

ON SHAPES

Dana H. Ballard and Daniel Sabbah

University of Rochester
Rochester, N.Y. 14627

Abstract

An important problem in computer vision is to detect the presence of a given rigid, 3-d object. Such objects can be detected from intrinsic image data by a Hough transform technique coupled with a polyhedral representation. More precisely, a known object is detected by finding changes in orientation, translation, and scale of the object from its canonical description.

1. Introduction

An important task in computer vision is to locate a given rigid, 3-d object. This paper describes a representation for rigid 3-d objects together with a Hough algorithm which allows for the detection of such objects from intrinsic image data. Since a description of each object is available, it is detected by finding changes in orientation, translation, and scale of the object from the canonical description.

A central concept of the method is the *object-centered coordinate frame*. Certain frames seem natural for given objects. For example, a cube is most naturally described with a coordinate frame aligned with its edges. However, there are other choices; one places an axis along the cube's longest diagonal. Experiments with humans have shown that the assignments of such systems can be made dynamically and that different choices often influence the questions that can be easily answered about the object [Hinton, 1979].

The general object recognition task can be thought of as building a description of the object that has two parts: (1) the description of the object itself (with respect to an object-centered frame); and (2) the transformation of the object-centered frame to the viewer-centered (image) frame. The reason for this decomposition is parsimony: different views of the object should have minimal impact on its description. This is achieved by factoring the object's description into two sets of parameters, one which is view-independent (the object-centered component) and one which is view-varying (the viewing transformation). Often a description of the object is known beforehand and the task reduces to finding the object-frame to viewer-frame transformation. This paper describes a method for handling this case.

The methodology assumes that the image has been segmented into planar components such as that shown in Figure 1. In the image, the planar component corresponds to a region of pixels, each of which corresponds to a point in the world which has the same surface normal direction.

Regions are bounded by linear edge segments which represent the boundaries between planes of different orientation. It might seem a demanding requirement to process image data into the form shown by Figure 1; however, this type of description is easy to compute from range data [Sugihara, 1979] and can also be computed when the surfaces have well-behaved reflectance properties [Ikeuchi, 1980] or once an image has been processed to line-drawing form [Kanade, 1979]. Thus it seems plausible to assume planar components as input.

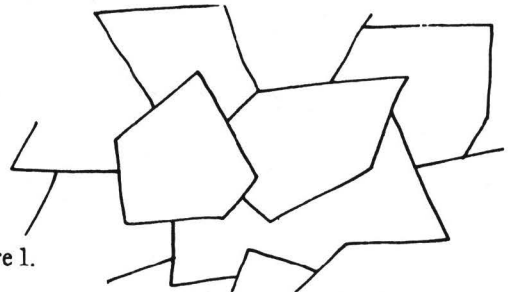


Figure 1.

The representation describes a rigid object with respect to an object-centered coordinate frame. Since the object description is known, the task of determining if the object is in an image can be posed as:

Is there a transformation of a subset of image features such that the subset can be explained as the transformed object?

If the answer to this question is no, then the object is not present. If yes, then the transformation provides all the necessary information about the object. It turns out that this transformation is completely specified by seven parameters: three for translation; three for orientation; and one for scale. This paper assumes orthographic projection, so z cannot be determined, leaving six parameters. The principal feature of our method is that it decouples the interdependence of these three groups of parameters. In other words, the orientation of the object can be detected without knowing its translation. In fact, there is a natural dominance of parameters:

scale > orientation > translation

The algorithm used is a form of Hough transform using *constraint tables* [Ballard, 1981]. Image features constrain the values for the transform parameters. Each image feature maps only a set of allowable parameter values. When all the features are taken into account, the mapping is many-to-one onto plausible parameter values. Since the cost of this method is exponential in the number of parameters

considered together, the decoupling of parameters mentioned before is very significant [Ballard, 1981].

The outline of the rest of the presentation is as follows. First we describe the location of a shape in 2-d, when scale and orientation are fixed. Then we describe how to detect the orientation of the shape without knowing its location. The method for orientation is then extended to 3-d. Finally, we show how the general, 3-d problem can be decomposed into a 3-d orientation problem with a set of 2-d location problems.

2. Representation

In representing shapes, one must keep in mind four different coordinate systems:

1. a two-dimensional coordinate system within which planar surfaces of an object are described (x_p, y_p)
2. a three-dimensional coordinate system (x_b, y_b, z_b) which is an object-centered system
3. a three-dimensional viewer-centered coordinate system (x_v, y_v, z_v) which describes the environment being imaged
4. an image coordinate system (x_i, y_i) which describes how the world is projected onto a retina.

Assuming orthographic projection, $x_i = x_v$ and $y_i = y_v$. The transformation T_{pb} is a known transformation that must be described. T_{bv} is a transformation which one would like to discover given image data.

2.1 Two-Dimensional Translation

In the 2-d problem assume a shape of fixed orientation and scale. The representation describes the shape as a set of pairs of vectors $((e_1, v_1), (e_2, v_2), \dots, (e_n, v_n))$. The first element of the set, e_i , is a directed edge of the boundary of the shape; that is, following e_i , e_{i+1} , etc., traverses the boundary of the object (see Figure 2).

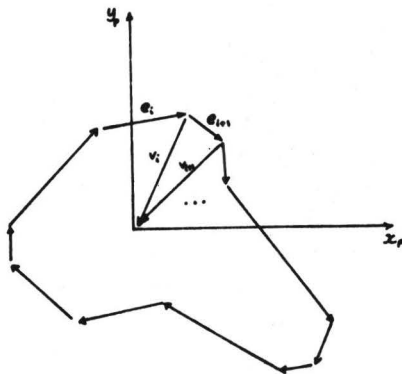


Figure 2.

Each v_i denotes a vector from the head of e_i to the origin of an object-centered coordinate frame. The representation is redundant since $v_i - e_{i+1} = v_{i+1}$ but this redundancy is necessary for the parallel algorithms described in Section 3.

Each e_i is represented as an angular orientation and a length (Θ_i, l_i) . v_i is represented as an offset $(\Delta x, \Delta y)$. Ordered pairs (e_i, v_i) form a translation offset table indexed by e_i .

For any orientation e_i one can quickly find the offset v_i which specifies the position of the body-centered origin.

2.2 Two-Dimensional Orientation

In the plane the angular orientation of a shape is specified by a single parameter Θ which denotes the rotation of the body-centered frame with respect to the viewer-centered frame. The representation encodes the orientation i of each edge e_i with respect to the body-centered frame. That way, given an edge Θ_i from an arbitrarily rotated shape which is rotated Θ_m from world coordinates we can compute the rotation of the local frame as $\Theta_m - \Theta_i$.

The orientation offsets Θ_i are sorted by length l_i and form a two-dimensional rotation-offset table of (l_i, Θ_i) index-value pairs. Figure 3 is a graphical display of this data where the e_i pairs are plotted as vectors emanating from the origin. The plot is a 2-d extended Gaussian map [Smith, 1979; Bajcsy, 1980].

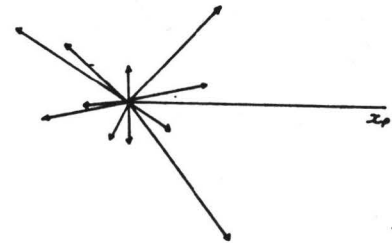


Figure 3.

2.3 Three-Dimensional Orientation

In three dimensions, surface areas and surface normals are analogous to edge lengths and edge orientations. The 3-d orientation representation is a form of extended gaussian map. Imagine an object with smooth surfaces where a surface unit normal vector is computed for each small value of unit area. The gaussian map is the histogram of such normals, plotted in spherical coordinates. The representation, termed an **orientation map**, has two important features which incorporate earlier ideas from [Horn, 1980; Smith, 1979; Bajcsy, 1980].

First, the representation is explicitly defined with respect to a body-centered coordinate frame whereas the gaussian map is usually defined with respect to a world coordinate frame. In other words, if the object is rotated with respect to the world, the gaussian map also rotates, but our orientation map does not. Second, there is a hierarchy of structure. The imaged surface of the object is assumed to be segmented into components which are locally smooth surfaces. Each component surface is represented in the

orientation map as a single orientation vector together with an associated surface area.

The specific form of orientation map is a spherical coordinate system for orientation, and sorting by area. An object is a set of equal-area surfaces

$$\begin{aligned} &((A_1 (\alpha_{11}, \beta_{11}), (\alpha_{12}, \beta_{12}), \dots, (\alpha_{1k_1}, \beta_{1k_1})), \\ &(A_2 (\alpha_{21}, \beta_{21}), \dots, (\alpha_{2k_2}, \beta_{2k_2})), \\ &(A_n (\alpha_{n1}, \beta_{n1}), \dots, (\alpha_{nk_n}, \beta_{nk_n}))). \end{aligned}$$

The above notation means that there are k_i surfaces in the set with area A_j . Furthermore $A_j > A_{j+1}$, $1 \leq j \leq n-1$. Both α_{ij} and β_{ij} are direction-cosines which describe the orientation of the ij^{th} surface. In the model representation, the direction cosines are measured with respect to the body-centered frame. In the input data, the direction cosines are measured with respect to the viewer-centered frame.

3. Algorithms

3.1 Detecting Translation in 2-D

In this case the reference origin coordinates (x_c, y_c) are the only parameters (remember, rotation and scaling have been fixed). Thus if an edge point (x, y) in an image with gradient orientation θ and length l is part of the shape, its reference origin must be at

$$(x + \Delta x(\theta, l), y + \Delta y(\theta, l)).$$

This constraint is incorporated into a generalized Hough algorithm as follows:

Algorithm A1: Generalized Hough for Shape Translation

Step 0. Make a table for the shape to be located like that shown in Figure 2.

Step 1. Form an accumulator array of possible reference points $A(x_{cmin}:x_{cmax}, y_{cmin}:y_{cmax})$ initialized to zero.

Step 2. For each edge point do the following:

Step 2.1. Compute $\theta, l(x)$

Step 2.2a. Calculate the possible centers, i.e.,
foreach table entry i for θ such that
 $\theta = \theta_i$ and $l = l_i$ compute

$$x_c := x + \Delta x_i$$

$$y_c := y + \Delta y_i$$

Step 2.2b. Increment the accumulator array

$$A(x_c, y_c) := A(x_c, y_c) + 1$$

Step 3. Possible locations for the shape are given by maxima in the array A .

The results of using this transform to detect a shape appear in Ballard[1979].

3.2 Detecting 2-d Orientation

First consider the problem of detecting a rotated two-dimensional object of known scale. As a simple example, take the shape shown in Figure 2, which appears together

with random, directed line segments in Fig. 4.

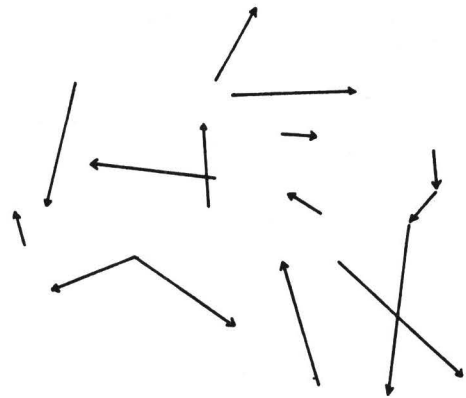


Figure 4: Object and random line-segment background.

Now for the shape define an intrinsic coordinate system. Then an orientation transform is defined as the following table: for each edge of the shape measure the angle required to rotate the edge such that it is aligned with the intrinsic x-axis. Figure 5 shows this for the shape.

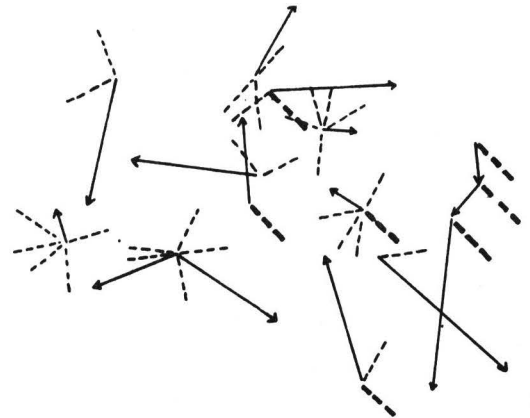


Figure 5: Orientation transform.

This graphical example can be expressed formally as the following Hough technique:

Algorithm A2: Orientation Detection

Step 0. Form an Accumulator array of possible orientations $A(0..2\pi)$.

Step 1. Foreach edge oriented at angle α , assume its part of the shape and compute the shape's three possible orientations β as

$$\alpha := \alpha - \theta_i \quad n = 0, 1, 2$$

and increment the appropriate accumulator array address

$$A(\beta) := A(\beta) + 1$$

Step 2. Maxima in A correspond to possible orientations for the shape.

If the orientation of the edges in the original "image" was random, the shape should stand out as a particular value of β . However, there is a potential problem with the method: the edges need not be touching each other's end points or even be anywhere near each other to be detected as a shape by the Hough technique. This problem is handled by examining the appropriate edges in the image domain by a kind of reverse Hough transform. Once we have detected a maximum in the accumulator array β^* we can eliminate all edges in the image which did not contribute to this maximum. Thus only edges whose orientation $\alpha = \theta_i + \beta^*$ (where θ_i is a table entry) are kept.

This pruning can be a parallel process if the edges are stored in a binary array $E(x,y,a)$ where

$$E(x,y,a) = 1 \text{ if there is an edge} \\ \text{at } xy \text{ with orientation } a \\ 0 \text{ otherwise}$$

Once the edge space has been pruned, isolated edges are eliminated leaving only groups of proximal, correctly oriented edges. This does not mean that each group is the shape, since a class of correctly oriented edges but which may be translated with respect to each other are detected by the algorithm. Figure 6 shows some examples of these.

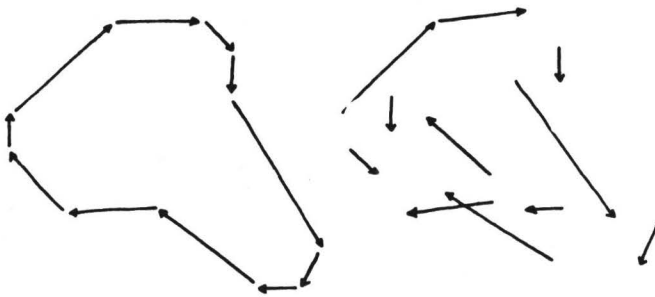


Figure 6: Shapes that would be detected by the transform.

However, the correct shapes can be distinguished from these other shapes by Algorithm A1 once the orientation is known. The key difference between this method and [Ballard, 1981] is that much less space is required since the parameters are detected *sequentially*. Only a small array of N_θ entries is required to detect orientation and another small array of N_x intrinsic origin locations is required to distinguish the triangular shape. The second array is smaller than the similar array required by [Ballard, 1981] since the reverse transformation localizes the possible origin locations.

3.3 Scale Changes

The method of the previous section can be extended to deal with scale. In this case we can change the table in Section 3.2 to record the relative length of each edge with respect to a standard intrinsic length. Now for each entry we record a (scale,orientation) pair where the scale is the difference in length of the edge from the reference and the orientation angle is as before. Now scale can be handled in the same way as was orientation. This is shown by the following algorithm.

Algorithm A3: Scale and Orientation Detection

Step 0. Form an Accumulator array of possible orientations and scale factors

$$A(0:2\pi:S_{\min}:S_{\max}).$$

Step 1. Foreach edge (l,α) and foreach triangle edge e_k ,

compute the orientation β as

$$\beta := \alpha + \Theta(k)$$

and compute the scale S as

$$S := l/l(k)$$

then increment the appropriate accumulator address

$$A(\beta,s) := A(\beta,s) + 1$$

Step 2. Maxima in the Array A correspond to instances of the shape.

Results of using this transform on an image of directed line segments are shown in Fig. 7. Figure 7a shows the example of Figure 4 which has been rotated by 180° and scaled by a factor of five. Figure 7b shows $A(\beta,s)$.

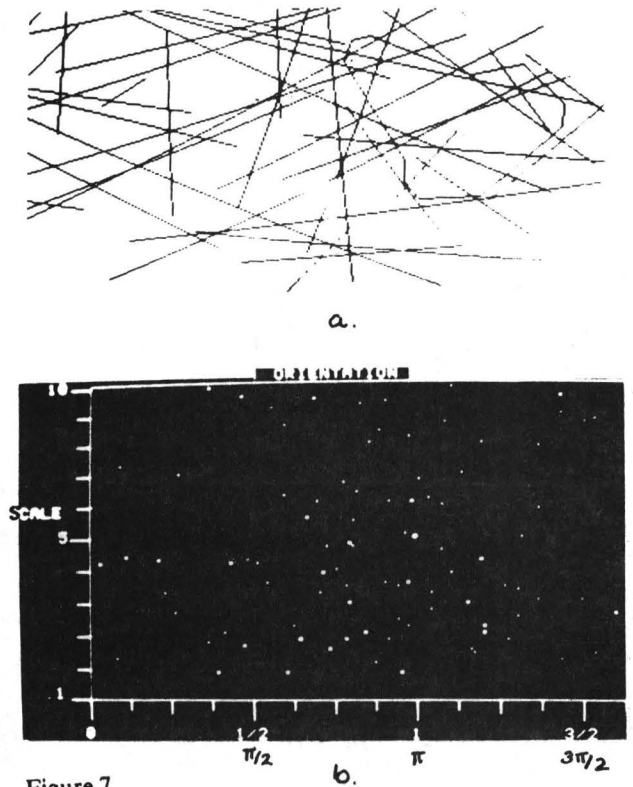


Figure 7

3.4 Detecting Orientation in 3-D

Detecting 3-d orientation is more difficult than 2-d orientation. This is because, given a single surface normal, one can only ascertain the orientation of the body-centered frame with respect to the normal. The frame may still be arbitrarily rotated with respect to that normal (see Figure 8); however, given a set of planes at different orientations, one can determine the final orientation parameter as the intersection of the loci of the individual normals.

Let us consider how the surface orientation constrains the object-centered frame orientation. Since the constraint is exploited in a subtle manner we develop it in stages. First, we examine the form of the basic constraint on the intrinsic frame given a surface normal in the viewer-centered frame. Next we show how this constraint could be implemented in general. Finally we show a short-cut that uses the matching normal in the body-centered frame.

Figure 8a and b summarizes the problem for a single normal. Given a normal, n_v in the viewer-centered frame ($x_v - y_v - z_v$), the problem is to determine the orientation of the object-centered frame ($x_b - y_b - z_b$). Figure 8a shows this case. Although the object-centered frame must in fact be at a specific orientation, this orientation cannot be determined from a single normal. Instead a single normal constrains the frame to be rotated about the normal at some arbitrary angle. Figure 8b shows this for the x_b -axis.

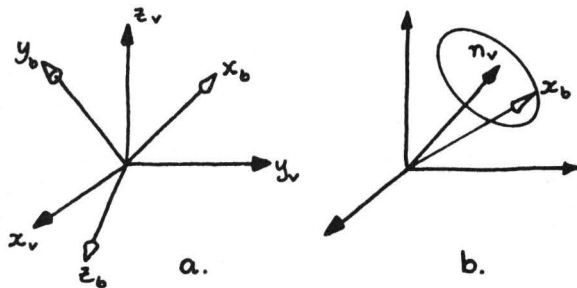


Figure 8.

While each individual normal only constrains the frame to a given locus, the frame loci of several distinct normals will be different, and will intersect at an orientation that is correct for the body-centered frame. This orientation can be specified by three parameters. Arbitrarily we use two direction cosines that specify the x_b axis orientation and one direction cosine that specifies the x -projection of the y_b axis. Once these are known the rest can be determined from the constraints that the vectors are unit vectors and form a right-handed cartesian frame. That is, where

$$x_b = (\alpha_x, \beta_x, \gamma_x)$$

and α_x and β_x are known,

$$\gamma_x = \sqrt{1 - \alpha_x^2 - \beta_x^2}$$

y_b is given by $(\alpha_y, \beta_y, \gamma_y)$ where

$$x_b \cdot y_b = 0$$

and

$$\|y_b\| = 1.$$

Finally, z_b is determined from

$$z_b = x_b \times y_b.$$

The three components α_x, β_x , and α_y are determined by a Hough technique. Each match between a normal n_v and a normal n_b determines a locus for a particular set of $\{\alpha_x, \beta_x, \alpha_y\}$. The algorithm is outlined as follows: Let $H(\alpha_x, \beta_x, \alpha_y)$ be the number of normals for which the location $(\alpha_x, \beta_x, \alpha_y)$ of the body-centered frame satisfies the fundamental constraint. The array H is initialized to zero and is generated by:

Foreach match between n_v and n_b
 increment all $H(\alpha_x, \beta_x, \alpha_y)$ where
 $\alpha_x \beta_x \alpha_y$ is on the appropriate locus.

The maximum in the array H corresponds to the orientation of the body-centered frame.

How can one determine the set $\{\alpha_x, \beta_x, \alpha_y\}$ for a particular match? If one possible orientation of the body-centered frame is known a standard technique for generating the other orientations is to: (1) rotate n_v and $(x_b y_b z_b)$ so that n_v is aligned with an axis (say z_v); (2) rotate the resultant $x_b y_b z_b$ about z_v ; and (3) rotate n_v and the rotated $(x_b y_b z_b)$ so that n_v is aligned with its original direction (Fig. 9).

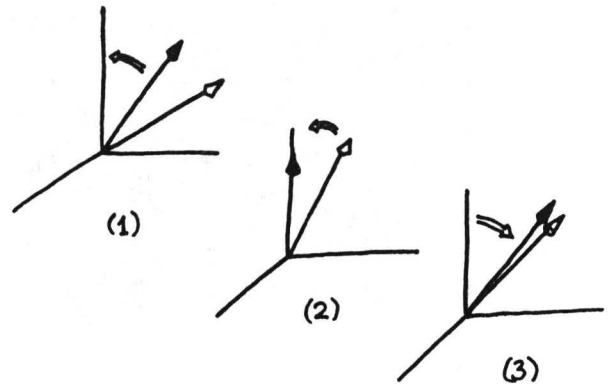


Figure 9.

A short-cut can be taken that takes advantage of the following fact:

If the normal n_b , along with $(x_b y_b z_b)$, is rotated so that n_b is aligned with the z_b axis, then the rotated $x_b y_b z_b$ lie on the appropriate locus.

In other words, the steps in the algorithm that determine the set $\{\alpha_x, \beta_x, \alpha_y\}$ corresponding to a match between n_b and n_v are as follows:

- (1) rotate $x_b y_b z_b$ so that n_b is aligned with z_b (the coordinates of n_b specify this rotation);
- (2) rotate the result around z_v (this generates points on the locus);
- (3) rotate the result of step (2) so that n_b is aligned with n_v (the coordinates of n_v specify this rotation).

Figure 10 shows the results of a simple test of the algorithm. An object with four facets was generated and converted to our surface normal encoding. Next this object was rotated about the z -axis by 30 degrees and this data was used as input. The grain of the array H is -10 units to +9 units. Figure 10 shows this array, mapping three dimensions onto two. The boxes represent successive planes through H with different values of α_y . Within the boxes the different values for $H(\alpha_x, \beta_x, \gamma_y)$ are shown. The maximum is at $\alpha_x = 9, \beta_x = 0, \gamma_y = 5$, as expected. The above algorithm assumes scale is known, but can be modified to use the method of Algorithm A3 to deal with unknown scale.

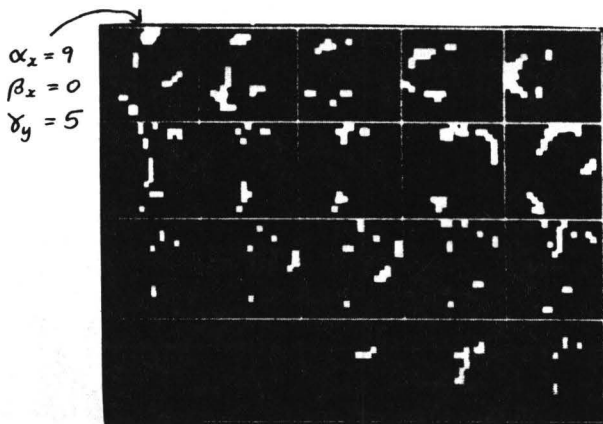


Figure 10.

3.6 Detecting 3-d Rigid Bodies

Three-dimensional rigid bodies can be detected by first detecting the orientation of the body. Once the orientation of the body-centered coordinate frame is known it is only necessary to see if the oriented faces have the correct shape. Furthermore this check can be accomplished by a two-dimensional Hough transform using the representation described in Section 3.1. Thus the five-parameter 3-d problem can be decomposed into a three-parameter orientation problem followed by a two-parameter translation problem.

Initially the representation is augmented by one more parameter which allows the complete specification of the transformation from planar coordinates (x_p, y_p) to body-centered coordinates (x_b, y_b, z_b) .

This final parameter is a rotation w about the out-of-

plane axis z_p . This specifies a transformation T_{pb} which transforms the coordinates of a planar face into the correct position with respect to the body-centered frame. The orientation representation already includes an area and two angles (A, a, b) for each face. Now append w so that the representation for each face is (A, a, b, w) . The solution to the face translation problem is to transform the planar shape into the viewer-centered frame by

$$\begin{aligned} e_i &:= T_{pb} T_{bv} e_i \\ v_i &:= T_{pb} T_{bv} v_i \end{aligned}$$

and then apply Algorithm A1 to find the appropriate x_w, y_w translation.

4. Discussion

The principal result of this paper is to extend the method of [Ballard, 1981] to 3-d. Three dimensions are harder than two as the number of parameters to describe a shape transformation is seven instead of four. This paper showed: (1) how the Hough technique could be applied sequentially to detect groups of parameters; and (2) how to derive constraints within these groups. The methodology is attractive as it can be implemented in parallel, connection-based architectures [Feldman, 1981; Feldman and Ballard, 1981].

References

- Bajcsy, R., "Three-dimensional scene analysis," *Proc., 5th IJ CPR*, Miami, December 1980.
- Ballard, D.H., "Generalizing the Hough Transform to detect arbitrary shapes," TR 55, Computer Science Dept., U. Rochester, 1979; *Pattern Recognition* 13, 2, 111-122, 1981.
- Ballard, D.H., "Parameter networks: Toward a theory of low-level vision," TR 75, Computer Science Dept., U. Rochester, April 1981.
- Feldman, J.A., "A connectionist model of visual memory," in G.E. Hinton and J.A. Anderson (Eds.), *Parallel Models of Associative Memory*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers, 1981.
- Feldman, J.A. and D.H. Ballard, "Computing with connections," TR72, Computer Science Dept, U. Rochester, 1981.
- Hinton, G., "Some demonstrations of the effects of structural descriptions in mental imagery," *Cognitive Science* 3, 1979.
- Horn, B.K.P. and B.G. Schunck, "Determining optical flow," M.I.T. AI Memo 572, April 1980.
- Ikeuchi, K., "Numerical shape from shading and occluding contours in a single view," M.I.T. AI Memo 566, February 1980.
- Kanade, T., "Recovery of the three-dimensional shape of an object from a single view," TR CS-79-153, Computer Science Dept., Carnegie-Mellon U., October 1979.
- Marr, D. and T. Poggio, "A theory of human stereo vision," M.I.T. AI Memo 451, November 1977.
- Smith, D., "Using enhanced spherical images," M.I.T. AI Memo 530, May 1979.
- Sugihara, K., "Range data analysis guided by a junction dictionary," *Artificial Intelligence* 12, 1979.