Some Experiments with the 3-D Hough Shape Transform[1]

Thomas C. Henderson and Wu So Fai

UUCS 83-002

5 August 1983

Department of Computer Science
Salt Lake City, Utah 84112

Abstract

The application of the Hough shape transform to the problem of the identification and localization of objects in 3-space is presented. The rotation-invariant Hough transform is defined which permits the determination of the geometric transformation from the model to the detected object. Given a sample of 3-D surface points, a set of special model points are derived which are used to form the model. Methods are presented for reducing the amount of computation and accumulator space required to perform the analysis.

---

[1]This work was supported in part by the System Development Foundation and NSF grants ECS-83-07483 and MCS-82-21750

Table of Contents

List of Figures

List of Tables

1. Introduction

The Hough shape transform has been proposed and developed by several workers [1, 2, 5]. A 3-D version has also been investigated [3, 4]. A performance analysis has been described by Shapiro [6], and special data structures have been investigated for use as the parameter space accumulators. We review here the method presented in Wu [7].

2. Hough Shape Models

An extention of the 2-D Hough shape transform to handle 3-D surfaces offers an alternative approach to object identification, and furthermore permits the localization of objects in space. That is, the exact transformation can be found which maps the reference object onto the detected object. The classic Hough transform is a method for detecting curves by using the duality between points on a curve and parameters of that curve. For instance, in the case of straight lines, we could parameterize them by their corresponding slopes and intercepts. The parameter space is then quantized, and an accumulator is associated with each point in the parameter space. The accumulator is incremented for each detected point whose associated curve in parameter space crosses that accumulator. Local maxima in the accumulators correspond to collinear points in the image space. The values in the accumulators measure the number of points on the line.

The Hough shape transform is a generalization of the classic Hough transform for handling objects which have no simple analytic forms, but have particular shapes. A reference point is picked. For each boundary point, compute the displacement vector from the boundary point to the reference point. Store the reference point and the displacement vectors. The basic strategy of the Hough shape transform is to compute the possible loci of the reference point given edge point data in an image, and is achieved by associating an accumulator with each point in space, and applying the following algorithm: for all e, a detected edge location, and for all d, a displacement vector, increment the accumulator at (e + d). Possible locations for the reference point are given by the maxima in the accumulator array. Figure 2-1 gives an example of the Hough shape definition. Figure 2-2 shows the corresponding Hough shape detection.

The extended 3-D Hough algorithm works as follows. A 3-D object is represented as a collection of n vertexes. These vertexes in turn are denoted by their (x,y,z) locations.

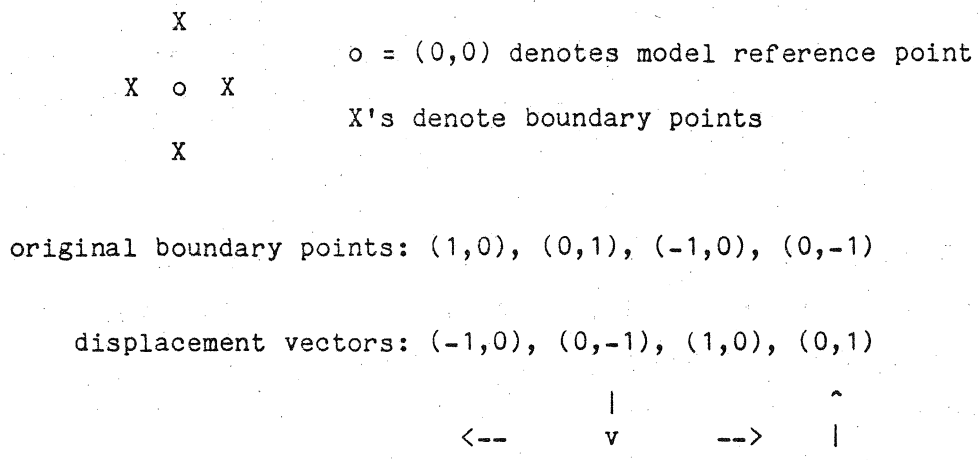


Figure 2-1: Example of Hough shape definition

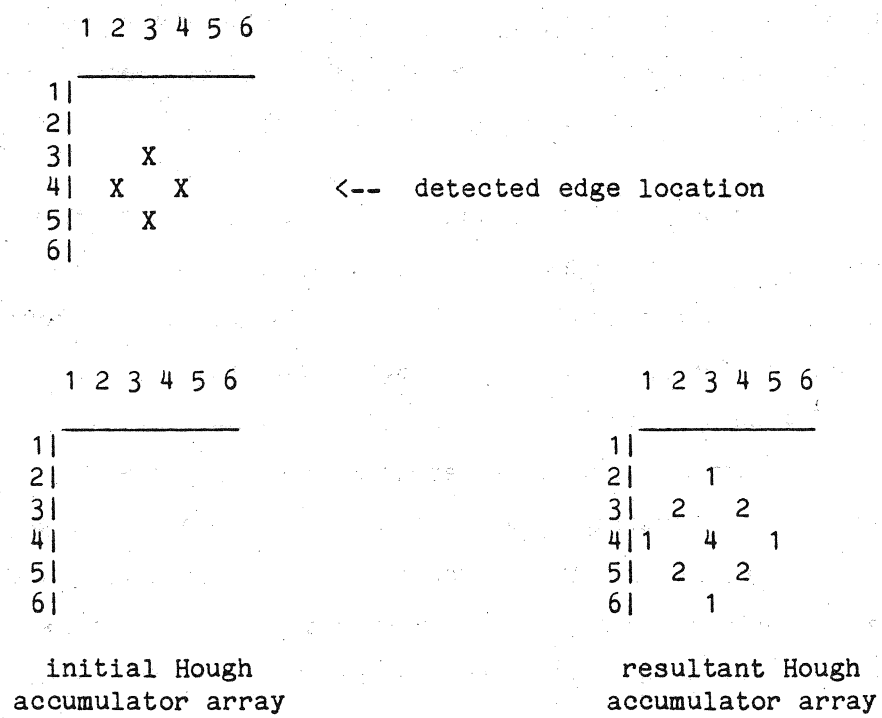


Figure 2-2: Example of Hough shape detection

Call this set of 3-D points P, where $P = \{ (x_i, y_i, z_i) \} i=1,...,n$. Choose some reference point, $P_0 = (x_0, y_0, z_0)$ e.g., the centroid of the set of triples. The object representation is given as a list of displacement vectors from each point in P to the reference point P_0. The model is then a characterization of P as a displacement from each vertex to the reference P_0.

Given the spatial proximity graph representation of a set of points sampled from the surface of a polyhedral object, the points in the graph can be grouped to find the planar regions. The planar faces of the detected object are then intersected to find the vertices of the object. The detection procedure is then to match the set of model vertexes with the detected vertexes as points in the transform space. From matching each point on the surface of an object to matching a small set of points representing the vertexes of the object, we arrive at a much reduced set to be matched.

For each detected vertex, we associate the n displacement vectors to compute the possible loci of the reference point, P_0, in parameter space. Accumulate counts for the possible locations of P_0. The location that has the maximum count in the 3-D space corresponds to the translated position of the reference point P_0 of the object model.

The above algorithm produces a unique maximum for any translated P, and the maximum value is equal to the number of detected object vertices. But if all the points in P are not in the detected object, then the maximum will be less than the number of points in the model object. Moreover, if there are several copies of the object, then there may not be a unique maximum. However, the reference point is always guaranteed to be one of the maxima. We could use the ratio maximum/d, where d is the number of detected points, to judge the likelihood that that maximum location does indeed correspond to P_0.

Since in general, objects are both translated and rotated, a more realistic Hough shape model will be the one which deals with rotation as well as translation. The following section outlines the algorithm to compute rotational invariant 3-D Hough transform.

3. Rotational Invariant 3-D Hough Shape Model

Given a set of points $P=\{(x_i,y_i,x_i)\}$, i=1,n, representing a 3-D object model, choose some reference point P_0, (x_0,y_0,z_0), such that the lengths of all the vectors of $R=\{(dx_i,dy_i,dz_i)\}$, where $dx_i=x_0-x_i$, $dy_i=y_0-y_i$, and $dz_i=z_0-z_i$, are distinct. The model representation is then in terms of the reference point P_0, and R.

Given a set of detected points $D=\{(x_i,y_i,x_i)\}$, i=1,m, use a 3-D array H, to accumulate counts for possible location of P_0 in space. Then the rotation invariant 3-D Hough transform is computed by:

for all p = (x,y,z) **in** D,

  for all r **in** R,

    increment H(sphere with center at (x,y,z) and radius r) by 1.

In actual implementation, the accumulators are defined in terms of intersections of spheres which represent possible loci of rotated and translated P_0's. Intuitively, the rotation invariant 3-D Hough transform is computed by keeping accumulators for points of intersection of the various spheres centered at all the detected points. Then the location in H having the maximum value corresponds to the translated and rotated position of the reference point P_0, of the object model. However, if there exists possible rotational symmetry of the object model, there is no guarantee of a unique maximum in the accumulator array H. In this case, any of the maximum may be chosen.

Recall P_0, the model reference point, is of distinct distance from all vectors in R. Call the possible location of the transformed point P_0', (x_0',y_0',z_0'). With the finding of the possible location of the transformed P_0, construct $R'=\{(dx_i,dy_i,dz_i)\}$, i=1,m, where $dx_i=x_0'-x_i$, $dy_i=y_0'-y_i$, and $dz_i=z_0'-z_i$, and (x_i,y_i,z_i) in D. Matching is done by finding for each r' in R', its counterpart r in R, such that r'=r. Then the detected point in D which gives rise to r' corresponds to the possible transformed location of the model point in P which gives rise to r. The match is reported in terms of the transformation that maps the model points to the detected points.

Section 3 gives the algorithmic implementation of a matcher based on the rotational invariant 3-D Hough shape model. Section 4 details a method to find the intersections of spheres, which is needed to process the accumulators.

4. Shape Recognition

We now give an algorithm based upon the Hough shape model for performing shape recognition.

```
procedure MATCHER(detected_object: a set of 3-D points);
{       /* match the points to all models */
        for m := 1 step 1 until all_models
        {
                /* produce maximum accumulator */
                max_count := MAX_ACCUMULATOR(detected_object,model);

                if OBJECT IDENTIFIED
                        DETERMINE ORIENTATION(detected_object,model);
        }
}
```

```
integer procedure MAX_ACCUMULATOR(detected_object,model);
{
        /* initialize accumulators */
        INITIALIZE_ACCUMULATORS;

        for d := 1 step 1 until all_detected_points
        {
                for a := 1 step 1 until all_accumulators
                {
                        for r := 1 step 1 until all_model_radii
                        {
                                if INTERSECT
                                        then UPDATE_ACCUMULATOR;
                        }
                }
        }

        return(FIND_MAX_ACCUMULATOR);
}
```

The procedure DETERMINE_ORIENTATION, when given the correspondence between the model reference point and the detected reference point, the model points and the detected points, finds the transformation which maps the model points to the detected points. The procedure INITIALIZE_ACCUMULATORS, when given the set of model radii, and

the pair of most distant points in the set of detected points, creates accumulators which represent all the intersections of spheres with radii taken from the set of model radii and centered at these two distant points. The procedure INTERSECT returns true if the sphere, centered at a given detected point with a radius equal to a given model radius, intersects a given accumulator. UPDATE ACCUMULATOR is a procedure that updates information pertaining to an existing accumulator which has been intersected, or creates new accumulators when necessary after intersection of an existing accumulator with a given sphere. FIND MAX ACCUMULATOR is a procedure that returns the maximum of all the accumulators.

5. Intersecting a set of spheres

We break this seemingly involved problem into a set of simpler problems. We approach the intersection problem in the following manner.

Let S_0 be the first sphere with radius r_0, centered at (x_0,y_0,z_0).

Let S_2 be the second sphere with radius r_2, centered at (x_2,y_2,z_2).

Let d be the distance between the centers of the spheres.

We have the following four possible cases:

1. No intersection if $r_0 + r_2 < d$ as shown in Figure 5-1,

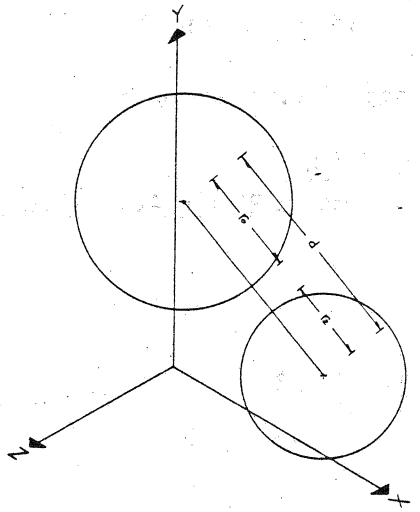


Figure 5-1: No intersection with $r_0 + r_2 < d$

2. Intersection at a single point if

   a. $r_0 + r_2 = d$, as shown in Figure 5-2, or

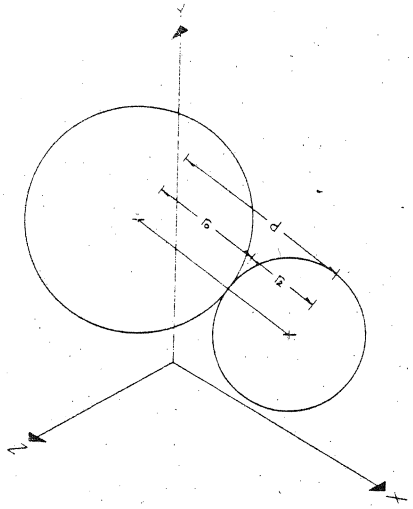


Figure 5-2: Single point intersection with $r_0 + r_2 = d$

b. $r_0 + d = r_2$, as shown in Figure 5-3, or

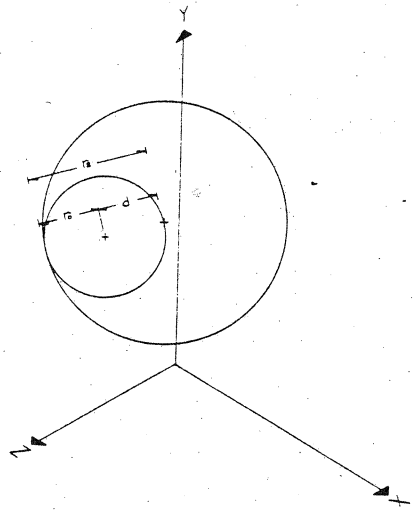


Figure 5-3: Single point intersection with $r_0 + d = r_2$

c. $r_2 + d = r_0$, as shown in Figure 5-4,

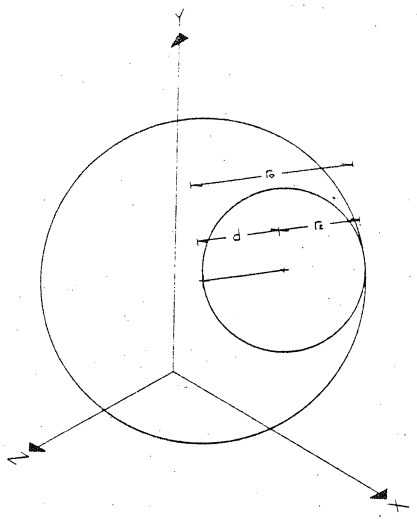


Figure 5-4: Single point intersection with $r_2 + d = r_0$

3. Intersection equal to either one of the spheres if

   a. $(x_0,y_0,z_0) = (x_2,y_2,z_2)$, and

   b. $r_0 = r_2$, as shown in Figure 5-5, and

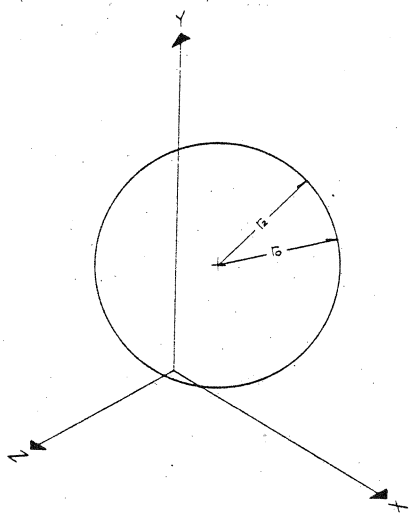


Figure 5-5: Spherical intersection with identical spheres

4. Intersection is a circle in 3-D space if $r_0 + r_2 > d$, as shown in Figure 5-6.

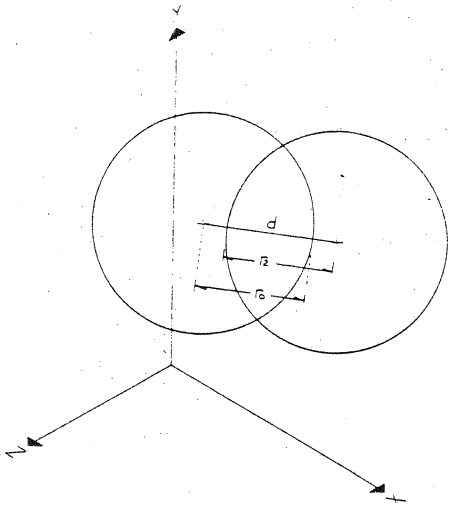


Figure 5-6: 3-D circle intersection with $r_0 + r_2 > d$

Since there are four possible outcomes for intersecting the first two spheres, in order to finish intersecting the entire set of spheres, we have to deal with each outcome separately.

In case 1, we continue by starting the whole intersection procedure again with a distinct pair of spheres from the set.

In case 2, further intersection means testing if the current point of intersection lies on the subsequent sphere.

In case 3, further intersection means picking another sphere from the remaining set, and repeat the process of intersecting two spheres.

The continuation of case 4 is relatively elaborate. We will consider this case in detail.

5.1. Intersection of a 3-D circle and a sphere

Now Figure 5-7 gives a closer view of the region of intersection between two spheres resulting in a 3-D circle.

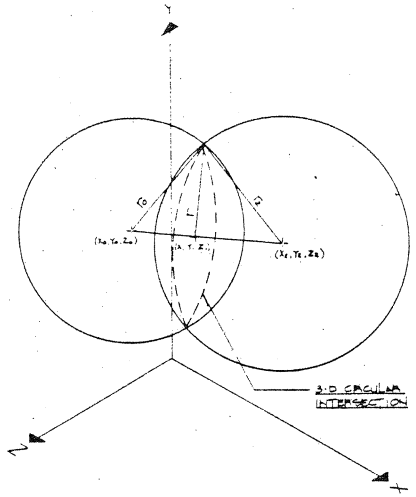


Figure 5-7: 3-D circle intersection with $r_0 + r_2 > d$

Let x be the distance between the center of the first sphere, and the center of the circular intersection.

Let r be the radius of the circular intersection.

We have as in Figure 5-8

$$d = ((x_2-x_0)^2 + (y_2-y_0)^2 + (z_2-z_0)^2)^{1/2}.$$

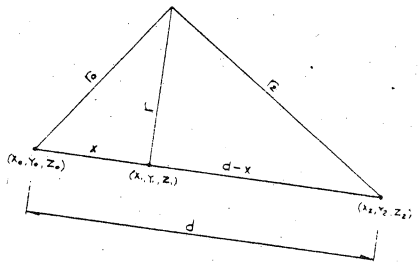


Figure 5-8: Details of 3-D circular intersection

Using the Pythagorian Theorem

$$r^2 = r_0^2 - x^2, \text{ and}$$

$$r^2 = r_2^2 - (d-x)^2.$$

After some rearrangements, finally

$$x = (r_0^2 - r_2^2 + d^2) / (2 * d), \text{ and}$$

$$r = (r_0^2 - x^2)^{1/2}.$$

As a result, the coordinates, (x_1,y_1,z_1), of the center of the 3-D circular intersection are:

$$x_1 = (x_2-x_0) * x / d + x_0,$$

$$y_1 = (y_2-y_0) * y / d + y_0, \text{ and}$$

$$z_1 = (z_2-z_0) * z / d + z_0.$$

The strategy to find the intersection between a 3-D circle and a sphere is as follows:

1. Transform the 3-D circle to the x-y plane, so that it is centered at (0,0,0). Call this transformation M.

2. Apply the same transformation M to the sphere to be intersected.

3. Find the intersection of the transformed sphere with the x-y plane. Call the intersection region U. U could be empty, a single point, or a circle.

4. Intersect the transformed 3-D circle with U. At this stage we are dealing with a simpler problem, intersecting one circle with a single point or another circle. The solution may be empty, a single point, a circle, or two points. Call the solution set W.

5. Apply the inverse transformation of M to W.

The overall intersection between the 3-D circle and the sphere will simply be the transformed W under the inverse of the original transformation M.

The following pages outline the steps to derive the transformation M which maps the 3-D circle onto the x-y plane and its center to the origin (0,0,0).

Figure 5-9 shows the original position of the 3-D circle. Since the plane containing the 3-D circular intersection is orthogonal to P_1P_2, M is equivalent to the transformation that maps P_1 to the origin and P_1P_2 on the negative z-axis, as shown in Figure 5-10.

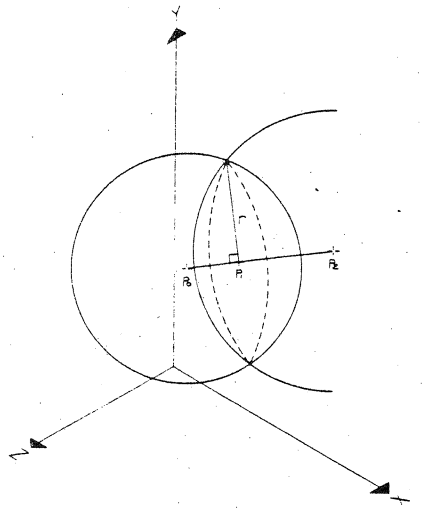


Figure 5-9: 3-D circular intersection

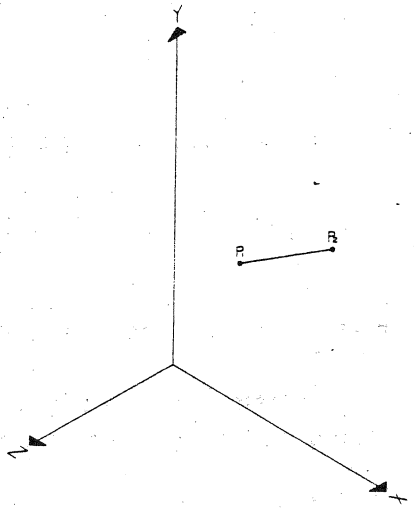


Figure 5-10: Symbolic representation of 3-D circular intersection

We formulate the transformation M in three steps.

1. Translate P_1 to the origin.
2. Rotate about the y-axis, so that P_1P_2 lies in the y-z plane.
3. Rotate about the x-axis, so that P_1P_2 lies on the negative z-axis.

Step 1: Translate P_1 to the origin

$$T(-x_1, -y_1, -z_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix}$$

Applying T to P_1, P_2 gives:

$$P_1' = P_1 \cdot T(-x_1, -y_1, -z_1) = [\ 0\ 0\ 0\ 1\ ],$$

$$P_2' = P_2 \cdot T(-x_1, -y_1, -z_1) = [\ x_2-x_1\ \ y_2-y_1\ \ z_2-z_1\ \ 1\ ].$$

Step 2: Rotate about the y-axis.

Figure 5-11 shows P_1P_2 after step 1, along with the projection of P_1P_2 onto the x-z plane. The rotation, R_y, is by the positive angle α, for which

$$\cos\ \alpha, = \frac{-(z_2-z_1)}{d_1}, \text{ and}$$

$$\sin\ \alpha, = \frac{x_2-x_1}{d_1},$$

where

$$d_1 = \left((z_2-z_1)^2 + (x_2-x_1)^2 \right)^{1/2}.$$

$$P_2'' = P_2' \cdot \alpha, = \begin{bmatrix} 0 & y_2-y_1 & -\frac{(x_2-x_1)^2}{d_1} - \frac{(z_2-z_1)^2}{d_1} & 1 \end{bmatrix}$$

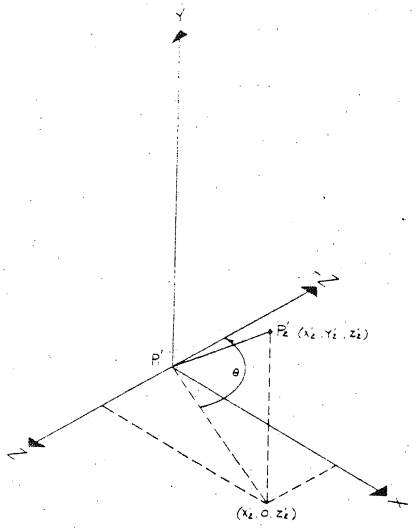


Figure 5-11: Rotation about the y-axis

Step 3: Rotate about the x-axis.

Figure 5-12 shows P_1P_2 after step 2. The rotation R is by the negative angle θ, for which

$$\cos(-\theta) = \cos\theta = \frac{-z_2''}{d-x}, \text{ and}$$

$$\sin(-\theta) = -\sin\theta = \frac{-y_2''}{d-x}.$$

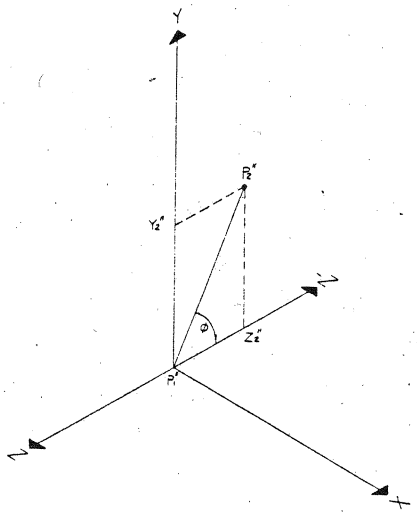


Figure 5-12: Rotation about x-axis

The result of the rotation in step 3 is

$$P_2''' = P_2'' \cdot R_x(\theta) = P_2' \cdot R_y(\alpha) \cdot R_x(-\theta)$$

$$= P_2 \cdot T \cdot R_y(\alpha) \cdot R_x(-\theta)$$

$$= \begin{vmatrix} 0 & 0 & -(d-x) & 1 \end{vmatrix} .$$

P_1P_2 now coincides with the negative z-axis.

The composite matrix M :

$$T(-x, -y, -z) \cdot R_y(\alpha) \cdot R_x(-\theta)$$

is the required transformation.

The inverse of matrix M is simply :

$$R_x(\theta) \cdot R_y(-\alpha) \cdot T(x, y, z)$$

Once the transformation M is obtained, we apply it to the sphere to be intersected. Let S_3 be the sphere which is centered at P_3, (x_3, y_3, z_3), with radius r_3.
Applying M to P_3 gives:

$$P_3' = P_3 \cdot M = \begin{vmatrix} x_3' & y_3' & z_3' \end{vmatrix} .$$

The equation of the transformed sphere becomes:

$$(x-x_3')^2 + (y-y_3')^2 + (z-z_3')^2 = r_3^2 .$$

Set z to 0 to find the intersection of the transformed sphere with the

x-y plane. We have

$$(x-x_3')^2 + (y-y_3')^2 + (-z_3')^2 = r_3^2 .$$

Expand and collect like terms:

$$x^2+y^2+(-2*x_3')*x+(-2*y_3')y+(x_3'^2+y_3'^2+z_3'^2-r_3^2) = 0 .$$

Recall the general equation of a circle:

$$x^2 + y^2 + a*x + b*y + c = 0$$

Such a circle is centered at (-a/2, -b/2),

with radius = 1/2 * ($a^2 + b^2 - 4*c$)$^{1/2}$.

Comparing the equation of the intersection between the transformed sphere and the x-y plane, and the general equation of a circle, we see that the intersection can be represented by a circle on the x-y plane centered at $(x_3', y_3', 0)$ with radius $r_4 = 1/2*(r_3^2 - z_3'^2)^{1/2}$.

Analyzing r_4, we observe the following about the intersection between the x-y plane and the transformed sphere:

1. Intersection is empty if r_4 is complex or $r_3^2 < z_3'^2$.
2. Intersection is a single point if r_4 is zero or $r_3^2 = z_3'^2$.
3. Intersection is a circle if r_4 is real or $r_3^2 > z_3'^2$.

So far, we have reduced the order of complexity of intersecting a 3-D circle with a sphere to the order of complexity of intersecting two circles. This finishes our discussion on intersecting a 3-D circle with a sphere.

Thus, we have examined all possible outcomes of the intersection between spheres by looking at the intersection between three spheres. Any further intersection between spheres will be an instantiation of one of the possible cases discussed.

6. Case Studies

6.1. Hough Shape Model

To directly apply the Hough shape transform to a model of a 3-D object in terms of its surface points would require a 3-D accumulator array, as discussed in section 2.2, which could easily exhaust the memory of a machine. Therefore, it is necessary to compress the size of the model representation. We recommend two approaches, illustrated by two examples, which drastically reduce the set of accumulators.

The first approach works well with polyhedral models. We reduce the size of the model representation by using the vertexes of the polyhedron to represent it.

The second approach is suitable for objects of irregular shape. This approach consists of choosing four control points which are recoverable from the type of data available, and determining the geometric transformation from the model to the detected control points. If the control points are not directly distinguishable, e.g., they are all vertexes of the same order, then the Hough shape transform can be used to label them; otherwise, the transformation can be determined directly.

Cube

We would like to use a cube as the model object to illustrate the first approach. The model is given in terms of :

reference point: (1.5, 1.3, 1.25),

and radii of magnitude :

4.09,
3.43,
3.38,
3.28,
2.54,
2.40,
2.33,
0.65.

Figures 6-1 and 6-2 show the detected surface points of the cube and the corresponding spatial proximity graph obtained, respectively, as displayed on the PS300.

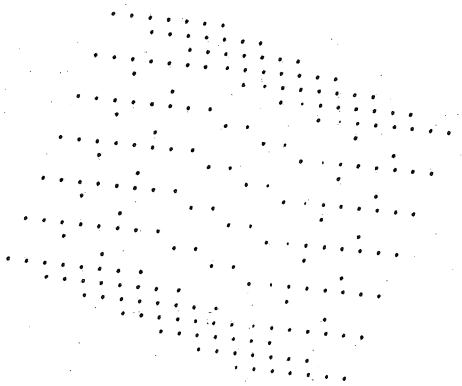


Figure 6-1: Surface points on the cube

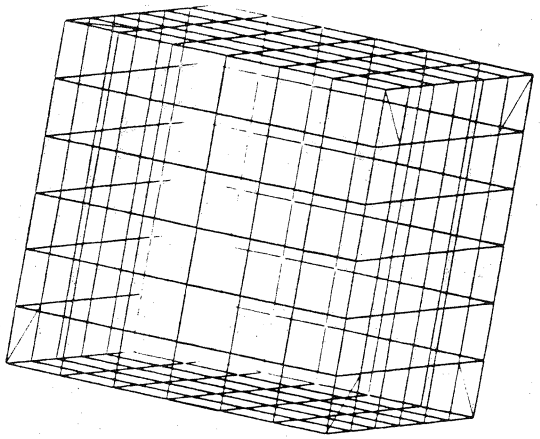


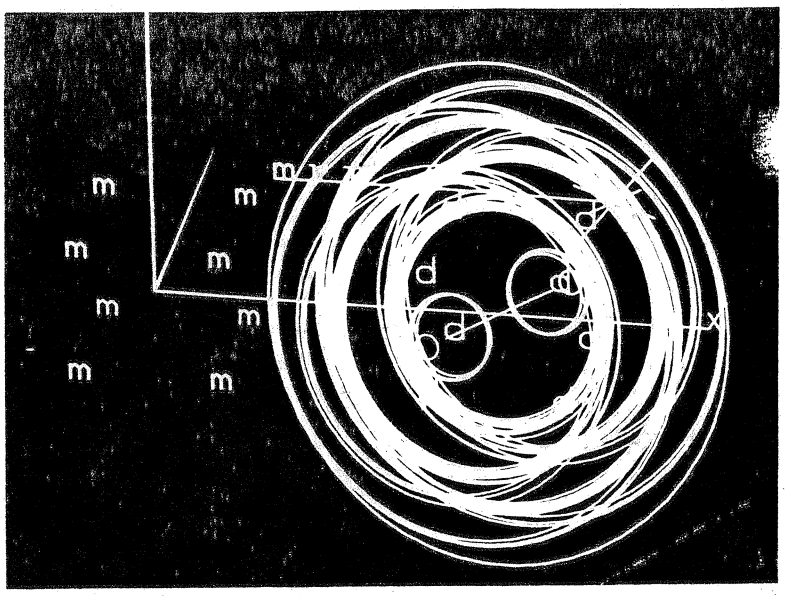
Figure 6-2: SPG for a cube with 4 nearest neighbors

After performing grouping on the spatial proximity graph to find the faces of the cube and intersecting the different faces, we found the vertexes given in Table 6-1.

x	y	z
4.0	-1.0	1.0
4.0	1.0	1.0
6.0	1.0	1.0
6.0	-1.0	1.0
4.0	1.0	-1.0
4.0	1.0	-1.0
6.0	1.0	-1.0
6.0	-1.0	-1.0

Table 6-1: Vertexes of the Cube

Figure 6-3 shows the initial set of accumulators when detecting the cube.



m's denote model vertexes

mrf denotes model reference point

d's denote detected vertexes

* denotes detected reference point

Figure 6-3: Initial circular accumulators when detecting cube

Since there are rotational symmetries in the cube with respect to the chosen model reference point, we actually detected eighteen possible locations for the model reference point. Here is the one we picked, and along with it is the corresponding transformation that maps the model cube to the detected cube:

detected reference point: (6.5, 1.3, 1.25)

transformation matrix:

$$\begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 5. & 0. & 0. & 1. \end{bmatrix}$$

In this case the detected cube is an instantiation of the model cube which has simply been translated by 5 units in the positive direction along the x-axis.

Renault Piece

We would like to use the industrial object shown below as the model object to illustrate the second approach. There are about 2000 point samples. We call this piece the Renault piece. Figure 6-4 shows the detected surface points of the Renault piece. (This data was digitized on a laser range finder developed at INRIA, Rocquencourt, France by F. Germane.) Figure 6-5 is the spatial proximity graph obtained as displayed on the Evans and Sutherland PS300.

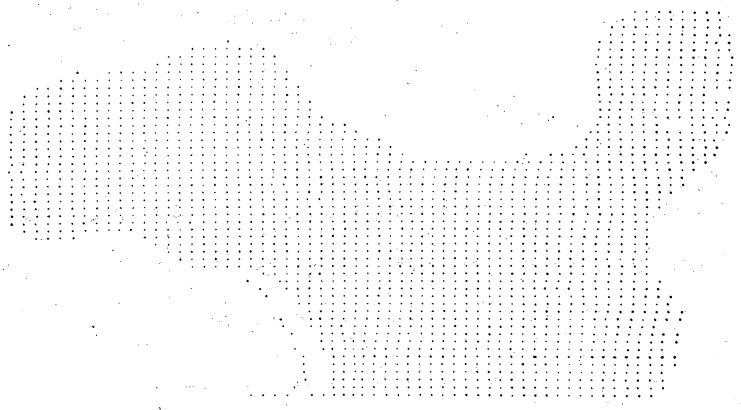


Figure 6-4: Detected surface points of the Renault piece

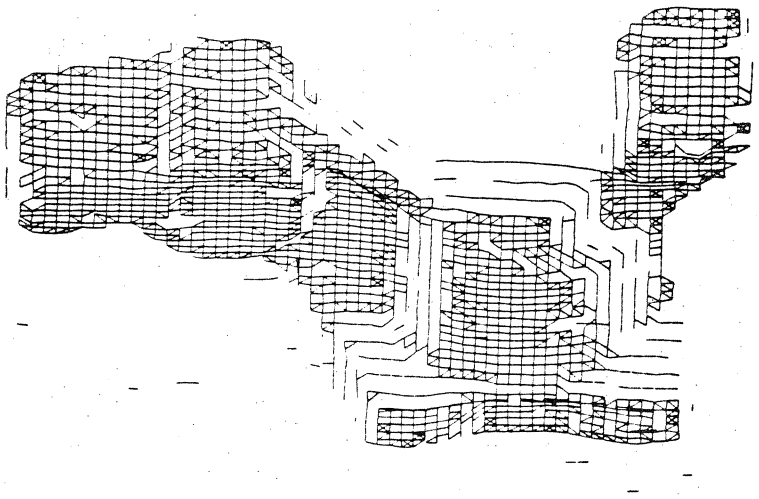


Figure 6-5: SPG of the Renault piece with 4 nearest neighbors

The four control points chosen as model points could be derived from the spatial proximity graph of the model. The model is given in terms of:

model reference point (10.344000, 19.333334, 25.122999)

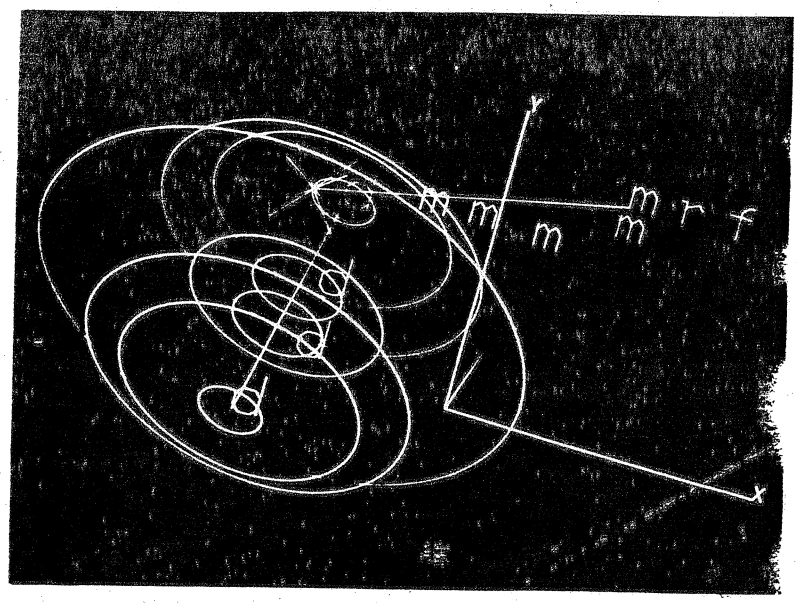
and the radii of magnitude:

29.900755,
18.041958,
13.079391,
3.379884.

The detected control points are:

(-21.000000,-7.246000,2.706000),
(-17.600000, -3.183000, 16.096001),
(-17.200001, 3.755000, 16.305000),
(-18.000000, 10.844000, 25.872999).

Figure 6-6 shows the initial set of accumulators using the four control points for detection.



m's denote model vertexes

mrf denotes model reference point

d's denote detected vertexes

* denotes detected reference point

Figure 6-6: Initial circular accumulators when detecting Renault piece

Here is the transformation that the maps the model object to the detected object:

```
| 0.000000   1.000000  -0.000000  0.000000 |
| -1.000000  -0.000002  0.000002  0.000000 |
| -0.000000  -0.000000  1.000001  0.000000 |
| -2.000000  1.000031   0.999962  1.000000 |
```

In this case, the detected object is an instantiation of the model object after it has been translated by (1.0, 2.0, 1.0) and rotated by 90 degrees with respect to the z-axis.

7. Conclusions and Future Research

We picked the Hough shape model for its simplicity. We would like to suggest a closer examination of a possible improvement to our current vertex-based implementation. When dealing with polyhedral models, an alternative to using vertexes to represent the polyhedral faces of a 3-D object is to map each face of the 3-D object into a 4-D

transform space, and model these points considered as an object. The 4-D points are the coefficients of the planes that contain the faces of the polyhedron. Since the number of faces is usually small, we can keep the storage needed for accumulators under control. However, the difficulty of such an approach lies in interpreting the geometric meaning and geometric correspondence of the detected reference point found to the model reference point. An undesirable consequence of lacking a clear understanding of the geometric meaning and geometric correspondence is that the orientation of the recognized object will remain unknown.

Finally, methods to limit the number of spheres to be intersected must also be found and applied to the appropriate situations. For example, if two vertexes are known to belong to the same object, then it is more reasonable to use those vertexes to form the initial set of accumulators. A very distant pair of vertexes known to belong to the same object would be even better, since the number of intersections would most probably be less.

References

[1] Ballard, D.H.
Generalizing the Hough Transform to Detect Arbitrary Shapes.
Pattern Recognition 13(2):111-122, 1981.

[2] Davis, L.S. and S. Yam.
A Generalized Hough-like Transformation for Shape Recognition.
Technical Report TR-134, U. of Texas, FEB, 1980.

[3] Henderson, T.C. and N. Keskes.
L'analyse des Objects Tridimensionnels.
In Proc. 3rd Conf. on Patt. Recog. and Art. Intell., pages 277-287. Nancy, France, September, 1981.

[4] Henderson, T.C. and A. Mitiche.
Modeling 3-D Structure.
In B. Radig (editor), Modelle und Strukturen, pages 112-116. Springer-Verlag, Berlin, October, 1981.

[5] Merlin, P.M. and D.J. Farber.
A Parallel Mechanism for Detecting Curves in Pictures.
IEEE Transactions on Computers C-24:96-98, 1975.

[6] Shapiro, S.D.
Use of the Hough Transform for Image Data Compression.
Pattern Recognition 12:333-337, 1980.

[7] Wu So Fai.
A Multi-sensor Integration and Data Acquisition System.
Master's thesis, University of Utah, June, 1983.