G. Bebis

Object Recognition Using Alignment

Daniel P. Huttenlocher and Shimon Ullman

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA

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, which yields relatively distinctive feature labels. Finally, the method is extended to the domain of non-flat objects as well.

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. The presence of more than one object in an image further 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 [9] [4].

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 [9].

Lowe's recent work [10] 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. 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. We then consider extending the method to the domain of rigid objects in general.

The current task cannot be handled by recognition systems that assume rigid objects with no distortion [9] [4] [7] [2], or by systems that only allow parameterized variation of rigid models [5]. The task is similar to that of Lowe,

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.

CH2465-3/87/0000/0102$01.00 © 1987 IEEE

Proc. of 1st Inter. Conf on Comp Vision

who addresses the problem of three-dimensional recognition from a single two-dimensional view [10]. 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 divide the recognition process 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 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.

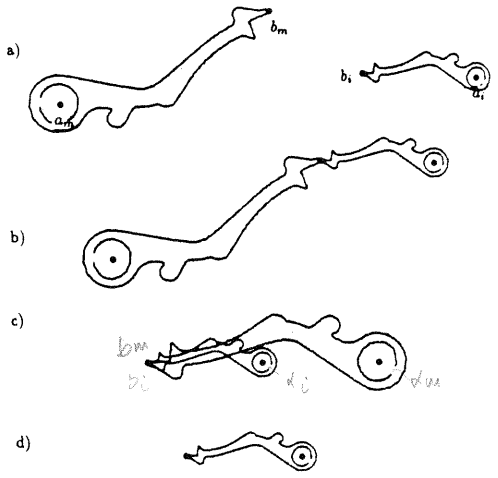

Figure 1. Two dimensional translation, rotation and scaling of one object to match another.

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 1a 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 1b. 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 b_i, 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.

The alignment method requires identifying potentially corresponding model and image points such as (a_m, a_i) in Figure 1. 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

This section briefly discusses the limitations of some recent recognition systems with respect to the recognition task described above (for a more general review see [3]). Recognition is generally structured as a search for the largest pairing of model and image features for which there exists a single transformation mapping each model feature to its corresponding image feature [9] [4] [10] [5] [7]. For i image features and m model features there are at most $p = i \times m$ pairs of model and image features. Because of occluded image points, and image points which 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 this space of possible matchings of model and image features.

i) The problem with using the identity of features in recognition is that there is a tradeoff between the distinctiveness of a feature and the robustness with which it can be recognized. Since systems which rely heavily on the iden-

tity of features must use relatively distinctive features, they tend to be sensitive to noise and occlusion in the image.

For instance, the LFF [4] recognition system forms feature descriptions using clusters of spatially proximate features. A "focus feature" in each cluster 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. The use of local feature clusters yields relatively distinctive features. However, it is difficult to ensure that each cluster is composed of features from a single object, making the system sensitive to the position and orientation of neighboring and occluding objects.

The second method of limiting the possible matches is to use relations between features to eliminate inconsistent pairs of model and image features [9] [7] [4] [2]. For instance, in order for two pairs of model and image features (m_1, i_1) and (m_2, i_2) to be part of a consistent set, the distance between the image features i_1 and i_2 must be the same as the distance between the model features m_1 and m_2, 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.

The 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 distances and angles cannot be measured in a two-dimensional image. Therefore these constraints cannot be exploited by 3D from 2D recognition systems. Relations which are invariant under projection tend to be much weaker than distance and angle relations.

The SCERPO system [10] [11] is the only one to address the problem of three-dimensional recognition from a single two-dimensional view. Image edges are grouped together using proximity and parallelism. Thus (similarly to LFF) the groups are sensitive to the positioning of neighboring objects. The local groupings are then used to form pairs of model and image features, and the space of pairs is searched for the maximally consistent subset. Under perspective projection, however, there are no simple pairwise checks such as distance and angle relations to incrementally check consistency. Instead, the perspective viewing equation is solved for each subset, using a Newton-Raphson iteration technique.

3 The Alignment Method of Recognition

As we have seen, 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.

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.

4 Alignment Points

The alignment operation requires finding pairs of corresponding model and image points, and model and image orientations. 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. The labels, however, must be relatively insensitive to partial occlusion, juxtaposition, and projective distortion, while being as distinctive as possible. If the number of pairs, p, is kept small, then little or no search is necessary to find the correct alignment.

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. Thus while we have chosen to use a shape description based on intensity edges to define labeled edge segments that are used for alignment, it is possible to augment this using other kinds of points. For instance, vertices where multiple edges come together make good alignment points. Other kinds of potential alignment points are cusps, tips, deep concavities and small blobs.

Forming a shape description based on intensity edges involves segmenting edge contours into primitive pieces. Many techniques (e.g., curvature primal sketch [1]) segment contours at maximal curvature points. Part of the reason for choosing maximal curvature points is their supposed special importance in recognizing line drawings. For instance, contours constructed by linear interpolation between maximal curvature points are easily recognizable. Lowe [10], however, points out that a contour constructed using the points midway between maximal curvature points is also easily recognizable. Maximal curvature points, per se, are not important.

Therefore, 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.

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

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 [6]. Given the array of edge points, the points are chained together into contours wherever there is an unambiguous eight-way neighbor. 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 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. 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.

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 [4] and SCERPO [10]).

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.

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 [14] hierarchy of segments. Since coarser scales of smoothing do not introduce zero crossings that were not present at finer scales [15], the hierarchy forms a tree of segments from coarser to finer scales.

Figure 2 shows a three-level scale-space curvature segmentation of the edge contours of a widget. Each part of the figure shows the same contour, segmented according to the curvature smoothed at different scales (using Gaussian filters of size $\sigma = 7$, 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).

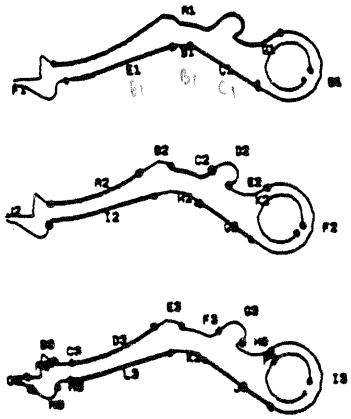

Figure 2. A scale-space segmentation of a widget, where the contours are segmented at inflections in the smoothed curvature. The coarsest scale is at the top..

Each segment of edge contour is classified according to whether it is `curved` or `straight`. The curved segments are also classified by the degree of closure: `open` or `closed`. The hierarchy of segmentations at different scales can also be viewed in terms of the correspondence between segments at neighboring scales, as show in Figure 3. Each segment at a coarse scale corresponds to one or more segments at the next finer scale. Each segment in the tree is indicated by its label from Figure 2 and by the type of segment: `straight`, `curve`, and `open-curve`.

Multi-scale descriptions are formed using the types of segments at a given scale, plus the structure of the hierarchy

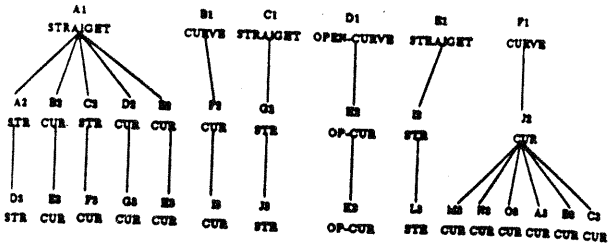

Figure 3. The tree corresponding to the curvature scale-space segmentation in Figure 2.

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 pattern of alternating direction of curvature.

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` segments at the next level whereas `C1` and `E1` are each composed of a `single` segment. 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`).

Since the coarse scale segments are relatively distinct, they are used to define alignment points, based on the type of the segment. 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 at the ends of zero curvature regions, they are relatively stable, making it reasonable to use endpoints and midpoints for matching.

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. The bulk of the time is for edge detection and feature extraction. 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 2 and Figure 3. The model is just the result of processing the image of the isolated widget in the same manner as any image. Thus for flat objects, models are formed directly from an image of an object.

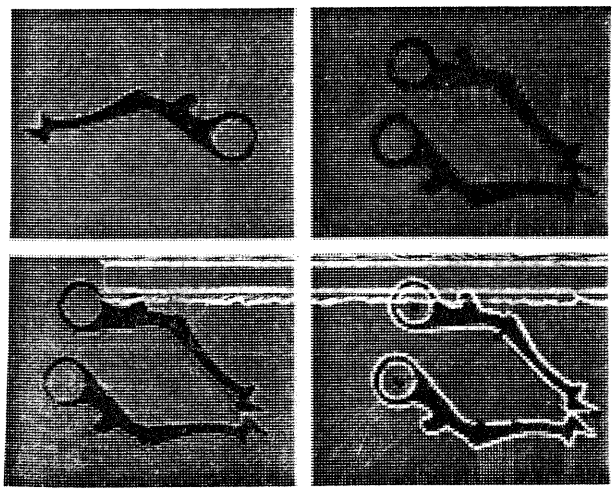

Figure 4. Matching a widget against an image of two widgets in the plane, a) the grey-level image and intensity edges of the model, b) the grey-level image, c) the image and the intensity edges, d) the edges of the aligned model superimposed on (c), with the alignment points marked.

The example in Figure 4 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.

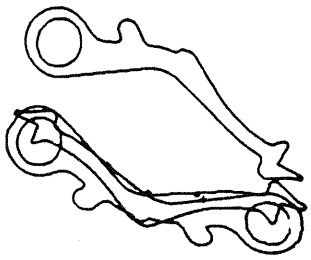

Figure 5. An alignment of a widget with an image that does not match the model edge contour with image edges.

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 5 superimposed with the image edges. The alignment is found because the two straight edges are indistinguishable, and the three points used in computing the alignment were the two straight edges and the bend.

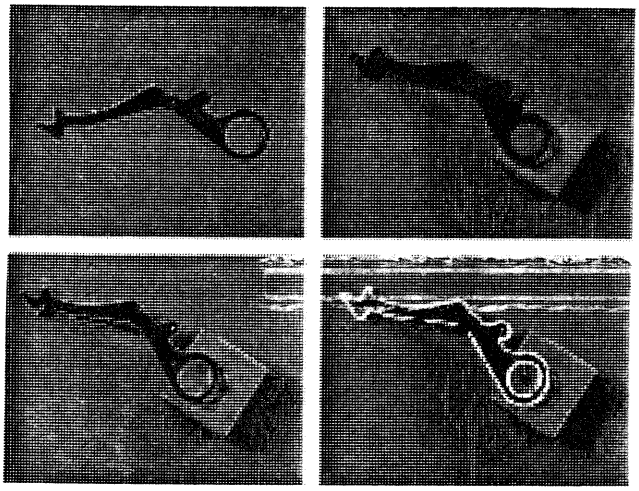

Figure 6. Matching a widget against an image of a foreshortened widget (see Figure 4 for an explanation).

Figure 6 shows a widget that has been tilted approximately 30 degrees by resting one end of it on a block, foreshortening

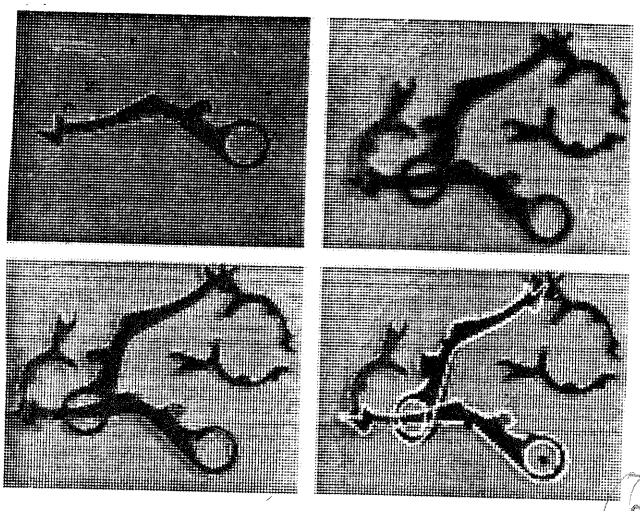

Figure 7. Matching a widget against an image of two partly occluded widgets (see Figure 4 for an explanation).

the image. The recognizer finds a single best position and orientation, which 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 7 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.

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 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 note that the alignment method can be simulated using only planar operations.

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

In contrast, using three pairs of model and image points to solve for *perspective* projection can yield up to four distinct solutions. It is interesting to note, however, that the ambiguous cases often involve solutions where all but one of the possibilities has a high perspective distortion. Thus three points may also be sufficient to solve for position and orientation under perspective viewing, if solutions with high perspective distortion are discarded.

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 8a.

Now it is necessary to rotate the model about three orthogonal axes to align b_m and c_m with their corresponding

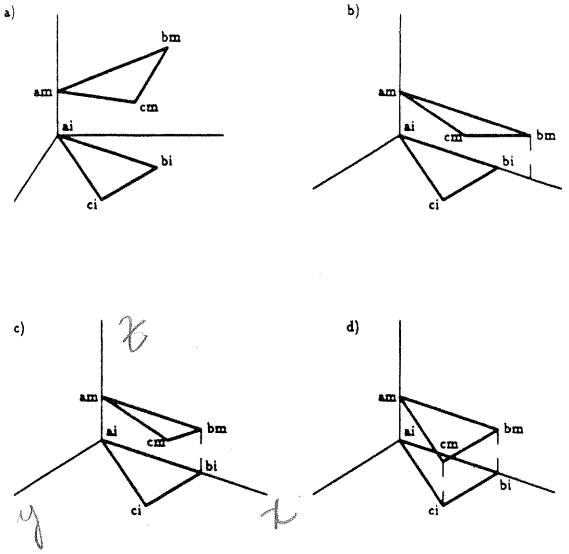

Figure 8. The alignment process: a) the points a_i and a_m are brought into correspondence, b) the ab edges are aligned, c) the points b_i and b_m are brought into correspondence, d) the points c_i and c_m are brought into correspondence.

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 8b.

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 8c.

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

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

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 8d. 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

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

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

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

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,

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

And similarly from (4),

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

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

$$s^2(x_bx_c' - x_cx_b')^2 = (s^2x_b^2 - x_b'^2)(s^2y_c^2 - y_c'^2).$$

Expanding out the terms we obtain

$$s^4(x_b^2y_c^2) - s^2(x_b^2y_c'^2 + x_b'^2y_c^2 + (x_bx_c' - x_cx_b')^2) + x_b'^2y_c'^2,$$

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 $\cos\theta$ [12].

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.

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, as described in [13].

7 Aligning Non-Flat Objects

Extending the alignment method to recognize non-flat objects is relatively straightforward. The only difference between using a three-dimensional model and a two-dimension model is the need to eliminate portions of the object which are not visible from a given position and orientation. Thus, after computing a possible alignment, model edges and surfaces which are not visible from the specified position and orientation must be removed before the model is mapped into the image.

For a three-dimensional model, it must also be ensured that the three points used in alignment are all visible given the position and orientation of the object which 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.

Rather than using a single three-dimensional model of an object, it is possible to use multiple planar views, with one view for each position from which a different set of object surfaces are visible. Using multiple local alignments, such a planar model can be matched to a variety of different images of the object.

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

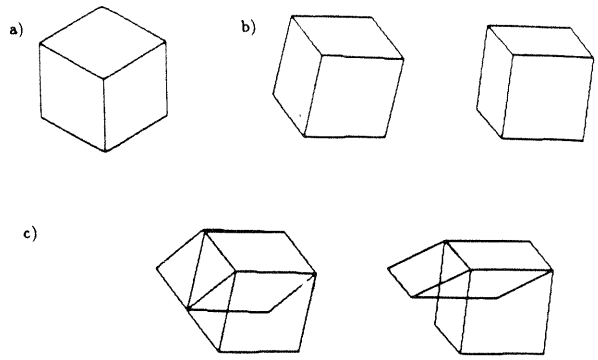

Figure 9. a) a planar model of a cube, b) two different views of the cube, c) planar alignments of the model with the views.

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. 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 9a 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.

Thus, using a single alignment, a planar model of a three-dimensional object will only match images taken from close to the same viewing angle as the model view. By using multiple alignments, however, a flat model can be matched to images from a wide range of different viewpoints. One method of finding regions for performing local alignments is to triangulate the set of model features, and separately align each triangular region. All the model points inside a given triangle are then mapped into the image 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 actually coplanar with 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.

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.

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.

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. Asada, H. and Brady, M. 1984. The Curvature Primal Sketch, MIT Artificial Intell. Lab. Memo, No. 758.
2. Baird, H. 1986. *Model-Based Image Matching Using Location.* Cambridge: MIT Press.
3. Besl, P.J. and Jain, R.C. 1985. Three-Dimensional Object Recognition, *ACM Computing Surveys* bf 17(1), pp. 75-154.
4. 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.
5. Brooks, R.A. 1981. Symbolic Reasoning Around 3-D Models and 2-D Images, *Artificial Intell. J.* **17**, pp. 285-348.
6. Canny, J. 1986. A Computational Approach to Edge Detection, *IEEE Trans. Pat. Anal. and Mach. Intel.* **8(6)**, pp. 679-698.
7. 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.
8. 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.
9. 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.*.
10. Lowe, D.G. 1985. *Perceptual Organization and Visual Recognition.* Boston: Kluwer.
11. Lowe, D.G. 1986. Three-Dimensional Object Recognition from a Single Two-Dimensional View, Courant Institute Robotics Report, No. 62.
12. Ullman, S. 1987. An Approach to Object Recognition: Aligning Pictorial Descriptions, MIT Artificial Intell. Lab. Tech. Report, No. 931.
13. Ullman, S. and Huttenlocher, D.P. 1987. Recognzing Rigid Objects by Aligning them With an Image, MIT Artificial Intell. Lab. Memo, No 937.
14. 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.
15. Yuille, A.L. and Poggio, T. 1986. Scaling Theorems for Zero Crossings, *IEEE Trans. Pat. Anal. and Mach. Intell.* **8(1)**, pp. 15-25.