BROWN UNIVERSITY
Department of Computer Science
Master's Project

CS-95-M17

"Robotic Object Recognition: Utilizing Multiple Views to Recognize Partially Occluded Objects"

by

Neil A. Jacobson

Robotic Object Recognition: Utilizing Multiple Views to Recognize Partially Occluded Objects

Neil A. Jacobson

Department of Computer Science
Brown University

June 1994

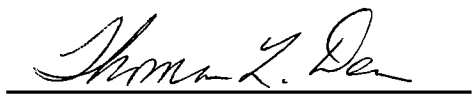
Robotic Object Recognition: Utilizing Multiple Views to Recognize Partially Occluded Objects

Neil A. Jacobson

Department of Computer Science
Brown University

Submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science at Brown University

June 1994



Professor Thomas L. Dean
Advisor

Abstract

To perform real world tasks, a robot needs to know detailed information about its environment. This information can be provided to the robot a priori, or the robot can determine this information by using its sensors and interacting with its environment. Since all but the most constrained environments are dynamic, the latter would be a more practical choice. One useful way of gathering information about the environment is by performing object recognition. This however, can be complicated by the existence of occluding objects and the need to make multiple observations to disambiguate among possible matches. This paper describes a method of robotic 2D object recognition in this type of environment. The robot is equipped with a laser light striper that returns a relatively accurate 2D (width and depth) view of its environment. Initially, the data retrieved from the laser light striper is passed to a curve matching algorithm which matches the data to each object in its database. The resulting hypothesis set is passed to sensing strategy algorithm which determines where the surface normals and distances to the hypothesized objects' poses differ, and returns the next place to look to gather the most information. The subsequent observation is then used to update the hypothesis set. This process continues until enough information is gathered to disambiguate among all possible matches, or until no further information can be gained by making additional observations.

Contents

1 Introduction

Object recognition is a useful way for a robot to learn about its surroundings. Most environments, however, contain objects of similar shape, thus multiple views of an object would be needed to disambiguate among the possible matches. Additionally, occlusion is very likely to occur in all but the most constrained environments. This paper addresses the following problem: given an unknown object, possibly occluded, and a database of known objects, determine whether the object is in the database, and if it is, determine its pose.

The solution proposed in the paper makes use of a curve matching algorithm that can match partially occluded objects [1], a sensing strategy algorithm that determines where the surface normals and distances to surfaces differ in a set of possible poses [2], and an algorithm that maintains a set of hypotheses given the sensor readings. This system was implemented on an autonomous robot equipped with a laser light striper. The laser light striper returns 2D (width and depth) information about it surroundings. In conjunction with the robot's dead reckoning, the robot is able to determine the (x, y) coordinates of objects within some error, which is a function of the dead reckoning and sensor error. The hardware is briefly described in section 2, the overall object recognition algorithm is described in section 3, the curve matching algorithm in section 3.1, the algorithm for updating the hypotheses in section 3.2, and the sensing strategy algorithm is explained in section 3.3. Section 4 contains the experimental results, and section 5 contains the summary.

Related Work

There has been a large amount of research completed in the area of matching partially occluded objects, including [1],[3]-[6]. In [1], shape signatures are used to match partially occluded objects. Each object and observed curve are converted into shape signatures and compared to come up with a set of candidate matches. The least-squares distance is then calculated for each candidate match, and the best match and its corresponding transformation is determined. This algorithm solves the matching problem: given any two curves, find the largest matching subcurve which appears in both curves.

In [3] and [4], curve matching is performed by computing the least squares fit of the observed curves to a database of previously known objects. The matching must be performed on proper subcurves of the database objects only. Therefore, to match partially occluded objects, the observed curves are split up along breakpoints, which are areas of the curve where the tangent changes rapidly. These sharp changes usually occur where objects occlude each other. These subcurves are then matched to the objects in the database. It's possible, however, that a breakpoint might not exists at a point where two or more objects are in contact. In such a case, the observed curve could not be split up into proper subcurves of the database objects, and the curve matching could fail.

In [5], a Landmark-based shape recognition is used. A landmark is a point of interest of an object that has an important shape attribute. Given a scene of occluded objects, the landmarks are extracted and matched to the landmarks of the database objects. A measure called sphericity is used to determine the similarity between landmarks and to determine the best match.

Recognition based on tracing feature points is described in [6]. In this method, objects are represented by linked lists of nodes composed of a consecutive pairs of feature points (corners) along with their distances and angles from each other. Initially, a set of candidate matches is extracted from the inputted scene by comparing pairs of feature points from the database object (nodes in the linked list) to pairs of feature points in the scene. If the distance and angle between the scene points lie within some error bounds of the feature point pair from the database object, then a candidate match is made. Each candidate match is then verified by tracing the remaining feature points. That is, given the candidate match node in the database object's linked list, iterate through the remaining list nodes while determining where the corresponding feature points should lie in the inputted scene, and test whether they exist and if their distance and angle are with some given error bounds.

Some research relating to planning multiple views can be found in [2],[7]-[9]. Given a set of sensory data points on an object and a set of possible poses, [2] disambiguates among the poses by determining where the surface normals and distances to the surfaces of the poses differ, and getting sensory data at these points. The sensory data is compared to the calculated values of the

transformed objects, and the correct pose is determined.

In [7], a Bayes Net and a maximum expected decision rule were used to decide where to position a camera over a scene. Their sensors were composed of a low resolution camera that provided a peripheral view and a small high resolution camera that could be positioned anywhere within the peripheral view, but neither camera could view the whole scene at one time.

The goal of [8] was to describe a random arrangement of unknown objects in a scene. They used a laser light striper and scanned the scene from above and the sides. The scene was rotated so that multiple views could be taken. Occluded regions were approximated by polygons, and based on the height information of the border of the occluded regions, the next view was determined.

A Vision Algorithm Compiler was used in [9] to analyze predicted object appearances and create an aspect resolution tree off-line. The resolution tree was composed of three types of nodes, congruent-set, move, and resolved nodes. Congruent-set nodes contained the possible poses of the object, move nodes contained positions to move to, and resolution nodes contained the correct object pose. The resolution tree was traversed based upon the observed aspects, and sensor moves were made until a resolution node (a leaf node) was reached.

2 Hardware Description

An autonomous robot was used for this experiment which had an on-board 486DX2 processor. Its sensors included infareds, sonars, a laser light striper, and dead reckoning. The dead reckoning has been calculated to be accurate within .015 x total distance traveled. The infareds and sonars were used for obstacle avoidance, and the laser light striper was used for object recognition. The laser light striper has an effective viewing angle of 45 degrees and distance over 3 meters, however only a distance of up to 1.5 meters was accurate enough to be used for the object recognition. The accuracy of the laser light striper within the limited area has been measured at ±3 cm. The laser light striper is composed of a laser emitter, which emits a horizontal plane of light, and a 512x256 resolution camera angled downward towards the laser. When the laser light hits objects at different distances, the light appears at different pixel heights in the camera image. A look up table is used to determine the distance of the object given the height of the brightest pixel (see figure 1). By combining the laser light striper information with the base's dead reckoning position, the (x,y) coordinates of object surfaces can be determined.

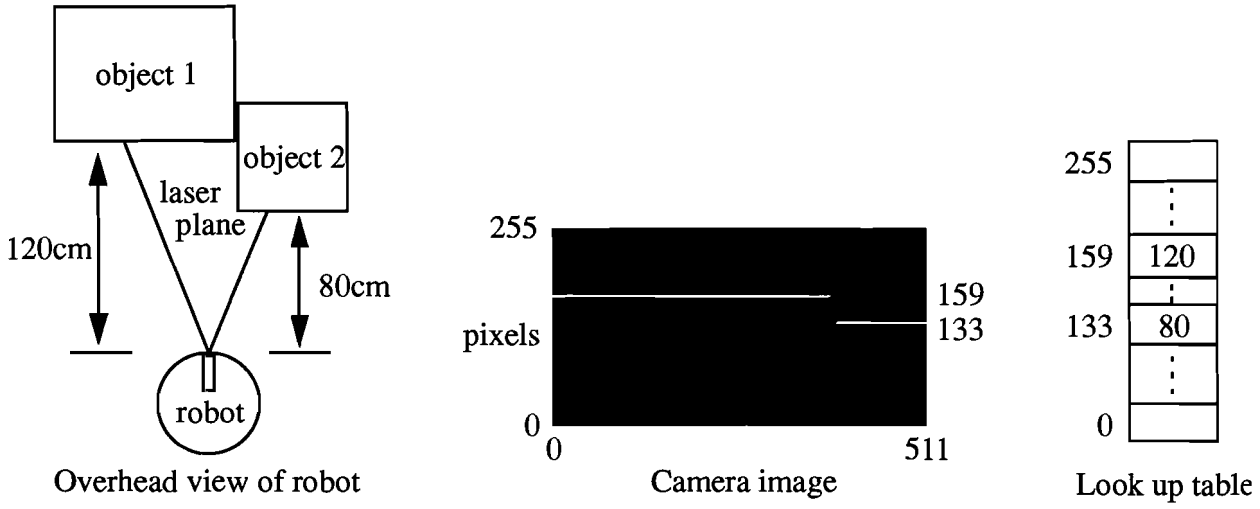


Figure 1: Calculating distances with the laser light striper.

3 Algorithm Descriptions

Initially the robot is supplied with a database of known objects. Each object is stored as a shape signature and a list of evenly spaced points (1 per cm) defining its curve. Both are used in the matching process. The shape signature is used to find a set of candidate matches, and a least squares fit method is used on the point list to find the best match and its transformation. For the purpose of this project, the algorithm discussed in [1] was modified to return a set of matches instead of a single best match. Section 3.1 describes the matching algorithm in greater detail.

The sensing strategy algorithm is used differently than described in [2]. In [2], the set of possible poses was disambiguated by determining places where their surface normals or the distances to their surfaces differ and taking sensor readings at those locations. The sensor readings were then compared to the expected values for each pose, and the inconsistent poses were eliminated. This process was continued until a single pose remained. If this algorithm were used in an environment with occluding objects, false positives and negatives could result. Therefore, for this project, the sensing strategy algorithm is used only to find the places were the poses differ, and does not use the sensor readings to disambiguate among the possible poses. Instead, a module called the Evidence Manager uses the sensor information to maintain a set of hypotheses. The hypothesis set contains a list of possible objects and their poses along with a measure of how well it fits to the viewed object. Section 3.2 describes the Evidence Manager, and the sensing strategy algorithm is described in section 3.3.

The initial position of the robot is (0,0) with a heading of 0 degrees, and all coordinates returned by the object recognition system are with respect to this coordinate system.

The general flow of control of the object recognition system is as follows:

1. Get sensor data from laser light striper
2. Convert sensor data into curves in the robot coordinate system
   (the sensor data is smoothed and the curves' shape signatures are calculated)
3. For each object in the database do
4. Match sensor curves to the database object

5. Update all hypotheses for this object to take the new data into account
6. Test ending conditions:
   If number of hypotheses == 1, return hypothesis's object and pose as a "match"
   If number of hypotheses == 0, return "object not in database"
   If one hypothesis's measure is an order of magnitude greater than all others,
   return that hypothesis's object and pose as a "match"
7. If any new hypotheses were added or deleted in step 5, compute new sensing strategy
8. Test ending conditions:
   If no more information gain is possible (no sensing positions returned),
   return all hypotheses as "ambiguous match"
8. Move the robot to the next position which has the most potential for information gain (the position where the greatest number of surface normals or distances differ)
9. Goto 1

3.1 Curve Matching

Before any curve matching can be performed the curves must be smoothed to remove any sensor noise and their shape signatures must be calculated. When the curves are created, either by reading them in from disk into the database, or by reading data from the laser light striper, this is performed. The curve matcher takes as input these curves and doesn't need to perform any curve smoothing or shape signature calculations. Section 3.1.1 describes the curve smoothing algorithm from [3], section 3.1.2 describes how the shape signatures are calculated, and 3.1.3 describes the matching algorithm.

3.1.1 Curve Smoothing

The curve smoother is used to help filter out any sensor noise that might exist. It calculates a polygonal approximation of the noisy curve by finding the shortest path lying within the curve's epsilon neighborhood. Since the basic smoothing algorithm in [3] only applies to curves monotonic in at least one direction, the first step is to break the noisy curve into maximal monotonic sections. In this implementation, the noisy curves were split up into segments monotonic in the x-axis and y-axis directions.

The next step in the smoothing process is to define the epsilon neighborhood. In [3], the epsilon

neighborhood is define as follows: Let y = c(x), $x \in [a, b]$ be the noisy curve with endpoints a and b, and let c1(x) = c(x) - ε and c2(x) = c(x) + ε be the epsilon neighborhood boundaries. This, however, can lead to inconsistent smoothing results since this epsilon neighborhood depends on the curve slope. Figure 2 shows an example of this on a x-axis monotonic curve. Therefore, for this project, it was decided to calculate the true epsilon neighborhood, also shown in figure 2. Doing so is computationally more expensive, however the smoothing algorithm still runs in linear time, and the results are improved.

The final step of the smoothing algorithm is to calculate the shortest distance within the epsilon neighborhood, which is fully explained in [3]. An epsilon value of 1.5cm was selected for this project based on sample tests.

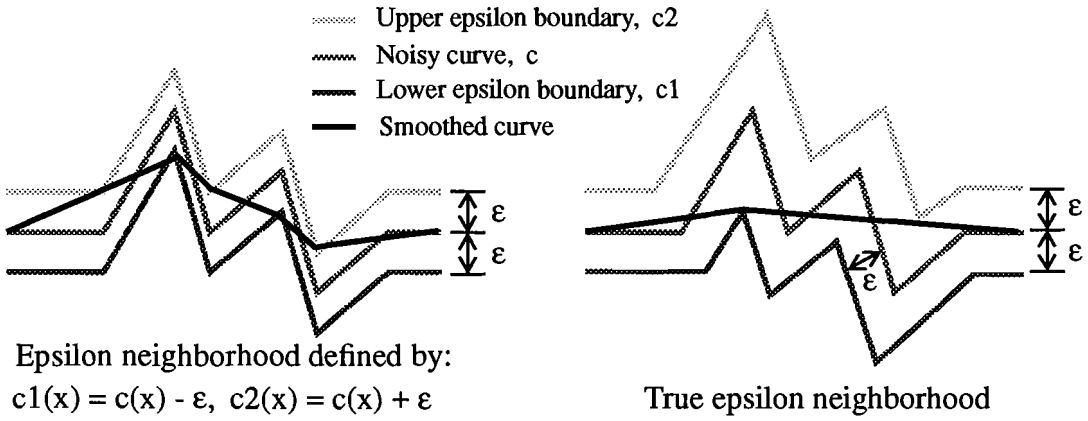


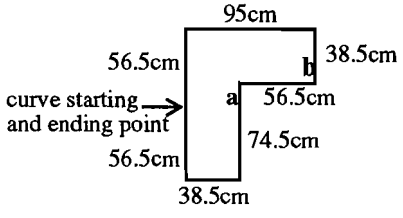
Figure 2: Effects of different epsilon neighborhood definitions on curve smoothing.

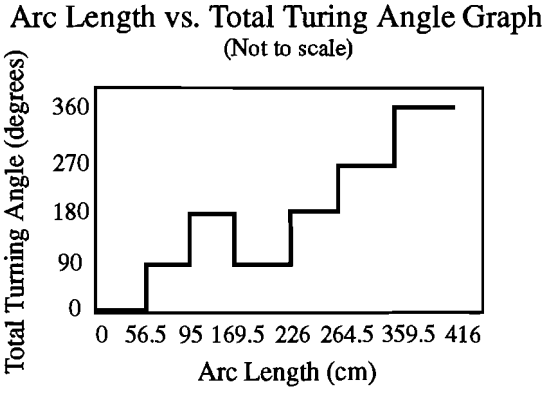
3.1.2 Calculating Shape Signatures

The shape signatures described here are strings of real numbers that represent curves. The shape signatures are translationally and rotationally invariant, and stable in the sense that small changes in the curves have small effects on the shape signatures [1]. The first step in calculating the shape signature is creating an arc length vs. total turing angle graph. Figure 3 shows this type of graph for an L-shaped object that was used in testing.

Denote the arc length vs. total turing angle graph by $\Theta(s)$. To calculate the shape signature, $\Theta(s)$ is sample at equally spaced points by arc length, and at every such point s_i $(i = 1, \ldots, n)$, the difference is calculated: $\Delta\Theta(s_i) = \Theta(s_i + \Delta s) - \Theta(s_i)$. Actually, to make the method more robust the averaged difference is used to calculate the shape signature ϕ: ϕ_i = Average $\Delta\Theta(s_i) = \frac{1}{k}\sum_{j=0}^{k-1} \Delta\Theta(s_i + j\delta)$.

For this project, the value for n was selected to be (curve's total arc length) / Δs, and the values, $k = 5$, $\delta = 1/k$ x (average length of polygonal approximation edge = 16cm), and $\Delta s = \delta$ were empirically determined to work well with polygonal objects, which were used for testing. Figure 3 also contains a portion of the shape signature for the L-shaped object.





s[80] = 0
s[81] = -0.314159
s[82] = -0.314159
s[83] = -0.314159
s[84] = -0.314159
s[85] = -0.314159
s[86] = 0
s[87] = 0
s[88] = 0
s[89] = 0
s[90] = 0
s[91] = 0
s[92] = 0
s[93] = 0
s[94] = 0
s[95] = 0
s[96] = 0
s[97] = 0
s[98] = 0
s[99] = 0.314159
s[100] = 0.314159
s[101] = 0.314159
s[102] = 0.314159
s[103] = 0.314159
s[104] = 0

This shape signature portion approximately corresponds t the L-shaped object's curve from points **a** to **b**.
The values are in radians.

Figure 3: Sample shape signature calculation.

3.1.3 Curve Matching Using Shape Signatures

Curve matching is performed is two main steps. First, a set of candidate matches are found by comparing the shape signatures of two curves, and second, the candidate matches are verified by

calculating the least squares fit on the real curve points corresponding to the shape signature match.

Since the shape signature strings may be noisy, a standard string matching algorithm can't be used to find matches. Instead, we need an algorithm that treats elements of a string as equivalent if their difference is at most ε. In [1], these differences are called ε-coincidences. Assume two strings have a long matching substring, starting at position *i* for the first string and position *j* for the second. If the first string is shifted *j-i* positions relative to the second, then many ε-coincidences would be expected. Therefore, the first step in finding candidate matches is to find shifts with a high number of ε-coincidences. A simple algorithm for doing this is explained in [1]. The next step is to take the *K* best shifts, align each pair of shape signature strings accordingly, and compare their elements looking for long substrings where the difference of the aligned elements is at most ε. If the length of this substring surpasses some minimum threshold, then the candidate match is passed to the next step, verification. For this project, the threshold for the minimum substring length was set at 3.

The verification of the candidate matches are performed on the actual curves rather than their less robust shape signatures. The first step is to calculate the actual curve segments given the starting and ending points of the matching shape signatures. Once this is performed, the least squares fit method is used to calculate the curves' fit as well as to determine the transformation of the first curve onto the second. This method from both [1] and [3] is described below:

> Let the first curve be denoted by C, and the second by C' (Note: C corresponds to the matching subcurve of the database object and C' to the sensor input), and let them be represented by a sequence of evenly spaced (by arclength) points denoted $(\boldsymbol{u}_j)|_{j=1}^{n}$ and $(\boldsymbol{v}_j)|_{j=1}^{n}$ respectively. Matching then amounts to finding an Euclidean transformation E which minimizes the least-squares distance between the sequences $(E\boldsymbol{u}_j)|_{j=1}^{n}$ and $(\boldsymbol{v}_j)|_{j=1}^{n}$: $\Delta = \min_{E} \sum_{j=1}^{n} |E\boldsymbol{u}_j - \boldsymbol{v}_j|^2$. To simplify the calculation, first translate C so that $\sum_{j=1}^{n} \boldsymbol{u}_j = 0$. Next write E as $E\boldsymbol{u} = R_\theta \boldsymbol{u} + \boldsymbol{a}$, R_θ denoting a counterclockwise rotation by θ and $\boldsymbol{a}$ a translation. In such a case, as it is shown in [3], the best match is obtained when, $\boldsymbol{a} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{v}_j$,

object's interior. Since there's error associated with the positioning of the observed curves and laser rays, points simply lying within an object or rays intersecting the interior of an object can't eliminate the possibility that the pose is still consistent. Therefore, the maximum error between each observed curve and the object being tested is calculated. Then an error circle is calculated for each point tested. This error circle is centered at the point's location and has a radius of the maximum error value. If this error circle lies completely within the object, then the hypothesis can be eliminated. This same method is used with the laser rays. Several points are sampled from each section of the ray that lies within the object and are tested as mentioned above.

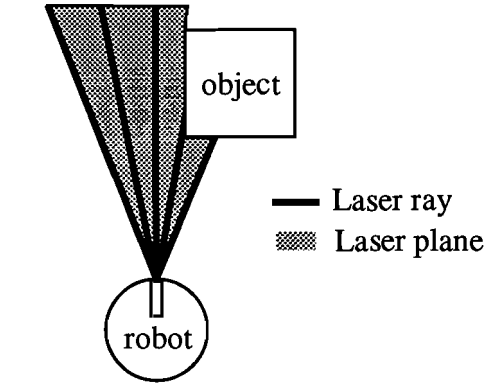


Figure 5: Laser plane and rays.

3.1.2 Planning Multiple Observations

After the sensor data is matched to the database objects and the hypotheses updated, the next viewing position is determined. A modified version of the sensing strategy algorithm from [2] is used to find the next best position to look. This algorithm works by determining where the surface normals and distances to the surfaces of the poses differ. The position with the greatest number of differences that hasn't been viewed before has the best potential for information gain, and therefore is selected as the next viewing position.

The original algorithm in [2] assumes that an initial set of sensing directions exists, and uses these

positions to compute sensing lines, which are lines perpendicular to the sensing directions and are used to determine where the poses differ. In this implementation, this is not assumed. Instead, the sensing directions are artificially computed by determining a circle that encloses the entire set of possible poses. Sensing directions are then selected from points along the enclosing circle. This was necessary since for this project only one sensing direction would exist initially. Therefore, only one sensing line would be created and only the object surfaces facing this sensing line would be used to determine next views. By artificially creating sensing directions surrounding the set of poses, all surfaces could be used. See figure 6a for an example of sensing directions and sensing lines for two possible poses of a T-shaped object.

Next, for each sensing line, select one pose and project the boundaries of all its visible surfaces facing the sensing line onto the sensing line, and label these resulting line segments with the surface normal and distance ranges (see figure 6b). This is repeated for each pose, giving different disjoint partitions of the sensing line for each pose (see figure 6c).

Next, the set of all such partitions are intersected and the resulting partition is labeled by the union of the labels of the corresponding segments of the individual partitions. The labels of the resulting segments are then compared to find differing surface normals or distances. Figure 6d shows the resulting segments which contain differing labels. The midpoints for each segment are selected for the viewing positions. This set is sorted by the number of differing poses per viewing position, and the position with the highest number is selected as the next sensing location.

to the L-shaped and T-shaped objects respectively. No matches were found for the square object, and all matches to the C-shaped and H-shaped objects were removed from the hypothesis set since the laser rays passed through their poses (see figure 9e). Figure 9f shows the sensing lines determined for the matches. The eight lines forming a circle around the poses are the sensing lines, and the lines perpendicular to them represent the set of viewing directions where the surface normals or distances to the surfaces differ. The small circle represents the next viewing position. The second view eliminates one of the T-shaped object matches, and a new sensing strategy is computed (figure 9g). With the third view, several new hypotheses are ascertained, and once again a new sensing strategy is computed (figure 9h). The following two observations deleted all but one hypothesis. Figure 9i shows this remaining hypothesis along with all the observed curves, laser rays, and robot viewing positions. The final results are listed in table 1. Due to some laser light striper noise that could not be smoothed out, part of the last curve viewed could not be matched to the hypothesis and was determined to be an occluding object. This effected the hypothesis's measure, but it didn't effect the overall outcome of the experiment. Although this experiment ended with only one possible hypothesis, it is not always possible for this to occur as is shown by experiment #2.

Experiment #2

Figure 10a shows the approximated scene and starting robot position for the second experiment. In figure 10b, the first observation is shown along with the three determined hypotheses, and the next viewing position. In figure 10c, one of the T-shaped hypotheses is deleted, and figure 10d shows all the observations made. This experiment ended with two hypotheses, and as can be seen from tables 2 and 3, the hypothesis for the T-shaped object had a higher measure than the hypothesis for the L-shaped object.

Experiment #3

This experiment had one occluding object in the scene and is displayed in figure 11a. Figure 11b shows the second observation and the sensing strategy computed after that view. The results of the first observation were similar to figures 9f and 10b. The third and fourth view was of the occlud-

ing object, and the last observation eliminated the T-shaped object hypothesis. The final results are shown in table 4. The clustering algorithm used actually clustered the nonmatching curves into two occluding objects instead of only one. This only effected the measure of the hypothesis and not the overall matching.

Experiment #4

Experiment 4 had three occluding obstacles lying around a L-shaped object (see figure 12a). Figure 12b shows the first and second observations. The third observation (shown in figure 12c) results in one of the T-shaped object hypotheses to be deleted. Based on the fifth observation, numerous new hypotheses are created and a new sensing strategy is computed (figure 12d). Since there were so many possible hypotheses, over 100 viewing positions were computed. For each iteration, the robot moves to the closest viewing position with the greatest potential for information gain. Additionally, each observation position is recorded so that the robot doesn't make more than one observation from the same general area. After a total of 13 observations, the hypothesis set was reduced back to the 2 hypotheses as shown in figure 12e along with all the observed surfaces. The results are listed in tables 5 and 6.

Experiment #5

In this experiment the matching algorithm failed to make the correct match. Figure 13a shows the approximate scene and figures 13b and 13c show the matches given the first observation. The occluding object makes it appear that the L-shaped object is at a different location than it actually is. Since the subsequent observations result in only line matches, no new hypotheses are created and an incorrect match results.

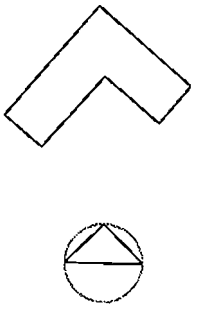


Figure 9a

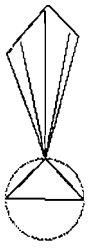


Figure 9b

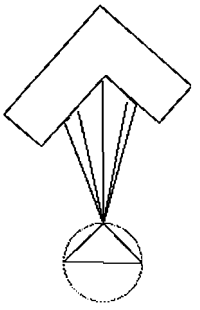


Figure 9c

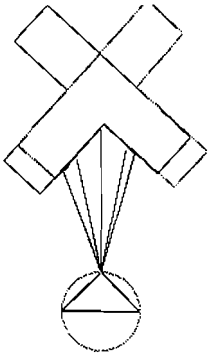


Figure 9d

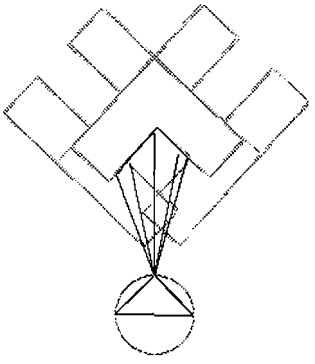


Figure 9e

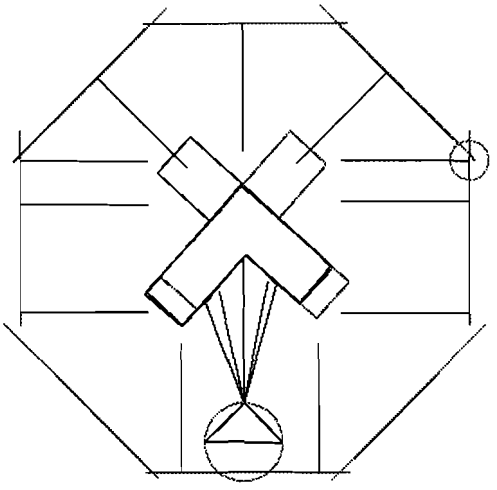


Figure 9f

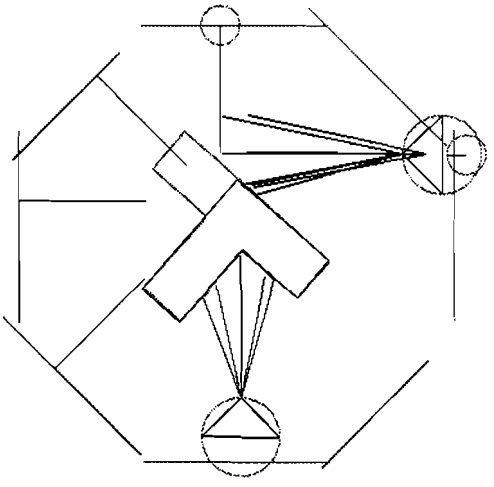


Figure 9g

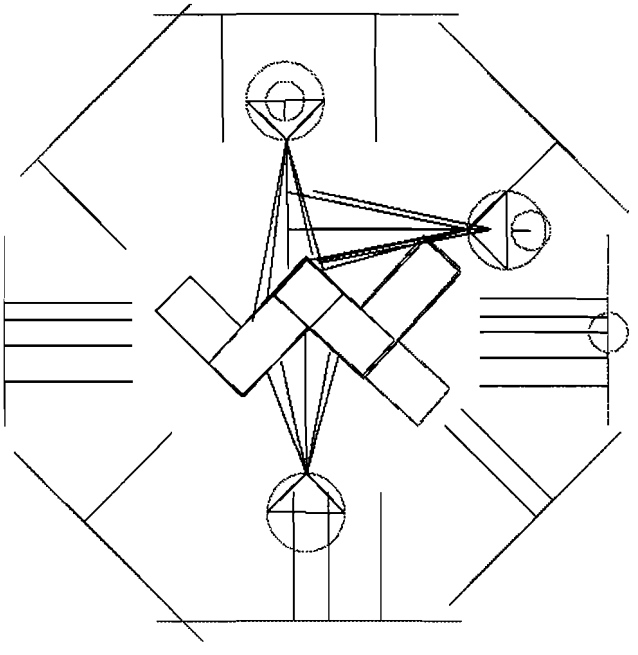


Figure 9h

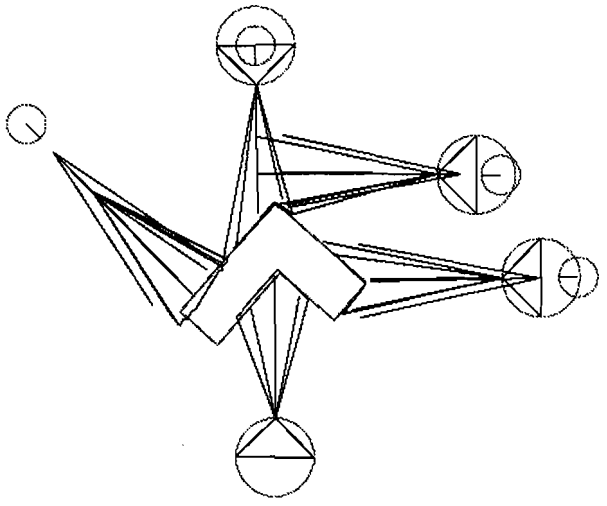


Figure 9i

Table 1: Experiment 1 results

Hypothesis 1: L-shaped object	
Percentage matched:	.606715
Average distance per point:	2.84435
Max average distance per point:	6.39442
Number of occluding objects:	1
Hypothesis measure:	.080841

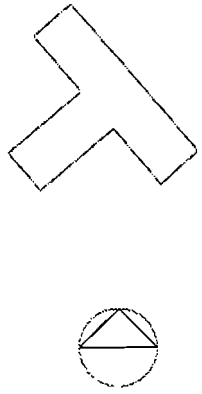


Figure 10a

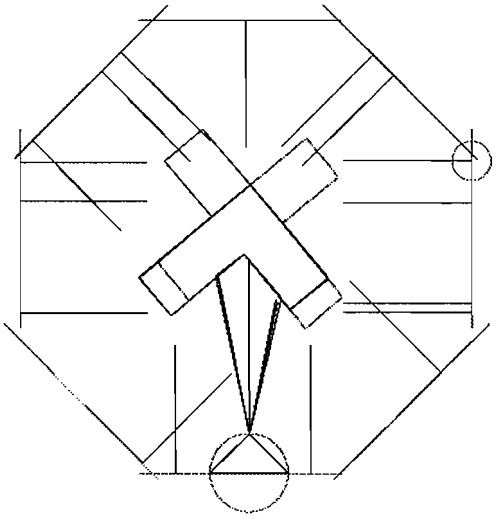


Figure 10b

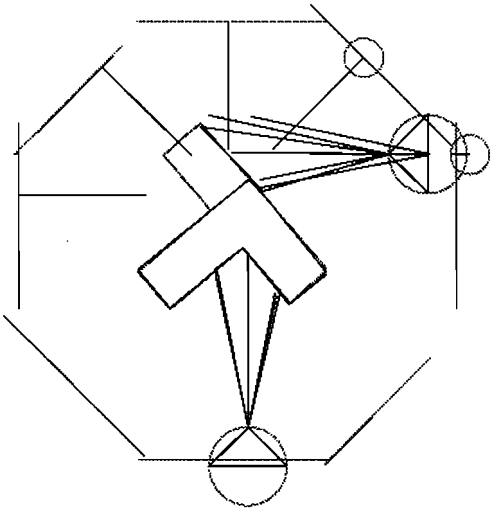


Figure 10c

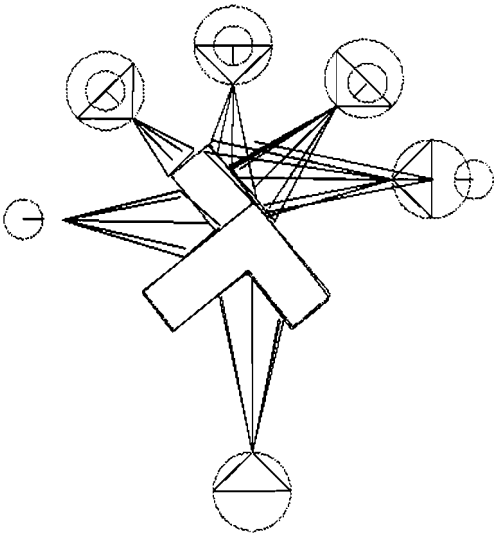


Figure 10d

Table 2: Experiment 2 results

Hypothesis 1: T-shaped object	
Percentage matched:	.55283
Average distance per point:	3.61282
Max average distance per point:	5.84602
Number of occluding objects:	0
Hypothesis measure:	.0591313

Table 3: Experiment 2 results

Hypothesis 2: L-shaped object	
Percentage matched:	.402878
Average distance per point:	3.49585
Max average distance per point:	5.66362
Number of occluding objects:	1
Hypothesis measure:	.0370087

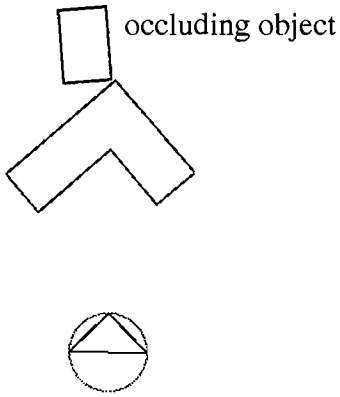


Figure 11a

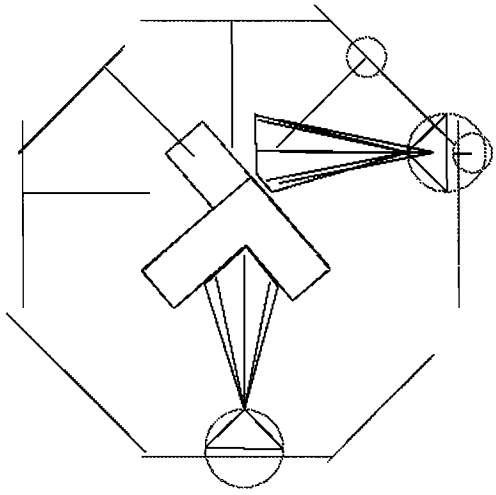


Figure 11b

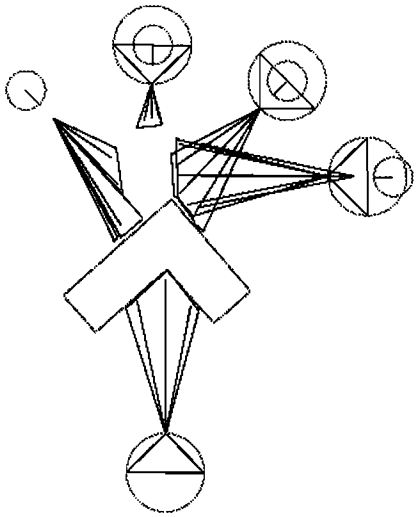


Figure 11c

Table 4: Experiment 3 results

Hypothesis 1: L-shaped object	
Percentage matched:	.258993
Average distance per point:	2.09154
Max average distance per point:	3.71033
Number of occluding objects:	2
Hypothesis measure:	.0412369

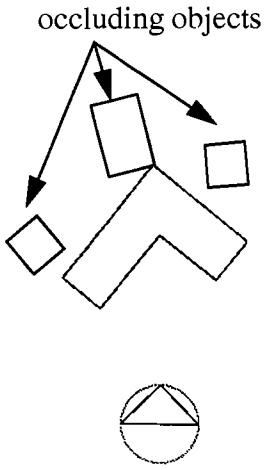


Figure 12a

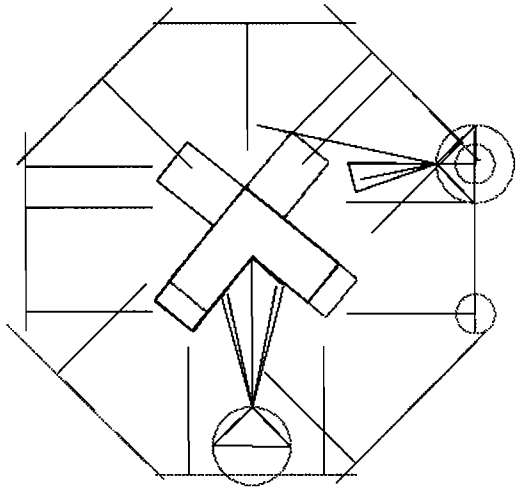


Figure 12b

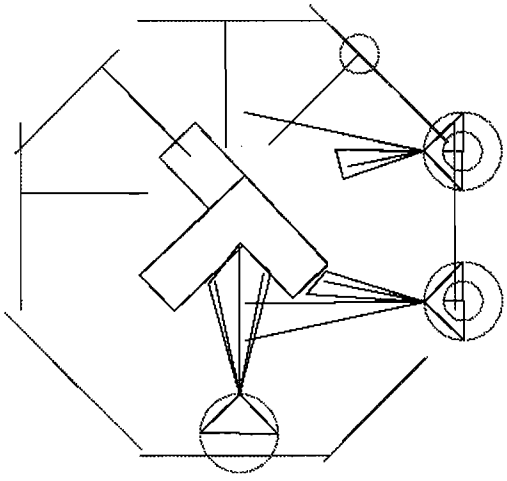


Figure 12c

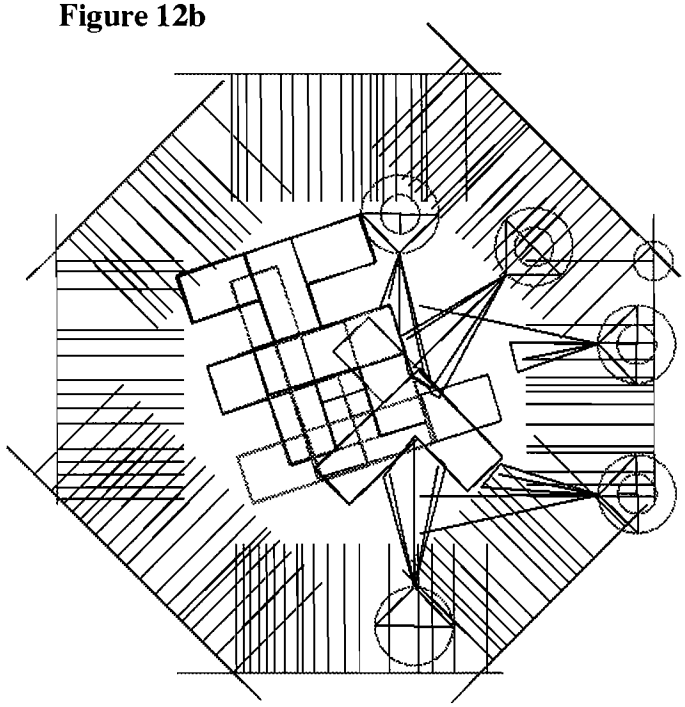


Figure 12d

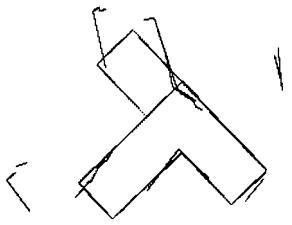


Figure 12e

Table 5: Experiment 4 results

Hypothesis 1: T-shaped object	
Percentage matched:	.262264
Average distance per point:	3.67816
Max average distance per point:	4.87336
Number of occluding objects:	6
Hypothesis measure:	.00257284

Table 6: Experiment 4 results

Hypothesis 2: L-shaped object	
Percentage matched:	.266187
Average distance per point:	3.05045
Max average distance per point:	4.53378
Number of occluding objects:	6
Hypothesis measure:	.00348356

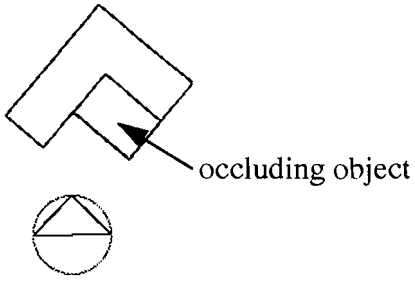


Figure 13a

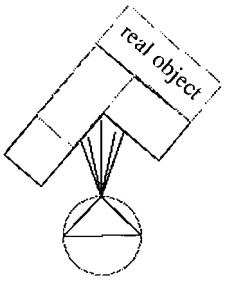


Figure 13b

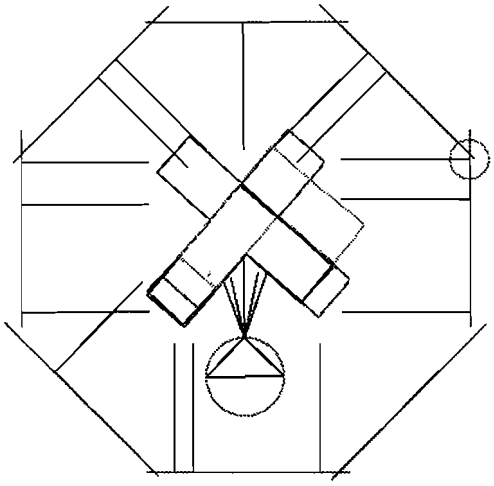


Figure 13c

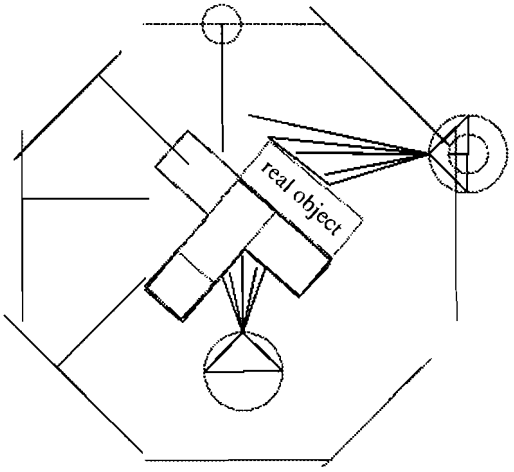


Figure 13d

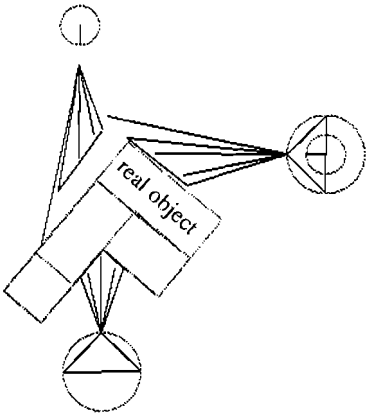


Figure 13e

6 Summary

The method presented in this paper allows a robot to plan multiple observations of partially occluded objects and combine the data from these views to preform object recognition. Although all tests were performed on polygonal objects, this method can be applied to any set of objects by using their polygonal approximations. In this method, hypotheses are only created when a curved segment returned from the sensor matches to an object in the database. This unfortunately can lead to incorrect matches if the initial data is of occluding objects that are in a formation that resembles real objects in the database and if no other data is retrieved that can create new hypotheses. This is a hard problem to avoid however, since it is infeasible to maintain a hypothesis for every possible line match.

References

[1] H. Wolfson. On Curve Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 5, pp. 483 - 489, 1990.

[2] W. E. Grimson. Sensing Strategies for Disambiguating Among Multiple Objects in Known Poses. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 4, 1986.

[3] J. T. Schwartz, and M. Sharir. Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves. *The International Journal of Robotics Research*, Vol. 6, No. 2, pp. 29 - 44, 1987.

[4] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir. Two Dimensional, Model Based, Boundary Matching Using Footprints. *The International Journal of Robotics Research*, Vol. 5, No. 4, pp. 38 - 55, 1986.

[5] N. Ansari and E. Delp. Partial Shape Recognition: A landmark-Based Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 5, pp. 470 - 483, 1990.

[6] J. Z. Lai, and J. M. Lin. Recognizing Partially Occluded 2-D Parts Based on Tracing of Feature Points. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pp. 1796 - 1801, 1991.

[7] R. D. Rimey, and C. M. Brown. Where to Look Next Using a Bayes Net: Incorporating Geometric Relations. *Proceedings of the European Conference on Computer Vision*, pp. 543 - 550, 1993.

[8] J. Maver, and R. Bajcsy. How to Decide From the First View Where to Look Next. *Proceedings of the Image Understanding Workshop*, pp. 482 - 496, 1990.

[9] K. D. Gremban, and K. Ikeuchi. Planning Multiple Observations for Object Recognition. *CMU-CS-92-146*, Carnegie Mellon University, 1992.