MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 937 January, 1987

Recognizing Rigid Objects by Aligning Them with an Image

Daniel P. Huttenlocher
Shimon Ullman

Abstract. This paper presents an approach to recognition where an object is first *aligned* with an image using a small number of pairs of model and image features, and then the aligned model is compared directly against the image. For instance, the position, orientation, and scale of an object in three-space can be determined from three pairs of corresponding model and image points. By using a small fixed number of features to determine position and orientation, the alignment method avoids structuring the recognition process as an exponential search. To demonstrate the method, we present some examples of recognizing flat rigid objects with arbitrary three-dimensional position, orientation, and scale, from a single two-dimensional image. The recognition system chooses features for alignment using a scale-space segmentation of edge contours. Segments are described in terms of both their shape and the structure of the scale-space hierarchy at the next finer level, producing distinctive features for use in finding possible alignments. Finally, the method is extended to the domain of non-flat objects as well.

Acknowledgments. This report 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 System Development Foundation and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract N0014-85-K-0124.

1 Introduction

Object recognition involves identifying a correspondence between part of an image and a particular view of a known object. This requires matching the image against stored object models to determine if any of the models could produce a portion of the image. For the general recognition problem, the number of possible models is very large. Most existing recognition systems, however, consider only one or a small number of object models (see [4] for a recent review of recognition systems).

Even for a single model, a given object can produce substantially different images depending on its position and angle with respect to the viewer. First, from a particular view, part of an object may be occluded. Second, an object may be distorted by projection into the image plane (e.g., foreshortening). Finally, an object may itself undergo transformations such as having parts that move independently, or being stretched or bent. Most recognition systems assume that objects are rigid, and do not undergo any transformation [16] [5] [11] [3]. Some systems allow for perspective projection [21], and some have parameterized models with parts that can articulate [8].

The presence of more than one object in an image also complicates the recognition problem. First, objects may occlude one another. Second, different objects in the image must somehow be individuated. In the case of touching and overlapping objects this generally cannot be done prior to recognition, but rather must be part of the recognition process itself.

Considerable attention has been paid to the problem of recognizing planar objects with two-dimensional positional uncertainty (we will refer to this as the 2D recognition problem). There are several 2D recognition systems that can find a given object in a grey-scale image, even when the object is partially occluded [16] [5].

Recognizing objects in three-space has generally been approached using a depth map, which specifies the distance from the sensor at each pixel (the 3D from 3D recognition problem). A depth map can be derived from a laser scanner, stereo matcher, or shape from motion, shading, contours, etc. Relatively successful systems have been developed for the 3D from 3D recognition task, using laser scanners to derive a depth map [17] [6].

Lowe's recent work [21] addresses the problem of recognizing objects with three-dimensional positional uncertainty given a single two-dimensional view (the 3D from 2D recognition problem). In 3D from 2D recognition, the sensory data only partially specifies the position of the object. Thus it appears to be more difficult than 3D from 3D recognition. People, however, seem to be good at this task, making it unclear whether three-dimensional sensory input is actually necessary for recognition.

The Task

In this paper we consider the problem of matching a two-dimensional view of a rigid object against a potential model. The viewed object can have arbitrary three-dimensional position, orientation, and scale, and may be touching or occluded by other objects. First we consider the domain of flat rigid objects such as the widget shown in Figure 1. While the viewed object is flat, the problem is not two-dimensional because a flat object positioned in three-space can undergo distortion such as foreshortening when projected into the image plane. This task, like the general recognition task, suffers from problems of occlusion and of individuating multiple objects in an image. There is also a limited kind of shape distortion caused by projecting a rigid object into the image. We then consider extending the method to the domain of rigid objects in general, such as the station wagon in Figure 2.

Figure 1. A widget used in recognition.

The current task cannot be handled by recognition systems that assume rigid objects with no distortion [16] [5] [11] [3], or by systems that only allow parameterized variation of rigid models [8]. The task is similar to that of Lowe, who addresses the problem of three-dimensional recognition from a single two-dimensional view [21]. The task considered by Lowe is more restricted, however, because it is assumed that objects are polyhedral, and are viewed such that parallel surfaces appear more or less parallel.

In order to solve this task, we present a new approach to recognition in which the process of matching a model to an image is divided into two stages. In the first stage, the model is *aligned* with the image using a small number of model and image features. In the second stage, the alignment is used to transform the model into image coordinates. Once the position and orientation have been determined, the model can be compared directly with the image. The key observation underlying the alignment operation is that the position and orientation of a rigid object can be

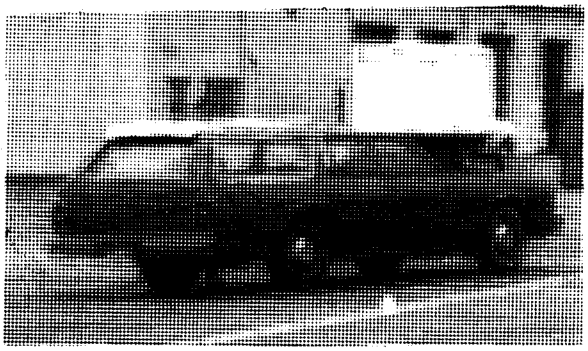

Figure 2. A station wagon used in recognition.

determined from a small number of position and orientation measures. In contrast, many recognition systems search for the largest set of model and image feature pairs that are consistent with a single position and orientation of a rigid object. The number of such sets is exponential, requiring the use of various techniques to limit the search.

Aligning a Model With an Image

For 2D recognition, only two pairs of corresponding model and image points are needed to align a model with an image. Consider two pairs, (a_m, a_i) and (b_m, b_i), such that model point a_m corresponds to image point a_i and model point b_m corresponds to image point b_i. Figure 3a shows the edge contours of two widgets, labeled with these four points. The two-dimensional alignment of the contours has three steps. First the model is translated such that a_m is coincident with a_i as shown in Figure 3b. Then it is rotated about the new a_m such that the edge $a_m b_m$ is coincident with the edge $a_i b_i$ as shown in part (c). Finally the scale factor is computed to make b_m coincident with a_m, as shown in part (d). These two translations, one rotation, and a scale factor make each unoccluded point of the model coincident with its corresponding image point, as long as the initial correspondence of (a_m, a_i) and (b_m, b_i) is correct.

For 3D from 2D recognition, the alignment method is similar, requiring three pairs of model and image points to perform a three-dimensional transformation and scaling of the model. Section 6 presents the 3D from 2D alignment method in detail, showing how to use three pairs of model and image points to position and orient a model in three-space given a single two-dimensional view, assuming orthographic projection. A transformation from the model to the image consists of two-dimensional translation, three-dimensional rotation, and a linear scale factor that is proportional to the viewing distance. Under normal viewing conditions, orthographic projection plus scale is a reasonable approximation to perspective viewing. Under high perspective distortion – when the object occupies much of the field of view – the approximation is poor. It appears, however, that under such

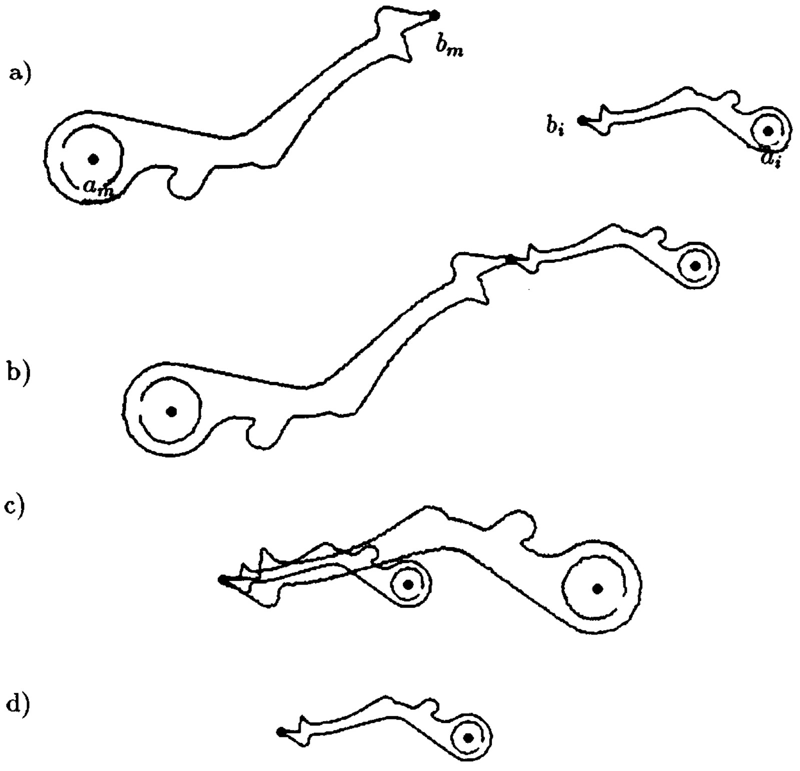

Figure 3. Two dimensional translation, rotation and scaling of one object to match another.

conditions recognition is also difficult for humans.

The alignment method requires identifying potentially corresponding model and image points such as (a_m, a_i) in Figure 3. These pairs are then used to determine possible alignments of the model with the image. Local orientation measures can also be used to solve for possible alignments. The problem of finding points and orientations for alignment is addressed in Section 3 and Section 4. In Section 5, a system for recognizing flat objects with three-dimensional positional freedom is described. Some recognition examples are also presented in that section. The details of the 3D from 2D alignment computation are given in Section 6, and the method is extended to handle non-planar objects in Section 7.

2 Matching Models and Images: Previous Approaches

Object recognition is generally viewed as a two stage process. First, possible object models are hypothesized. Second, each hypothesis is tested to determine if the model can be positioned and oriented such that it matches the image data. Most recognition systems address the problem of matching a model to an image, assuming that an appropriate model has already been hypothesized [8] [5] [11]. Some recent work has addressed the problem of hypothesizing relevant models, but only for two-dimensional recognition, and relatively small object libraries [20].

Assuming that one or a small number of possible objects have been hypothesized, there are several systems for matching a model of a rigid object to an image (cf. [4]). These systems all exploit rigidity by noting that for a given position and orientation of a rigid object, there must be a single transformation that maps each model feature onto its corresponding image feature. This transformation consists of a three-dimensional rotation and translation in 3D from 3D recognition, and a solution to the perspective viewing equation in 3D from 2D recognition [21].

In this framework, recognition is generally structured as a search for the largest pairing of model and image features for which there is a single transformation mapping each model feature to its corresponding image feature [11] [5] [16] [8] [21]. For i image features and m model features there are at most $p = i \times m$ pairs of features. Because of occluded image points, and image points that do not correspond to the model, any subset of these p pairs could be the largest set of matching model and image points, and thus the number of possible matches is exponential in the size of p.

Two methods are used to limit the number of possible matchings of model and image features. The first is to use the identity of features to specify which model features can match which image features, thereby reducing the number of pairs, p [5] [21]. Even in the case that each model feature has only a single corresponding image feature, however, there may be multiple matches because the image features may actually correspond to some other object.

A problem with using the identity of features in recognition is that there is a tradeoff between the uniqueness of a feature and the robustness with which it can be recognized. Because systems that rely heavily on the identity of features must use highly distinctive features, they tend to be sensitive to noise and occlusion in the image.

The second method for limiting the search is to use relations between features to eliminate pairs of model and image features that are inconsistent [16] [5] [11]. For instance, in order for two pairs of model and image features (a_m, a_i) and (b_m, b_i) to be part of a consistent match, the distance between the image features a_i and b_i must be the same as the distance between the model features a_m and b_m, within some error bound. Similarly, the angle between orientation measures for any pair of image features must match the angle between the corresponding pair of model features.

A problem with using relations between features in recognition is that the relations must be measurable in the image. Since relations such as distance and angle are not invariant under projection, three-dimensional recognition systems that use these relations require three-dimensional data. Relations that are invariant under projection tend to be much weaker than distance and angle relations.

Using the Identity of Features

This section considers several recognition systems that rely heavily on the identity of individual features. The features used in recognition are generally local shape properties, or parts, of an object. Global properties such as moments of inertia (cf. [4]) are also used as features for matching. Global properties of objects are highly sensitive to occlusion, however, making them inappropriate for tasks such as the ones considered here.

The LFF [5] and 3DPO [6] systems look for a variety of pre-determined features such as corners and holes, which are grouped together based on proximity. A "focus feature" in each group is chosen for use in matching. This feature is described in terms of its type (e.g., corner, hole), and the type, distance, and angle of the other features in the cluster. By clustering local features, the LFF system produces highly distinctive labels for the focus features. Using proximity to cluster image features, however, may yield clusters composed of features from multiple objects. Such image clusters will generally not match the correct model cluster. This makes the system sensitive to the position and orientation of neighboring and occluding objects.

The SCERPO 3D from 2D recognition system [21] [22] uses proximity and parallelism to group edge fragments together into simple features. As in the LFF system, the use of proximity can be problematic when there are neighboring or occluding objects. Since parallelism is not invariant under viewing position, "almost parallel" segments are also grouped together. This too, however, may fail under some viewing conditions. The strong reliance on parallelism restricts this classification scheme to polyhedral objects, where parallel surfaces appear almost parallel.

Various shape descriptions have been proposed that use local features to describe objects (SLS [7], LRS [14], curvature primal sketch [2], codons [19]). Some of these representations have been used in recognition systems (e.g., SLS [10]), and others have been discussed as possible representations for recognition (e.g., codons [19]). These shape descriptions are intended to produce highly unique features for use in matching models against images. Thus an object model is matched to an image using primarily the identity of individual image and model features, rather than structural relations between features.

The curvature primal sketch [2] is a shape description composed of a hierarchy of approximations to edge contours at various scales. At a given scale, the curvature primal sketch consists of a cubic spline approximation to a two-dimensional edge contour. The endpoints of the splines occur at local maxima in the smoothed curvature of the edge contour. There are two major problems with using the curvature primal sketch for recognition. First, splines connecting maximal curvature points are only a good approximation at relatively fine scales, when the local maxima in curvature do not occur very far apart. Second, maximum curvature points are unstable under three-dimensional rotation and projection. This issue is considered in Section 4 on finding points for use in alignment.

Symmetry-based shape descriptions [7] [14], including generalized cones or cylinders [23] [8], discard a lot of shape information. Symmetry is a shape attribute that can only be computed at relatively large scales, making it difficult to encode detailed shape information in such a representation. Because symmetry is a relatively global shape property it is also quite sensitive to occlusion. The information preserved by symmetry representations, such as lengths and orientations of axes, is not invariant under projection, making such representations inappropriate for 3D from 2D recognition tasks. Symmetry axes can, however, be used as orientation measures for alignment, as described in Section 4.

Using Relations Between Features

At the other extreme are systems that do not use feature identity at all, but rather rely exclusively on structural relations between image and model points. Grimson and Lozano-Pérez [16] [17] structure recognition as a search through a tree of all pairs of model and image points. A given level of the tree pairs a particular image point with each model point. Distance and angle relations between pairs of points are used to prune the tree. If at any point along a path, a node specifies a pair of points that are inconsistent with some previous node on that path, then the remainder of the path is not explored because it cannot lead to a consistent set of pairs. In order to handle image points from other objects, there is a special model point called the null face that will match any image point. The null face is expanded last in searching the tree, and the longest path (not counting null face branches) is always expanded first.

It has been demonstrated that this tree search converges rapidly for both 2D from 2D [16] and 3D from 3D [17] recognition tasks. The success of the algorithm is due to the power of the distance and angle constraints to prune the exponential tree of possible matches. The fact that any model and image points are allowed to match makes the system robust in the face of partial information, such as when there is substantial occlusion. The strong reliance on distance and angle relations, however, makes the method inapplicable to the problem of 3D from 2D recognition.

In addition to empirical demonstrations of the power of distance and angle relations among features, it has been shown that given a noise bound for the sensory data, pairwise distance relations can be used to develop an $O(n^2)$ time algorithm for recognizing an isolated object with n features, which can undergo two-dimensional rotation and translation [3].

The LFF and 3DPO systems [5] [6] use distance and angle relations between features as well as the identity of individual features. Model and image features of the same type are paired together. The space of possible subsets of these pairs is then searched using a graph matching algorithm. Pairwise distance and angle relations are used to form a graph structure. Each pair of model and image features is represented by a node, and each consistent pair of nodes is connected by an arc. A maximum clique of this graph constitutes a largest pairwise consistent match of model and image features.

This method has been shown to find a maximally consistent match for a variety of images in a two-dimensional recognition task. Much of the success of the algorithm is attributable to the use of local feature clusters to restrict the number of possible initial pairs of model and image features. As discussed above, this reliance on local context makes the system sensitive to overlapping objects.

ACRONYM [8] uses both the identity of features and the relations between features in recognition. The features are generalized cones, which are described in terms of axes and cross sections. The presence of a given feature is used to predict the positions and orientations of other features. Therefore, like the other systems discusssed in this section, given two-dimensional data ACRONYM can perform only 2D recognition. While ACRONYM's geometric modeling component is heavily three-dimensional, the system was tested using aerial photographs, which are two-dimensional in nature.

The SCERPO system [21] [22] is the only one that addresses the problem of three-dimensional recognition from a single two-dimensional view. The problem of finding a best match is structured as a tree search (similar to [16]), however the check for a consistent match is different from previous systems. A given set of model and image feature pairs are consistent if there is a solution to the perspective viewing equation that maps each model feature onto its corresponding image feature. There are no simple pairwise checks such as distance and angle relations that can be used to test the consistency of a set of pairs incrementally, given an added pair. Instead, the perspective viewing equation is solved for each added pair of model and image points. This is done using Newton-Raphson iteration, which allows a solution to be modified to account for a new pair.

None of the recognition systems discussed in this section can effectively handle the task of recognizing objects that have arbitrary three-dimensional position, orientation, and scale, from a single two-dimensional view. Each system solves a more restricted recognition task, in order to limit the space of possible matchings of model and image features (e.g., requiring distance and angle relations to be measurable, or parallel surfaces to appear parallel). The next section describes how to recognize an object by first aligning it with an image. This use of alignment avoids having to structure the recognition process as an exponential search.

3 The Alignment Method of Recognition

We have seen above that recognition can be viewed as a search through the space of all possible positions and orientations of all possible objects. The idea of the *alignment* approach is to separate this search into two stages. In the first stage, the position, orientation, and scale of an object are found using a minimal amount of information, such as three pairs of model and image points. In the second stage, the alignment is used to map the object model into image coordinates for comparison with the image.

There are two major advantages of this approach. First, by using a small fixed number of model and image features, the method avoids structuring recognition as a search through an exponential space. Second, a given alignment can be used to match multiple models against the image. If a group of objects are stored such that they are aligned with one another, then a single alignment computation will map the entire group of objects into the image.

The key observation behind the approach is that the alignment can be performed with a small amount of information. For example, three image points and three corresponding model points are sufficient to determine the position, orientation and scale of a rigid object in three-space. Similarly, two points and an orientation measure can also be used to solve for the three-dimensional alignment.

Consider an object, O, with three-dimensional positional freedom, and a two-dimensional image, I, which contains a view of O (perhaps along with other objects). We are interested in using the alignment computation to find O in the image. Assume that a feature detector returns a set of potentially matching model and image feature pairs, P (one such detector is described in Section 5). Since three pairs of model and image features specify a potential alignment of a model with an image, any triplet in P may specify the position and orientation of the object. In general, some small number of triplets will specify the correct position and orientation, and the rest will be due to incorrect matchings of model and image points. Thus the recognition problem is to determine which alignment in P defines the transformation that best maps the model into the image.

Given a set of pairs of model and image features, P, we solve for the alignment specified by each triplet in P. For some triplets, there will be no way to position and orient the three model points such that they project onto their corresponding image points. Such triplets do not specify a possible alignment of the model and the image. The remaining triplets each specify a transformation mapping model points to image points. An alignment is scored by using the transformation to map the model edges into the image, and comparing the transformed model edges with the image edges. The best alignment is the one that maps the most model edges onto image edges.

For m model features and i image features, the number of pairs of model and image features, p, is at most $i \times m$. With a good labeling scheme, the number of pairs, p, will be much smaller, approaching m when each model point has one corresponding image point. Given p pairs of features, there are $\binom{p}{3}$, or an upper bound of $O(p^3)$, triplets of pairs. Each triplet specifies a possible alignment of the model and the image. An alignment is scored by mapping the model edges into the image. If the model edges are of length l, then the worst case running time of the algorithm is $O(lp^3)$. Thus by structuring the recognition process as an alignment stage followed by a comparison stage, it is transformed from the exponential problem of finding the largest consistent set of model and image points, to the polynomial problem of finding the best triplet of model and image points.

Since the number of possible alignments is cubic in the number of model and image feature pairs, it is important to label features distinctively in order to limit the number of pairs. If the number of pairs, p, is small, then little or no search is necessary to find the correct alignment. For instance, if $p = 3$ there is only one possible alignment, and if $p = 5$ there are $\binom{5}{3}$, or 10, possible alignments. The problem of labeling features is discussed in Section 4 and Section 5.

Limiting the Number of Possible Alignments

We have formulated the recognition problem as finding the alignment that best matches the model edges to image edges. Since any triplet of model and image points could specify that alignment, each one must be considered. Having solved for a possible alignment, additional pairs of model and image points or orientations can be used to check the validity of the alignment, before projecting the model into the image. The fact that a given alignment does not map a model point onto a corresponding image point does not, however, mean that the alignment is incorrect. It could also be the case that the additional pair of model and image points is incorrect.

For certain kinds of model and image pairs, it is fairly certain that the correspondence of the model and image features is correct. Thus, a failure to align these additional measures indicates an incorrect alignment. For instance, if the feature detector returns not only a location and a label, but also an orientation of each feature, then it is fairly certain that a given feature point and orientation belong to the same viewed object.

When an orientation measure is associated with each model and image point, the alignment computation is modified so that each of the three model orientations is mapped into the image, and compared with the corresponding image orientation. If all three match, then the triplet specifies a possible alignment, otherwise it is discarded.

The use of local orientation measures to filter possible alignments is different from the use of orientation constraints in existing recognition systems (e.g., [16] [11] [5]). Here, each model orientation undergoes a three-dimensional alignment transformation to map it into the image, where it is compared with a corresponding image orientation. In contrast, existing systems compare the angle between a pair of model features with the angle between a pair of image features. This requires that the image measures specify three-dimensional orientation in order to match them with a three-dimensional model.

Alignment and Identification are Separate Operations

Finding a match of a model to an image requires both *aligning* the object with the image, determining its position and orientation, and *identifying* the object, determining that it is actually present. However, the operations of alignment and identification have substantially different data requirements.

Aligning a model with an image requires only a small number of data points. For instance three pairs of model and image points are sufficient to determine the three-dimensional position and orientation of an object, plus a linear scale factor. A small number of data points are not only sufficient for alignment, it is advantageous to have fewer points, because the number of possible alignments of a model and an image is cubic in the number of model and image point pairs.

In contrast to alignment, identification requires a fairly large number of data points. In the worst case, identification can require almost arbitrary amounts of data, when differentiating between two similar objects.

The difference between alignment and identification is most pronounced in 3D from 2D recognition. In these tasks, the position and orientation should ideally be known before attempting to identify an object, as projection distorts an object's shape. Because the alignment method solves for position and orientation using a minimal amount of information, it is particularly well suited to this problem. The differing requirements of alignment and identification have led us to propose the alignment method, where alignment precedes comparison.

When alignment and identification are combined, moderately detailed and dense features must be used in order to be able to distinguish objects from one another. The large number of features makes the number of possible positions and orientations very large, because different n-tuples of points will produce slightly different positions even though they correspond to the "same" interpretation of the object. This is illustrated by Grimson and Lozano-Pérez's system [16], which finds a large number of highly similar interpretations, differing only slightly in position or orientation. Since these interpretations are essentially the same, they are grouped together into a single interpretation. In contrast, by using a small number of coarse features to first align an object with an image, there will be relatively few possible positions and orientations (some of which may still be very similar).

Another consequence of combining alignment and identification is the difficulty of determining that an object is *not* in the image. All possible interpretations must be considered and found inconsistent before an object can be rejected. Since the features are relatively dense, their number is large, and many inconsistent interpretations must be considered and rejected. If alignment and identification are separated, then the alignment operation alone will indicate that there is no possible match in cases of a substantial mismatch. In other cases, some comparison will have to be performed as well.

On Alignment in Human Recognition

There is some evidence that people align two objects before comparing them. For instance, people appear to be slower at judging if two edge contours are the same when they are presented at different orientations compared to when they are presented at the same orientation [26]. The major exception to this is edge contours that define a region with an "intrinsic axis". Such contours can be compared rapidly even when presented at different orientations [29].

The effect is illustrated in Figure 4. Part (a) of the figure shows a contour without a good axis, and part (b) shows a contour with a good axis. Comparison is rapid for both contours when the orientation is the same. When the orientation is different, however, recognition is substantially slower for contours without an axis, as in (a), than for ones with an axis, as in (b) [29].

As discussed in Section 1, in two dimensions two points are sufficient for aligning a model with an image. Similarly, it is possible to use a point and an orientation

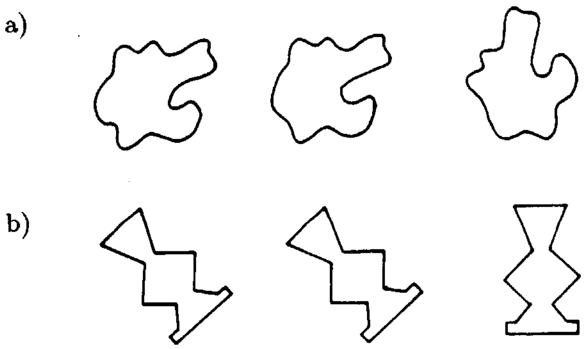

Figure 4. Orientation affects human matching speed, a) an object without a good axis which is hard to match, b) an object with a good axis which is easy to match.

if no change in scale is allowed. The point is used to translate the model, and the orientation is used to rotate it. The contour in (b) has a distinguished point (e.g., the base) and orientation for alignment, whereas the one in (a) does not.

The results of these studies suggest that without sufficient information to perform alignment, recognition is more difficult, perhaps requiring explicit rotation of an object to compare many possible orientations.

4 Alignment Points

The alignment operation requires finding pairs of corresponding model and image points, and model and image orientations. Because the number of potential alignments is cubic in the number of pairs, it is desirable to limit the number of pairs as much as possible. There must be at least three pairs that correspond to the image of the object, however, or else the method will fail to find an alignment.

To limit the number of pairs of model and image points (or orientations) it is necessary to associate labels with these measures, and only pair together model and image data with the same label. The labels must be relatively insensitive to partial occlusion, juxtaposition, and projective distortion, while being as distinctive as possible.

There are several sources of information in an image that can be used to define and label points and orientations for alignment. We have chosen to use a shape description based on intensity edges to define labeled edge segments that are used for alignment. The method is described in Section 5, below. These descriptions can be augmented using color, texture, shading and depth information to form more distinctive descriptions of the alignment points. We intend to exploit such sources of information in the next version of the system.

It is also possible to define other kinds of alignment points based on intensity edges. For instance, vertices where multiple edges come together can be used as alignment points. Other kinds of alignment points are cusps, tips, deep concavities and small blobs. In the current implementation, however, we have concentrated on points and local orientations derived from inflections in edge contours.

As long as different kinds of alignment points are identified by distinct labels, the more kinds of information the better in terms of being able to quickly identify the correct alignment of a model with an image. A large amount of data is not a problem, only a large amount of indistinguishable data.

Edge Contour Shape Features

Forming a shape description based on intensity edges requires breaking edge contours into primitive pieces. Many techniques for deriving shape descriptions segment contours at maximal curvature points, partly because of their supposed psychological importance. The studies [1] [12] demonstrating the importance of maximal curvature points, however, do not address the problem of what information is *necessary* to recognize an object. Rather, the studies demonstrate that certain information is *sufficient* to recognize an object.

For instance, Attneave's cat, shown in Figure 5a, is constructed by linearly interpolating between the maximal curvature points in a drawing of a cat. The fact that the drawing is still easily recognizable has been used to claim that maximal curvature points are of special significance. Lowe [21], however, points out that the contour in Figure 5b is also easily recognizable as a cat. This contour is constructed using the points midway between maximal curvature points.

Thus, it appears that there is sufficient information in a contour that any one of a variety of sparse descriptions is sufficient for people to be able to recognize an object. Maximal curvature points, per se, are not important.

Because it is possible to represent contour shape using various sparse representations, the choice of points should be motivated by the requirements of the recognition task being addressed. Maximal curvature points are highly unstable under three-dimensional rotation and projection, both appearing and disappearing (without being occluded by other parts of the object), making them inappropriate for 3D from 2D recognition. For example an ellipse can be rotated about its minor axis to obtain a circle – illustrating maximal curvature points that disappear. This circle can then be rotated around another axis to obtain a different ellipse – illustrating maximal curvature points that appear.

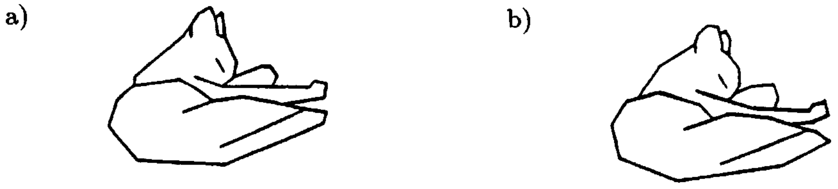

Figure 5. Attneave's cat, which is intended to demonstrate the importance of maximal curvature points for representing shape, and Lowe's cat which shows other points work as well.

In contrast to curvature maxima, zero crossings of curvature (or inflection points in the contour) are relatively stable under projection, only disappearing when the contour is projects to a straight line. False inflection points only appear when one piece of an object partly occludes another.

Low curvature regions pose a problem because very small changes in curvature may yield "inflections". We define significant inflections to occur only in regions where the curvature is not in the range $[-\epsilon, \epsilon]$. The recognition system described in the next section uses significant inflection points and low-curvature regions to segment edge contours.

5 A 3D from 2D Recognition System

This section describes a system that implements the alignment method of recognition. The system is consistent with the requirements outlined by the theory, but is more specific in many respects and thus reflects certain choices of implementation that are only one of several possible ways of doing things.

In the current implementation, the features used by the system are obtained from inflections in the intensity edges in a two-dimensional image. These features, however, can be augmented using other sources of information, as mentioned above.

Shape Features: Segmenting Edge Contours

The recognition system forms edge-based shape features for use in aligning models with images. The input to the system is a grey-level image, which is processed by an edge detector [9]. The output of the edge detector is noisy, containing edges due to shadows and texture as well as object edges. This causes two problems. First, the presence of many weak edges will make it difficult to group edges together into larger shape units. Second, missing pieces of edge contour will make it unclear whether two pieces of edge contour are part of the same underlying edge in the real world. The second problem is somewhat easier to deal with, so the strategy utilized here is to discard relatively weak edges.

Given an array of edge points, the points must be chained together into contours. A simple method is to chain together neighboring points whenever there is an unambiguous eight-way neighbor. Otherwise, a new chain is started. Chains with low overall edge strength are then discarded. Thresholding whole chains rather than individual edge points produces a more stable output. Finally, edge chains with unambiguous nearest neighbors are merged together if they can be connected by a smooth spline, without intersecting another edge contour.

Once pieces of edge contour have been chained together, simple shape descriptors are derived using the local curvature of the edge contours.† The edge contours are segmented by breaking the contour at zero crossings of curvature (inflection points in the contour), and at the ends of low-curvature regions; producing straight, positive curvature and negative curvature segments. As discussed above, inflections were chosen as segmentation points because they are relatively stable under three-dimensional rotation and projection.

Multi-Scale Descriptions

The purpose of labeling the edge segments is to produce distinctive labels for use in pairing together potentially matching model and image points. Most recognition systems form distinctive labels by using local context to describe a given feature. The problem with this, however, is that an image feature may be labeled using context which is not part of the object being recognized (e.g., as in LFF [5] and SCERPO [21]).

We use a more limited form of context, in which the edge contour is smoothed at various scales, and the finer scale descriptions are used to label the coarser scale segments. In other words, the coarser scale segments are used to group finer scale segments together. This produces distinctive labels without the problem of accidentally using context from a different object, because the "context" is part of the same edge contour.

†Curvature is computed as the change in angle (per unit arclength) between local tangent vectors at neighboring pixels. The tangents are computed using the least squares best fit line over a small local neighborhood.

A hierarchy of curve segmentations can be obtained by smoothing the curvature at different scales, and using the smoothed curvature to segment the edge contour. Smoothing the curvature preserves only those inflection points in the edge contour (zero crossings of curvature) that are significant at a given scale. Thus coarser scale segments correspond to merging neighboring segments at finer scales, producing a scale-space [30] hierarchy of segments. Since coarser scales of smoothing do not introduce zero crossings that were not present at finer scales [31], the hierarchy forms a tree of segments from coarser to finer scales.

Figure 6 shows a three-level scale-space curvature segmentation of the edge contours of the widget from Figure 1. Each part of the figure shows the same contour, segmented according to the curvature smoothed at different scales (using Gaussian filters of size σ = 7, 20, and 40 pixels, respectively). The coarsest scale is at the top of the figure and the finest scale is at the bottom. The endpoints of each segment are delimited by a dot, and straight regions (at that scale) are shown in bold. Each segment is labeled with a letter, and a number denoting the level (1 is coarsest and 3 is finest).

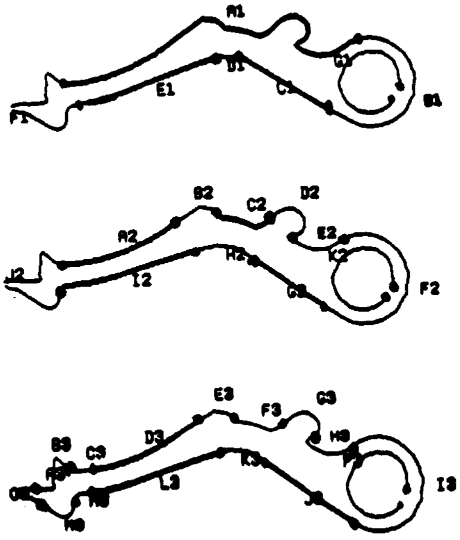

Figure 6. A scale-space segmentation of a widget, where the contours are segmented at inflections in the smoothed curvature. The coarsest scale is at the top..

Each segment of edge contour is classified according to whether it is `curved` or `straight`. The curved segments are further classified by the degree of closure: open or `closed`, and the smoothness of the contour: `smooth` or `unsmooth`, yielding a total of five types of segments. Richer descriptions are then obtained by combining the classifications at multiple scales of smoothing.

This (multi-rooted) segmentation tree can also be viewed in terms of the correspondence between regions at neighboring scales, as show in Figure 7. Each region at a coarse scale corresponds to one or more regions at the next finer scale. Each segment in the tree is indicated by its label from Figure 6 and by the type of segment: **straight**, **curve**, and **open-curve**.

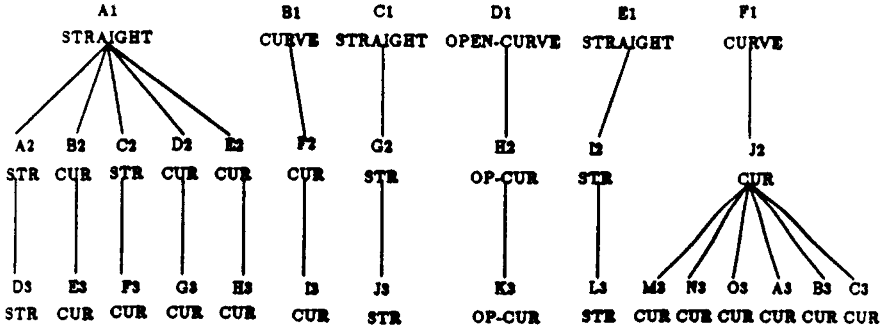

Figure 7. The tree corresponding to the curvature scale-space segmentation in Figure 6.

Multi-scale descriptions are formed using the types of segments at a given scale, plus the structure of the hierarchy at the next finer scale. Two aspects of the tree structure are used. First, a segment is classified according to whether it corresponds (primarily) to one or many segments at the next finer scale: **single** or **multiple**. Second, a **multiple** segment is classified according to whether or not the finer scale segments form a regular pattern. A pattern consists of a repetition of the same type of segment, in either the same or opposite directions of curvature. This yields the classification **irregular**, **regular-same**, and **regular-opposite**.

For example, using this multi-scale description, the **straight** segment A1 at level 1 in the tree is differentiated from the other **straight** segments C1 and E1 at the same level, because A1 is composed of **multiple**, **irregular** segments at the next level whereas C1 and E1 are each composed of a **single** segment.

Using the multi-scale description, at the coarsest scale the widget is composed of seven segments, only two of which can be confused with one another (the two straight segments C1 and E1). The seven segments are, A1: **straight**, **multi-irregular**; B1: **curve**, **single**, **smooth**; C1: **straight**, **single**; D1: **open-curve**, **single**; E1: **straight**, **single**; F1: **curve**, **multi-irregular**; G1: **closed-curve**, `single`, `smooth`. Thus, the labels resulting from this multi-level description are highly distinctive even though the individual features are coarse, and relatively sparsely distributed in the image.

Since the coarse scale segments are relatively distinct, alignment points are chosen using a coarse scale segmentation. Each segment defines a point used for alignment, based on its type. For closed segments of contour, the center of the region defined by the contour is used. This point is found from the intersection of the major and minor axes of the region. For straight segments the endpoints are used, and for other curved segments the middle of the curve is used. Since the endpoints of the segments are at inflection points or the ends of zero curvature regions, they are relatively stable, making it reasonable to use endpoints and midpoints for matching.

Each alignment point is labeled with the type of its coarse scale segment. In addition, local orientation measures are defined for `straight` and `open-curve` segments, for use in eliminating the alignments specified by inconsistent triplets (as described in Section 3).

Using coarse scale regions for choosing alignment points reduces the number of points, while retaining relatively distinctive labels. It is often possible, however, to achieve a more accurate alignment by using intermediate or finer scale descriptions. Therefore, the recognizer first finds the best alignment using coarse-scale features, and then attempts to improve upon it by performing a second alignment with finer scale features. Since the model and the image are already almost aligned, this secondary alignment is relatively fast. In general, each partially aligned model feature has only one corresponding image feature, and the correspondence of model and image features is correct.

Recognition Examples

The recognizer is implemented on a Symbolics 3650, and takes from 2-5 minutes for each of the examples shown in this section, using a pre-computed model. First a multi-scale description of the edge segments is formed and used to define alignment points in the image, as described in the previous section. Possible alignments are then computed using triplets of model and image point pairs, as discussed in Section 3. For each alignment, the model is mapped into the image and the transformed model edges are correlated with the image edges. The alignments are ranked based on the percentage of the model edge contour for which there is a corresponding image edge contour. The recognizer returns the best alignment accounting for each part of the image which is matched by the model.

We present several examples of the recognizer processing grey-scale images of widgets. The model is the multi-scale description of the widget shown in Figure 6 and Figure 7. The model is just the result of processing the image of the isolated widget in Figure 1 in the same manner as any image. Thus for flat objects, models are formed directly from an image of an object.

Each example is illustrated by a figure with four parts: a) the grey-level image of the model widget, with the intensity edges superimposed, b) the grey-level image, c) the image with the intensity edges superimposed, and d) the edges of the aligned model superimposed on the image. Part (d) also indicates the points which were used in computing the alignment of the model with the image.

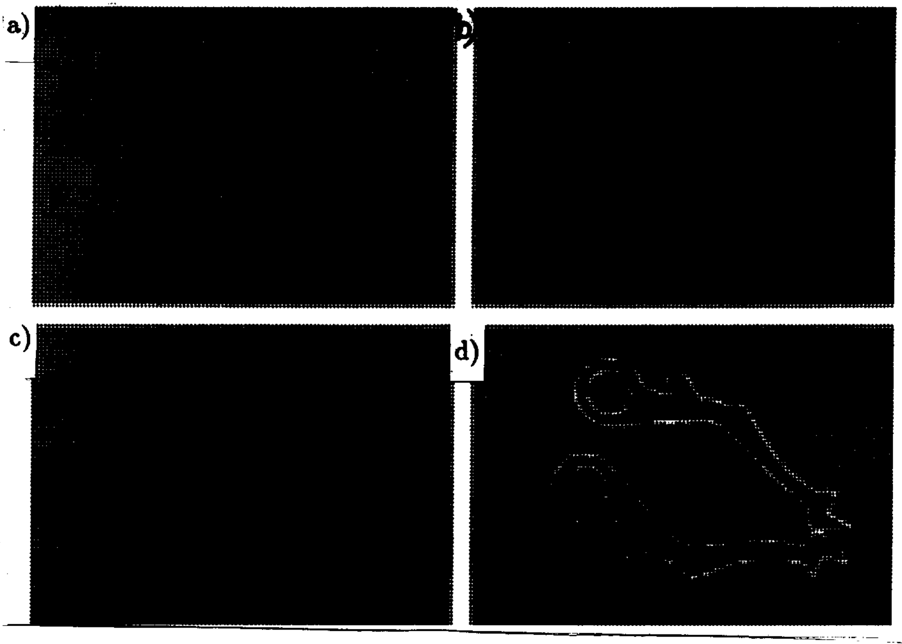

Figure 8. Matching a widget against an image of two widgets in the plane, a) the grey-level image and intensity edges of the model, b) the grey-level image, c) the image and the intensity edges, d) the edges of the aligned model superimposed on (c), with the alignment points marked.

The example in Figure 8 shows two widgets in the plane. The top widget has been flipped over, and thus cannot be recognized using only two-dimensional transformations. The recognizer finds two distinct positions and orientations of the model that match 99% and 98% of the model edge contour to image edges. These two matches are shown superimposed on the image in part (d) of the figure.

Another position and orientation is found at the alignment stage, but is eliminated because the correlation with the image is poor, and the image edges are accounted for by a better alignment. This alignment is shown in Figure 9 superimposed with the image edges. The alignment is found because the two straight edges

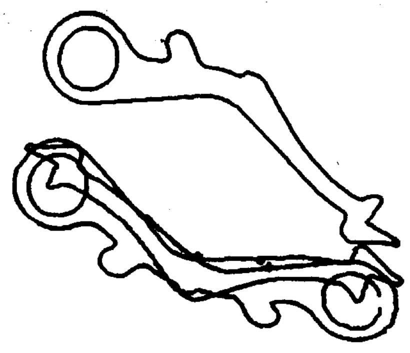

Figure 9. An alignment of a widget with an image that does not match the model edge contour with image edges.

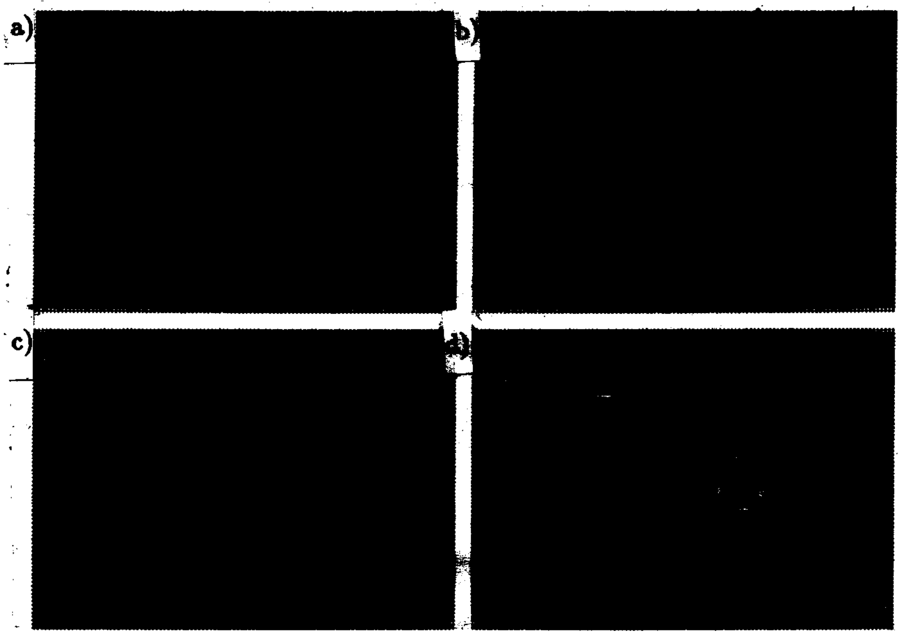

Figure 10. Matching a widget against an image of a foreshortened widget, a) the grey-level image and intensity edges of the model, b) the grey-level image, c) the image and the intensity edges, d) the edges of the aligned model superimposed on (c), with the alignment points marked.

are indistinguishable, and the three points used in computing the alignment were the two straight edges and the bend.

Figure 10 shows a widget that has been tilted approximately 30 degrees by resting one end of it on a block, foreshortening the image. The recognizer finds a single best position and orientation, which is shown in part (d) of the figure.

The next example, in Figure 11, shows another widget that has been tilted out

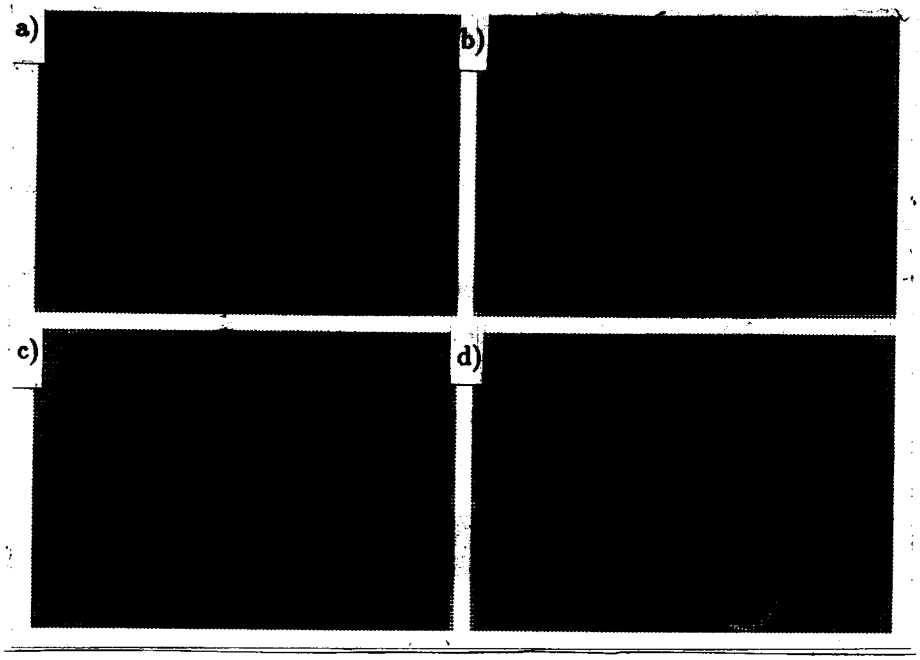

Figure 11. Matching a widget against an image of a tilted widget, a) the grey-level image and intensity edges of the model, b) the grey-level image, c) the image and the intensity edges, d) the edges of the aligned model superimposed on (c), with the alignment points marked.

of the image plane — for instance the circular end is now an oval. The best match is shown in part (d) of the figure.

Finally, we demonstrate the ability of the recognizer to find partly occluded objects. As long as three features are visible in the image, the alignment algorithm will be able to align the model with the image. Figure 12 shows a widget that has a pile of smaller widgets obscuring the circular end. The best alignment matches 80% of the model edge contour to image edges, and is shown in part (d) of the figure. Figure 13 shows two widgets obscured by each other and several smaller objects. The matcher finds two distinct positions and orientations, which are shown in part (d) of the figure.

From these examples we see that the alignment algorithm finds a small number of reasonable matches of widgets to images, even when the widget is foreshortened, scaled, and partly occluded. The scoring method of transforming the model edges and correlating them with the image edges provides a simple method for finding the best alignment. While this scoring method suffices for the examples considered here, it may be too simple in the general case. For instance, it may be desirable to have different parts of the model carry different weights in scoring the goodness of a match.

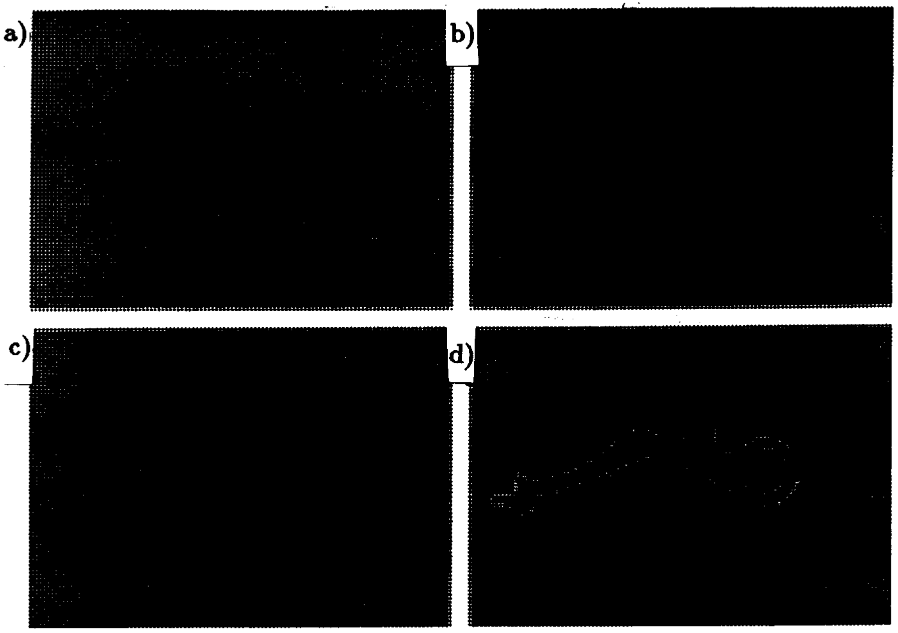

Figure 12. Matching a widget against an image of a partly occluded widget, a) the grey-level image and intensity edges of the model, b) the grey-level image, c) the image and the intensity edges, d) the edges of the aligned model superimposed on (c), with the alignment points marked.

6 The 3D from 2D Alignment Method

This section presents the alignment method in detail. It is shown that the position, orientation, and scale of an object in three-space can be determined from a two-dimensional image using three pairs of corresponding model and image points.

The description of the alignment method is divided into three parts. First we discuss the use of orthographic projection and a linear scale factor to approximate perspective viewing. Then we present the alignment method using explicit three-dimensional rotation. Finally we present a version of the alignment method that simulates three-dimensional rotation using planar operations.

Perspective Projection

Consideration of how to align an object with a two-dimensional image raises the issue of what model of projection to use. The imaging characteristics of cameras and human eyes are well approximated by the perspective projection model. Under perspective projection, imaging size is inversely proportional to the distance from an object to the center of projection, as illustrated in part (a) of Figure 14. Imaging

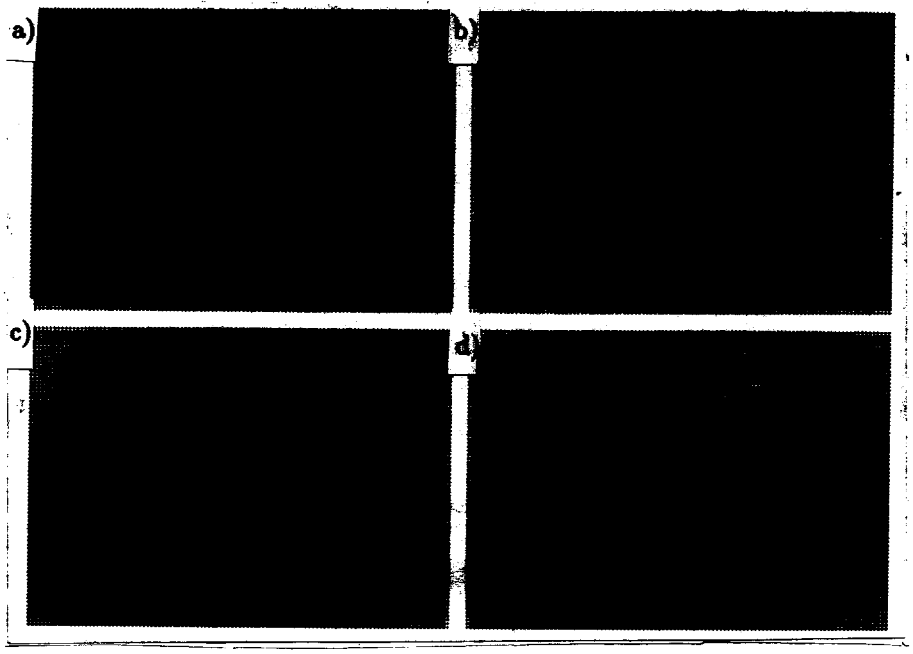

Figure 13. Matching a widget against an image of two partly occluded widgets, a) the grey-level image and intensity edges of the model, b) the grey-level image, c) the image and the intensity edges, d) the edges of the aligned model superimposed on (c), with the alignment points marked.

size is also dependent on f, the focal length of the device, which can be assumed to be constant for a given sensor.

Under orthographic projection, on the other hand, imaging size is independent of distance. Therefore all positions along the viewing axis are indistinguishable from one another, as illustrated in part (b) of Figure 14. Orthographic projection is simpler to solve for than perspective projection, but is not necessarily an adequate model of actual viewing conditions.

There are two major practical consequences of perspective viewing. The first is that objects that are further away look smaller. The second is that objects that are large relative to the viewing distance appear distorted, because the distant parts of an object project smaller images than the closer parts do. People appear to have difficulty recognizing objects under conditions of high projective distortion.

For objects that are not large relative to the viewing distance, the major effect of perspective projection is scaling proportional to the distance of the object. Therefore, in these cases perspective projection can be well approximated by orthographic projection plus a linear scale factor.

Orthographic projection plus scale can be solved for unambiguously using three pairs of model and image points, as described in the next section. Under ortho-

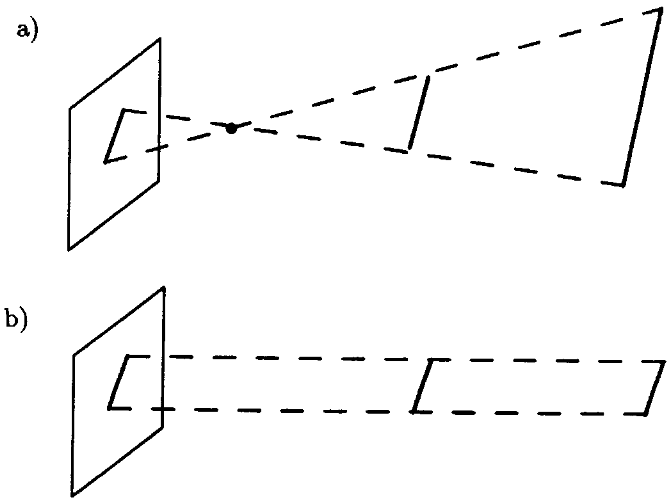

Figure 14. a) perspective, and b) orthographic projection.

graphic projection, position along the viewing axis and reflection about the view plane are indistinguishable. Thus the solution is unique up to position in z and reflection about the $z = 0$ plane.

An unambiguous solution for the position and orientation of an object under perspective projection can require up to six pairs of model and image points [13]. Using three pairs of points there can be up to four distinct perspective solutions. It is interesting to note that the ambiguous cases involve positions and orientations where part of the object is extremely close to the viewer and part is far away – cases where there is high perspective distortion. Thus three points are sufficient to solve for position and orientation under perspective viewing if solutions with high perspective distortion are discarded.

The Explicit Three-Dimensional Method

Consider three model points a_m, b_m and c_m and three corresponding image points a_i, b_i and c_i, where the model points specify three-dimensional positions, (x, y, z), and the image points specify positions in the image plane, $(x, y, 0)$. The alignment task is to find a transformation that maps the plane defined by the three model points onto the image plane, such that each model point coincides with its corresponding image point. If no such transformation exists, then the alignment process must determine this fact.

Since the viewing direction is along the z-axis, an alignment is a transformation that positions the model such that a_m projects along the z-axis onto a_i, and similarly for b_m onto b_i, and c_m onto c_i. The transformation consists of translations in x and y, and rotations about three orthogonal axes. There is no translation in z because all points along the viewing axis are equivalent under orthographic projection. Instead, distance from the viewer is reflected by a change in scale. First we show how to solve for the alignment assuming no change in scale, and then modify the computation to allow for a linear scale factor.

The first step in finding an alignment is to translate the model points so that one point projects along the z-axis onto its corresponding image point. Using the point a_m for this purpose, the model points are translated by $(x_{a_i} - x_{a_m}, y_{a_i} - y_{a_m}, 0)$, yielding the model points a'_m, b'_m and c'_m. This brings a'_{mi}, the projection of a'_m into the image plane, into correspondence with a_i, as illustrated in Figure 15a.

Now it is necessary to rotate the model about three orthogonal axes to align b_m and c_m with their corresponding image points. First we align one of the model edges with its corresponding image edge by rotating the model about the z-axis. Using the $a'_m b'_m$ edge we rotate the model by the angle between the image edge $a_i b_i$, and the projected model edge $a'_{mi} b'_{mi}$, yielding the new model points b''_m and c''_m, as illustrated in Figure 15b.

To simplify the presentation, the coordinate axes are now shifted so that a_i is the origin, and the x-axis runs along the $a_i b_i$ edge.

Because b''_{mi}, the projection of b''_m into the image plane, lies along the x-axis, it can be brought into correspondence with b_i by simply rotating the model about the y-axis. The amount of rotation is determined by the relative lengths of $a_m b_m$ and $a_i b_i$, because the model must be rotated such that the projected model edge is the same length as the image edge. If the model edge is shorter than the image edge, then there is no such rotation, and hence the model cannot be aligned with the image.

Thus, the model points b''_m and c''_m are rotated about the y-axis by ϕ to obtain b'''_m and c'''_m, where

$$\cos\phi = \frac{\|b_i \cdot (1,0,0)\|}{\|b_m \cdot (1,0,0)\|}$$

for $0 \leq \cos\phi \leq 1$. The result of this rotation is illustrated in Figure 15c.

Finally, c'''_m is brought into correspondence with c_i by rotation about the x-axis. The degree of rotation is again determined by the relative lengths of model and image edges. In the previous case, however, the edges were parallel to the x-axis, and therefore the length was the same as the x component of the length. In this case, the edges need not be parallel to the y axis, and therefore the y component of the lengths must be used. Thus, the rotation about the x-axis is θ, where

$$\cos\theta = \frac{\|c_i \cdot (0,1,0)\|}{\|c_m \cdot (0,1,0)\|} \qquad (2)$$

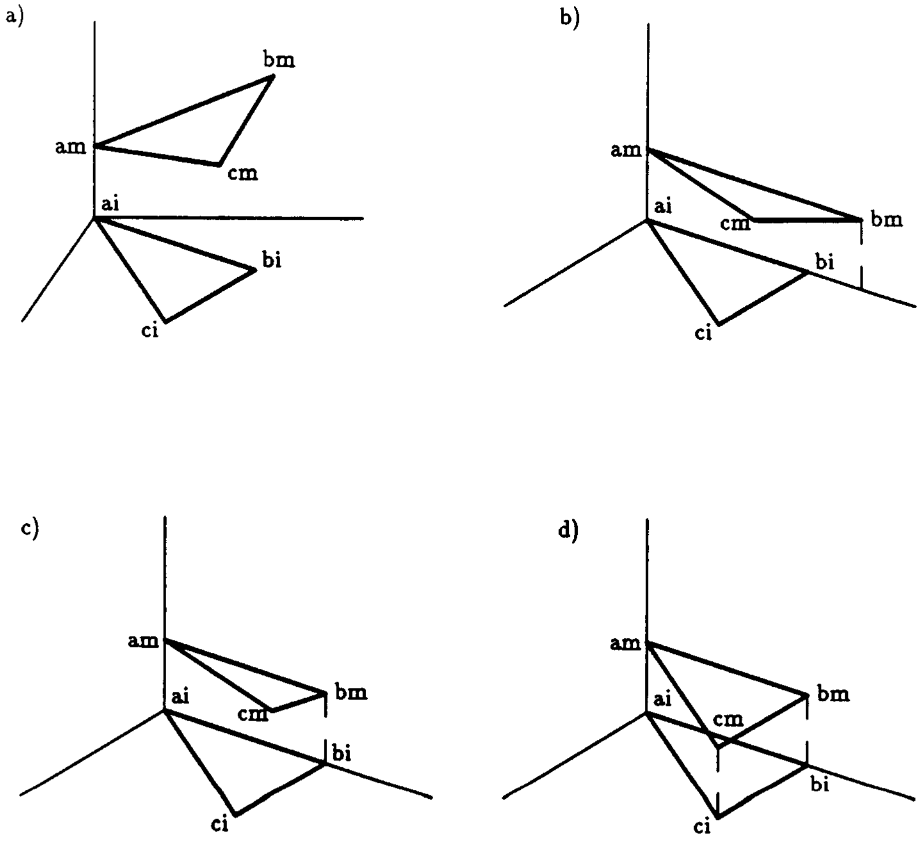

Figure 15. The alignment process: a) the points a_i and a_m are brought into correspondence, b) the ab edges are aligned, c) the points b_i and b_m are brought into correspondence, d) the points c_i and c_m are brought into correspondence.

for $0 \leq \cos\theta \leq 1$.

If the model distance is shorter than the image distance, there is no transformation that aligns the model and the image. Furthermore, if the rotation does not actually bring c''''_{mi} into correspondence with c_i, then there is also no alignment. This latter case can result because the rotations are those that will bring the points into alignment if there actually is a consistent solution. If there is no solution then it may still be possible to solve for the rotations, but they will not bring all three points into alignment.

The final rotation brings the plane defined by the three model points into correspondence with the image plane, as illustrated in Figure 15d. This combination of translations and rotations can now be used to map the model into the image, in order to determine if the object is in fact in the image at this position and orientation.

Now it is necessary to solve for a linear scale factor as a sixth unknown, in order to simulate distance from the viewer by a scale factor. The final two rotations which align b_m with b_i, and c_m with c_i are the only computations affected by a change in scale. The alignment of b_m involves movement of b_{mi} along the x-axis, whereas the alignment of c_m involves movement of c_{mi} in both the x and y directions.

Because the movement of b_{mi} is a sliding along the x-axis, only the x-component, x_b, changes. The change is given by the rotation ϕ about the y-axis, as in (1). With a scale factor, s, this becomes

$$x'_b = sx_b(\cos\phi). \tag{3}$$

Similarly the movement of c_{mi} in the y direction is given by the rotation θ about the x-axis, as in (2). With a scale factor this becomes

$$y'_c = sy_c(\cos\theta). \tag{4}$$

The movement of c_{mi} in the x direction is given by the rotations about both the x- and the y-axis. From the matrix for a combined rotation about the x- and y- axis we obtain

$$x' = (x\cos\theta + y\sin\phi\sin\theta).$$

Thus with the scale factor, the x component of c_m is

$$x'_c = s(x_c\cos\theta + y_c\sin\phi\sin\theta). \tag{5}$$

Now we have three equations in the three unknowns, s, θ, and ϕ. One method to solve for s is to substitute for $\cos\theta$, $\sin\theta$, and $\sin\phi$ in (5). From (3) we know that,

$$\sin\phi = \frac{1}{sx_b}\sqrt{s^2x_b^2 - x_b'^2}. \tag{6}$$

And similarly from (4),

$$\sin\theta = \frac{1}{sy_c}\sqrt{s^2y_c^2 - y_c'^2}. \tag{7}$$

Substituting (6) and (7) into (5) and simplifying yields

$$s^2(x_b x'_c - x_c x'_b)^2 = (s^2 x_b^2 - x_b'^2)(s^2 y_c^2 - y_c'^2).$$

Expanding out the terms we obtain

$$s^4(x_b^2 y_c^2) - s^2(x_b^2 y_c'^2 + x_b'^2 y_c^2 + (x_b x'_c - x_c x'_b)^2) + x_b'^2 y_c'^2,$$

a quadratic in s^2. While there are generally two possible solutions, it can be shown that only one of the solutions will specify possible values of $\cos\phi$ and $\cos\theta$ [28].

Having solved for the scale of an object, the final two rotations ϕ and θ can be computed using (1) and (2) modified to account for the scale factor,

$$s(\cos\phi) = \frac{\|b_i \cdot (1,0,0)\|}{\|b_m \cdot (1,0,0)\|}$$

and

$$s(\cos\theta) = \frac{\|c_i \cdot (0,1,0)\|}{\|c_m \cdot (0,1,0)\|}.$$

Thus solving for scale involves the computation of s, and slight a modification to the computation of the final two rotations.

The Planar Method

For planar models, the three-dimensional alignment task only involves mapping points from one plane to another. Therefore, a planar model can be aligned with an image using only planar operations. In effect, the actual three-dimensional rotation and translation are simulated in the plane.

Initially the model points are translated and rotated such that a_i and a_m are coincident, and the $a_i b_i$ and $a_m b_m$ edges are aligned. Then a coordinate shift is performed to place a_i and a_m at $(0,0)$, and b_i and b_m on the x-axis, as shown in Figure 16a. This is the two-dimensional analog of the translation and z-axis rotation done in the three-dimensional alignment algorithm of the previous section.

Now, rather than performing three-dimensional rotations to align b_m with b_i, and c_m with c_i, the model is simply scaled and then sheared. In the three-dimensional case, there is a rotation about the y-axis that moves b_m in the x-direction, a rotation about the x-axis that moves c_m in the y-direction, and a combined rotation about the two axes that moves c_m in the x-direction. Thus the planar operations must simulate these three motions.

To make b_m coincident with b_i, simulating the rotation about the y-axis, it is only necessary to scale the x-coordinates of the model points by

$$\frac{\|b_i\|}{\|b_m\|}$$

obtaining b'_m and c'_m as illustrated in Figure 16b.

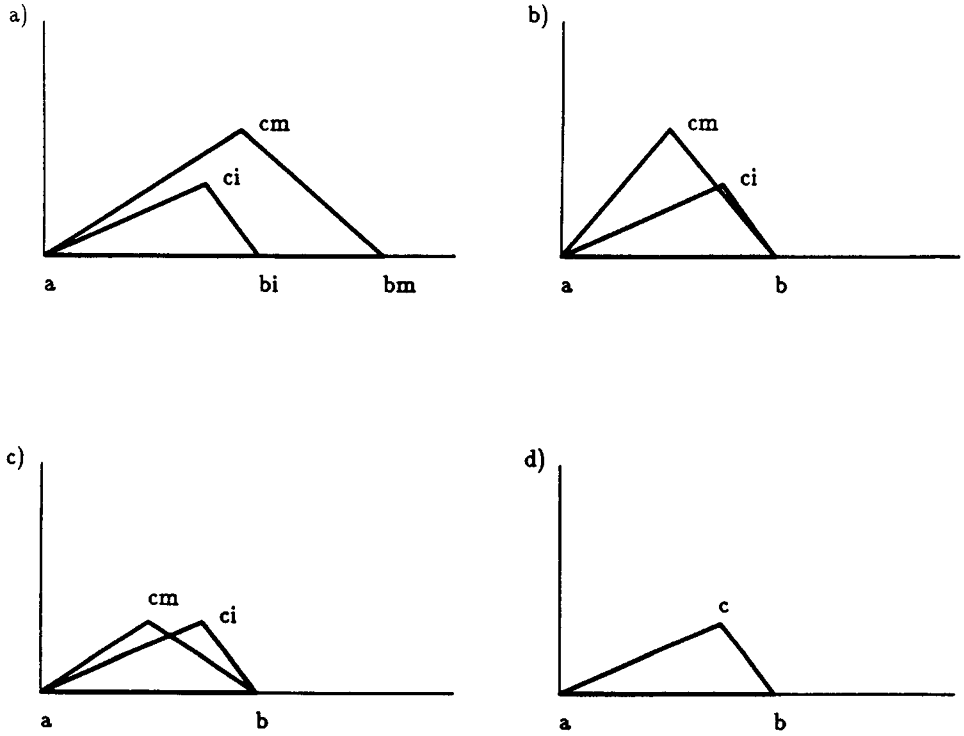

Figure 16. The planar alignment process: a) the points a_i and a_m are brought into correspondence and the ab edges are aligned, b) the points b_i and b_m are brought into correspondence, c) the point c_m is brought onto the line through c_i and parallel to the x-axis, and d) the points c_i and c_m are brought into correspondence.

Making c_m coincident with c_i involves two steps. First, simulating the rotation about the x-axis, the y-coordinates of the model points are scaled by

$$\frac{\|c_i \cdot (0,1)\|}{\|c_m \cdot (0,1)\|}$$

This puts c''_m on the line through the point c_i, and parallel to the x-axis, as illustrated in Figure 16c.

Now c''_m moves in the x-direction such that it is coincident with c_i. For the point c_m, the distance d, is

$$x_{c_i} - x_{c_m}\frac{x_{b_i}}{x_{b_m}}. \tag{8}$$

In general, a point will move in the x-direction proportional to its y-position, because points further from the x-axis will move more due to the combined effect of the two three-dimensional rotations. Thus the general form for computing the new x-coordinate of a point is

$$x' = x + d\frac{y}{y_{c_i}}$$

for d given in (8). This final shearing brings all three points into correspondence, as shown in Figure 16d.

Unlike the three-dimensional alignment algorithm, the planar alignment algorithm finds an alignment for any triplet of model and image points, because the x and y scaling and shearing can bring any two pairs of points into alignment. Thus an additional point must be used to ensure that the alignment is consistent. This also provides an added flexibility in matching, however, because operations such as stretching of an object will not cause the matching to fail. Then the fact that there are separate scale factors for the x and y directions will allow other points in a planar model to be aligned with a stretched image of an object.

7 Aligning Non-Flat Objects

We have seen how to align and match flat objects which have three-dimensional positional uncertainty. In this section, we briefly consider several means of extending the alignment method to the domain of non-flat objects.

At one extreme, an object can be represented by a single three-dimensional model. Using such a representation, the alignment computation is the same as for a flat object. A triplet of corresponding model and image points are used to determine the position and orientation of the model. In order to project the aligned model into the image, however, it is necessary to determine which portions of the object are visible given its position and orientation. This is accomplished using standard hidden line and surface elimination algorithms (cf. [15]).

In order for an alignment of a three-dimensional model with an image to be valid, it must be the case that the three points used in alignment are all visible given the position and orientation of the object that they specify. Thus for each possible alignment, the three model points must be checked to make sure that they are not on hidden edges or surfaces.

A single three-dimensional model is not well suited to the problem of visual recognition, because it does not explicitly represent information about viewpoint. Thus, before a three-dimensional model can be matched to an image, hidden lines and surfaces be removed.

Furthermore, it is difficult to construct a single three-dimensional model from image data, because multiple views must be integrated together. While there are systems for building models from multiple views [18], the problem is quite difficult, and often requires a large number of images. Therefore, most vision systems require object models to be entered explicitly, rather than forming them automatically from images. Moreover, forming a single three-dimensional model discards the viewpoint information which is necessary for recognition. Thus, viewpoint must be reconstructed by eliminating hidden surfaces.

It also appears to be the case that people do not use a single three-dimensional model of an object for recognition. Matching such a model against an image requires rotating the model in three-space to bring it into correspondence with the image (or vice versa). There is evidence, however, that people are quite slow at three-dimensional mental rotation [27]. Recognition, on the other hand, is extremely rapid. This suggests that people may not manipulate three-dimensional models in recognition.

Planar Models

At the opposite extreme, it is possible to model an object in terms of planar views. In other words, to use the 2D projection of an object from a given viewing position as a flat model. For this kind of representation, it is necessary to have multiple models of a single object, corresponding to different possible views.

A planar view of an object will only match images that are taken at exactly the same viewing angle as the model view. At any other angle, the object is only approximated. Therefore the use of planar models comes at the cost of some inaccuracy in the matching process. The use of planar models, however, simplifies the recognition process. First, a flat model can be aligned with an image using only the planar operations of translation, rotation, scaling and shearing – simulating three-dimensional motion in the plane. Second, it is not necessary to eliminate hidden surfaces in order to match the model with the image.

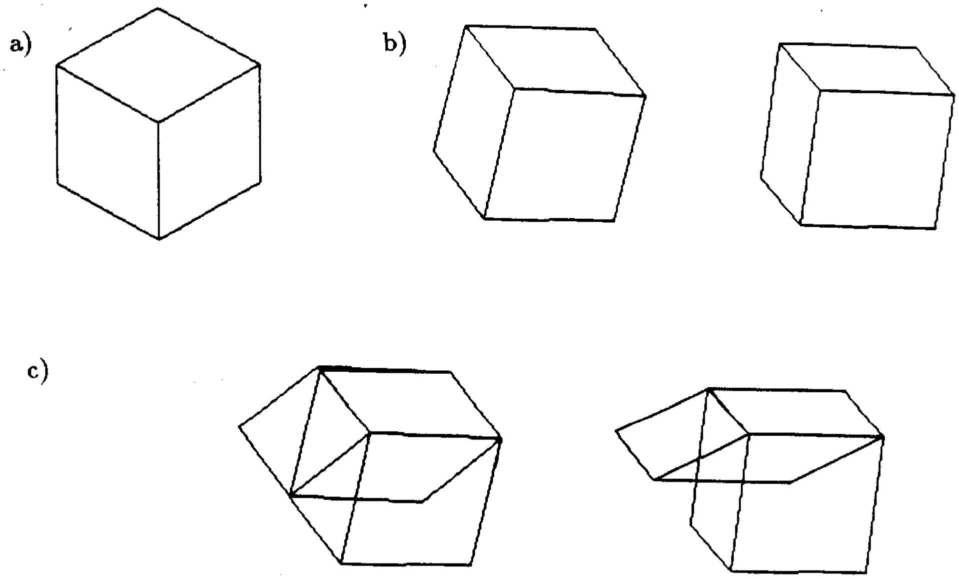

Figure 17. a) a planar model of a cube, b) two different views of the cube, c) planar alignments of the model with the views.

A planar view of a non-planar object specifies only two-dimensional information about points that actually have three-dimensional position. Therefore, an alignment of such a model with an image will only correctly transform points that are coplanar (in three-dimensions) with the three model points used for alignment. Other model points will be treated as if they are the same plane even though they are not. The amount of error this introduces is proportional to both the distance from the point to the match plane, and the difference in angle between the model viewpoint and the image viewpoint. For an object like a cube, which is poorly approximated by a single plane, the error rapidly becomes substantial as the object is rotated. Figure 17a shows an isometric view of a cube which will serve as a planar model. Part (b) of the figure shows two views of the cube, rotated by $\frac{\pi}{9}$ and $\frac{\pi}{6}$, respectively. In part (c), the model has been aligned with each of the views, using the three alignment points marked by dots. Since the model is planar, the alignment is only correct for those edges which are coplanar (on the actual object) with the alignment points.

Multiple Planar Alignments

An intermediate possible representation is to use multiple views, rather than a single object-centered model, but not have the views be simple two-dimensional projections of the object. For instance, it is possible to use a "2.5D" representation which encodes a given view of an object in terms of almost-planar pieces. Thus each planar section of the model could be aligned with the image separately. Using such a representation, the cube model in Figure 17a would have each face marked as being a different planar piece.

This representation requires a model for each viewpoint from which a different set of planar pieces are visible. A multiple-view model of a cube, for example, would consist of three views: a view when one surface is visible, a view when two surfaces are visible, and a view when three surfaces are visible. A cube with each surface distinctly marked would have twenty-six different views: eight views with three visible surfaces (one for each vertex), twelve views with two visible surfaces (one for each edge), and one view of each of the six surfaces alone.

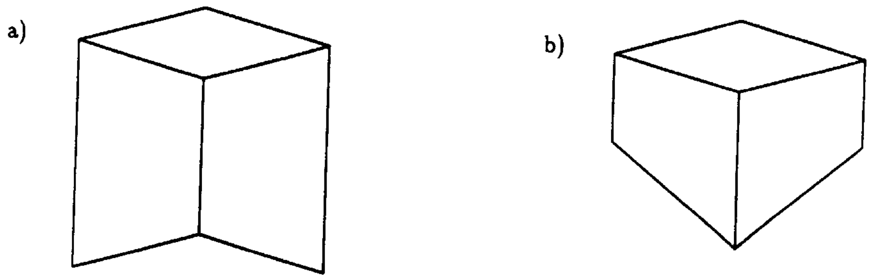

Figure 18. Two non-cubes that match a cube using the locally-rigid alignment algorithm.

Using this representation, and aligning each model plane separately, the cube model can be matched perfectly against the views in parts (b) and (c) of Figure 17. The problem with this matching scheme is that it is too general, accepting the non-cubes in Figure 18 as cubes.

This over-generality can be restricted by noting that the alignments for the various planes must be related. Even without knowing the true three-dimensional angles between faces, if the object is convex then it cannot be rotated such that both external edges become lower than the internal edge without exposing a new surface on the bottom. The convexity of the external contour must be maintained. This eliminates the non-cube in Figure 18a. We are currently investigating the use of such relations to limit the flexibility of the multiple-alignment method of matching.

Human recognition of three-dimensional objects from two-dimensional images is also characterized by overly general matching. For instance, the partial pyramid in Figure 19 cannot actually be a projection of a solid object, even though it appears to be one [24]. In order to correspond to a real object the three numbered edges must come together at a point, but they do not. This over-generality is probably not due to acuity limitations, as evidenced by another set of experiments.

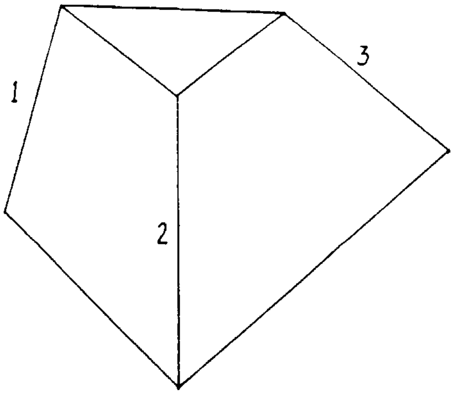

Figure 19. While appearing to be a partial pyramid, this figure cannot correspond to an actual solid object (from [24]).

Even for common objects, people are quite lax about what contours they accept as being real projections of an object. For instance, in deciding whether cube-like figures are actually projections of cubes people are only about 85% correct [24]. The errors are apparently due to the way people match models to images, rather than limitations in visual acuity. This is demonstrated by the fact that performance can be greatly improved using special tricks. In order for a figure to be a projection of a cube, the angles about the central vertex must all be at least 90 degrees. Using this rule, people can score nearly 100% correct on the cube classification task [24].

Local Alignment

In order to perform separate alignments for each object plane, it is necessary to know what parts of the object are nearly coplanar. One possibility is to use "2.5D" models that indicate which parts of the object are on the same local plane. Thus the models still specify only two-dimensional positional information, but they group edges together according to what actual object plane they lie on.

Another possibility is to determine where the different object planes are at recognition time. One method of doing this is to use the pairs of model and image features to define regions for local alignment. For instance, the set of model points can be triangulated, and then each triangular region can be aligned with the image separately. To map the model to the image, all the model points inside a given triangle are transformed using the alignment for that triangle. To the extent that a triangle falls on a nearly planar part of an object, this will produce a correct alignment for that region. By aligning each locally neighboring triple of model features found in the triangulation, rigidity is preserved for each triangle, but not for the object as a whole.

Points in a given triangle will not be mapped into the image correctly if they are not on the same plane of the object as the three points defining the triangle. Such points, however, will usually be transformed to a point near their correct position because the alignment will be partially correct. Therefore the locally-rigid alignment process can be iterated, starting with a three-point alignment, and using the partial match to pick potential corresponding model and image points for computing more refined, and less globally rigid, alignments.

This algorithm starts by computing a standard three-point rigid alignment. If the initial alignment does not match any portion of the model to the image, then no further action is taken. If there is a match, however, then the alignment is used to pick additional corresponding pairs of model and image points. The model points are triangulated and each triangle is aligned separately. This process is repeated until either a good match of the model to the image is obtained, or the triangles become small.

To show how the local alignment algorithm works, we consider aligning two slightly different views of a station wagon. First, we show that a single rigid alignment is not sufficient to align the two views. Figure 20 shows a single three-dimensional alignment, using three points chosen on the rear of the station wagon. Part (a) shows the "model" image that will be transformed, and part (b) shows the image. The three corresponding points in the two images are marked. Part (c) shows the synthetic image formed by aligning the "model" points with the image points – translating, rotating and scaling the "model" according to the alignment points. Part (d) shows the superposition of the image with the synthetic image. It can be seen from the superposition that the side of the wagon has been substantially foreshortened in the two images. Therefore, similarly to the cube in the previous section, the alignment brings the rear plane of the wagon into good agreement, but does not do so for the side.

Figure 21 shows the same two views with a set of corresponding model and image points marked. The points are triangulated, and separate local alignments are performed for each triangle. The synthetic image in part (c) is formed by mapping each pixel from the "model" image according to the transformation specified by the triangle that the pixel falls in. Pixels that fall in no triangle are transformed using triangles formed between the four corners of the image and the convex hull of the point set. The superposition of the image with the synthetic image in part (d) shows that the alignment is very good.

Thus by using a relatively small number of corresponding points, two different planar views of an object can be matched to one another with a high degree of accuracy. This multiple alignment method allows three-dimensional objects with three-dimensional positional freedom to be recognized from a single two-dimensional view using planar models.

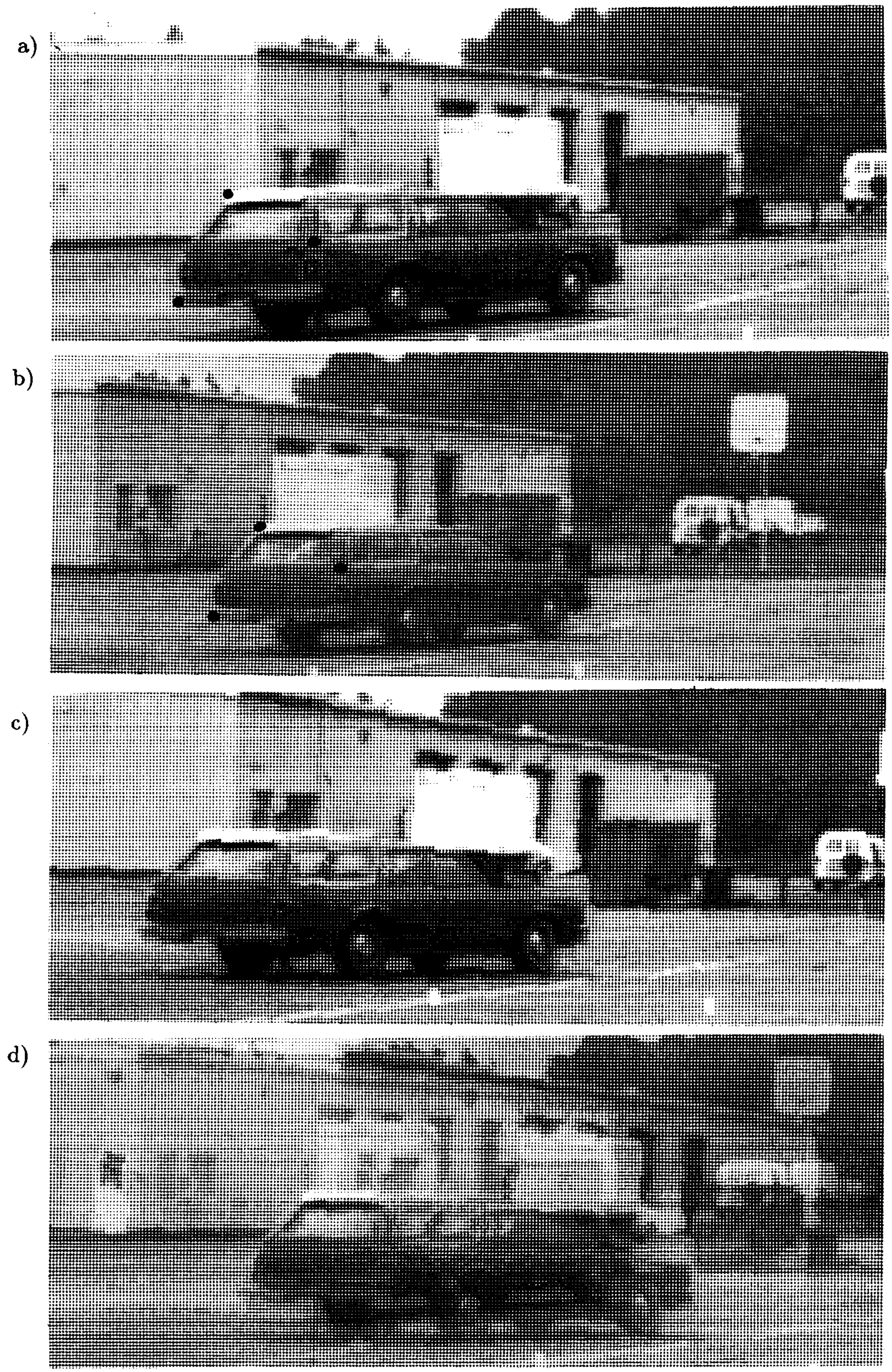

Figure 20. Three-dimensional alignment of two views of a station wagon: a) "model" with three points marked, b) image with three points marked, c) synthetic image created by aligning "model" with image, d) superposition of image and synthetic image.

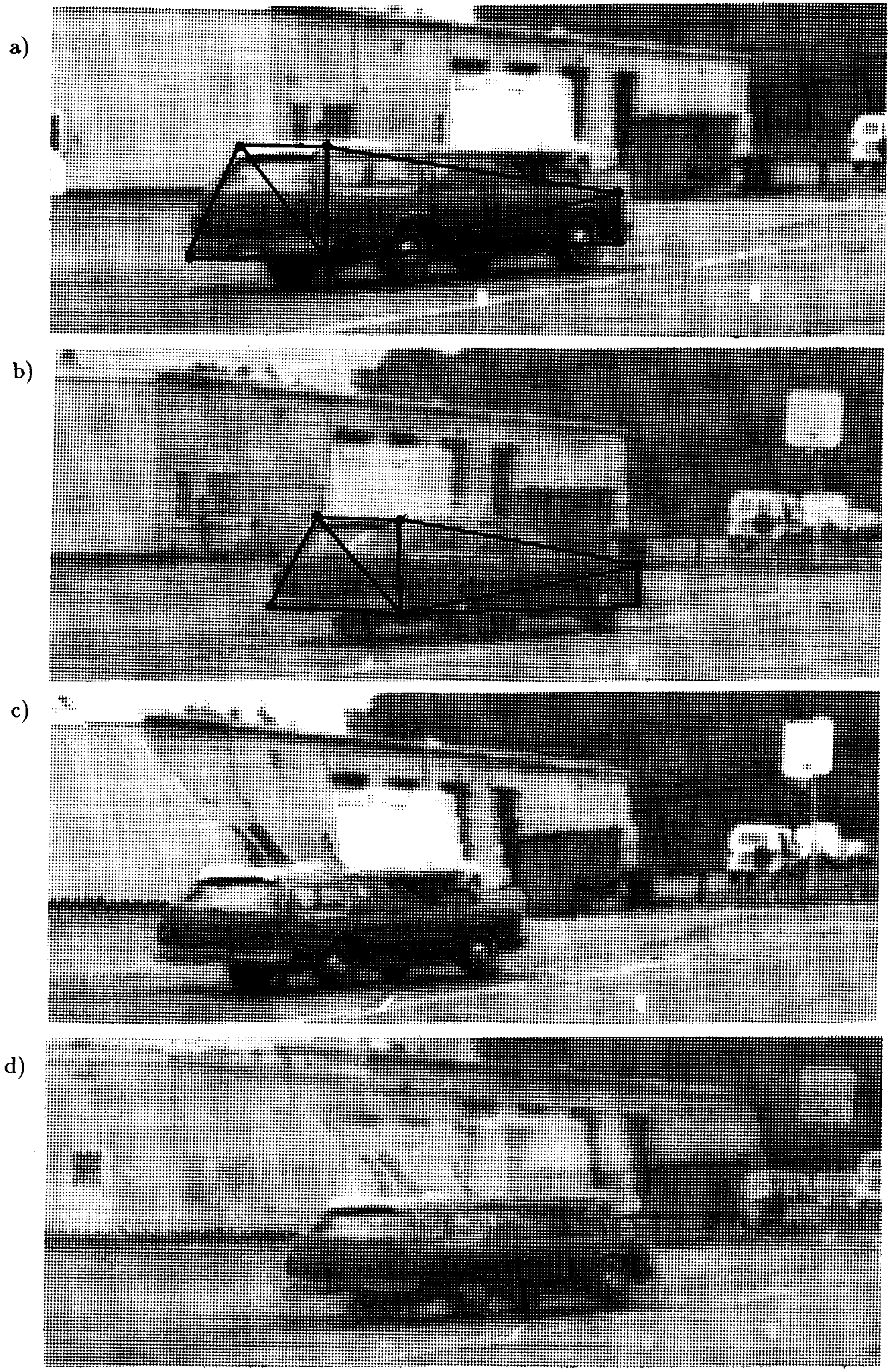

Figure 21. Multiple local alignment of two views of a station wagon: a) “model” with three points marked, b) image with three points marked, c) synthetic image created by aligning “model” with image, d) superposition of image and synthetic image.

8 Summary

Recognition is generally viewed as a search through the space of possible positions and orientations of objects. The idea of the *alignment* approach is to separate this search into two stages. In the first stage, the position, orientation, and scale of an object are found using a minimal amount of information, such as three pairs of model and image points. In the second stage, the alignment is used to map the object model into image coordinates for comparison with the image.

There are two major advantages of this approach. First, by using a small fixed number of model and image features, we avoid structuring recognition as search through an exponential space. Second, by storing object models such that they are aligned with one another, a single alignment computation can be used to map multiple objects into the image.

The key observation behind the approach is that the alignment can be performed with a small amount of information. For example, three points are sufficient to determine the position, orientation and scale of a rigid object in three-space from a single two-dimensional image. Similarly, two points and an orientation measure can also be used to solve for this alignment.

We have implemented a recognizer using the alignment method. This system chooses features for alignment using a scale-space segmentation of edge contours. The multiple scale description is used for choosing reliable alignment points, and for associating descriptive labels with them. Coarse scale segments are described both in terms of their shape, and the structure of the scale-space hierarchy at the next finer level. This produces relatively distinctive features for use in finding possible alignments of a model and an image.

To demonstrate the recognition method, several examples were shown of the recognizer finding a widget with arbitrary three-dimensional position and orientation, in a two-dimensional image. From these examples it can be seen that the alignment algorithm finds a small number of reasonable matches of widgets to images, even when the widget is foreshortened, scaled, and partly occluded.

Finally, we have briefly discussed how the alignment method can be extended to recognize rigid objects in general, either by using a three-dimensional model, or by using two-dimensional models and multiple local alignments.

References

1. Attneave, F. 1954. Some Informational Aspects of Visual Perception, *Psych. Review* **61**, pp. 183-193.

2. Asada, H. and Brady, M. 1984. The Curvature Primal Sketch, MIT Artificial Intell. Lab. Memo, No. 758.

3. Baird, H. 1986. ***Model-Based Image Matching Using Location.*** Cambridge: MIT Press.

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

5. Bolles, R.C. and Cain, R.A. 1982. Recognizing and Locating Partially Visible Objects: The Local Feature Focus Method, ***Int. J. Robotics Res.*** **1(3)**, pp. 57-82.

6. Bolles, R.C. and Horaud, P. 1986. 3DPO: A Three-Dimensional Part Orientation System ***Int. J. Robotics Res.*** **5(3)**, pp. 3-26.

7. Brady, M. and Asada, H. 1984. Smoothed Local Symmetries and Their Implementation, MIT Artificial Intell. Lab. Memo, No. 757.

8. Brooks, R.A. 1981. Symbolic Reasoning Around 3-D Models and 2-D Images, *Artificial Intell. J.* **17**, pp. 285-348.

9. Canny, J. 1986. A Computational Approach to Edge Detection, *IEEE Trans. Pat. Anal. and Mach. Intel.* **8(6)**, pp. 679-698.

10. Connell, J.H. 1985. Learning Shape Descriptions: Generating and Generalizing Models of Visual Objects, MIT Artificial Intell. Lab. Tech. Report, No. 853.

11. Faugeras, O.D. and Hebert, M. 1986. The Representation, Recognition, and Locating of 3-D Objects, ***Int. J. Robotics Res.*** **5(3)**, pp. 27-52.

12. Fischler, M.A. and Bolles, R.C. 1986. Perceptual Organization and Curve Partitioning, ***IEEE Trans. Pat. Anal. and Mach. Intel.*** **8(1)**, pp. 100-104.

13. 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)**, pp 381-395.

14. Fleck, M.M. 1985. Local Rotational Symmetries, MIT Artificial Intell. Lab. Tech. Report, No. 852.

15. Foley, J.D. and VanDam A. 1984. *Fundamentals of Interactive Computer Graphics.* Reading, Mass: Addison.

16. Grimson, W.E.L. and Lozano-Pérez, T. 1984. Model-Based Recognition and Localization from Sparse Range or Tactile Data, *Int. J. Robotics Res.* **3(3)**, pp. 3-35.

17. Grimson, W.E.L. and Lozano-Pérez, T. 1987. Localizing Overlapping Parts by Searching the Interpretation Tree, to appear in *IEEE Trans. Pat. Anal. and Mach. Intel.*.

18. Herman, M., Kanade, T. and Kurve, S. 1984. Incremental Acquisition of a Three-Dimensional Scene Model from Images, *IEEE Trans. Pat. Anal. and Mach. Intell.* **6(3)**, pp. 331-339.

19. Hoffman, D.D. and Richards, W.A. 1986. Parts of Recognition, in *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, edited by A.P. Pentland. Norwood N.J.: Ablex.

20. Kalvin, A., Schonberg, E., Schwartz, J.T. and Sharir, M. 1985. Two Dimensional Model Based Boundary Matching Using Footprints, Courant Institute.

21. Lowe, D.G. 1985. *Perceptual Organization and Visual Recognition.* Boston: Kluwer.

22. Lowe, D.G. 1986. Three-Dimensional Object Recognition from a Single Two-Dimensional View, Courant Institute Robotics Report, No. 62.

23. Nevatia, R. and Binford, T.O. 1977. Description and Recognition of Curved Objects, *Artificial Intell. J.* **8**, pp 77-98.

24. Perkins, D.N. 1983. Why the Human Perceiver Is a Bad Machine, in *Human and Machine Vision* edited by J. Beck, B. Hope and A. Rosenfeld, pp. 341-364. Orlando: Academic Press.

25. Preparata, F.P. and Shamos, M.I. 1985. *Computational Geometry An Introduction*, New York: Springer-Verlag.

26. Rock, I. and Leaman, R. 1963. An Experimental Analysis of Visual Symmetry, *Acta Psychologica* **23**, pp. 171-183.

27. Shepard, R.N. and Cooper, L.A. 1982. *Mental Images and Their Transformations*, Cambridge: MIT Press.

28. Ullman, S. 1987. An Approach to Object Recognition: Aligning Pictorial Descriptions, MIT Artificial Intell. Lab. Tech. Report, No. 931.

29. Wiser, M. 1981. The Role of Intrinsic Axes in Shape Recognition, in Proc. 3rd Annual Conf. of Cog. Sci. Soc.

30. Witkin, A.P. 1985. Scale-Space Filtering, in *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, edited by A.P. Pentland, pp. 5-19. Norwood N.J.: Ablex.

31. Yuille, A.L. and Poggio, T. 1986. Scaling Theorems for Zero Crossings, *IEEE Trans. Pat. Anal. and Mach. Intell.* **8(1)**, pp. 15-25.

CS-TR Scanning Project
Document Control Form

Date : 10 / 12 / 95

Report # AIM - 937

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
□ Laboratory for Computer Science (LCS)

Document Type:

□ Technical Report (TR) ☒ Technical Memo (TM)
□ Other:

Document Information

Number of pages: 43(49-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
□ Double-sided

Intended to be printed as :
□ Single-sided or
☒ Double-sided

Print type:
□ Typewriter □ Offset Press ☒ Laser Print
□ InkJet Printer □ Unknown □ Other:

Check each if included with document:

☒ DOD Form (2) □ Funding Agent Form □ Cover Page
□ Spine □ Printers Notes □ Photo negatives
□ Other:

Page Data:

Blank Pages(by page number):

Photographs/Tonal Material (by page number): 2,4,20-24,37-38

Other (note description/page number):

Description : Page Number:

(A) IMAGE MAP: (1-43) UN#'ED TITLE PAGE, 1-42
(44-49) SCANCONTROL, DOD(2), TRGT'S(3)

(B) CUT&PASTE FIG.S ON PAGES 2-4, 13, 15, 17-18, 20-25, 27, 30, 33-35, 37-38

Scanning Agent Signoff:
Date Received: 10 / 12 / 95 Date Scanned: 11 / 27 / 95 Date Returned: 11 / 30 / 95

Scanning Agent Signature: Michael W. Cook

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AIM-937	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Recognizing Rigid Objects by Aligning Them with an Image		5. TYPE OF REPORT & PERIOD COVERED AI Memo
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Daniel P. Huttenlocher Shimon Ullman		8. CONTRACT OR GRANT NUMBER(s) N0014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE January, 1987
		13. NUMBER OF PAGES 42
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer vision, object recognition, model-based vision, alignment

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents an approach to recognition where an object is first aligned with an image using a small number of pairs of model and image features, and then the aligned model is compared directly against the image. For instance, the position, orientation, and scale of an object in three-space can be determined from three pairs of corresponding model and image points. By using a small fixed number of features to determine

20. Abstract (cont'd)

position and orientation, the alignment method avoids structuring the recognition process as an exponential search. To demonstrate the method, we present some examples of recognizing flat, rigid objects with arbitrary three-dimensional position, orientation, and scale, from a single two-dimensional image. The recognition system chooses features for alignment using a scale-space segmentation of edge contours. Segments are described in terms of both their shape and the structure of the scale-space hierarchy at the next finer level, producing distinctive features for use in finding possible alignments. Finally, the method is extended to the domain of non-flat objects as well.