referencing.

Sharing a Reference Parameter An important characteristic of a particular grouping technique is whether reference parameters are shared by multiple features, i.e., if many features have identical references. As an example, consider once more features that are triplets of point features ϕ, $\Phi = (\phi_1, \phi_2, \phi_3)$, where every possible triplet forms a distinct feature. Then for $\mathcal{T}$ the group of similarity transformations, the coordinates of ϕ_1 and ϕ_2 may serve as the reference parameter, i.e.,

$$\mathcal{R}(\Phi) = (\mathcal{R}(\phi_1), \mathcal{R}(\phi_2)),$$

with $\mathcal{R}(\phi)$ the coordinates of point feature ϕ. If an object contains n feature points then the $n-2$ triplet-features,

$$\{(\phi_1, \phi_2, \phi_i) \mid 3 \le i \le n\},$$

all have the same reference parameters. Also, since ϕ_1 and ϕ_2 are arbitrary features, each reference is shared by $n-2$ features. Fig. 4 shows four tuple features, $\Phi_1, \ldots, \Phi_4$, all having an identical reference parameter—the coordinates of the two points (ϕ_1, ϕ_2).

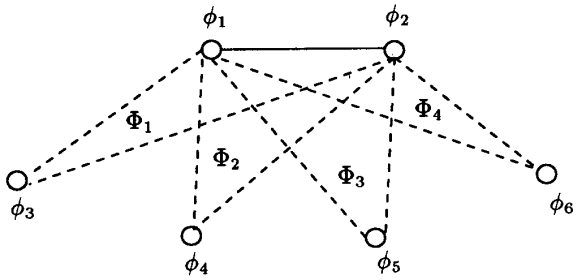

Fig. 4. Four features sharing one reference.

An example of tuple features that do not have common references are feature pairs, $\{(\phi_i, \phi_j)\}$ The distance between the feature points serves as the index parameter and the coordinates of both feature points serve as the reference parameter; hence, each feature pair $\Phi = (\phi_i, \phi_j)$ has a distinct reference. It is important to note that in both these examples, the reference parameters are identical—coordinates of two points. However, in the latter case, each reference is associated with one feature, while for the former, each reference is associated with $n-2$ features.

In geometric hashing, feature tuples Φ are created in such a way that many of the tuples Φ share an identical reference. This results in multiple indices for each reference, which are capable of generating strong evidence for a particular model feature candidate (Section V). Thus, although both examples of grouping features in the previous paragraph are suitable for

representing objects using indices and references, when $\mathcal{T}$ is the group of translations and rotations, only the first can be used by geometric hashing. Both may be used by generalized Hough algorithms.

C. Examples of Features, Indices and References

Ballard, who introduced the index/reference object-representation technique, matches object boundaries (2-D curves) [12]–[13], as described earlier; points along the curve are used as point features ϕ. Index parameters are approximated gradient directions at these points, and reference parameters are the coordinates of the points. This is most suitable for translations. If rotations and scaling are also considered, the object model has to be rotated and scaled for all possible angles and scale factors; and each resulting instance treated as a separate model. This is a very inefficient process, both from a storage and from a computational point of view. Because of the large number of object models, recognition performance degrades.

Hong and Wolfson [7] introduce a better representation for 2-D curve segments. It is used for matching curve segments detected in an image to a library of model curves. Here, $\mathcal{T}$ is the group of 2-D rigid body transformations. Features ϕ are again curve points (and their neighborhoods). The indices are various translation and rotational invariant parameters, such as the curvature κ at point feature ϕ. The reference parameter s for point feature ϕ on a curve segment is the arc length from the starting point of the segment to the feature location. Matching two curve feature points is based on similar indices and yields the location of the starting point of the scene curve on the model curve (expressed in arc length). The Euclidean transformation $\mathbf{T}$ between the model curve and scene curve is computed from the translation along the arc length in a straightforward way. Kishon and Wolfson [6] extend these techniques to 3-D curves.

A representation of 2-D curves, for $\mathcal{T}$ affine transformations, is available in [9]. A feature ϕ is selected to be a subcurve. The index $\mathcal{I}(\phi)$ is the parameterized shape of the normalized subcurve (normalized such that the shape parameters are affine invariant). The reference $\mathcal{R}(\phi)$ is the triplet of curve points which consists of the two endpoints of the subcurve and the point furthest away from the line segment connecting the endpoints. Since every subcurve is a feature, the number of features is very large. This number can be much reduced if the objects contain concavities. Then only the subcurves that form the concavities are considered as features.

Lamdan et al. [8]–[10] extract (about 15–20) point features ϕ from more complicated objects and use the above described grouping technique to represent these objects. This is what we called in our review section 'basis geometric hashing'. The size of the feature tuples Φ that they form depends on $\mathcal{T}$. The features ϕ can also be lines, line segments, etc.

Polygons can be represented with the grouping technique by treating each vertex as a point feature ϕ. A reduction in the number of tuple features Φ is possible. Only those tuples that contain an edge, i.e., at least two of the tuple points are connected by an edge of the polygon, are considered as features. This significantly reduces the number of features Φ that represent a polygon; hence, storage and search requirements are reduced. Yet, the number of features having a common reference is not reduced while the number of references is reduced. This is an important issue in iteratively selecting scene features (Section V).

A slight variant of an early (generalized Hough) example of modeling 2-D and 3-D polytopes [14] is the following: the edges of the polytope are the features ϕ; the edge length is the index parameter; and the coordinates of one of two vertices together with the edge normal serve as the reference. This representation is suitable for recognition under translations and rotations. Of course, obscured edges in the scene will then produce false indices.

V. Approaches to Recognition

An object usually contains many features. Each feature can be used to find the viewing transformation, or at least to constrain the set of possible transformations $\mathcal{T}$ to some subset $\mathcal{T}'$. The question arises as to how many features to match. Computing the viewing transformation from a match of only *one* feature will possibly yield too many candidate transformations. On the other hand, integrating information from various matches requires additional capabilities. The generalized Hough and basis geometric hashing are at opposite extremes regarding this issue. Generalized Hough transforms match *all* scene features and then try to find clusters among the transformations associated with these matches. Basis geometric hashing *iteratively* matches only *small subsets* of scene features. The estimate of the transformation for a given match is either improved, or the match is abandoned in a verifying step. The key issue in this approach is that if the subsets of features are chosen "correctly," no clustering is required.

The next section discusses the approach to recognition common to the generalized Hough transform and curve geometric hashing. Following that, we discuss the contributions that basis geometric hashing offers to recognition.

A. Transform Clustering

The generalized Hough transform and curve geometric hashing have a similar approach to object recognition. Object recognition consists of two stages. In the first stage, every single scene feature ϕ_S is matched with all model features $\phi_\mathcal{M}$ that possess similar invariant characteristics (indices). The output of this stage is a set of pairs $\{(\mathcal{M}, \mathbf{T})\}$. Here $\mathcal{M}$ denotes a model that contains feature $\phi_\mathcal{M}$, which is matched to a scene feature ϕ_S, and $\mathbf{T}$ denotes the transformation that maps $\phi_\mathcal{M}$ onto ϕ_S. In Fig. 5, T_{oi}, the transformation that maps $\phi_{\mathcal{M}_1}$ to ϕ_S, is computed from $\mathcal{R}_o$ and $\mathcal{R}_i$, the scene and model reference parameters.

The second stage assigns a meaning to the output of the first stage. For each model $\mathcal{M}$ that has at least one feature $\phi_\mathcal{M}$ that is matched to a scene feature ϕ_S, the second stage searches for large clusters of transformations $\mathbf{T}$ that map features of $\mathcal{M}$ onto scene features. The data structures that are used to

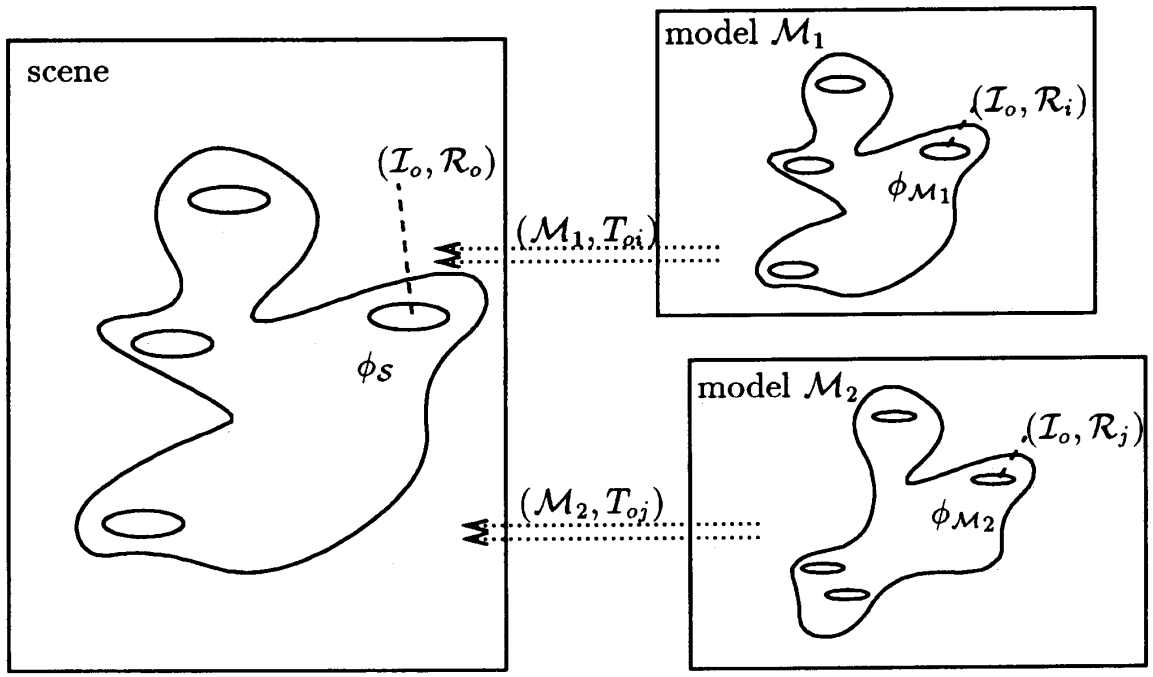

Fig. 5. Matching scene and model features.

store and detect these clusters are called parameter spaces. These are histograms of transformations $\mathbf{T}$—one histogram per model $\mathcal{M}$ in the object model database. Hence, the first stage amounts to building these histograms and the second stage to finding peaks in the histograms.

The matching process in the first stage of recognition works as follows. Scene feature ϕ_S is paired with every model feature $\phi_{\mathcal{M}}$ such that $\mathcal{I}(\phi_S) = \mathcal{I}(\phi_{\mathcal{M}})$. This can be achieved very quickly because model features are stored in a lookup table with the index parameters as addresses. For every pair $(\phi_S, \phi_{\mathcal{M}})$, a transformation $\mathbf{T}$ that maps the model feature $\phi_{\mathcal{M}}$ onto scene feature ϕ_S is computed. This transformation is computed directly from the respective reference parameters $\mathcal{R}(\phi)$. This information is accumulated in the histogram containing various transformations associated with model $\mathcal{M}$.

B. Iterating over Feature Subsets

Regarded at the level of *individual* feature matching, basis geometric hashing and the generalized Hough transform (and curve geometric hashing, of course!) are actually equivalent. First, a scene feature ϕ_S is chosen, and the index and reference parameters are computed. Next, by indexing into the lookup table, the index parameter is used to extract those models $\mathcal{M}$ that have similar features $\phi_{\mathcal{M}}$. In turn, each model feature reference is paired with the scene reference. From each such pair, the transformation $\mathbf{T}$ that maps the model onto its instance in the scene is computed.

The important difference between the two schemes is found in the global strategy of selecting scene features. The generalized Hough transform matches *every* scene feature ϕ_S in *one step*, leaving it up to the analysis of the histogram to make sense out of all the matches. Basis geometric hashing *iteratively* matches *subsets* of scene features Φ_S. Each subset is chosen such that the resulting candidates $(\mathcal{M}, \mathbf{T})$ can be analyzed without the use of grouping techniques. There is no need to define a distance measure on $\mathcal{T}$. Clustering in high-dimensional space is often troublesome [22]; additionally, clustering implicitly defines a distance measure, which usually is meaningless.

Selecting Feature Subsets In this section, we assume the group of transformations $\mathcal{T}$ is a one-to-one mapping between references. All the above mentioned transformations satisfy this condition.

We describe two methods for selecting subsets of scene features ϕ_S. The simplest method is to select singletons; at each step in the iteration process, *one* scene feature ϕ_S is chosen. This feature is matched with model features $\phi_{\mathcal{M}}$ of equal index and the mapping transformations computed from the respective references. The number of generated pairs $(\mathcal{M}, \mathbf{T})$ is a function of the index type, the noise level, and the popularity of the scene feature ϕ_S in the object-model database.

Transformations for a particular model $\mathcal{M}_0$ are mutually inconsistent, because two features belonging to one model cannot map onto one scene feature. That is, one scene feature ϕ_S cannot instantiate the model $\mathcal{M}_0$ more than once. Therefore,

transformation histograms are unnecessary, and each $(\mathcal{M}, \mathbf{T})$ must be verified independently. This technique is used for affine (concave and convex) curve matching [9].

The second method of selecting feature subsets, used by 'basis geometric hashing', is employed when features are tuples Φ formed, as previously discussed, by grouping basic features ϕ. In this case, multiple features Φ have identical reference parameters (see Section IV.B). At each step of the iterative recognition process, exactly *one* scene reference parameter is chosen from among all unmatched reference parameters. This then defines a set of features, namely, those scene features that contain the one selected reference parameter. This feature set is matched with the object-model features.

In general, a scene reference can only be associated with at most one reference in each model. Therefore, for each model, a scene reference can generate at most one correct transformation $\mathbf{T}$ that maps this particular model onto the scene. So, if after an iteration among the model/transformation candidates we have both

$$(\mathcal{M}_0, \mathbf{T}_1) \quad \text{and} \quad (\mathcal{M}_0, \mathbf{T}_2),$$

with $\mathbf{T}_1 \neq \mathbf{T}_2$, then it cannot be the case that both $\mathbf{T}_1$ and $\mathbf{T}_2$ are correct transformations of model $\mathcal{M}_0$. Therefore, unequal transformations surely do not represent noisy instances of the same transformation. That is, there is no need for a distance measure between transformations. The exceptions to this correspond to different instances of the same object in a scene that share an identical scene reference. For this to be the case, the transformation between these instances is large (e. g., a 180 degree rotation around some point in the image), and hence can be readily detected.

Each iteration generates a multiset $\{(\mathcal{M}_i, \mathbf{T}_j)\}$ of candidates. These candidates can be ordered according to the number of times they appear in the set, producing the list

$$(\mathcal{M}_{i_1}, \mathbf{T}_{j_1}), (\mathcal{M}_{i_1}, \mathbf{T}_{j_2}), \ \ldots \ (\mathcal{M}_{i_n}, \mathbf{T}_{j_n})).$$

The number of times a particular candidate appears in the list can be thought of as scene features agreeing to this model candidate—called "voting" in geometric hashing terminology. Verification proceeds in the order of this list. The highest confidence candidates, typically candidates that correspond to objects in the scene with many features, are verified first. Heuristics can be used to stop verification of the list [8]. This hypotheses/verification cycle is continued until the scene is completely identified, including scene features labeled as 'unknown'. For verification, easily accessible individual object representations are used.

Geometric hashing uses groupings of basic features ϕ such that *many* tuple features Φ share the same reference. Let us assume that a reference is a subtuple containing basic features. If a scene reference corresponds to a number of scene features that all belong to the same object in the scene, then many of the scene features that share this reference will generate the same candidate $(\mathcal{M}_i, \mathbf{T}_j)$. This has the effect that per iteration few candidates will be generated with high confidence. Consequently, the scene is analyzed more efficiently since relatively little processing time is spent on verification of low-confidence candidates.

The major implication of the subset selection methods is that there is no need for transformation histograms and cumbersome histogram analysis. This is a pronounced difference between geometric hashing and the generalized Hough transform.

VI. Conclusion

We observe that the generalized Hough transform can be divided into two classes. The first class precomputes geometric relations in the models and stores them in lookup tables. The second class does not perform any preprocessing.

Comparing the first class of generalized Hough algorithms to geometric hashing yields the following observations:

- Both paradigms use similar object representation schemes. This is of key importance since the success of these schemes is very much due to the intricate model representation.
- Some geometric hashing algorithms (the early curve matching algorithms) accumulate evidence for all matches in a parameter space, representing transformations and search for clusters in this space. This is identical to the generalized Hough approach.
- Most geometric hashing algorithms iteratively match a small subset of scene features until a verification step succeeds. The important implication of this approach is that the clustering step, which is a central but weak element of generalized Hough algorithms, can be totally avoided.
- Geometric hashing introduces a systematic technique for grouping of simple features to obtain features that contain enough information for matching scene features to model features and computing transformations (object positions) from these matches.

References

[1] T. O. Binford, "Survey of model-based image analysis systems." *The Int. Journal of Robotics Research*, vol. 1, no. 1, pp. 18–64, Spring 1982.
[2] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *Computing Surveys*, vol. 17, no. 1, pp. 75–145, Mar. 1985.
[3] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Computing Surveys*, vol. 18, no. 1, pp. 67–108, Mar. 1986.
[4] Y. Lamdan, "Object recognition by geometric hashing," *Ph.D.* Thesis, New York University, Aug. 1989.
[5] H. J. Wolfson, "On curve matching," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 12, no. 5, pp. 483–489, 1990.
[6] E. Kishon and H. J. Wolfson, "3-d curve matching," In *Proc. AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, pp. 250–261, Oct. 1987.
[7] J. Hong and H. J. Wolfson, "An improved model-based matching method using footprints," In *Proc. 9th Int. Conf. on Pattern Recognition*, Nov. 1988.
[8] Y. Lamdan, J. T. Schwartz and H. J. Wolfson, "On recognition of 3-D objects from 2-D images," In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1407–1413, Apr. 1988.
[9] Y. Lamdan, J. T. Schwartz and H. J. Wolfson, "Object recognition by affine invariant matching," In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 335–344, Jun. 1988.
[10] Y. Lamdan and H. J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," In *Proc. Second Int. Conf. on Computer Vision*, pp. 238–249, Dec. 1988.

[11] Y. Lamdan and H. J. Wolfson, "On the error analysis of geometric hashing," *Technical Report 467*, Courant Institute of Mathematical Sciences, New York University, Oct. 1989.
[12] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.
[13] D. H. Ballard and C. M. Brown, *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
[14] D. H. Ballard and D. Sabbah, "Viewer independent shape recognition," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 5, no. 6, pp. 653–660, Nov. 1983.
[15] P. V. C. Hough. *Methods and Means for Recognizing Complex Patterns*, U.S. Patent 3069654, Dec. 1962.
[16] R. O. Duda and P. E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *Comm. of the CAM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.
[17] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.
[18] C. Kimme, D. H. Ballard and J. Sklansky, "Finding circles by an array of accumulators," *Comm. of the ACM*, vol. 18, no. 2, pp. 120–122, Feb. 1975.
[19] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 87–116.
[20] P. M. Merlin and D. J. Farber, "A parallel mechanism for detecting curves in pictures," *IEEE Trans. on Computers*, vol. 24, no. 1, pp. 96–98, Jan. 1975.
[21] G. S. Stockman, "Three-dimensional pose computations from multiple views," In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice II*, pp. 233–242. North-Holland, Amsterdam, 1986.
[22] W. E. L. Grimson and D. P. Huttenlocher, "On the sensitivity of the Hough transform for object recognition," In *Proc. Second Int. Conf. on Computer Vision*, pp. 700–706, Dec. 1988.
[23] W. E. L. Grimson and D. P. Huttenlocher, "On the sensitivity of geometric hashing," In *Proc. 3rd Int. Conf. on Comp. Vision*, Dec. 1990.
[24] M. Costa, R. M. Haralick and L. G. Shapiro, "Optimal affine-invariant point matching," In *Proc. 6th Israel Conf. on AI*, pp. 35–61, 1990.
[25] J. T. Schartz and M. Sharir, "Identification of partially obscured objects in two dimensions by matching of noisy characteristic curves," *The Int. Journal of Robotics Research*, vol. 6, no. 2, pp. 29–44, 1987.
[26] A. Kalvin, E. Schonberg, J. T. Schartz and M. Sharir, "Two dimensional model based boundary matching using footprints," *The Int. Journal of Robotics Research*, vol. 5, 4, pp. 38–55, 1988.
[27] Y. C. Hecker and R. M. Bolle, "Invariant feature matching in parameter space with applications to line features," In *Proc. Geometric Methods in Computer Vision, SPIE 36 Annual Symposium*, 1991.

Yaron C. Hecker is a Ph.D. candidate in computer science at the Courant Institute of Mathematical sciences at New York University. He received the B.S. degree in mathematics and computer science from the Hebrew University, Jerusalem, Israel, in 1986, and the M.S. degree in computer science from NYU in 1989. From 1989 to 1991 he performed research in computer vision at the IBM T.J. Watson Research Center in Yorktown Heights, NY. His research interests include various aspects of multimedia systems: computer graphics and user interfaces, computer vision and image processing.

Ruud M. Bolle (S'82–M'84–SM'89) was born in Voorburg, The Netherlands. He received the Bachelor's Degree in Analog Electronics in 1977 and the Master's Degree in Electrical Engineering in 1980, both from Delft University of Technology, Delft, The Netherlands. In 1983 he received the Master's Degree in Applied Mathematics and in 1984 the Ph.D. in Electrical Engineering from Brown University, Providence, RI. In 1984 he became a research staff member at the IBM T.J. Watson Research Center in the Artificial Intelligence Department of the Computer Science Department. In 1988 he became manager of the newly-formed Exploratory Computer Vision Group; since September, 1990 he has been manager, Computer Vision and Mobile Robotics.

Currently, his research interests are focused on object modeling and matching for image database browsing, active and real-time vision, and computer vision for user interfaces.

He is a Senior Member of the IEEE, and associate editor of IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and the *Journal of Mathematical Imaging and Vision*.