International Journal of Computer Vision, 5:2, 195-212 (1990)


# Recognizing Solid Objects by Alignment with an Image

DANIEL P. HUTTENLOCHER
*Computer Science Department, Cornell University, 4130 Upson Hall, Ithaca, NY 14850*

SHIMON ULLMAN
*Department of Brain and Cognitive Science, and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139*

**Abstract**
In this paper we consider the problem of recognizing solid objects from a single two-dimensional image of a three-dimensional scene. We develop a new method for computing a transformation from a three-dimensional model coordinate frame to the two-dimensional image coordinate frame, using three pairs of model and image points. We show that this transformation always exists for three noncollinear points, and is unique up to a reflective ambiguity. The solution method is closed-form and only involves second-order equations. We have implemented a recognition system that uses this transformation method to determine possible *alignments* of a model with an image. Each of these hypothesized matches is verified by comparing the entire edge contours of the aligned object with the image edges. Using the entire edge contours for verification, rather than a few local feature points, reduces the chance of finding false matches. The system has been tested on partly occluded objects in highly cluttered scenes.

## 1 Introduction

A key problem in computer vision is the recognition and localization of objects from a single two-dimensional image of a three-dimensional scene. The model-based recognition paradigm has emerged as a promising approach to this problem (e.g., [Ayache and Faugeras 1986; Brooks 1981; Huttenlocher and Ullman 1987; Lamdan et al. 1988; Lowe 1987; Roberts 1965; Silberberg et al. 1986; Thompson and Mundy 1987]; see also [Chin and Dyer 1986] for a comprehensive review). In the model-based approach, stored geometric models are matched against features extracted from an image, such as vertexes and edges. An interpretation of an image consists of a set of model and image feature pairs, such that there exists a particular type of transformation that maps each model feature onto its corresponding image feature. The larger this corresponding set, the better the match.

In this paper we present a model-based method for recognizing solid objects with unknown three-dimensional position and orientation from a single two-dimensional image. The method consists of two stages: (i) computing possible *alignments*—transformations from model to image coordinate frames—using a minimum number of corresponding model and image features, and (ii) verifying each of these alignments by transforming the model to image coordinates and comparing it with the image. In the first stage, local features derived from corners and inflection points along edge contours are used to compute the possible alignments. In the second stage, complete edge contours are then used to verify the hypothesized matches.

Central to the method is a new means of computing a transformation from a three-dimensional model coordinate frame to the two-dimensional image coordinate frame. This method determines a possible *alignment* of a model with an image on the basis of just three corresponding model and image points (or two corresponding points and unit orientation vectors). The method is based on an affine approximation to perspective projection, which has been used by a number of other researchers (e.g., [Brooks 1981; Cyganski and Orr 1985; Kanade and Kender 1983; Silberberg et al. 1986; Thompson and Mundy 1987]). Unlike earlier approaches, however, we develop a simple closed-form solution for computing the position and orientation of an object under this imaging model.

There are two key observations underlying the new transformation method. First, the position and orientation of a rigid, solid object is determined up to a reflective ambiguity by the position and orientation of some plane of the object (under the affine imaging model). This plane need not be a surface of the object, but rather can be any "virtual" plane defined by features of the object. Second, the three-dimensional position and orientation of an object can be recovered from the affine transformation that relates such a plane of the model to the image plane.

We have implemented a system that uses this transformation method to determine possible alignments of a model with an image, and then verifies those alignments by transforming the model to image coordinates. Local features are extracted from an image, where each feature defines a position and an orientation (e.g., corners and inflections of edge contours). Pairs of two such model and image features are used to compute possible alignments of an object with an image. Thus, in the worst case, each pair of model features and each pair of image features will define a possible alignment of a model with an image. Perceptual grouping (e.g., [Lowe 1987]) can be used to further limit the number of model and image feature pairs that are considered, but we do not consider such techniques in this paper.

### *1.1 Model-Based Recognition by Alignment*

Model-based recognition systems vary along a number of major dimensions, including: (i) the types of features extracted from an image, (ii) the class of allowable transformations from a model to an image, (iii) the method of computing or estimating a transformation, (iv) the technique used to hypothesize possible transformations, and (v) the criteria for verifying the correctness of hypothesized matches. A number of existing approaches are surveyed in section 6. In this section we briefly discuss the design of the alignment recognition system in terms of these five dimensions.

In our recognition system, local features are extracted from intensity edge contours. These features are based on inflections and corners in the edge contours. The location of a corner or inflection defines the *position* of the feature, and the orientation of the tangent to the contour at that point defines the *orientation* of the feature. While extended features such as symmetry axes can also be used to align a model with an image, they are relatively sensitive to partial occlusion. Therefore, the current system relies only on local features.

A valid transformation from a model coordinate frame to the image coordinate frame consists of a rigid three-dimensional motion and a projection. We use a "weak-perspective" imaging model in which true perspective projection is approximated by orthographic projection plus a scale factor. The underlying idea is that under most viewing conditions, a single scale factor suffices to capture the size change that occurs with increasing distance (such a model has also been used by [Brooks 1981; Lamdan et al. 1988; Thompson and Mundy 1987]). Under this affine model, a good approximation to perspective projection is obtained except when an object is deep (i.e., large in the viewing direction) with respect to its distance from the viewer (e.g., railroad tracks going off to the horizon).

The recognition system we have implemented operates by considering the possible alignments of a model with an image. The maximum possible number of such alignments is polynomial in the number of model and image features. If each feature defines just a single point, for $m$ model points and $n$ image points there are $\binom{m}{3}\binom{n}{3}3!$ possible different alignments, resulting from pairing each triple of model points with each triple of image points. Thus for point features the worst-case number of hypotheses is $O(m^3n^3)$. The implementation that we describe below relies on features that each consist of a point and an orientation. For such features, each pairing of two model and image features defines a transformation, resulting at most $O(m^2n^2)$ possible hypotheses. No labeling or grouping of features is performed by the current implementation, thus reliable perceptual grouping methods could be used to improve the expected-time performance of the system.

Each hypothesized alignment can be verified in time proportional to the size of the model, by comparing the transformed model with the image. Thus the overall worst-casee running time of the system is $O(m^3n^2 \log n)$ —$O(m \log n)$ time to verify each of $O(m^2n^2)$ hypotheses. This relatively low-order polynomial running time contrasts with a number of other major methods of hypothesizing transformations: search for sets of corresponding features [Grimson and Lozano-Pérez 1987; Lowe 1987], and search for clusters of transformations [Silberberg et al. 1986; Thompson and Mundy 1987]. The former type of method is exponential in the number of features in the worst case. The latter type of method involves multi-dimensional clustering, for which the solution methods are either approximate (e.g., the generalized Hough transform) or iterative (e.g., $k$-means). Methods of the latter type are often not well suited to recognition in cluttered scenes (see [Grimson and Huttenlocher 1990]).

In the final stage of the recognition process, each hypothesized alignment of a model with an image is verified by comparing the transformed edge contours of the model with nearby edge contours in the image. This reduces the likelihood of false matches by using a more complete representation of an object than just the local features from which an alignment is hypothesized. In contrast, many existing recognition systems accept hypotheses that bring just a few local model features into approximate correspondence with image features [Fischler and Bolles 1981; Lamdan et al. 1988; Silberberg et al. 1986].

### *1.2 The Recognition Task*

The recognition problem addressed in this paper is that of identifying a solid object at an unknown position and orientation, from a single two-dimensional view of a three-dimensional scene (3D from 2D recognition). Thus an object has three translational and three rotational degrees of freedom, which must be recovered from two-dimensional sensory data. The input to the recognizer is a grey-level image and a set of three-dimensional models. The models are represented as a combination of a wire frame and local point features (vertexes and inflection points). The output of the recognition process is a set of model and transformation pairs, where each transformation maps the specified model onto an instance in the image.

The imaging process is assumed to be well approximated by orthographic projection plus a scale factor, as discussed in the following section. It is also assumed that the transformation from an object to an image is rigid, or is composed of a set of locally rigid parts. Under these assumptions there are six parameters for the image of an object under a rigid-body motion: a three-dimensional rotation, a two-dimensional translation in the image plane, and a scale factor.

Edge contour features are used for hypothesizing and verifying alignments of a model with an image, so it is assumed that objects are identifiable on the basis of the shape of their edge contours (surface discontinuities). Thus an object must have edge contours that are relatively stable over small changes in viewpoint, which is true of most objects that do not have smoothly changing surfaces. Recent work by Basri and Ullman [1988] addresses the problem of aligning smooth solid objects with a two-dimensional image.

There are relatively few constraints on the kinds of scenes in which objects can be recognized. An object can be partly occluded and highly foreshortened, and the scene can be very cluttered. The major limitation on clutter is the performance of the edge detector on images that have many intensity edges in close proximity. The level of image complexity is illustrated in figure 1, which contrasts with the relatively simple images on which a number of 3D from 2D recognition systems have been tested (e.g., [Lamdan et al. 1988; Silberberg et al. 1986; Linainmaa et al. 1985]).

## 2 The Transformation Method

A central problem in model-based recognition is the efficient and accurate determination of possible transformations from a model to an image. In this section, we show that three pairs of corresponding model and image points specify a unique (up to a reflection) transformation from a three-dimensional object coordinate frame to a two-dimensional image coordinate frame. Based on this result, a closed-form solution is developed for computing the transformation that aligns a model with an image using three pairs of corresponding model and image points. The method only involves solving a second-order system of two equations.

### *2.1 The Imaging Model*

3D from 2D recognition involves inverting the projection that occurs from the three-dimensional world into a two-dimensional image, so the imaging process must be modeled in order to recover the position and orientation of an object with respect to an image. We assume a coordinate system with the origin in the image plane, $I$, and with the $z$-axis (the optic axis) normal to $I$.

Under the perspective projection model of the imaging process, a transformation from a solid model to an image can be computed from six pairs of a model and image points. The equations for solving this problem are relatively unstable, and the most successful methods use more than six points and an error minimization procedure such as least squares [Lowe 1987]. When the parameters of the camera (the focal length and the location of the viewing center in image coordinates) are known, then it has been conjectured that three corresponding model and image points can be used to solve for up to four possible transformations from a model

*Fig. 1.* The type of scene of interest. The object to be recognized is the partly occluded polyhedron at the center.

plane to the image plane [Fischler and Bolles 1981]. There are two drawbacks to this approach. First, a calibration operation is required to determine the camera parameters, so if the camera has a variable focal length it is necessary to recalibrate when the focal length is changed. Second, the solution method is complicated and has hence not actually been used in an implementation [Fischler and Bolles 1981].

While perspective projection is an accurate model of the imaging process, the magnitude of the perspective effects in most images is relatively small compared with the magnitude of the sensor noise (such as the uncertainty in edge location). This suggests that a more robust transformation may be obtained by ignoring perspective effects in computing a transformation between a model and an image. Furthermore, even when a perspective transformation is computed, other properties of a recognition method may prevent perspective effects from being recovered. For instance, the SCERPO system [Lowe 1987] uses approximately parallel line segments as features, but parallelism is not preserved under perspective transformation. Thus the choice of features limits the recognition process to cases of low perspective distortion, even though a perspective transformation is being computed.

Under orthographic projection, the direction of projection is orthogonal to the image plane (see figure 2), and thus an object does not change size as it moves further from the camera. If a linear scale factor is added to orthographic projection, then a relatively good approximation to perspective is obtained. The approximation becomes poor when an object is deep with respect to its distance from the viewer, because a single scale factor cannot be used for the entire object. Thus if $z_{min}$ is the distance to the closest part of the object and $z_{max}$ is the distance to the farthest part of the object, we require that $z_{max} - z_{min} \ll (z_{max} + z_{min})/2$.

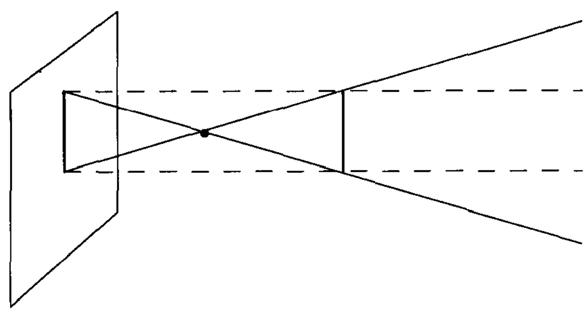

*Fig. 2.* Perspective projection approximated by orthographic projection plus a scale factor.

Under this imaging model, transformation from a three-dimensional model to a two-dimensional image consists of a two-dimensional translation (in $x$ and $y$ in the image plane), a three-dimensional rotation, and a scale factor that depends on the distance to and size of the object. Several recognition systems [Brooks 1981; Cyganski and Orr 1985; Lamdan et al. 1988; Thompson and Mundy 1987] have used this imaging model. Unlike the current method, however, these systems have not solved the problem of how to compute a three-dimensional transformation directly from corresponding model and image points. Instead, they either use a heuristic approach to finding a transformation [Brooks 1981], store views of an object from all possible angles in order to approximate the transformation by table lookup [Thompson and Mundy 1987], or restrict recognition the planar objects [Cyganski and Orr 1985].

## 2.2 The Alignment Transformation Exists and Is Unique

The major result of this section is that the correspondence of three non-collinear points is sufficient to determine a transformation from a three-dimensional model coordinate frame to a two-dimensional image coordinate frame. This transformation specifies a three-dimensional translation, rotation, and scale of an object such that it is mapped into the image by an orthographic projection. The result is shown in two stages. First we note that an affine transformation of the plane is uniquely defined by three pairs of non-collinear points. Then we show that an affine transformation of the plane uniquely defines a similarity transformation of space, specifying the orientation of one plane with respect to another, up to a reflection.

The results reported by Ullman [1987] and Huttenlocher and Ullman [1988] are precursors to the result established here. A similar method was presented by Cyganski and Orr [1985], but was not shown to yield a transformation that always exists and is unique. The recovery of skewed symmetry [Kanade and Kender 1983] relies on a restricted version of the same result, but the existence and uniqueness of the transformation was not established there. A related result is the determination of at most four distinct perspective transformations from three corresponding points [Fischler and Bolles 1981], however the equation counting argument presented there does not guarantee the existence and four-way uniqueness of a solution.

LEMMA 1. *Given three non-collinear points* $\mathbf{a}_m$, $\mathbf{b}_m$, *and* $\mathbf{c}_m$ *in the plane, and three corresponding points* $\mathbf{a}_i$, $\mathbf{b}_i$, *and* $\mathbf{c}_i$ *in the plane, there exists a unique affine transformation,* $A(\mathbf{x}) = \mathbf{L}\mathbf{x} + \mathbf{b}$, *where* $\mathbf{L}$ *is a linear transformation and* $\mathbf{b}$ *is a translation, such that* $A(\mathbf{a}_m) = \mathbf{a}_i$, $A(\mathbf{b}_m) = \mathbf{b}_i$, *and* $A(\mathbf{c}_m) = \mathbf{c}_i$.

This fact is well known in affine geometry, and its proof is straightforward (for example, see [Klein 1939]).

DEFINITION 1. *A transformation,* $\mathbf{T} : V \to V$, *is* symmetric *when*

$$\|\mathbf{T}\mathbf{v}_1\| = \|\mathbf{T}\mathbf{v}_2\| \Leftrightarrow \|\mathbf{v}_1\| = \|\mathbf{v}_2\| \tag{1}$$

$$\mathbf{T}\mathbf{v}_1 \cdot \mathbf{T}\mathbf{v}_2 = 0 \Leftrightarrow \mathbf{v}_1 \cdot \mathbf{v}_2 = 0 \tag{2}$$

*for any* $\mathbf{v}_1$, $\mathbf{v}_2$ *in* $V$.

PROPOSITION 1. *Given a linear transformation of the plane,* $\mathbf{L}$, *there exists a unique (up to a reflection) symmetric transformation of space,* $\mathbf{U}$, *such that* $\mathbf{L}\mathbf{v} = \Pi(\mathbf{U}\mathbf{v}^*)$ *for any two-dimensional vector* $\mathbf{v}$, *where* $\mathbf{v}^* = (x, y, 0)$ *for any* $\mathbf{v} = (x, y)$, *and* $\Pi$ *is orthographic projection into the x-y plane.*

The structure of the equivalence between $\mathbf{L}$ and $\mathbf{U}$ stated in the proposition is,

$$\begin{array}{ccc} \mathbf{V}^2 & \xrightarrow{\mathbf{L}} & \mathbf{V}^2 \\ \downarrow * & & \uparrow \Pi \\ \mathbf{V}^3 & \xrightarrow{\mathbf{U}} & \mathbf{V}^3 \end{array}$$

where $\mathbf{V}^2$ and $\mathbf{V}^3$ are two- and three-dimensional vector spaces, respectively. The $*$ operator lifts a vector $(x, y) \in \mathbf{V}^2$ into $\mathbf{V}^3$ by adding a third coordinate $z = 0$. The $\Pi$ operator projects a vector $(x, y, z) \in \mathbf{V}^3$ into $\mathbf{V}^2$ by selecting the $x$ and $y$ coordinates.

*Proof.* Clearly some transformation $\mathbf{U}$ satisfying $\mathbf{L}\mathbf{v} = \Pi(\mathbf{U}\mathbf{v}^*)$ exists for any given $\mathbf{L}$, for instance just embed $\mathbf{L}$ in the upper-left part of a $3\times3$ matrix, with the remaining entries all zero. What must be shown is that there is always a $\mathbf{U}$ that is symmetric, and that this $\mathbf{U}$ is unique up to a reflection.

In order for $\mathbf{U}$ to be symmetric, it must satisfy the two properties of definition 1. We show that these properties are equivalent to two equations in two unknowns, and that these equations always have exactly two solutions differing only in sign. Thus a symmetric $\mathbf{U}$ always exists and is unique up to a reflection corresponding to the sign ambiguity.

Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be orthonormal vectors in the plane, with

$$\mathbf{e}_1' = \mathbf{L}\mathbf{e}_1$$

$$\mathbf{e}_2' = \mathbf{L}\mathbf{e}_2$$

If $\mathbf{v}_1 = \mathbf{U}\mathbf{e}_1^*$ and $\mathbf{v}_2 = \mathbf{U}\mathbf{e}_2^*$, then by the definition of $\mathbf{U}$ we have,

$$\mathbf{v}_1 = \mathbf{e}_1' + c_1\mathbf{z}$$

$$\mathbf{v}_2 = \mathbf{e}_2' + c_2\mathbf{z}$$

where $\mathbf{z} = (0, 0, 1)$, and $c_1$ and $c_2$ are unknown constants.

U is symmetric iff

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$$

$$\|\mathbf{v}_1\| = \|\mathbf{v}_2\|$$

because $\mathbf{e}_1^*$ and $\mathbf{e}_2^*$ are orthogonal and of the same length.

From

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$$

$$(\mathbf{e}_1' + c_1\mathbf{z}) \cdot (\mathbf{e}_2' + c_2\mathbf{z}) = 0$$

$$\mathbf{e}_1' \cdot \mathbf{e}_2' + c_1c_2 = 0$$

and hence

$$c_1c_2 = -\mathbf{e}_1' \cdot \mathbf{e}_2'$$

The right side of this equality is an observable quantity, because we know $\mathbf{L}$ and can apply it to $\mathbf{e}_1$ and $\mathbf{e}_2$ to obtain $\mathbf{e}_1'$ and $\mathbf{e}_2'$. Call $-\mathbf{e}_1' \cdot \mathbf{e}_2'$ the constant $k_1$.

In order for

$$\|\mathbf{v}_1\| = \|\mathbf{v}_2\|$$

it must also be that

$$\|\mathbf{v}_1\|^2 = \|\mathbf{v}_2\|^2$$

$$\|\mathbf{e}_1'\|^2 + c_1^2 = \|\mathbf{e}_2'\|^2 + c_2^2$$

$$c_1^2 - c_2^2 = \|\mathbf{e}_2'\|^2 - \|\mathbf{e}_1'\|^2$$

Again the right side of the equality is observable, call it $k_2$.

It remains to be shown that these two equations,

$$c_1c_2 = k_1$$

$$c_1^2 - c_2^2 = k_2$$

always have a solution that is unique up to a sign ambiguity. Substituting $c_2 = k_1/c_1$ into the latter equation and rearranging terms we obtain

$$c_1^4 - k_2c_1^2 - k_1^2 = 0$$

a quadratic in $c_1^2$. Substituting $u$ for $c_1^2$ yields

$$u = \frac{1}{2}(k_2 \pm \sqrt{k_2^2 + 4k_1^2})$$

We are only interested in positive solutions for $u$, because $u = c_1^2$. There can be only one positive solution, as $4k_1^2 \geq 0$, and thus the discriminant, $\delta$ satisfies $\delta \geq k_2$. Hence there are exactly two real solutions for $c_1 = \pm\sqrt{u}$. For the positive and negative solutions to $c_1$ there are corresponding solutions of opposite sign to $c_2$ because $c_1c_2 = k_1$.

The equation for $u$ does not have a solution when $c_1 = 0$, because the substitution for $c_2$ is undefined. When $c_1 = 0$, however, $c_2 = \pm\sqrt{-k_2}$, which always has two solutions because $k_2 \leq 0$. Recall

$$\|\mathbf{e}_1'\|^2 + c_1^2 = \|\mathbf{e}_2'\|^2 + c_2^2$$

therefore, because $c_1 = 0$,

$$\|e_1'\|^2 \geq \|e_2'\|^2$$

and hence

$$k_2 = \|e_2'\|^2 - \|e_1'\|^2 \leq 0$$

Thus there are always exactly two solutions for $c_1$ and $c_2$ differing in sign. These equations have a solution iff the symmetric transformation, U, exists. So there are always exactly two solutions for U which differ in the sign of the z component of $\mathbf{x}'$, where $\mathbf{x}' = \mathbf{U}\mathbf{x}$, and hence the sign difference corresponds to a reflective ambiguity in U. ■

We now establish proposition 2, the major result of this section.

PROPOSITION 2. *Given three non-collinear points* $\mathbf{a}_m$, $\mathbf{b}_m$, *and* $\mathbf{c}_m$ *in the plane, and three corresponding points* $\mathbf{a}_i$, $\mathbf{b}_i$, *and* $\mathbf{c}_i$ *in the plane, there exists a unique similarity transformation (up to a reflection),* $Q(\mathbf{x}) = \mathbf{U}\mathbf{x} + \mathbf{b}$, *where* U *is symmetric and* b *is a translation, such that* $\Pi(Q(\mathbf{a}_m^*)) = \mathbf{a}_i$, $\Pi(Q(\mathbf{b}_m^*)) = \mathbf{b}_i$, *and* $\Pi(Q(\mathbf{c}_m^*)) = \mathbf{c}_i$, *where* $\mathbf{v}^* = (x, y, 0)$ *for any* $\mathbf{v} = (x, y)$, *and* $\Pi$ *is orthographic projection into the x-y plane.*

*Proof.* From lemma 1 three non-collinear points define a unique affine transformation such that $A(\mathbf{a}_m) = \mathbf{a}_i$, $A(\mathbf{b}_m) = \mathbf{b}_i$, and $A(\mathbf{c}_m) = \mathbf{c}_i$. By the definition of an affine transformation, $A$ consists of two components, a translation vector b and a linear transformation L. By proposition 1, L defines a unique (up to a reflection) symmetric transformation of space, U, such that $\mathbf{L}\mathbf{v} = \Pi(\mathbf{U}\mathbf{v}^*)$. ■

### *2.3 Computing the Transformation*

The previous section established the existence and uniqueness of the similarity transformation $Q$, given three corresponding model and image points. This section shows how to compute $Q$ (and the two-dimensional affine transformation $A$) given three pairs of points $(\mathbf{a}_m, \mathbf{a}_i)$, $(\mathbf{b}_m, \mathbf{b}_i)$ and $(\mathbf{c}_m, \mathbf{c}_i)$, where the image points are in two-dimensional sensor coordinates and the model points are in three-dimensional object coordinates.

**Step 0:** Rotate and translate the model so that the new $\mathbf{a}_m$ is at the origin, (0, 0, 0), and the new $\mathbf{b}_m$ and $\mathbf{c}_m$ are in the $x$-$y$ plane. This operation is performed offline for each triple of model points.

**Step 1:** Define the translation vector $\mathbf{b} = -\mathbf{a}_i$, and translate the image points so that the new $\mathbf{a}_i$ is at the origin, the new $\mathbf{b}_i$ is at $\mathbf{b}_i - \mathbf{a}_i$ and the new $\mathbf{c}_i$ is at $\mathbf{c}_i - \mathbf{a}_i$.

**Step 2:** Solve for the linear transformation

$$\mathbf{L} = \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{pmatrix}$$

given by the two pairs of equations in two unknowns

$$\mathbf{L}\mathbf{b}_m = \mathbf{b}_i$$

$$\mathbf{L}\mathbf{c}_m = \mathbf{c}_i$$

These first two steps yield a unique affine transformation, $A$, where $\mathbf{b}$ is the translation vector, and $\mathbf{L}$ is the linear trasformation, as long as the three model points are not collinear.

**Step 3:** Solve for $c_1$ and $c_2$, using

$$c_1 = \pm\sqrt{\frac{1}{2}\left(w + \sqrt{w^2 + 4q^2}\right)}$$

and

$$c_2 = \frac{-q}{c_1}$$

where

$$w = l_{12}^2 + l_{22}^2 - (l_{11}^2 + l_{21}^2)$$

and

$$q = l_{11}l_{12} + l_{21}l_{22}$$

**Step 4:** From proposition 1 we know that there are two possible symmetric matrixes, call them $s\mathbf{R}^+$ and $s\mathbf{R}^-$, differing by a reflection. These 3×3 matrixes have the 2×2 matrix $\mathbf{L}$ embedded in the upper left, because they effect the same two-dimensional transformation as $\mathbf{L}$. Using equations (1) and (2) we can solve for the remaining entries of $s\mathbf{R}^+$ and $s\mathbf{R}^-$ given the 2×2 matrix $\mathbf{L}$, resulting in

$$s\mathbf{R}^+ = \begin{pmatrix} l_{11} & l_{12} & (c_2l_{21} - c_1l_{22})/s \\ l_{21} & l_{22} & (c_1l_{12} - c_2l_{11})/s \\ c_1 & c_2 & (l_{11}l_{22} - l_{21}l_{12})/s \end{pmatrix}$$

where

$$s = \sqrt{l_{11}^2 + l_{21}^2 + c_1^2}$$

This yields the complete transformation, $Q$, with translation vector $\mathbf{b}$ and scale and rotation $s\mathbf{R}$. The image coordinates of a transformed model point, $\mathbf{x}' = s\mathbf{R}\mathbf{x} + \mathbf{b}$ are then given by the $x$ and $y$ coordinates of $\mathbf{x}'$.

The reflected solution $s\mathbf{R}^-$ is identical to $s\mathbf{R}$ except the entries $r_{13}$, $r_{23}$, $r_{31}$, and $r_{32}$ are negated.

This method of computing a transformation is relatively fast, because it involves a small number of terms, none of which are more than quadratic. Our implementation using floating point numbers on a Symbolics 3650 takes about 3 milliseconds.

If there are more than three corresponding model and image points, then there is an equation of the form

$$\mathbf{L}\mathbf{x}_m = \mathbf{x}_i$$

for each pair $(\mathbf{x}_m, \mathbf{x}_i)$, and the system of equations for $\mathbf{L}$ is overdetermined. A solution can be obtained using an error minimization technique such as least squares as long as the model points used for computing $\mathbf{L}$ are coplanar.

### *2.4 Using Virtual Points and Orientations*

In addition to computing alignments from triples of edge contour points, it is possible to use the orientations of edge contours to induce virtual points. For example, suppose we are given two model points $\mathbf{a}_m$ and $\mathbf{b}_m$, with three-dimensional unit orientation vectors $\hat{\mathbf{a}}_m$, and $\hat{\mathbf{b}}_m$, respectively. Let $A_m$ be the line through $\mathbf{a}_m$ in the direction $\hat{\mathbf{a}}_m$, and $B_m$ be the line through $\mathbf{b}_m$ in the direction $\hat{\mathbf{b}}_m$. If $A_m$ and $B_m$ intersect, then they define a third point for alignment, $\mathbf{c}_m$, as illustrated in figure 3. Given similar conditions on two image points and orientations, a third image point, $\mathbf{c}_i$ can be defined. Now the alignment computation can be performed using three corresponding model and image points.

The stability of this method depends on the distance from the two given points, $\mathbf{a}_m$ and $\mathbf{b}_m$, to the intersection point, $\mathbf{c}_m$. If this distance is large, then a small amount of error in either of the two orientation vectors will cause a large positional error in the location of $\mathbf{c}_m$. In real images, the two orientation vectors are generally edge fragments that have a distinguished point at one end, but an uncertain endpoint at the other. A useful heuristic is to use only intersection points that

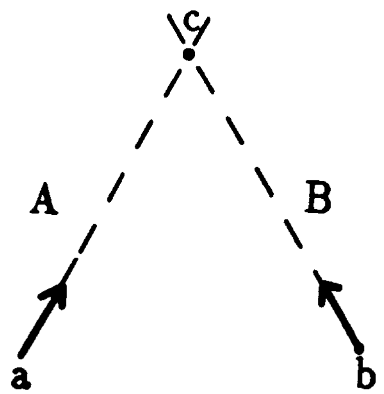


*Fig. 3.* Defining a third point from two points and two orientations.

are within some allowable distance of the endpoints of the actual segments. The alignment transformation can also be computed directly from a combination of points and lines, as described in [Shoham and Ullman 1988].

## 3 Feature Extraction

The previous section describes a method that uses three corresponding model and image points to compute a transformation from a model coordinate frame to the image coordinate frame. In this section we present a method for extracting simple, local point information from the intensity edges of an unknown image. The method relies on connected intensity-edge contours, so a grey-level image is first processed using a standard edge detector [Canny 1986] and the edge pixels are then chained together into singly-connected contours. For each resulting contour, local orientation and curvature of the contour are computed. The curvature is used to identify two types of local events along a contour: an inflection or a corner. The location of each inflection or corner and the tangent orientation at that point are then used as features for computing possible alignments of a model with an image.

### *3.1 Chaining Edge Contours*

The output of an edge detector is a binary array indicating the presence or absence of an edge at each pixel. This section describes a simple method of chaining neighboring eight-connected pixels together into contours. When a given pixel has only one or two neighbors the chaining process is unambiguous. Otherwise, a decision must be made as to which pixels belong together as part of an edge. The chaining algorithm works in two stages. First, all the unambiguous chains are formed; any pixel that has more than two neighbors is ambiguous and thus defines the start of a new chain. The second stage uses a process that we call *mutual favorite pairing* in order to merge neighboring chains into a single chain. This merging process continues until no two neighboring chains can be joined together.

The mutual favorite pairing procedure works as follows. Given a set of edge contours $\{x_i\}_{i=1,\ldots,n}$, each contour $x_i$ has an ordered sequence of possible matching contours. This ordering is computed using a cost function that measures how bad the resulting contour would be if the two contours were joined. Each contour $x_i$ is considered in turn. A contour $x_i$ is only merged with its best matching contour $x_j$ if $x_j$ in turn has $x_i$ ranked as its best match (hence the name "mutual favorite"). This process continues until no two contours (including the resulting merged contours) can be joined together. One slight complication over the above description is that each contour has two endpoints that are considered separately, because merging two contours actually occurs at a specific endpoint.

The cost function for sorting the candidate merges is based on the distances and tangent orientations between endpoints of a contour. For a given endpoint $\mathbf{e}_i$ of a contour, the cost of matching with another endpoint $\mathbf{e}_j$ of some other contour is defined as

$$D = \|\mathbf{e}_i - \mathbf{e}_j\|(\angle \mathbf{e}_i\mathbf{v} + \angle \mathbf{e}_j\bar{\mathbf{v}})$$

where $\|\mathbf{e}_i - \mathbf{e}_j\|$ is the distance between the endpoint, $\mathbf{v}$ is the unit tangent vector in the direction $\mathbf{e}_i - \mathbf{e}_j$, and $\bar{\mathbf{v}}$ is the unit vector in the opposite direction of $\mathbf{v}$.

An iteration of the mutual favorite pairing procedure considers the two endpoints of each chain separately. If two chains each rank the other as the best match (based on $D$) then they are merged together, and the merged ends of the chains are removed from further consideration. The iteration stops when no more endpoints to be merged are found. Clearly the process terminates, because there are a finite number of chains, and at each step either two chains are merged into one, or there are no merges and the iteration stops. Each iteration takes $O(n \log n)$ time for $n$ edges, and at least one chain is removed at each step, so the worst-case running time is $O(n^2 \log n)$. This worst case occurs only if all the chains have an endpoint near the same place in the image. In general, there are a small number of chains with endpoints near any given point, so the expected running time is effectively linear in the number of chains, rather than quadratic.

*3.2 Edge Orientation and Curvature*

For a curve $g$, parameterized by its arclength $s$, the local orientation at a point $g(s)$ along the curve is defined by the unit vector $\mathbf{T}_s$ that is tangent to the curve at that point. There are several ways of estimating the local orientation of a curve. The simplest method is to define a local neighborhood $d$, and estimate the orientation as $\mathbf{T}_s = g(s - d) - g(s + d)$. The major shortcoming of this method is sensitivity to noise, because each tangent is estimated using only two points. A method that is less sensitive to noise is to estimate the orientation at $g(s)$ by fitting a line to the points in the neighborhood, $g(s - d), \ldots, g(s + d)$.

A standard line-fitting technique is the least-squares method. In this case, we are concerned with minimizing the squared distance in the direction normal to the best-fitting line, rather than in the direction of one of the coordinate axes [Duda and Hart 1973]. The best-fitting line for a set of points must go through the centroid of that point set. Therefore, it is equivalent to translate the set of points to be centered at the origin, and then solve for the best-fitting line through the origin as characterized by its unit-normal vector, $\mathbf{N}$. This unit vector can be shown to be the $\mathbf{N}$ that minimizes

$$d^2\mathbf{N} = \mathbf{N}^T\mathbf{S}\mathbf{N}$$

where

$$\mathbf{S} = \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T$$

for the given points $\mathbf{v}_1, \ldots, \mathbf{v}_n$. This form is minimized by taking $\mathbf{N}$ to be the eigenvector of $\mathbf{S}$ that is associated with the smallest eigenvalue. A simple approximation to this is obtained by minimizing the $x$ or $y$ distance, depending on the slope of the best-fitting line [Huttenlocher and Ullman 1988].

Given the local tangent orientations, the curvature, $\kappa$, of a path in space $x = g(s)$, parameterized by arc length, $s$, is given by

$$\kappa\mathbf{N} = \frac{d\mathbf{T}}{ds}$$

where $\mathbf{T}$ is the unit tangent vector and $\mathbf{N}$ is the unit normal vector in the direction in which the path is turning. This formula has a special case for plane curves,

$$\kappa = \frac{d\psi}{ds} = \frac{1}{\rho}$$

where $\psi$ is the angle of the slope of the tangent line, and $\rho$ is the radius of the osculating circle, tangent to the curve. We use this method to compute the curvature from the tangent vectors.

Most computational vision algorithms do not compute curvature using the tangent vector, but rather parameterize $g(s)$ in terms of separate functions of $x$ and $y$, $x(s)$ and $y(s)$, so that the curvature is given by

$$\kappa = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}$$

using Newton's dot notation for the derivatives of $x$ and $y$ with respect to $s$. This method of computing curvature is quite sensitive to noise, however, due to the computation of second-derivative quantities from pointwise image measurements.

A number of methods have been proposed for smoothing curvature [Mokhtarian and Mackworth 1986; Horn and Weldon 1985; Lowe 1988]. These methods are concerned with reconstructing a smoothed version of a curve. In contrast, we are interested only in extracting certain local interest points along a curve, such as corners and inflections. Thus the following rather simple smoothing procedure is sufficient. The curvature of a path parameterized by arclength, $\kappa(s)$, is smoothed. Events such as zeros of curvature and high curvature points are then identified in the smoothed curvature. Each of these events occurs at some particular location along the arc, $s$, For each event, the corresponding location $s$ along the original contour is marked as the point where the event occurs.

In areas of low curvature, there may be a number of small inflections that are relatively unimportant. To filter out these insignificant zeros of curvature we use a method based on the magnitude of the area of $\kappa(s)$ between two successive zero crossings. When the area is smaller than a threshold value, the neighboring zero crossings are discarded as insignificant (see figure 4a). This method is less sensitive to noise than using a threshold on the height of the peak between two zero crossings (shown in part (b) of the figure), because it does not rely on a value at a single point. Also, unlike using a threshold on the slope at the zero crossing (shown in part (c) of the figure), the method does not exclude low-slope zero crossings for which there is still a large peak between the two zero crossings. It should be noted that the computation of the area of $\kappa(s)$ between two successive zero crossings is trivial. It is just the total turning angle of the contour over that region of arclength (i.e., the angle between the tangent orientations to the original curve at those two corresponding locations).

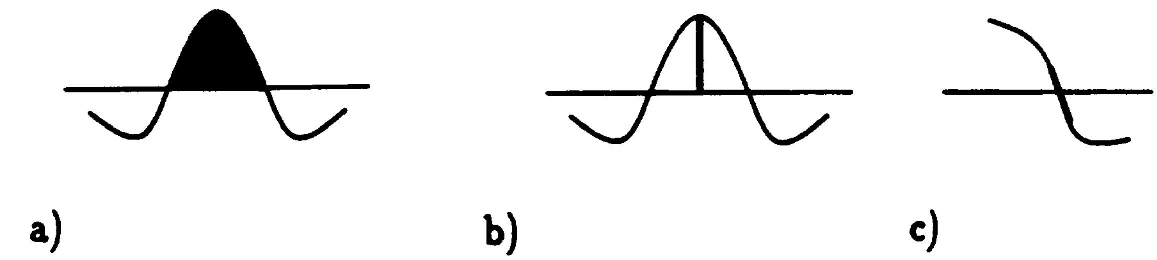


*Fig. 4.* Methods of filtering zero crossings: (a) the area of the curve between two zeros; (b) the height of the largest peak between two zeros; and (c) the slope of the curve at the zero crossing.

### *3.3 Alignment Features*

Using the edge curvature information, inflection points and corners are identified along the edge contours. A *feature* consists of the location of a corner or inflection, and the tangent orientation at that point. Inflections are located where the curvature changes sign, and corners are located at high curvature points. Figure 5a shows a contour and the inflection points found by marking the significant zero crossing of the curvature (as described in the previous section). Figure 5b shows the same contour and the corners identified by marking the peaks in the curvature function.

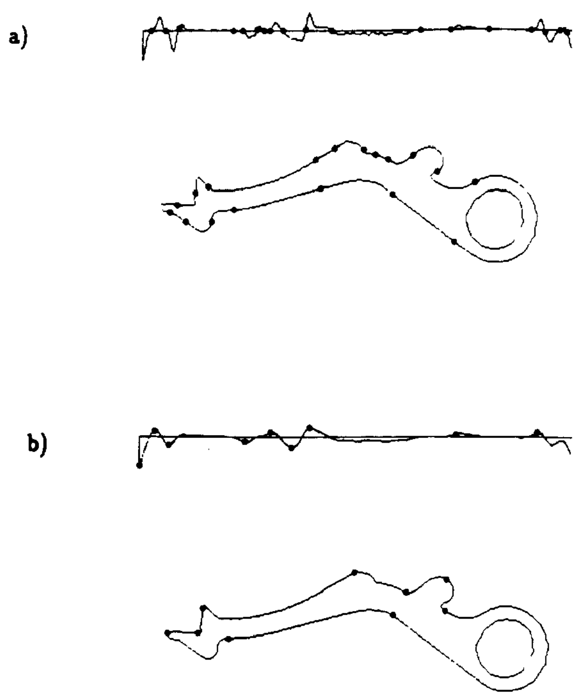


*Fig. 5.* Features consist of the locations and tangent orientation of corners and inflection points: (a) the inflections in a contour, (b) the corners.

In order to align a model with an image, two pairs of corresponding model and image features are required (note that in the model the tangent vectors must be coplanar with the two points). For a model with $m$ features and an image with $n$ features, there are $\binom{m}{2}$ pairs of model features and $\binom{n}{2}$ pairs of image features, each of which may define a possible alignment of the model with the image. Each of these $O(m^2n^2)$ alignments can be verified by transforming the model to the image coordinate frame and checking for additional evidence of a match, as described in the following section.

## 4 Verification

Once a potential alignment transformation has been computed, it must be determined whether or not the transformation actually brings the object model into correspondence with an instance in the image. This is done by transforming the model to image coordinates, and checking that the transformed model edges correspond to edges in the image. The verification process is of critical importance because it must filter out incorrect alignments without missing correct ones.

The verification method is hierarchical, starting with a relatively simple and rapid check to eliminate many false matches, and then continuing with a more accurate and slower check. The initial verification procedure compares the segment endpoints of the model with the segment endpoints of the image. Recall that a segment endpoint is defined at inflections and at corners. Each point thus has an associated orientation, up to a reflective ambiguity, defined by the orientation of the edge contour at that point. A model point correctly matches an image point if both the position and orientation of the transformed model point are within allowable error ranges of a corresponding image feature (currently 10 pixels and $\pi/10$ radians). Those alignments where a certain percentage (currently half) of the model points are correctly matched are then verified in more detail.

The detailed matching procedure compares the entire edge contours of a transformed model with the image. Each segment of the transformed model contour is matched against the image segments that are nearby. The comparison of contours is relatively time consuming, but reduces the likelihood that an incorrect match will be accidentally accepted. The initial verification generally eliminates many hypotheses, however, so this more time-consuming procedure is applied only to a small number of hypotheses.

The comparison of a transformed model segment with an image is intended to distinguish between an accidental and a nonaccidental match. For instance, an accidental match is highly unlikely if a transformed model segment exactly overlaps an image segment such that they coterminate. If a transformed model segment lies near some image segments that do not coterminate with it, then there is a higher likelihood that the match is accidental. We define three types of evidence for a match: positive evidence, neutral evidence, and negative evidence, as illustrated in figure 6 (model edges are shown in bold). A model edge that lies near and approximately coterminates with one or more image edges is positive evidence of a match, as shown in the first part of the figure. A model edge that lies near an image edge that is much longer or much shorter than the model edge is neutral evidence of a match, and is treated as if there were no nearby edge. A model edge that is crossed by one or more image edges, at a sufficiently different orientation than the model edge, is negative evidence of a match.

The verification process uses a table of image-edge segments stored according to $x$ position, $y$ position, and orientation. A given image segment, $i$, is entered into

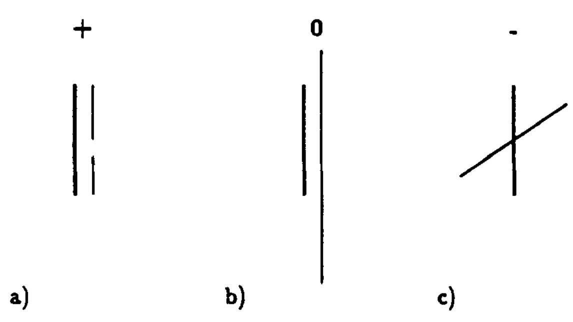


*Fig. 6.* Matching model segments to image segments, (a) positive evidence, (b) neutral evidence, similar to no matching edge, (c) negative evidence.

the table by taking each point, $g(s)$, along its edge contour and the corresponding tangent orientation, $\mathbf{T}_s$, and storing the triple ($i$, $g(s)$, $\mathbf{T}_s$) in the table using quantized values of $g(s)$ and $\mathbf{T}_s$. A positional error of 10 pixels and an orientation error of $\pi/10$ radians are allowed, so a given point and orientation will generally specify a set of table entries in which to store a given triple.

In order to match a model segment to an image, each location and orientation along the transformed model contour is considered in turn. A given location and orientation are quantized and used to retrieve possible matching image segments from the table.

## 5 The Matching Method

We have implemented a 3D from 2D recognition system based on the alignment method. The system uses the simple, local features described in the preceding sections: corners and inflections extracted from intensity edges. Each pair of two model and image features defines a possible alignment of the model with the image. These alignments are verified by transforming the edge contours of the model into image coordinates, and comparing the transformed model edges with nearby image edges. Alignments that account for a high percentage of a model's contour are kept as correct matches.

As described in section 3, a grey-level image is processed to extract intensity edges [Canny 1986]. The edge pixels are chained into contours by following unambiguous 8-way neighbors; and neighboring chains are merged together using a mutual favorite-pairing procedure. High curvature points along the contour are identified as corners, and zero crossings of curvature are identified as inflection points. The possible alignments defined by these local features are verified by transforming the edge contours of the model to image coordinates and comparing the transformed edges with nearby image edges, as described in section 4.

A model consists of the three-dimensional locations of the edges of its surfaces (a wire-frame), and the corner and inflection features that are used for computing possible alignments. In the current implementation we use several different models of an object from different viewpoints. It is also possible to use a single three-dimensional model and a hidden-line elimination algorithm. For example, in the current system a cube is represented by 9 line segments in 3D coordinates and by 12 corner features at the vertexes (one feature for each

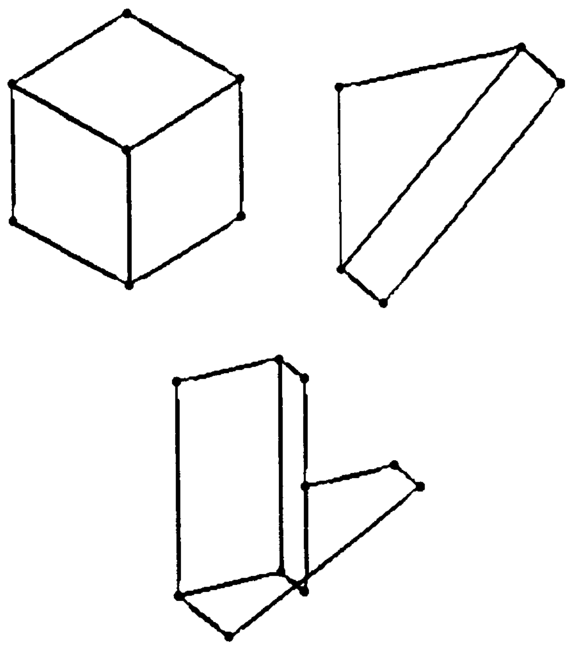

*Fig. 7.* The three solid object models that were matched to the images.

corner of each of the three faces). Three such models are shown in figure 7. The edge contours of each model are represented as chains of three-dimensional edge-pixel locations. Properties of a surface itself are not used (e.g., the curvature of the surface), only the bounding contours.

### 5.1 The Search Algorithm

Once the features are extracted from an image, all pairs of two model and image features are used to solve for potential alignments, and each alignment is verified using the complete model edge contour. If an alignment maps a sufficient percentage of the model edges onto image edges, then it is accepted as a correct match of an object model to an instance in the image. In order to speed up the matching process on a serial machine the image feature pairs are ordered, and once an image feature is accounted for by a verified match it is removed from further consideration. The ordering of features makes use of two principles. First, those pairs of image features that lie on the same contour are considered before pairs that do not. Second, feature pairs near the center of the image are tried before those near the periphery.

The exact matching algorithm is given below, where the variable TRANSES is used to accumulate the accepted transformations, IMAT is a Boolean array indicating the image features that have already been matched to some model feature, MFEAT returns all the features of a model, and IFEAT returns the features in an image. The procedure ALIGN computes the possible alignment transformations from a pair of two model and image features, using the method developed in section 2.2. VERIFY returns all the matching model and image edges when the given transformation is applied to the model, using the method described in section 4. GOOD-ENOUGH checks that a match accounts for sufficiently many model edges, and MATCHED-FEATURES returns the image features accounted for by a match.

```
for m1,m2 in mfeat(model) do
 for i1,i2 in ifeat(image) do
  if not imat(ifeat)
   then for trans in align(m1,m2,i1,i2) do
    match ← verify(trans,model,image)
     if good-enough(match)
      then put trans in transes
       for ifeat in matched-features(match) do
         imat(ifeat) ← true
```

An alignment specifies two possible transformations, that differ by a reflection of the model about the three model points. A further ambiguity is introduced by not knowing which of the two model features in a pair corresponds to which of the two image features. Thus each feature pair defines four possible alignment transformations.

To verify an alignment (procedure VERIFY), each model edge segment is transformed into image coordinates in order to determine if there are corresponding image edges. The image edges are stored in a table according to position and orientation, so only a small number of image edges are considered for each model edge, as discussed in section 4. When an alignment matches a model to an image, each alignment feature in the image that is part of an accounted for edge is removed from further consideration by the matcher (by marking it in IMAT). This can greatly reduce the set of matches considered, at the cost of missing a match if image features are incorrectly incorporated into verified alignments of other objects. This is unlikely, however, because the verifier has a low chance of accepting false matches.

The worst-case running time of the matching process is $O(m^3n^2 \log n)$ for $m$ model features and $n$ image features. This is because there are $O(m^2n^2)$ possible alignments, and the verification of each model can be done in time $O(m \log n)$ (each of the $m$ model features must be transformed and looked up in the image; by sorting the image features this lookup can be done in worst-case logarithmic time). In practice, however, it is not necessary to consider all pairs of features, because each correct alignment will eliminate a number of image features from consideration. To the extent that any reliable perceptual grouping can be done, the average running time can be further improved. Finally, each alignment computation is independent of all the others, so the method is well suited to implementation on a massively parallel machine.

### *5.2 Examples*

The recognizer has been tested on images of simple polyhedral objects in relatively complex scenes, and on piles of laminar parts. The images were taken under normal lighting conditions in offices or laboratories. The method is not restricted to the recognition of polyhedral objects, only to objects with well-defined local features (see also [Basri and Ullman 1988] for extensions to matching smooth solids).

In the examples presented here, three different objects: a cube, a wedge, and an object consisting of a rectangular block and a wedge glued together, were matched to each of the images. These models are shown in figure 7, with the corner features indicated by dots. Each corner feature has an associated orientation, and thus each dot in figure 7 may actually represent more than one feature. For example the central vertex of the cube corresponds to three features, one on each face. There are seven features for the wedge, and twelve features each for the other two objects.

Figures 8–10 illustrate the performance of the recognizer on several images of solid objects. Part (a) of each figure shows the grey-level image; part (b) shows the Canny edges; part (c) shows the features (marked by

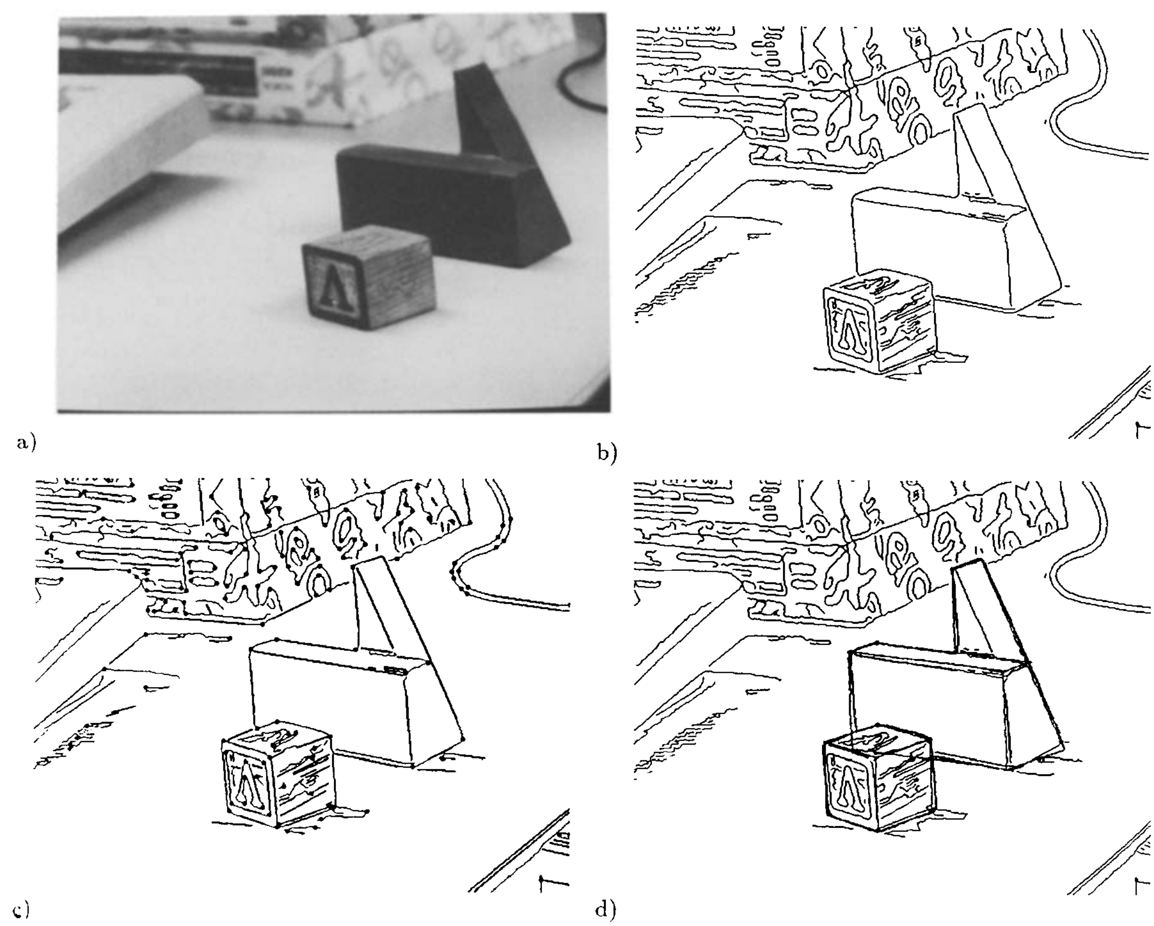


*Fig. 8.* The output of the recognizer: (a) grey-level image input, (b) Canny edges, (c) edge segments, (d) recovered instances.

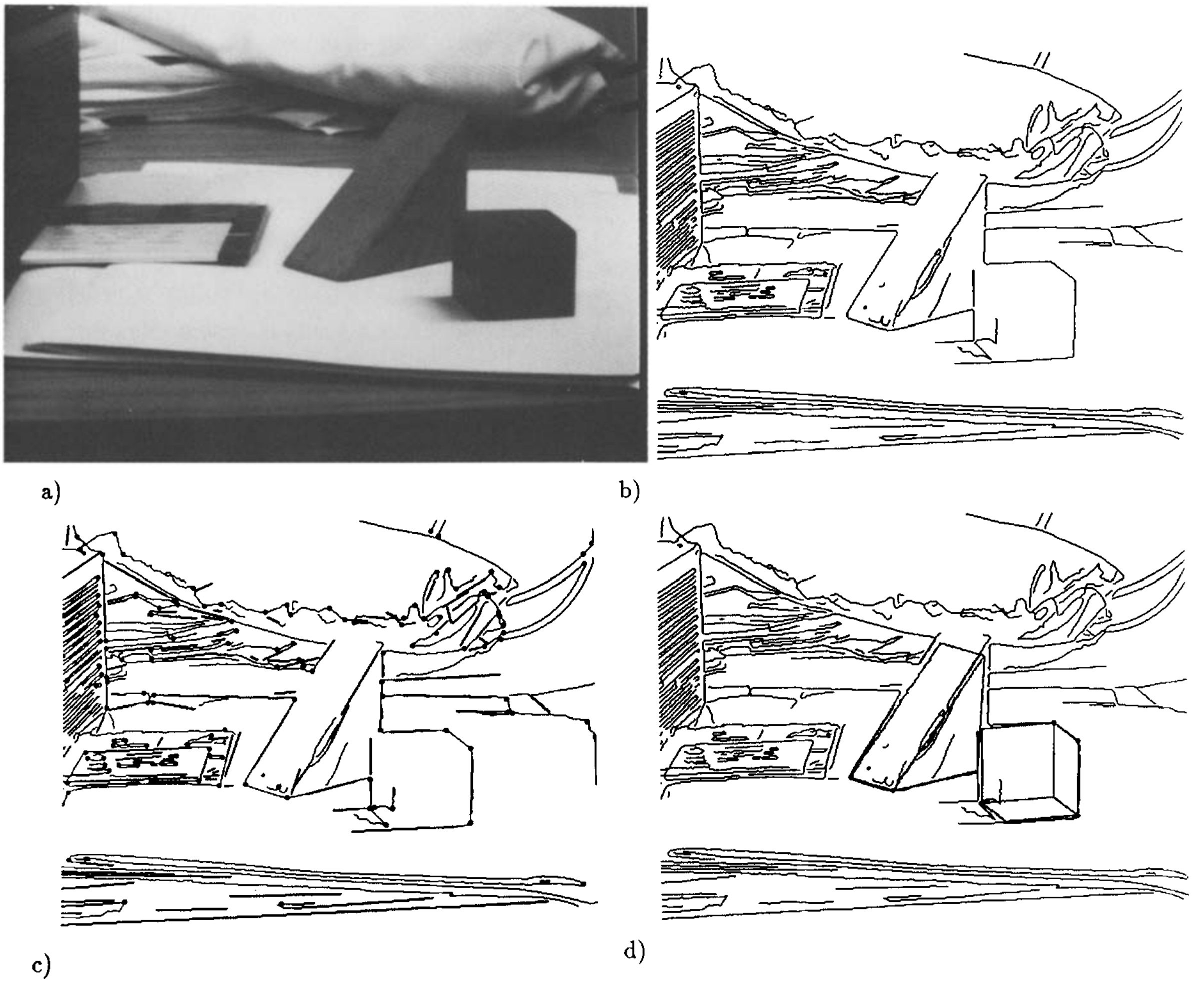


*Fig. 9.* The output of the recognizer: (a) grey-level image input, (b) Canny edges, (c) edge segment, (d) recovered instances.

dots); and part (d) shows the verified instances of the models in the image. Each model is projected into the image, and the matched image edges are shown in bold. All of the hypotheses that survived verification are shown in part (d) of the figures. The image in figure 8 contains about 100 corners, yielding approximately 150,000 matches of the model to the image. The matching time (after feature extraction) for these images is between 4 and 8 minutes on a Symbolics 3650. The initial edge detection and feature extraction operations require approximately 4 minutes.

It can be seen from these examples that the method can identify a correct match of a model to an image under a variety of viewing positions and angles. Furthermore, a correct match can be found in highly cluttered images, with no false matches.

The alignment errors in the examples are due to noise in the locations of the features. An alignment is computed from only three corresponding points, so there are not many sample points over which to average out the sensor noise. Thus it may be desirable to first compute an estimate of the transformation from three corresponding points, as is done here, and if the initial match is good enough then perform some sort of least squares fit to improve the final position estimate. One possibility is to compute a least-squares matching under perspective projection (e.g., [Lowe 1987]). However the magnitude of the perspective effects is generally smaller than the magnitude of the sensor noise. Thus another possibility is to compute a least-squares matching under a degenerate three-dimensional affine transformation (an axonometric transformation [Klein 1939]). Finding

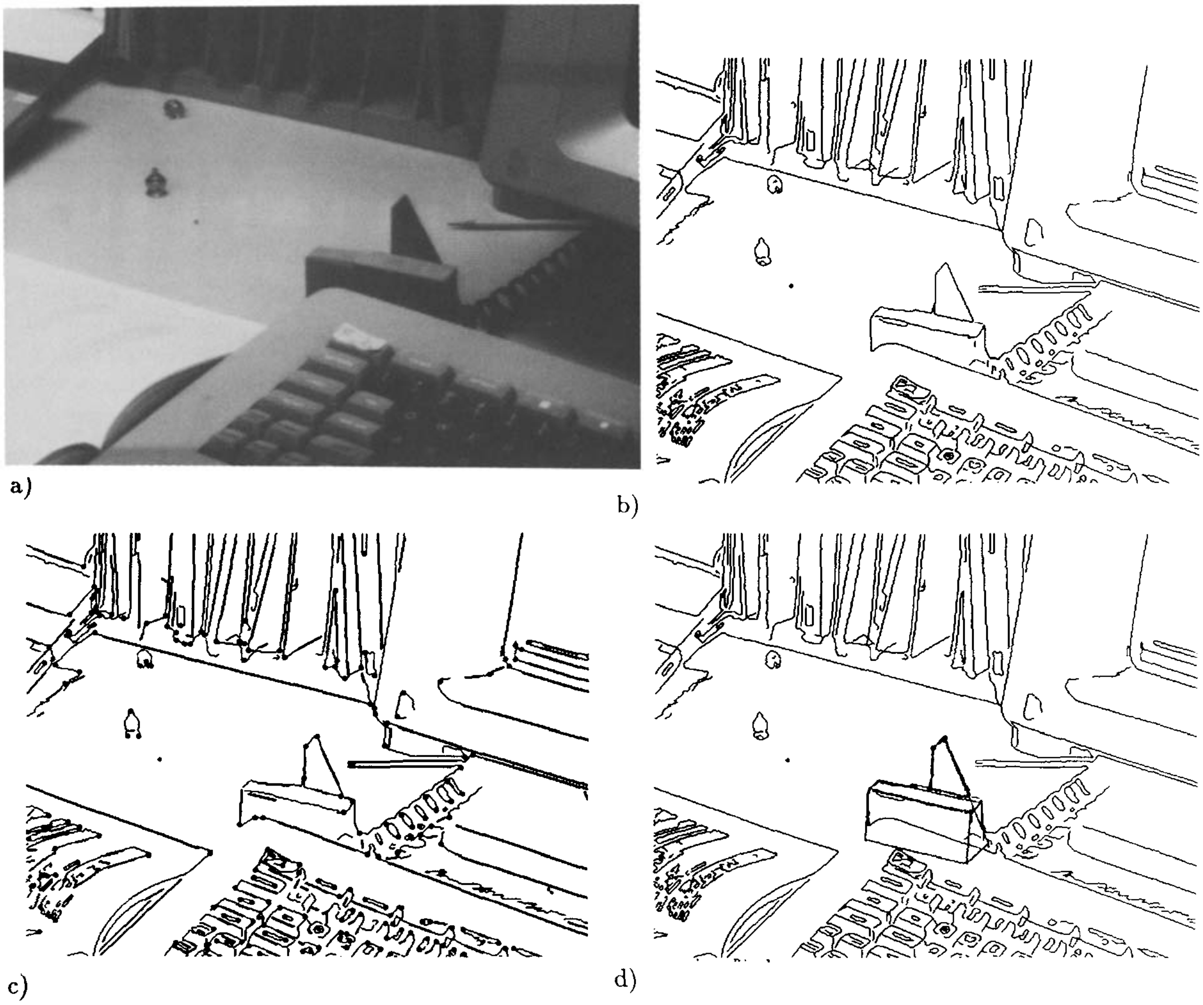


*Fig. 10.* The output of the recognizer: (a) grey-level image input, (b) Canny edges, (c) edge segments, (d) recovered instances.

corresponding features using our transformation method minimizes the number of transformations that must be considered, and simplifies the computation of each transformation. Then solving for a least-squares transformation provides a more accurate match.

## 6 Related Work

A number of researchers have addressed various aspects of the recognition problem (see [Chin and Dyer 1986] for a review). In this section we consider some of the major model-based recognition methods, divided into three classes based on the type of matching algorithm that is used. Methods in the first class compute possible transformations using a fixed number of corresponding features, and then verify those transformations. Methods in the second class search for large sets of corresponding model and image features. Those in the third class search for clusters of similar transformations.

The pioneering work of Roberts [1965] considered the problem of recognizing polyhedral objects in a line drawing or photograph. The major limitation of the system was its reliance on finding complete convex polygons. The recognizer computed a singular three-dimensional affine transform mapping the three-dimensional coordinate system of the model into two-dimensional image coordinates. Possible transformations were computed by matching each group of four connected image vertexes to each group of four connected model vertexes.

The RANSAC system [Fischler and Bolles 1981] solves for a perspective transformation, assuming that the camera parameters are known. Triples of corresponding points are used to solve for possible transformations that map a planar model onto an image. It is argued, on the basis of equation counting, that three corresponding points determine only four distinct transformations. This argument does not, however, guarantee the existence and four-way uniqueness of a solution. A closed-form method of solving for a transformation is described, but the method is complex. The implementation of RANSAC uses a heuristic method for computing the transformation, suggesting that the closed-form method is not robust. RANSAC verifies each possible transformation by mapping the set of (coplanar) model points into image coordinates. When more than some minimum number of model points are accounted for, a transformation is accepted as a correct match.

The HYPER system [Ayache and Faugeras 1986] uses geometric constraints to find matches of data to models. An initial match between a long data edge and a corresponding model edge is used to estimate the transformation from model coordinates to data coordinates. This estimate is then used to predict a range of possible positions for unmatched model features, and the image is searched over this range for potential matches. Each potential match is evaluated using position and orientation constraints, and the best match within error bounds is added to the current interpretation. The additional model-data match is used to refine the estimate of the transformation, and the process is iterated.

The second type of matching method is to search for the largest sets of corresponding model and image features that are consistent with a single pose of an object. This is a subset-matching problem, and thus there are potentially exponential number of solutions. The ACRONYM system [Brooks 1981] approximates perspective viewing by orthographic projection plus a scale factor, and predicts how a model would appear given partial restrictions on its three-dimensional position and orientation. This prediction is then used to further constrain the estimated pose. A constraint manipulation component decides whether the set of constraints specified by a given correspondence is satisfiable. The decision procedure is only partial, and takes time exponential in the number of variables in the constraint equations. As a result, ACRONYM is limited in its ability to recognize objects with six degrees of positional freedom; and the system was evaluated using aerial photographs, which are essentially two-dimensional in nature.

While the space of possible corresponding model and image features is exponential, a number of recognition systems have successfully used pairwise relations between features, such as distances and angles, to prune the search. The LFF system [Bolles and Cain 1982] forms a graph of pairwise consistent model and image features, and then searches for a maximum clique of that graph to find good correspondences between a model and an image. The largest clique above some minimum size is taken as the best match. Simple, local features are used to minimize with occlusion and noise.

A related approach is the interpretation tree method [Grimson and Lozano-Pérez 1987; Ikeuchi 1987], where a tree of possible matching model and image features is formed. Each level of the tree pairs one image feature with every model feature, plus a special branch that accounts for image features that do not correspond to the model. Pairwise consistency among the nodes along each path from the root is maintained, and any inconsistent path is pruned from further consideration. A consistent path that accounts for more than some minimum number of model features is accepted as a correct match.

The SCERPO system [Lowe 1987] solves for a perspective transformation from a model to an image, by searching for sets of model and image features that are consistent with a single view of a rigid object. Primitive edge segments are grouped together into features such as (nearly) parallel lines and corners (proximate edges). The use of approximate parallelism restricts the system to an affine viewing approximation, where perspective distortion is not significant.

SCERPO starts with an initial guess of the transformation. This guess is used to map model features into the image in order to locate additional correspondences. To provide a reasonable initial guess, SCERPO requires groups of image segments that are composed of at least three line segments. Recognition proceeds by refining a given estimate of the transformation using the additional correspondences to compute a new transformation (by an iterative least-squares technique).

The third major class of matching methods search for clusters of similar transformations. The clustering is most often done using a generalized Hough transform, where quantized values of the transformation parameters serve as indexes into a table that has one dimension corresponding to each parameter. It is assumed that clusters of similar transformations are unlikely to arise at random, so a large cluster is taken to correspond to an instance of the object in the image.

These Hough transform techniques are most applicable to recognition problems where there are a small number of transformation parameters, and where the number of model and image features is relatively small.

When there are a large number of possible transformations, the peaks in transformation space tend to be flattened out by the bucketing operation. Only a small percentage of the transformations correspond to correctly located features, so the buckets tend to become full of incorrect pairings, and the true "peaks" become indistinguishable from the incorrect pairings. For the size tables commonly used in recognition systems, the number of image features must be relatively small [Grimson and Huttenlocher 1990].

Silberberg, Harwood, and Davis [1986] use a Hough transformation approach in a restricted 3D from 2D recognition task, where there are only three degrees of freedom: two translations and one rotation. The recognizer pairs corners of a polyhedral model with each junction of edges in the image. The method for computing a transformation can yield one, two, or an infinity of possible solutions. If one or two solutions are obtained then they are entered into the clustering table. The largest clusters are taken to be the best possible transformations from the model to the image, and are used to project the model features into the image to verify and refine the correspondence.

Linainmaa, Harwood, and Davis [1985] use a similar matching algorithm, but with a more general method of computing a transformation from a model to an image. They use three corresponding model and image points to compute a set of possible transformations under perspective projection. Similarly to the method in Fischler and Bolles [1981], the computation is quite complex. Each transformation is then entered into a six-dimensional Hough table, and clusters of similar transformations are found.

Thompson and Mundy [1987] use parameter clustering for a task with six transformation parameters: 3 rotations, 2 translations and a scale factor. Their system uses a feature called a vertex pair that specifies two positions and two orientations. The two vertexes of a pair need not be (and generally are not) connected by an edge. Rather than solving directly for possible transformations using corresponding model and image vertex pairs, the 2D views of a model are stored from each possible orientation, sampled in 5-degree increments. A table maps the (quantized) planar location and angle of each vertex pair at each sampled orientation to the actual vertex pair and the 3-dimensional position. This limits the accuracy of recognition to at best 5 degrees.

The two rotation parameters about the $x$ and $y$ axes are used for an initial clustering. Each vertex pair in an image is used to index into the model table, recovering the possible model orientations for that pair. These orientations are then clustered, and those transformations in large clusters are further discriminated by clustering using translation, scale, and $z$ rotation.

Lamdan, Schwartz, and Wolfson [1988] recently presented a method for recognizing rigid planar patch objects in three-space using a two-dimensional affine transform. This computation is similar to that used in an early version of the alignment method [Huttenlocher and Ullman 1987]. In Lamdan et al.'s method, a model is processed by using each ordered non-collinear triple of the model points as a basis to express the coordinates of the remaining model points. Each of these newly expressed points is quantized and serves as an index into a hash table, where the appropriate basis triplet is recorded. This preprocessing is done offline.

At recognition time an ordered non-collinear triple of the image points is chosen as an affine basis to express the other image points. Each new image point is used to index into the hash table that was created in the offline processing, and the corresponding model bases are retrieved. A tally is kept of how many times each model basis is retrieved from the table. A basis that occurs many times is assumed to be unlikely to occur at random, and is hence taken to correspond to an instance of the model in the image. In the worst case, all triples of image points may need to be considered. A similar approach was suggested by Ullman [1987].

## 7 Summary

We have shown that when the imaging process is approximated by orthographic projection plus a scale factor, three corresponding model and image points are necessary and sufficient to align a rigid solid object with an image (up to a reflective ambiguity). We then used this result to develop a simple method for computing the alignment from any triple of non-collinear model and image points. Most other 3D from 2D recognition systems either use approximate solution methods [Thompson and Mundy 1987], are restricted to recognizing planar objects [Cyganski and Orr 1985; Lamdan et al. 1988], or solve the perspective viewing equations [Fischler and Bolles 1981; Lowe 1987], which are relatively sensitive to sensor noise because the perspective effects in most images are small relative to the errors in feature localization.

The recognition system described in this paper uses two pairs of corresponding model and image features to align a model with an image, and then verifies the alignment by transforming the model edge contours into image coordinates. The alignment features are local and are obtained by identifying corners and inflections in edge contours. Thus the features are relatively insensitive to partial occlusion and stable over changes in viewpoint. By identifying the minimum amount of information necessary to compute a transformation, the system is able to use simple local features, and only considers $O(m^2n^2)$ possible matches. We have shown some examples of recognizing partially occluded rigid objects in relatively complex indoor scenes, under normal lighting conditions.

## References


Ayache, N., and Faugeras, O.D. 1986. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. Patt. Anal. Mach. Intell.* 8 (1): 44–54.

Basri, R., and Ullman, S. 1988. The alignment of objects with smooth surfaces. *Proc. 2nd Intern. Conf. Comput. Vision*, pp. 482–488.

Bolles, R.C., and Cain, R.A. 1982. Recognizing and locating partially visible objects: The local feature focus method. *Intern. J. Robotics Res.* 1 (3): 57–82.

Brooks, R.A. 1981. Symbolic reasoning around 3-D models and 2-D images. *Artificial Intelligence J.* 17: 285–348.

Canny, J.F. 1986. A computational approach to edge detection. *IEEE Trans. Patt. Anal. Mach. Intell.* 8 (6): 34–43.

Chin, R.T., and Dyer, C.R. 1986. Model-based recognition in robot vision. *ACM Computing Surveys* 18 (1): 67–108.

Cyganski, D., and Orr, J.A. 1985. Applications of tensor theory to object recognition and orientation determination. *IEEE Trans. Patt. Anal. Mach. Intell.* PAMI-7 (6): 662–673.

Duda, R.O., and Hart, P.E. 1973. *Pattern Classification and Scene Analysis*. Wiley: New York.

Fischler, M.A., and Bolles, R.C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Comm. Assoc. Comput. Mach.* 24 (6): 381–395.

Goad, C. 1986. Fast 3D model-based vision. In *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*. A.P. Pentland, ed. Ablex: Norwood, N.J.

Grimson, W.E.L., and Lozano-Pérez, T. 1987. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Patt. Anal. Mach. Intell.* 9 (4): 469–482.

Grimson, W.E.L., and Huttenlocher, D.P. 1990. On the sensitivity of the Hough transform for object recognition. *IEEE Trans. Patt. Anal. Mach. Intell.* 12 (3): 255–274.

Horn, B.K.P., and Weldon, E.J. 1985. Filtering closed curves. *Proc. Conf. Comput. Vision Patt. Recog.*, pp. 478–484.

Horn, B.K.P. 1986. *Robot Vision*. MIT Press: Cambridge, MA.

Huttenlocher, D.P., and Ullman, S. 1987. Object recognition using alignment. *Proc. 1st Intern. Conf. Comput. Vision*, pp. 102–111.

Huttenlocher, D.P., and Ullman, S. 1988. Recognizing solid objects by alignment. *Proc. DARPA Image Understanding Workshop*. Morgan Kaufman Publishers: San Mateo, CA, pp. 1114–1124.

Ikeuchi, K. 1987. Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks. *Proc. DARPA Image Understanding Workshop*. Morgan Kaufmann Publishers: San Mateo, CA, pp. 321–338.

Kanade, T., and Kender, J.R. 1983. Mapping image properties into shape constraints: Skewed symmetry, affine transformable patterns, and the shape-from-texture paradigm. In J. Beck et al. (eds.), *Human and Machine Vision*, Academic Press: Orlando, FL.

Klein, F. 1939. *Elementary Mathematics from an Advanced Standpoint: Geometry*. MacMillan: New York.

Lamdan, Y., Schwartz, J.T., and Wolfson, H.J. 1988. Object recognition by affine invariant matching. *Proc. IEEE Conf. Comput. Vision Patt. Recog.*

Linainmaa, S., Harwood, D., and Davis, L.S. 1985. Pose determination of a three-dimensional object using triangle pairs. CAR-TR-143, Center for Automation Research, University of Maryland.

Lowe, D.G. 1987. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence J.* 31: 355–395.

Lowe, D.G. 1988. Organization of smooth image curves at multiple scales. *Proc. 2nd Intern. Conf. Comput. Vision*, pp. 558–567.

Mokhtarian, F., and Mackworth, A. 1986. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Trans. Patt. Anal. Mach. Intell.* 8 (1).

Roberts, L.G. 1965. Machine perception of three-dimensional solids. J.T. Tippet et al. eds. MIT Press, Cambridge, MA.

Shoham, D., and Ullman, S. 1988. Aligning a model to an image using minimal information. *Proc. 2nd Intern. Conf. Comput. Vision*.

Silberberg, T., Harwood, D., and Davis, L.S. 1986. Object recognition using oriented model points. *Comput. Vision, Graphics and Image Process.* 35: 47–71.

Thompson, D., and Mundy, J.L. 1987. Three-dimensional model matching from an unconstrained viewpoint. *Proc. IEEE Conf. Robotics and Automation*, p. 280.

Ullman, S. 1987. An approach to object recognition: Aligning pictorial descriptions. MIT Artificial Intelligence Lab., Memo No. 931.


## Acknowledgments


This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76-85-C-0010, in part by the Office of Naval Research University Research Initiative Program under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract N00014-85-K-0124. Shimon Ullman was also supported by NSF Grant IRI-8900267.