LOCATING POLYHEDRAL OBJECTS FROM EDGE POINT DATA

Gary Silverman
IBM Scientific Center, Los Angeles, CA 90025

Roger Tsai and Mark Lavin
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

ABSTRACT

A method is presented to locate a general polyhedron in space with six degrees of freedom. A light stripe sensor is assumed to be capable of providing three-dimensional data points on known edges of the polyhedron to within a known accuracy. A representation is found for all the ways three line segments of fixed length can fall between the edges of a polyhedron. This representation, called an edge interval, is used in a tree search to find the location of each data point to within an interval on the edge. The final location of the object is determined by a one-dimensional line search. The method is illustrated with simulated data from a unit tetrahedron and results from a computer implementation are presented. The tree search takes a negligible amount of time and the one-dimensional line search takes a small additional increment.

I INTRODUCTION

The general problem is to determine the identity and location of a part in 3-D space to within a known error in real time. Gordon and Seering '87 describe a method which uses light stripes between known edges of a polyhedral object. When three light stripes intersect a right corner, they are able to locate the object in negligible time. Bolles and Horaud '86 "concentrate on edges because they contain more information than surface patches and are relatively easy to detect in range data," page 8. Grimson and Lozano-Perez '84 use points on a known face and an estimate of the surface normal at each point to locate the object. On page 9, they suggest that if a point is on an edge of an object, "the recognition process is greatly simplified."

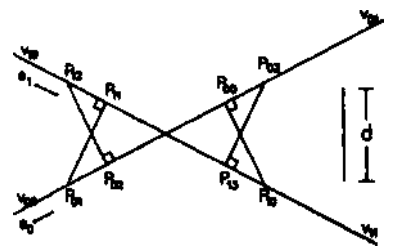
The problem treated in this paper is to locate a general polyhedral object with six degrees of freedom using data from a light stripe sensor. We assume that the light stripes locate 3-D data points on edges of the object to within a known error. Call each of these data points an edge point. Research is currently underway to develop effective sensors for this kind of application, see Tsai '85, Gordon and Seering '87 and Bolles and Horaud '86. There is a further assumption that each edge point is on a known edge of the object. This may occur in an application such as in Gordon and Seering '87, where the geometric relation between the sensor and the object has been specified in advance or as in Arbab '87, where the edge points are identified with an edge via an interpretation method.

In the next section the main results are provided on how the ends of a line segment of fixed length may intersect two edges of an object in 3-D space. The Edge Interval Tree section shows how the edge intervals may be propagated as constraints in a tree search to find the location of each edge point to within an interval on the edge. The section on Locating The Object shows how to locate the object within error by a one-dimensional line search.

II EDGE INTERVALS

Consider a line segment of fixed length, *d,* whose end points are known to intersect each of two edges, *e0* and e1, see Figure 1. Now add an orientation to each line, so that a minus (-) and a plus (+) direction is defined along the line. Edge intervals are defined as a pair of intervals, one interval on each line, with the following property. As the point of intersection on the first line moves in a monotonic direction from beginning to end of the first interval, the point of intersection on the second line moves in a monotonic direction on the second interval. These intervals are mutually exclusive and include all possible intersections of the line segment with the two edges.

Figure 1. Monotonic edge intervals on two edges.



To describe the edge intervals introduce vertices "00' 01which define a finite edge €*Q* in Figure 1 and a positive direction on *e0.* Define a parameterization of a point along the edge using a convex combination of the vertices,

(1) $$P_0(\alpha_0) = (1 - \alpha_0)v_{00} + \alpha_0 v_{01},$$

so that,

(2) $P_0(0) = v_{00}$

(3) $P_0(1) = v_{01}$.

Similar vertices and parameterization are defined for edge

Consider the line segment $\overline{P_{00}, P_{10}}$ of length *d.* This line segment is constructed to be perpendicular to edge *e0.* As one end of the line segment moves from poo to *PQI* on edge *e0* the other end moves from P_{10} to P_{11} on eage e\. One end of the line segment is moving along *e§* in the negative direction and the other end is moving along ej in the negative direction. At P_{01}, P_{11} a sense of direction must change. This is because at *P11* the line segment is perpendicular to edge q. It is impossible to continue moving in the negative direction on both edges and still maintain contact. Therefore, as the line segment moves from $\overline{P_{01}, P_{11}}$ to $\overline{P_{02}, P_{12}}$ the sense of direction along e_0 must change from - to + Similarly, from $\overline{P_{02}, P_{12}}$ to $\overline{P_{03}, P_{13}}$ the senses are +, +. Finally, from $\overline{P_{03}, P_{13}}$ returning to PQQ, PJQ the senses are - , +. To convince yourself, cut out the line and re-trace this path on Figure 1.

To calculate values of *OLQ* and aj that correspond to edge intervals, the orthogonality condition

(4) $(P_0(\alpha_0) - P_1(\alpha_1)) \bullet (v_{01} - v_{00}) = 0$

and the distance condition

(5) $||P_0(\alpha_0) - P_1(\alpha_1)|| = d$

must be satisfied.

Equations (4) and (5) may be reduced to a standard quadratic equation in *a$.* The solution is the following equations for aQ and a1, where a_1, b_1, c_1, g_1, g_2 and g3 are dependent only on the edge vertices and the distance. These equations are too long to include here, but are included in Silverman et. a187.

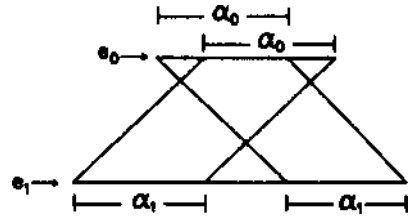
$$\alpha_0 = \frac{-b_1 \mp \sqrt{b_1^2 - 4a_1c_1}}{2a_1} \quad (6)$$

$$\alpha_1 = \frac{g_1\alpha_0 + g_2}{g_3}$$

Cases where equations (6) and (7) are degenerate are classified as follows. If the distance, *d,* is shorter than the minimum distance between the edges, then the discriminant term in (6) is negative. If the edges are orthogonal, the discriminant in (6) is equal to zero. If the edges are parallel, the term a_1 in (6) equals zero and the edge intervals are illustrated in Figure 2. In all of these cases, degenerate or non-degenerate, it is possible to compute the edge intervals by solving simple quadratic equations.

This is a complete representation of edge intervals along edges e0 and e1, and although Figure 1 and Figure 2 are planar, this representation extends immediately to arbitrary edges in 3-D space. Errors in the edge

Figure 2. Edge intervals on parallel edges.



point data will cause error in the final location of the object. For the parallel case shown in Figure 2 if the distance, *d,* is within sensor error of the distance between the edges, edge interval calculation may fail.

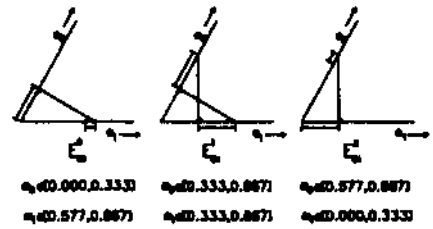


Figure 3. Edge intervals on edges 0 and 1 of unit tetrahedron.

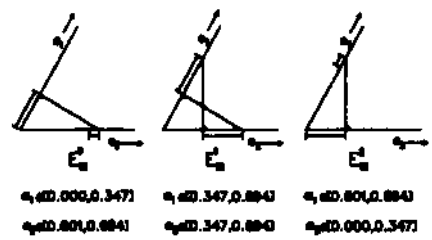


Figure 4. Edge intervals on edges 1 and 2 of unit tetrahedron.

III THE EDGE INTERVAL TREE

The edge interval tree has three levels: Oth level, 1st level and 2nd level, one for each edge point. A node at the *ith* level represents an interval on *e:.* There is an arc emanating to the next level for each edge interval. The monotonic character of the edge intervals enables an edge interval tree search for a set of three valid intervals, one on each edge. The valid intervals are subsets of the re-

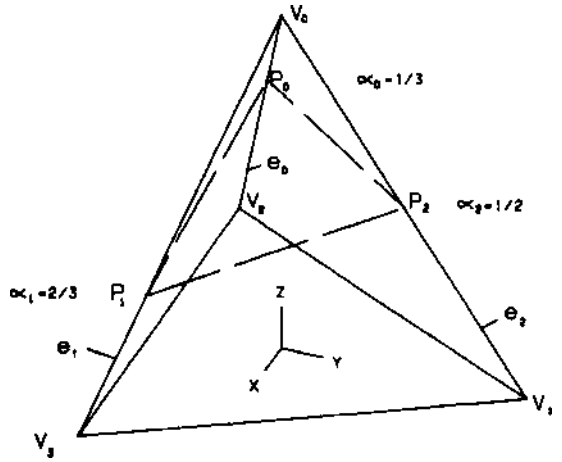


Figure 5. Simulated light stripe on unit tetrahedron.

spective edge intervals.* A*set of valid intervals contains all sets of values $\{\alpha_0^*, \alpha_1^*, \alpha_2^*\}$ such that; if each edge point is located at the value of a, on its respective edge, then the three mutual distances are consistent with the sensor data to within error. This is a location where the edge point data fit the object.

Valid intervals are determined by propagating the edge intervals from e_0 to e_1 to e_2 and back to e0. The monotonic property oT the edge intervals implies there is a continuous, monotonic mapping from any point ai within the edge interval on *e^* which with <£„ maps to a unique point a- on *e:.* Let this mapping and its inverse be denoted by the transfer functions,

(8) $\alpha_j = T_{ij}(\alpha_i)$

and

(9) $\alpha_i = T_{ji}(\alpha_j)$.

The transfer function *Tij* is the location of a point on an edge at a given distance from a point on another edge, where both points are expressed parametrically as in (1).

The equation for *T* can be derived by solving

(10) $\|P_i(\alpha_i) - P_j(\alpha_j)\| = d_{ij}$

for (8) or (9). This leads to a quadratic equation

(11) $$\alpha_j = T_{ij}^{\mp}(\alpha_i) = \frac{-b_2(\alpha_i) \mp \sqrt{b_2(\alpha_i)^2 - 4a_2(\alpha_i)c_2(\alpha_i)}}{2a_2(\alpha_i)}$$

where a% ^2* °2[are] functions of a/, see Silverman et. al/87 for details. This quadratic equation is non-degenerate for any a^- in the edge interval on <?,. The function *Tiji* indicates there are two points on *ej;* at a given distance from a,-, one each for the plus and minus root in (11). At any node in the edge interval tree search the correct sign in (11) is determined by the edge interval on e,-. In the discussion below, let *T* denote a specific transfer function where the sign is determined by the edge interval being propagated.

Searching the Edge Interval Tree

A unit tetrahedron is illustrated in Figure 5. Three edges, e_0, e_1 and e2, are cut by a simulated light stripe. Three edge points, P_0, P_1, P_2, are known to within a given sensor error. The three mutual distances between edge points may be computed, do$_1$' d12[and] d20- A[mode]l of the object provides the vertices for each edge in some canonical location. The edge intervals for the the first two unit tetrahedron edge pairs are displayed in Figure 3 and Figure 4. This example will be used to illustrate the tree search.

The tree is propagated by intersecting intervals on common edges and transferring the resulting intervals to the next edge with (11). Pruning occurs when the intersected interval is empty. The propagation and pruning steps for the tree differ at each level. Figure 6 shows the edge interval constraint propagation from edge interval E_{01}^{0} in Figure 3 to E_{12}^{1} in Figure 4. In Figure 6 the unit tetrahedron has been "unfolded" along ej to make a planar figure illustrating the edge interval tree search.

The Oth level: This level propagates intervals on *eo* to intervals on e using the edge intervals. Let this interval be noted by $J_1 = [\alpha_1^l, \alpha_1^u]$. In Figure 6, 1*J* is actually the edge interval E_{01}^{0} on e_1, $\alpha_1 \in [0.577, 0.667]$.

The 1st level: This level propagates intervals on *e1* to intervals on e2 using *d12-* For the edge intervals from *e1* to e2, intersect the interval on e1 with I1, the interval passed in from the Oth level. Let this intersected interval be noted by $H_1 = [\beta_1^l, \beta_1^u]$. If H1 is empty, then the tree is pruned at this point. In Figure 6, this results in /\, being intersected with E_{12}^{1}. Here $H_1 = I_1$ because $I_1 \subset E_{12}^{1}$.

If H] is equal to *E1 2* then propagate H_1 to *e2* by using the respective edge interval. If not, propagate H1 to *e^* using the transfer function *T.* That is $I_2 = [\alpha_2^l, \alpha_2^u]$, where

(12) $\alpha_2^l = T_{12}(\beta_1^l)$,

(13) $\alpha_2^u = T_{12}(\beta_1^u)$.

This is illustrated in Figure 6, where $I_2 = [0.500, 0.622]$.

The 2nd level: This level propagates intervals on e2 back to intervals on *eo* using d20- The propagation and pruning steps at level 1 are repeated at level 2 using the edge intervals from e2 to e0 and the distance, d_{20}. The result is an interval, $H_0 = [\beta_0^l, \beta_0^u]$, on e_0.

Now propagate H_0 to e_1 to e_2 d back to e o to obtain another interval $I_0 = [\alpha_0^l, \alpha_0^u]$. That is,

(14) $\alpha_0^l = T_{20}(T_{12}(T_{01}(\beta_0^l)))$

and

(15) $\alpha_0^u = T_{20}(T_{12}(T_{01}(\beta_0^u)))$.

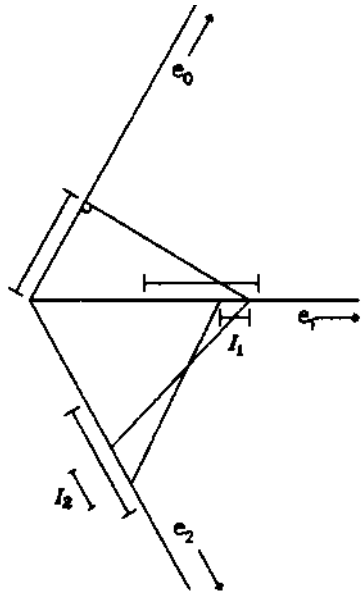


Figure 6. Edge interval constraint propagation on unit tetrahedron.

Errors in the sensor data may cause localizations to be missed unless both intervals are increased. Let *VQ* be the intersection of the increased intervals, Ho and /Q. If Vo is empty the tree may be pruned at this point. Any non-empty interval, *VO,* is a valid interval. This final pruning step is possible because the line segments *d20* and *d01* must intersect *eo* at the same point, *PO,* to within error.

IV LOCATING THE OBJECT

If the solution is known to lie in $V_0 = [\alpha_0^l, \alpha_0^u]$ a one-dimensional line search for a fixed point to within error, of the function,

(16) $$\beta_0 = T_{20}(T_{12}(T_{01}(\alpha_0)))$$

will find all solutions within the valid interval. The line search is required because while all three distances are maintained by (16), both starting and ending points of intersection on *eo* must be the same to within error.

The object localization problem is solved by locating the edge point, P_i, at α_i^* on each edge, e_i, for $i = 0, 1, 2$ Once a particular triple $(\alpha_0^*, \alpha_1^*, \alpha_2^*)$, is known a transformation to within error from model space to image space may be computed to fit the object.

A program has been written implementing the ideas presented here and it has been tested on the unit tetrahedron data. The edge interval calculation involves only a few quadratic equations to solve and takes negligible time. The edge interval tree search requires a few comparisons and some evaluations of (11) per node. This too takes only a very small amount of time. Any edge interval tree with three levels has a maximum of 84 nodes, because the maximum number of branches at any node is four. The edge interval tree search for the unit tetrahedron has to examine only 15 out of the possible 84 nodes. The rest are pruned from the tree. Experience with the current program indicates a few seconds of execution time on the IBM RT PC programmed in C. This includes line search of two valid intervals. It should be possible to substantially improve upon this.

V ACKNOWLEDGMENTS

We would like to thank Bob Cypher of the University of Washington and our colleagues, Bijan Arbab, Fernando Vicuna and Jim Moore for many helpful discussions. The authors are also grateful to the referee whose comments resulted in a number of important clarifications.

REFERENCES

1. Arbab. B. "Object Identification by Parallel Light Stripes/ In Proc. IJCAI-87. Milano, Italy, August 1987.

2 Bolles R.C. and P. Horaud, "3DPO: A Three-Dimensional Part Orientation System." The International Journal of Robotics Research, 5:3 (1986) 3-26.

3. Gordon, S.J. and W.P. Seering, "Locating Polyhedral Features from Sparse Light Stripe Data." In Proc. IEEE International Conference on Robotics and Automation, Raleigh, NC, April, 1987, pp. 801-806.

4. Grimson, W.E.L. and T. Lozano-Perez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data." The International Journal of Robotics Research, 3:3 (1984) 3-35.

5. Silverman, G. J., R. Tsai and M. Lavin, "Locating Polyhedral Objects from Edge Point Data," LASC Report, IBM Scientific Center, 11601 Wilshire Blvd., Los Angeles, CA, 90025 August 1987.

6. Tsai, R.Y. 1985 "A Versatile Camera Calibration Technique for High Accuracy 3-D Machine Vision Metrology using off-the-shelf TV Cameras and Lenses," IBM T.J. Watson Research Report RC 11413, Yorktown Heights, NY, September 1985.