[WC92] Aaron Wallack and John Canny. A geometric matching algorithm for beam scanning. (preprint, University of California, Berkeley), 1992.

[WCM93] Aaron Wallack, John Canny, and Dinesh Manocha. Object localization using cross-beam sensing. In *IEEE International Conference on Robotics and Automation*, 1993.

[KSSS86] Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir. Two-dimensional model-based boundary matching using footprints. *International Journal of Robotics Research*, 5(4):38–55, 1986.

[LSW88] Yehezkel Lamdan, Jacob T. Schwartz, and Haim J. Wolfson. Object recognition by affine invariant matching. *International Conference of Computer Vision*, 1988.

[LW88] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. Technical Report No. 368, New York University, Robotics Research Laboratory, Department of Computer Science, May 1988.

[Mac02] F.S. Macaulay. On some formula in elimination. *Proceedings of London Mathematical Society*, pages 3–27, May 1902.

[Man92] D. Manocha. *Algebraic and Numeric Techniques for Modeling and Robotics.* PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1992.

[MD92] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves. Computer Science Dept. Technical Report 698, University of California at Berkeley, October 1992.

[Mur91] Richard Murray. *Lymph Reference Manual.* Berkeley Robotics Laboratory, 1991. Version 2.3.

[Nic92] Edward Nicolson. *Talisker Reference Manual.* Berkeley Robotics Laboratory, 1992. Version 2.0.

[Ols] Clark Olson. Probabilistic indexing: Recognizing 3d objects from 2d images using the probabilistic peaking effect. (preprint, University of California, Berkeley).

[Sal85] G. Salmon. *Lessons Introductory to the Modern Higher Algebra.* G.E. Stechert & Co., New York, 1885.

[SS87] Jacob T. Schwartz and Micha Sharir. Identification of partially obscured objects in three dimensions by matching noisy characteristic curves. *International Journal of Robotics Research*, 6(2):29–44, 1987.

[AF86] Nicholas Ayache and Oliver D. Faugeras. Hyper: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 8(1):44–54, 1986.

[ATT67] "Description of a Robot Workspace Based on a Linear Stepper Motor". *AT&T Technical Journal*, 67(2):6–11, 1967.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning.* ACM Doctoral Dissertation Award. MIT Press, 1988.

[CG] J. Canny and K. Goldberg. RISC robotics. (preprint, University of California, Berkeley).

[CJ91] D. T. Clemens and D. W. Jacobs. Space and time bounds on indexing 3-d models from 2-d images. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 13(10):1007–1017, 1991.

[FH86] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[FM90] O.D. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4:225–246, 1990.

[GL89] G.H. Golub and C.F. Van Loan. *Matrix Computations.* John Hopkins Press, Baltimore, 1989.

[Hor89] Bernard Klaus Paul Horn. *Robot Vision.* McGraw-Hill, seventh edition, 1989.

[Hor91] B.K.P. Horn. Relative orientation revisited. *Journal of Optical Society of America*, 8(10):1630–1638, 1991.

[HU90] Daniel P. Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[JJ90] C. Jerian and R. Jain. Polynomial methods for structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1150–1166, 1990.

formulation ([MD92]).

$$M = \{\{2\mathtt{ax2} + 4\mathtt{ax2}T^2 + 2\mathtt{ax2}T^4, \mathtt{ax1y1} + 2\mathtt{ax1y1}T^2 + \mathtt{ax1y1}T^4, \quad (34)$$
$$\mathtt{ax1t1} - 4\mathtt{ax1}T + 2\mathtt{ax1t2}T - 3\mathtt{ax1t1}T^2 + 3\mathtt{ax1t3}T^2 - 2\mathtt{ax1t2}T^3 + 4\mathtt{ax1t4}T^3 - \mathtt{ax1t3}T^4\},$$
$$\{\mathtt{ax1y1} + 2\mathtt{ax1y1}T^2 + \mathtt{ax1y1}T^4, 2\mathtt{ay2} + 4\mathtt{ay2}T^2 + 2\mathtt{ay2}T^4,$$
$$\mathtt{ay1t1} - 4\mathtt{ay1}T + 2\mathtt{ay1t2}T - 3\mathtt{ay1t1}T^2 + 3\mathtt{ay1t3}T^2 - 2\mathtt{ay1t2}T^3 + 4\mathtt{ay1t4}T^3 - \mathtt{ay1t3}T^4\},$$
$$\{\mathtt{ax1} + \mathtt{ax1t1}T + \mathtt{ax1t2}T^2 + \mathtt{ax1t3}T^3 + \mathtt{ax1t4}T^4, \mathtt{ay1} + \mathtt{ay1t1}T + \mathtt{ay1t2}T^2 + \mathtt{ay1t3}T^3$$
$$+\mathtt{ay1t4}T^4, \mathtt{at1} - 4\mathtt{a}T + 2\mathtt{at2}T - 3\mathtt{at1}T^2 + 3\mathtt{at3}T^2 - 2\mathtt{at2}T^3 + 4\mathtt{at4}T^3 - \mathtt{at3}T^4\}\}$$

The sum squared error between points and linear and circular features is a rational function which can be multiplied by $(1+T^2)^2$ to arrive at symbolic algebraic equation 35. The partials of this equation (divided by $(1+T^2)^2$) are the system of equations which is identically zero at the global minimum of the error function.

$$FF_{circle} = \mathtt{a} + X\mathtt{ax1} + XY\mathtt{ax1y1} + XY^2\mathtt{ax1y2} + X^2\mathtt{ax2} + X^2Y\mathtt{ax2y1} + X^3\mathtt{ax3} + Y\mathtt{ay1} +$$
$$Y^2\mathtt{ay2} + Y^3\mathtt{ay3} + T\mathtt{at1} + XT\mathtt{ax1t1} + XYT\mathtt{ax1y1t1} + XY^2T\mathtt{ax1y2t1} + X^2T\mathtt{ax2t1}$$
$$+X^2YT\mathtt{ax2y1t1} + X^3T\mathtt{ax3t1} + YT\mathtt{ay1t1} + Y^2T\mathtt{ay2t1} + Y^3T\mathtt{ay3t1} + T^2\mathtt{at2}$$
$$+XT^2\mathtt{ax1t2} + XYT^2\mathtt{ax1y1t2} + XY^2T^2\mathtt{ax1y2t2} + X^2T^2\mathtt{ax2t2} + X^2YT^2\mathtt{ax2y1t2}$$
$$+X^3T^2\mathtt{ax3t2} + YT^2\mathtt{ay1t2} + Y^2T^2\mathtt{ay2t2} + Y^3T^2\mathtt{ay3t2} + T^3\mathtt{at3} + XT^3\mathtt{ax1t3}$$
$$+XYT^3\mathtt{ax1y1t3} + XY^2T^3\mathtt{ax1y2t3} + X^2T^3\mathtt{ax2t3} + X^2YT^3\mathtt{ax2y1t3} + X^3T^3\mathtt{ax3t3}$$
$$+YT^3\mathtt{ay1t3} + Y^2T^3\mathtt{ay2t3} + Y^3T^3\mathtt{ay3t3} + T^4\mathtt{at4} + XT^4\mathtt{ax1t4} + XYT^4\mathtt{ax1y1t4}$$
$$+XY^2T^4\mathtt{ax1y2t4} + X^2T^4\mathtt{ax2t4} + X^2YT^4\mathtt{ax2y1t4} + X^3T^4\mathtt{ax3t4}$$
$$+YT^4\mathtt{ay1t4} + Y^2T^4\mathtt{ay2t4} + Y^3T^4\mathtt{ay3t4} + (1+T^2)^2(X^2+Y^2)^2\mathtt{ax4y4}$$

References

[ABB+92] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, and D. Sorensen. *LAPACK User's Guide, Release 1.0.* SIAM, Philadelphia, 1992.

- sensor delay: on the order of 1 millisecond
  - sensors are polled discontinuously at frequent intervals
  - the latency associated with polling the robot's position
  - beam sensors have a 500 $\mu second$ delay

5.3 Sources of Error

Due to the fact that we used a reflective sensor instead of a through beam sensor, the sensor functioned partially as an object detector, and partially as a beam scanning sensor. As a result, the sensor measurements were not scan line measurements.

6 Conclusion

In this paper we demonstrated the use of hash tables for recognition of objects using beam sensors and optimal pose computation for localization. Localization was precise to a few thousandths of an inch due to the high accuracy of these sensors. Precomputed indexed hash tables were used to solve the correspondence problem in constant time. Given the set of points and corresponding features, the global minimum least squared pose was computed exactly in almost constant time for given objects with linear boundaries. The approach is directly applicable to objects with nonlinear boundaries with an increased overhead in running time. This is due to the algebraic complexity of finding the global minima corresponding to the optimal pose.

Acknowledgements

The authors wish to acknowledge: Dr. Isabelle Mazon for incorporating the light beam sensors into the RISKY-SPARA environment, Ed Nicolson and Eric Paulos for continually upgrading the RobotWorld environment, Dr. Richard Murray for his assistance, and Brian Mirtich for helpful criticisms and suggestions on this report.

Appendix

The resultant M for computing t between a set of points and associated linear features is given in equation 34. The determinant is a 12^{th} degree polynomial which separates into $(1+T^2)^2$ and an eighth order polynomial. There are at most 8 interesting solutions for t for any set of points and associated linear features. Given that 4 eigenvalues are $+i, -i$, we can improve the algebraic

x-pos. (mm)	computed pos.	Err
53.999	(53.944 , 296.689)	19.84
54.499	(53.957 , 296.653)	17.34
54.999	(53.857 , 296.718)	13.25
55.499	(53.888 , 296.619)	5.08
55.999	(53.926 , 296.700)	9.34
56.499	(53.955 , 296.576)	31.74
56.999	(53.998 , 296.629)	5.02
57.499	(54.005 , 296.602)	4.55

Figure 13: The computed positions (mm) and least squared error, from a $\frac{5}{8}''$ circle scanned by three beams

x-pos. (mm)	computed pos.	Err
53.999	(53.933 , 296.695)	29.36
54.499	(53.955 , 296.615)	53.39
54.999	(53.940 , 296.677)	28.70
55.499	(53.959 , 296.611)	13.64

Figure 14: The computed positions (mm) and least squared error, from a $\frac{5}{8}''$ circle scanned by six beams

5.2 Sources of Uncertainty

The total uncertainty is conservatively estimated by the variable δ, and is incorporated into the correspondence algorithm. We use δ to validate the best least squared error fit pose. Uncertainty arises because:

- parts are imperfect with respect to the model
- imperfect beam sensors
  - hysteresis
  - non-rigidity
  - non-zero beam width

technique's accuracy improves to $\pm.025mm$ or $\pm\frac{2}{1000}''$ (see figure 12).

x-pos. (mm)	computed pos.	orientation	Err
54.499	(53.267 , 296.715)	-0.238	0.06
54.999	(53.247 , 296.762)	-0.834	0.03
55.499	(53.317 , 296.768)	-0.516	0.01
55.999	(53.284 , 296.724)	-0.370	0.00
56.499	(53.298 , 296.694)	-0.354	0.02
56.999	(53.278 , 296.708)	-0.033	0.03
57.499	(53.321 , 296.720)	-0.180	0.02
57.999	(53.367 , 296.718)	0.0768	0.22

Figure 11: The computed positions (mm) and least squared error, from a $\frac{5}{8}''$ square being scanned by three beams

x-pos. (mm)	computed pos.	orientation	Err
53.999	(53.282 , 296.703)	-0.336	0.09
54.499	(53.263 , 296.750)	-0.482	0.09
54.999	(53.319 , 296.752)	-0.401	0.04
55.499	(53.326 , 296.725)	-0.101	0.25

Figure 12: The computed positions (mm) and least squared error, from a $\frac{5}{8}''$ square being scanned by six beams

5.1.2 Scanning a $\frac{5}{8}''$ Circle

The $\frac{5}{8}''$ circle is not as "perfect" as the square, it is ridged and frayed. With a simulated three beam scanner, the technique computed the circle's position to $\pm.10mm$ or $\pm\frac{4}{1000}''$ (see figure 13). With a simulated six beam scanner (based upon the same data), the technique's accuracy improves to $\pm.025mm$ or $\pm\frac{2}{1000}''$ (see figure 14). The computed errors are much larger for the circular object than for the polygon; this result can be attributed to the circle's imperfections.

(the highest condition number for random values of t and the symbolic coordinates). The roots of the determinant of the 31×31 matrix polynomial is solved using the eigendecomposition algorithms on companion matrices highlighted earlier.

4.8 Solving for X

We compute X and Y after solving for T using resultant methods again because all three of the equations are nonlinear in X and Y. The resultant is constructed using the equations: $\frac{\partial FF}{X}$ and $\frac{\partial FF}{Y}$.

5 Experiment and Results

In this section we highlight the performance of recognition and localization algorithms on some cases. The experiments measure the accuracy of the localization technique by simulating the scanning of stationary objects and comparing the results. In these experiments, the simulated scanning sensor is stationary, and the objects are scanned by passing them over the sensor. We are still designing the scanning sensor, so we simulated scanning by repeatedly using a stationary reflective sensor. Two different objects were used to measure the technique's accuracy: a $\frac{5}{8}''$ square and a $\frac{5}{8}''$ circle.

The positional accuracy is measured by computing the position of an an object given the data from simulatedly scanning the object at different positions. The objects were passed over the reflective sensor at a rate of 0.1 inches per second in order to achieve precise measurements; after upgrading the RobotWorld controller hardware (from multiprocessing Motorola 68020s [Mur91] to a SPARC 1 [Nic92]), we should be able to accurately scan objects at the rate of 2 inches per second.

5.1 Localization Accuracy Results

The positional accuracy of this localization technique is measured by simulating the scanning of a $\frac{5}{8}''$ square and a $\frac{5}{8}''$ circle at different positions. Simulated scanning is achieved by grasping the object in the robot gripper and passing the object over a fixed reflective sensor multiple times.

5.1.1 Scanning a $\frac{5}{8}''$ Square

With a simulated three beam scanner, the technique computed the square's position to $\pm.10mm$ or $\pm\frac{4}{1000}''$ (see figure 11). With a simulated six beam scanner (based upon the same data), the

i. $\forall$ Y, if Y is completely real, (ImY 0),

ii. Solve for X given Y and T because $\frac{\partial Error}{\partial Y}$ is quadratic in X

iii. Compute $Error(X, Y, T)$, and compare it to the minimal value found so far

As in the linear feature case, the system of equations is algebraically simpler by transforming the points to match the features, so we solve this system and apply the transformations given in equations (14) through (16).

The squared error between a set of points and a set of linear and circular features is still a fourth order function of T. (In equation 30, a,x refer to points matching circular features and a',x' refer to points matching linear features).

$$(\|(\mathcal{T}(X,Y,\theta)(\mathtt{X},\mathtt{Y},\mathtt{1})^{\mathsf{T}} - (\mathtt{a},\mathtt{b},\mathtt{1}))\|^2 - \mathtt{c})^2 = \quad (29)$$
$$(\frac{\|(Mat(X,Y,T)(\mathtt{X},\mathtt{Y},\mathtt{1})) - (1+T^2)(\mathtt{a},\mathtt{b},\mathtt{1})^{\mathsf{T}}\|^2}{(1+T^2)^2} - \mathtt{c})^2$$

$$Error(X,Y,\theta) = \sum(\|(\mathcal{T}(X,Y,\theta)\vec{x}^{\mathsf{T}}) - \vec{a}^{\mathsf{T}}\|^2 - \mathtt{c})^2 + \sum \|\vec{a'}^{\mathsf{T}} \cdot (\mathcal{T}(X,Y,\theta)\vec{x'})\|^2 = \quad (30)$$

$$\frac{\sum(\|(Mat(X,Y,T)\vec{x} - (1+T^2)\vec{a})\|^2 - \mathtt{c}(1+T^2)^2)^2}{(1+T^2)^4} + \frac{\sum \|\vec{a'}^{\mathsf{T}} \cdot (Mat(X,Y,T)\vec{x'})\|^2}{(1+T^2)^2} \quad (31)$$

Since $\cos^2\theta + \sin^2\theta = 1$, $(1+T^2)^2$ can be factored out of the numerator and denominator in all the algebraic functions using Mat.

$$\frac{(\frac{\|(Mat(X,Y,T)(\mathtt{X},\mathtt{Y},\mathtt{1})) - (1+T^2)(\mathtt{a},\mathtt{b},\mathtt{1})\|}{1+T^2}^2 - \mathtt{c}(1+T^2))^2}{(1+T^2)^2} \quad (32)$$

$$\frac{\sum(\frac{\|(Mat(X,Y,T)\vec{x} - (1+T^2)\vec{a})\|}{1+T^2}^2 - \mathtt{c}(1+T^2))^2}{(1+T^2)^2} \quad (33)$$

The sum squared error between points and linear and circular features is a rational function, multiplied by $(1+T^2)^2$ is given in equation (35).

4.7 Solving for T

The resultant of the system of equations: $\nabla FF_{circle}(X,Y,T) = 0$ is a 36×36 matrix of which the first 3 rows and columns can be discarded because they only add a constant solution to the matrix. The rank of the 33×33 submatrix is 31 because there are two linearly dependent columns. After eliminating these, we keep the 31 rows which together provide the most stability

Then we compute the eigenvalues of E which are the roots of $\det(M(T)) = 0$. The distinct roots are -0.5, 0.618033988750, ∞, -1.618033988750, i, -i. Many of them occur with multiplicity greater than one. Finally we compute the X and Y values corresponding to each T and return the set of values with the smallest error. In this case, there are two poses with minimal error, we choose $(X, Y, T) = (1, 0, \infty)$. In fact, it corresponds to the exact solution of the system of equations.

T	X	Y	Error	Type
-0.5	5	1.6	0.0	global minima
0.618033988750	2.537704574643	2.371804446214	2.992038844286	local maxima
∞	1	0	0.0	global minima
-1.618033988750	3.176581139643	0.199624125214	0.436532584286	local minima

The values $X = 1$,$Y = 0$, and $T = \infty$ ($\theta = \pi$) correspond to the transformation which best maps the set of points to the set of linear features. Since we are interested in the inverse transformation, we use equations 14 through 16 and compute $X = 1$, $Y = 0$, and $\theta = \pi$. (In this case X', Y', and θ' are the same as X_{opt}, Y_{opt}, and θ_{opt}, but in general this is not true.)

4.6 Points and Circular and Linear Features

In the earlier section we had highlighted the algorithm for optimal pose determination for objects with linear features. The algorithm can be extended onto any objects whose objects can be described by algebraic functions (as polynomials). In this section, we show how the algorithm is modified to handle objects with linear and circular features using resultants. The resultant formulation can also find the best affine transformation between a set of points and a set of circular arcs and linear features. The method is similar to the points to linear features case:

1. Eliminate X and Y to construct a resultant matrix for T using symbolic variables. It is expressed as a matrix polynomial.

2. Eliminate X to construct a resultant matrix for Y using symbolic variables, assuming that T has already been computed

3. At run time, compute the symbolic variables from the set of points and features

4. Solve for T using eigenvalue computations described in section 4.2

   (a) $\forall$ T, if T is completely real, (Im(T) 0)

   (b) Solve for Y given T using eigenvalue computations described in section 4.2.

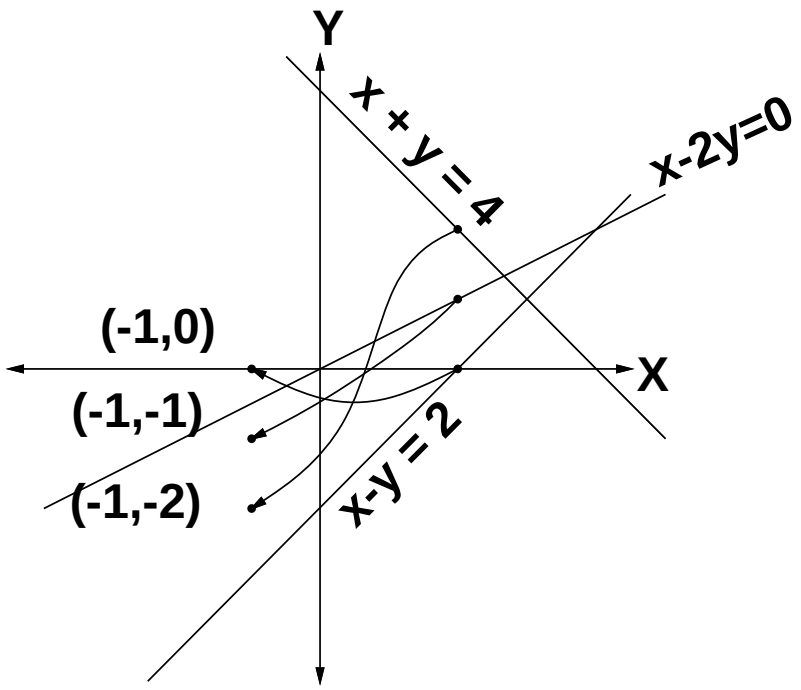

Figure 10: The set of points and corresponding features used to example the optimal pose algorithm

These algebraic error functions are summed to produce FF_Σ:

$$\begin{aligned} FF_\Sigma = 59 - 18X + 3X^2 - 24Y + 6Y^2 - 4T + 12XT \\ -24YT + 62T^2 - 24XT^2 + 6X^2T^2 - 24YT^2 + 12Y^2T^2 - \\ 12T^3 + 12XT^3 - 24YT^3 + 3T^4 - 6XT^4 + 3X^2T^4 + 6Y^2T^4 \end{aligned} \tag{28}$$

Next, we extract the monomial coefficients (ie. `a`, `ax1`, ...). Using these monomial coefficients, we construct the M resultant matrix, and then its companion matrix.

$$\texttt{a} = 59; \texttt{at1} = -28; \texttt{at2} = 62; \texttt{at3} = -20; \texttt{at4} = 3; \texttt{ay1} = -12; \texttt{ay1t1} = -24; \texttt{ay1t2} = -8;$$
$$\texttt{ay1t3} = -24; \texttt{ay1t4} = 4; \texttt{ay2} = 6; \texttt{ax1} = -18; \texttt{ax1t1} = 20; \texttt{ax1t2} = -24; \texttt{ax1t3} = 20;$$
$$\texttt{ax1t4} = -6; \texttt{ax2} = 3; \texttt{ax1y1} = -4;$$
$$M = \{\{6T^4 + 12T^2 + 6, -4T^4 - 8T^2 - 4, -20T^4 + 24T^3 + 24T + 20\},$$
$$\{-4T^4 - 8T^2 - 4, 12T^4 + 24T^2 + 12, 24T^4 + 32T^3 + 32T - 24, \}, \{-6T^4 + 20T^3$$
$$-24T^2 + 20T - 18, 4T^4 - 24T^4 - 8T^2 - 24T - 12, 20T^4 - 112T^3 + 24T^2 - 112T - 28\}\}$$

nomial in the following manner. Let

$$C_1 = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & M_4 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0 & -I & 0 & 0 \\ 0 & 0 & -I & 0 \\ 0 & 0 & 0 & -I \\ M_0 & M_1 & M_2 & M_3 \end{bmatrix} \tag{27}$$

and the eigenvalues of $C_1T + C_2$ correspond exactly to the roots of $\mathrm{Det}(M(T)) = 0$.

4.4 Computing X and Y

Given the value of T computed using eigenvalue routines, the corresponding X and Y coordinates of the roots of the algebraic equations are found in constant time since $\frac{\partial Error(X,Y,T)}{\partial X}$ and $\frac{\partial Error(X,Y,T)}{\partial Y}$ are two equations linear in X and Y.

4.5 Example

In this section we illustrate the algorithm for optimal pose determination on an example consisting of a set of points and linear features. In this example the set of three points is: $Points = \{(-1,0),(-1,-1),(-1,-2)\}$, and the set of associated linear features is: $Features = \{x - y = 2, x - 2y = 0, x + y = 4\}$ (see figure 10).

The first step is to write down the algebraic squared error for each point and associated feature as a function of the pose parameters (X, Y, T).

$$\begin{aligned} FF_{\mathtt{X}_1=-1,\mathtt{Y}_1=0,\mathtt{a}_1=1,\mathtt{b}_1=-1,\mathtt{c}_1=2}(X,Y,T) &= 9 - 6X + X^2 + 6Y - 2XY + Y^2 - 12T + 4XT \\ &-4YT + 10T^2 - 8XT^2 + 2X^2T^2 + 8YT^2 - 4XYT^2 + 2Y^2T^2 - 4T^3 + 4XT^3 - 4YT^3 \\ &+T^4 - 2XT^4 + X^2T^4 + 2YT^4 - 2XYT^4 + Y^2T^4 \\ FF_{\mathtt{X}_2=-1,\mathtt{Y}_2=-1,\mathtt{a}_2=1,\mathtt{b}_2=-2,\mathtt{c}_2=0}(X,Y,T) &= 1 + 2X + X^2 - 4Y - 4XY + 4Y^2 + 12T \\ &+12XT - 24YT + 34T^2 + 2X^2T^2 - 8XYT^2 + 8Y^2T^2 - 12T^3 + 12XT^3 - 24YT^3 + T^4 \\ &-2XT^4 + X^2T^4 + 4YT^4 - 4XYT^4 + 4Y^2T^4 \\ FF_{\mathtt{X}_3=-1,\mathtt{Y}_3=-2,\mathtt{a}_3=1,\mathtt{b}_3=1,\mathtt{c}_3=4}(X,Y,T) &= 49 - 14X + X^2 - 14Y + 2XY + Y^2 - 28T \\ &+4XT + 4YT + 18T^2 - 16XT^2 + 2X^2T^2 - 16YT^2 + 4XYT^2 + 2Y^2T^2 - 4T^3 + 4XT^3 \\ &+4YT^3 + T^4 - 2XT^4 + X^2T^4 - 2YT^4 + 2XYT^4 + Y^2T^4 \end{aligned}$$

solutions of the algebraic equations. Given the T coordinates, we can compute the X and Y coordinates of the roots by back substituting and solving a linear system.

A simple procedure to compute the roots involves expanding the symbolic determinant and computing the roots of the resulting univariate polynomial. This procedure can involves a significant amount of symbolic computation (in determinant expansion), which makes it relatively slow. Moreover, a major problem comes from the fact that finding roots of polynomials of degree greater than 10 is frequently ill-conditioned. As a result, the solutions obtained using IEEE floating point arithmetic (or any other model of finite precision arithmetic) are numerically inaccurate. To circumvent these problems we use matrix computations.

Solving $\det(M(t)) = 0$ is reduced to determining the eigenvalues of the block companion matrix E [Man92]. The construction of the block companion matrix E is highlighted in equation (25). It involves matrix inversion and matrix products. In this section we assume that the leading matrix, M_4, is numerically well-conditioned. The eigenvalues of E are exactly the roots of the equations $\det(M(T)) = 0$ and are computed using routines from LAPACK [ABB+92].

$$M(T) = M_4T^4 + M_3T^3 + M_2T^2 + M_1T + M_0 \tag{25}$$

$$E = \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ M_4^{-1}M_0 & M_4^{-1}M_1 & M_4^{-1}M_2 & M_4^{-1}M_3 \end{bmatrix} \tag{26}$$

4.3 Generalized Eigenvalue Formulation

In the algorithm highlighted in the previous section, it is quite possible that the matrix M_4 is singular or numerically ill-conditioned. It is possible to use linear transformations on the matrix polynomial, $M(t)$, such that the leading matrix of the resulting matrix polynomial is numerically well conditioned [Man92]. However, that may not be always possible and in such cases we reduce the problem of finding roots to a generalized eigenvalue problem of a matrix pencil [GL89].

The matrix pencil for the generalized eigenvalue problem is constructed from the matrix poly-

$$2X^2T^2\texttt{ax2} + YT^2\texttt{ay1t2} + 2Y^2T^2\texttt{ay2} + T^3\texttt{at3} + XT^3\texttt{ax1t3} + YT^3\texttt{ay1t3} +$$
$$T^4\texttt{at4} + XT^4\texttt{ax1t4} + XYT^4\texttt{ax1y1} + X^2T^4\texttt{ax2} + YT^4\texttt{ay1t4} + Y^2T^4\texttt{ay2}$$

To solve for the global minima of $FF * X, Y, T)$, we solve for the common solutions of the three algebraic equations corresponding to the partial derivatives of $FF(X, Y, T)$ with respect to X, Y and T. This results in three algebraic equations in three unknowns X, Y and T. It follows that all minima of FF, both local and global, satisfy these three equations simultaneously. As a result, the resulting algorithm involves solving for all roots of these three equations and back substituting the roots into $FF(X, Y, T)$ to compute the global minima among them.

The system of equations is rewritten algebraically; the partials of $Error$ with respect to X, Y are quotients of the partials of FF with respect to X, Y divided by $(1 + T^2)^2$ (by applying chain rule). The partial of $Error$ with respect to T is shown in equation 22)

$$\frac{\partial Error(X, Y, \theta)}{\partial T} = \frac{\partial \frac{FF(X,Y,T)}{(1+T^2)^2}}{\partial T} = \frac{((1 + T^2)\frac{\partial FF(X,Y,T)}{\partial T} - 4TFF(X, Y, T))}{(1 + T^2)^3} \tag{22}$$

$$\nabla \cdot Error(X, Y, \theta) = \vec{0} \tag{23}$$

The resultant of these system of equations is obtained by eliminating X and Y from these equations and the resulting expression is a polynomial in T [Man92]. In particular, we used Macaulay's formulation of resultant and using it we can express the resultant as determinant of a matrix, whose entries are polynomials in T [Mac02]. We refer to the later expression as a matrix polynomial and the problem reduces to finding common roots of the determinant of a matrix polynomial. Let us denote the matrix polynomial as:

$$M(T) = Resultant(\frac{\partial Error}{\partial X}, \frac{\partial Error}{\partial Y}, \frac{\partial Error}{\partial T}) \tag{24}$$

This matrix polynomial is described in detail in the appendix at the end of the paper.

4.2 Solving the Matrix Polynomial

$M(T)$, a matrix polynomial, (equation 34) is obtained by eliminating X, Y from the equations corresponding to the partial derivatives of $FF(X, Y, T)$. In this section, we highlight a numerically accurate and efficient algorithm to find the roots of Determinant$M(T) = 0$. It follows from the properties of resultants that the roots correspond to the T coordinate of the common

$$\theta' = -\theta_{optimal} \tag{16}$$

$$\mathcal{T}(X,Y,\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & X \\ \sin(\theta) & \cos(\theta) & Y \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$Mat(X,Y,T) \stackrel{T=\tan(\frac{\theta}{2})}{=} \begin{bmatrix} 1-T^2 & -2T & X(1+T^2) \\ 2T & 1-T^2 & Y(1+T^2) \\ 0 & 0 & 1+T^2 \end{bmatrix} \tag{18}$$

4.1 Points and Linear Features

In this section we highlight the method for mapping to linear features. Given a point `X,Y` and an associated linear feature $\mathtt{a}X + \mathtt{b}Y = \mathtt{c}$, the error is: `aX + bY - c`, or $(\mathtt{a},\mathtt{b},-\mathtt{c}) \cdot (\mathtt{X},\mathtt{Y},\mathtt{1})$ (Let $\vec{a}$ be $(\mathtt{a},\mathtt{b},-\mathtt{c})^{\mathsf{T}}$ and $\vec{x}$ be $(\mathtt{X},\mathtt{Y},\mathtt{1})^{\mathsf{T}}$). The squared error, as a function of the transformation $\mathcal{T}$ is shown in equation (19). The total least squared error for all of the points and associated features is highlighted in equation (20).

$$\|(\mathtt{a},\mathtt{b},-\mathtt{c})^{\mathsf{T}} \cdot (\mathcal{T}(X,Y,\theta)(\mathtt{X},\mathtt{Y},\mathtt{1})^{\mathsf{T}})\|^2 = \frac{\|(\mathtt{a},\mathtt{b},-\mathtt{c})^{\mathsf{T}} \cdot (Mat(X,Y,T)(\mathtt{X},\mathtt{Y},\mathtt{1})^{\mathsf{T}})\|^2}{(1+T^2)^2} \tag{19}$$

$$Error(X,Y,\theta) = \sum \|\vec{a}^{\mathsf{T}} \cdot (\mathcal{T}(X,Y,\theta)\vec{x}^{\mathsf{T}})\|^2 = \frac{\sum \|\vec{a}^{\mathsf{T}} \cdot (Mat(X,Y,T)\vec{x}^{\mathsf{T}})\|^2}{(1+T^2)^2} \tag{20}$$

For each point and associated linear feature (beam), we express the error term as a symbolic polynomial in X, Y and T. The symbolic coefficients are represented as (`a, ax1, ax1y1, ...`). Given the point sets and the associated features, these coefficients are being specialized. The resulting approach involves symbolic preprocessing for a given class of object features (say linear edges in this case) and numeric substituting at the run time.

The sum squared error between a set of points and corresponding lines is an algebraic function of translation and rotation (X,Y,θ), and the coefficients $\mathtt{ax_i y_j t_k}$ which depend upon the model points and beams. Given the set of points and associated lines, the global error function (equation (20)) is computed by substituting $T = \tan(\frac{\theta}{2})$, and the resulting algebraic express is polynomial in X, Y and T (as expressed in equation 21).

$$FF(X,Y,T) = \sum \|\vec{a}^{\mathsf{T}} \cdot (Mat(X,Y,T)\vec{x}^{\mathsf{T}})\|^2 = \mathtt{a} + X\mathtt{ax1} + XY\mathtt{ax1y1} + X^2\mathtt{ax2} + Y\mathtt{ay1} + \tag{21}$$
$$Y^2\mathtt{ay2} + T\mathtt{at1} + XT\mathtt{ax1t1} + YT\mathtt{ay1t1} + T^2\mathtt{at2} + XT^2\mathtt{ax1t2} + 2XYT^2\mathtt{ax1y1} +$$

object	disc.	# coordinates checked	# different coordinates	time (seconds)
1 cm circle	0.1 cm	97	49	6
1 cm circle	0.05 cm	189	95	12
1 cm circle	0.01 cm	929	465	45
$1''$ hexagon	0.1 cm	4450	1148	368
$1''$ hexagon	0.05 cm	15276	3875	1209
$\frac{5}{8}''$ square	0.1 cm	10253	2638	726
$\frac{5}{8}''$ square	0.05 cm	36788	9327	2640

Figure 9: The efficiency of adaptively constructing the hash table

4 Optical Pose Determination

In the last section we described the construction and use of a hash table for solving the correspondence problem of an optical beam scanner. In this section we present an algorithm for determining the pose of an object O which best matches the sensed data. The problem of pose determination is reduced to solving a system of overconstrained algebraic equations obtained by matching points to features on the boundary (linear or curved edges). The coefficients of these equations are inaccurate because of noise and the optimal solution is obtained by computing the global least square method. We show how resultants and matrix computations can be utilized to accurately and efficiently compute the global minima of such systems.

The points are matched to their associated features using a rigid motion transformed in a plane, denoted by the matrix $\mathcal{T}(X, Y, \theta)$. We reduce the least square minimization problem to a system of algebraic equations and these equations are solved using resultant. We initial present a symbolic representation of the error term (to be minimized) and later on specialize the coefficients for a given point set and the features of a given object.

Although we are interested in the pose of the edges, the system of equations is algebraically simpler by transforming the points to match the linear features. The transformation of the edges is simply the inverse of the computed optimal transformation for the points given in equations (14) through (16).

$$X' = -(\cos(\theta_{optimal}) * X_{optimal} + \sin(\theta_{optimal}) * Y_{optimal}) \quad (14)$$

$$Y' = -(-\sin(\theta_{optimal}) * X_{optimal} + \cos(\theta_{optimal}) * Y_{optimal}) \quad (15)$$

3.3.2 Partials for Points on Circles

This section describes the partial derivatives of circular arc contact points as functions of Y and θ. Equations 12 and 13 refer to figure 8.

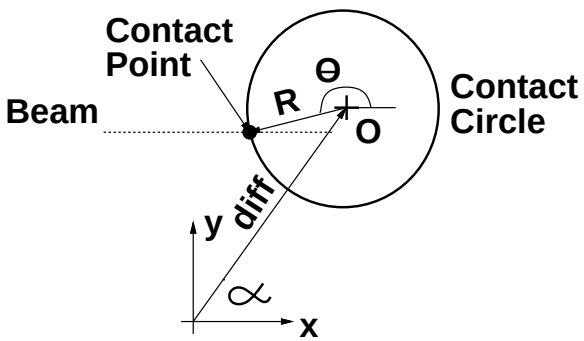

Figure 8: A contact point and its associated circular arc

$$\frac{\partial Contact\ Pt_i}{\partial Y} = |diff| * \tan(\theta) \tag{12}$$

$$\frac{\partial Contact\ Pt_i}{\partial \theta} = -|diff|(\sin(\alpha) - R\cos(\alpha)\tan(\theta)) \tag{13}$$

We determine the neighbors by iteratively using conservative estimates ($\frac{1}{2}$ the estimated change to the next boundary) of the step sizes for $d\theta^+$,$d\theta^-$,dY^+,dY^- until we cross a discretization boundary. θ and Y should always be varied by at least a small minimum amount at each iteration in order to overcome the problem of decreasing derivatives close to a discretization boundary ($d\theta_{min}$ and dY_{min}).

Using the construction, we constructed hash tables for some objects (shown in figure 9). The table contains the following information for each $\langle$ object, discretization resolution $\rangle$: the number of coordinates checked, the total size of the hash table, and the time to construct the hash table.

This construction builds the table in linear time with respect to the size of the table. If the discretization resolution is coarser than the anticipated sensor error, then noisy sensor data will often be discretized into the same hash table entry as exact sensor data.

3.4 Sensor and Object Error

The hash table approach should handle imperfect data point values: the sensors are susceptible to noise, and the objects may differ slightly from the model. The best approach is to make the hash table resolution comfortably larger than a conservative error estimate.

$$dist_{i,b}^{y-} = \begin{cases} Coordinate_b(y,\theta) \bmod \lambda & \text{if } \frac{\partial Coordinate_b(y,\theta)}{\partial y} < 0 \\ \lambda - Coordinate_b(y,\theta) \bmod \lambda & \text{otherwise} \end{cases} \quad (6)$$

$$d\theta^+_to_collision = \min_{b \in axes} \frac{dist_{i,b}^{\theta+}}{\frac{\partial Coordinate_{i,b}}{\partial \theta}} \qquad d\theta^-_to_collision = \min_{b \in axes} \frac{dist_{i,b}^{\theta-}}{\frac{\partial Coordinate_{i,b}}{\partial \theta}} \quad (7)$$

$$dY^+_to_collision = \min_{b \in axes} \frac{dist_{i,b}^{Y+}}{\frac{\partial Coordinate_{i,b}}{\partial Y}} \qquad dY^-_to_collision = \min_{b \in axes} \frac{dist_{i,b}^{Y-}}{\frac{\partial Coordinate_{i,b}}{\partial Y}} \quad (8)$$

3.3 Computing the Coordinates and their Partial Derivatives

The coordinates are the normalized scan line endpoints, and they are the intersections of horizontal lines with a copy of the object rotated by θ and translated by $(0, y)$. The partial derivatives of the coordinates are the differences between the partials derivatives of the intersections and the partial derivatives of the minContactPoint (equation 9).

$$\partial Coordinate_i(y,\theta) = \partial(ContactPoint_i.x(y,\theta) - \min ContactPoint.X(y,\theta)) \quad (9)$$

3.3.1 Partials for Points on Edges

This section describes the partial derivatives of edge contact points as functions of Y and θ. Equations 10 and 11 refer to figure 7.

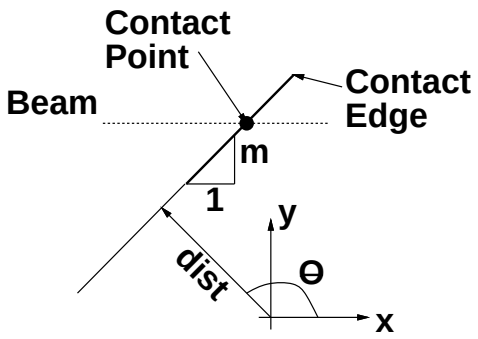

Figure 7: A contact point and its associated edge

$$\frac{\partial Contact\ Pt_i}{\partial Y} = Slope(Contact\ Edge_i) \quad (10)$$

$$\frac{\partial Contact\ Pt_i}{\partial \theta} = \frac{-YIntercept(Beam) + dist\sin(\theta)}{\cos^2(\theta)} \quad (11)$$

$(y, \theta-)$, $(y, \theta+)$. The brush fire keeps searching neighbors unless the point has been searched previously. This success of this technique depends upon the ability to find the most immediately adjacent coordinate neighbor in each of those directions.

3.2.1 Determining Neighbors

Each coordinate may be treated as a function of Y and θ. The partial derivatives with respect to Y and θ predict which discretization boundary is crossed as y and θ increase or decrease. Discretization boundaries, and steps, occur at multiples of λ. The distance to each discretization boundary of axis b , and $dist_{i,b}^{dir}$, are given in equations 3 through 6. A coordinates neighbors are equivalent to its closest immediate discretization boundaries in $\pm y, \pm\theta$ directions.

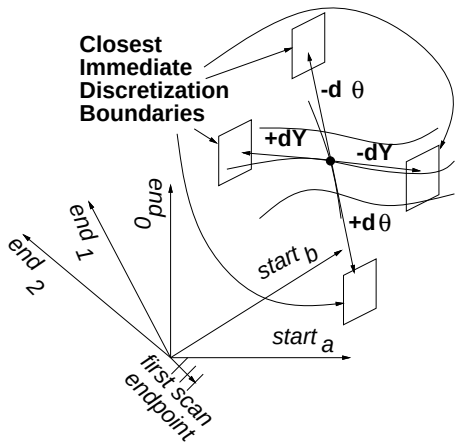

Figure 6: The adaptive step sizes of $d\theta$ and dY depend upon the closest immediate discretization boundaries which is crossed as the coordinates vary with θ and Y

$$dist_{i,b}^{\theta+} = \begin{cases} Coordinate_b(y,\theta) \bmod \lambda & \text{if } \frac{\partial Coordinate_b(y,\theta)}{\partial \theta} > 0 \\ \lambda - Coordinate_b(y,\theta) \bmod \lambda & \text{otherwise} \end{cases} \tag{3}$$

$$dist_{i,b}^{\theta-} = \begin{cases} Coordinate_b(y,\theta) \bmod \lambda & \text{if } \frac{\partial Coordinate_b(y,\theta)}{\partial \theta} < 0 \\ \lambda - Coordinate_b(y,\theta) \bmod \lambda & \text{otherwise} \end{cases} \tag{4}$$

$$dist_{i,b}^{y+} = \begin{cases} Coordinate_b(y,\theta) \bmod \lambda & \text{if } \frac{\partial Coordinate_b(y,\theta)}{\partial y} > 0 \\ \lambda - Coordinate_b(y,\theta) \bmod \lambda & \text{otherwise} \end{cases} \tag{5}$$

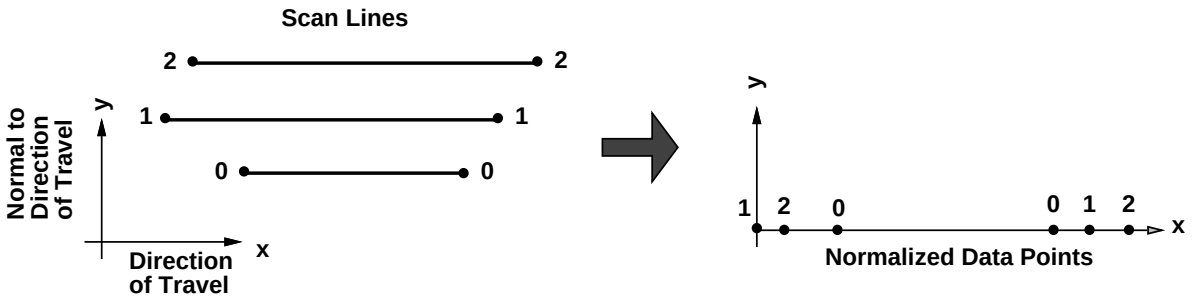

Figure 4: The scan line points are *normalized* by shifting all of the data points such that the *minContactPoint* (the closest point in the direction of travel) is shifted to the origin

3.2 Constructing the Hash Table

The hash table entries are filled in by simulating the scanning sensors for various y-offsets and orientations, and computing the normalized scan line endpoints. The values of y and θ should not be stepped through using fixed size steps dy and $d\theta$ because we could miss some valid coordinates. dy and $d\theta$ should be functions of the $Coordinate_i(y, \theta)$ and the derivatives of $Coordinates_i(y, \theta)$

Similarly, we should not step through θ for specific values of y (or step through y for specific values of θ) because gaps may develop between once adjacent paths of constant y (or θ).

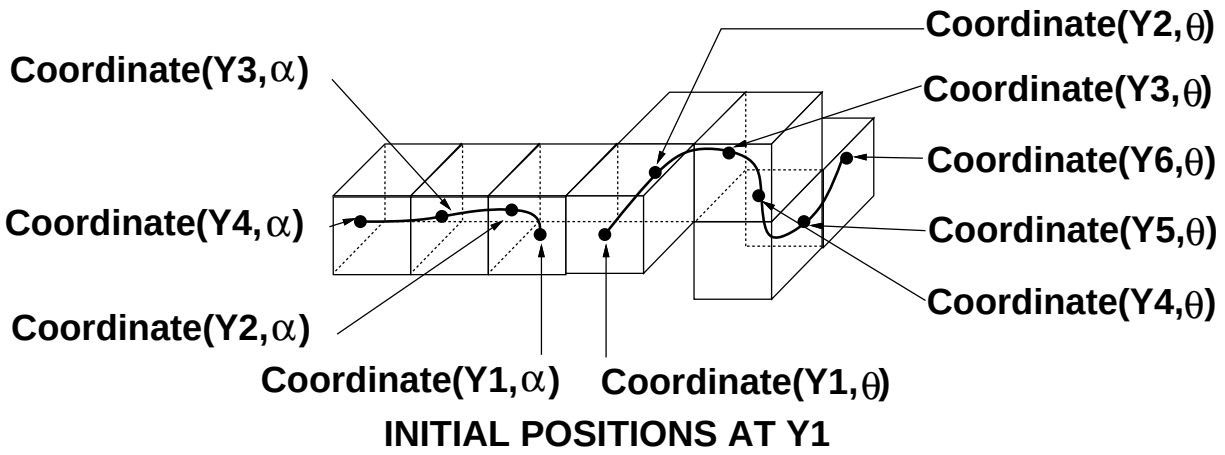

Figure 5: $Coordinate_i(\theta)$ and $Coordinate_i(\alpha)$ are continuous functions of Y, but the adjacency of $Coordinate_i(Y1, \theta)$ and $Coordinate_i(Y1, \alpha)$ does not guarantee that the paths remain adjacent

We use a brush fire search technique to fill in all of the valid coordinates. In this case, given a starting point (y, θ) the brush fire technique checks all of its adjacent neighbors $(y-, \theta)$, $(y+, \theta)$,

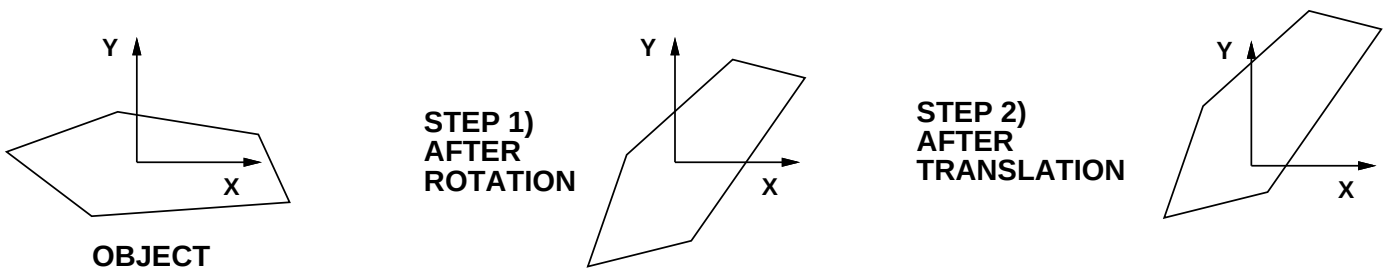

Figure 2: The hash table is filled in by simulating horizontal scan lines on a rotated and translated object

minimum contact point from one of the n beams always shifts to zero after normalization. In this case (3 scanning beams), the coordinate space is $3 \times \mathbb{R}^5$. The valid coordinates parameterized by Y and θ form a two-dimensional manifold in the coordinate space (see figure 3).

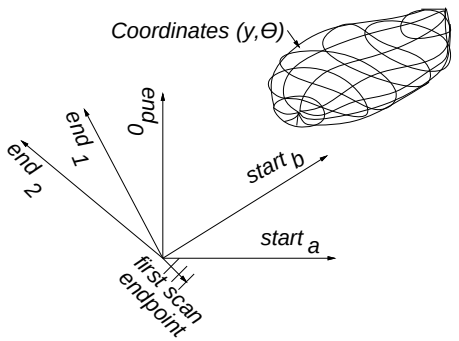

Figure 3: The values $first\,scan\,endpoint$, $start_a$, $start_b$, end_0, end_1, end_2 depend upon the y-offset and θ

3.1.1 Normalizing the Scan Line Endpoints

Normalizing the scan line endpoints removes any positional information along the direction of travel, and therefore reduces the complexity of the correspondence problem's by 1 dimension (refer figure 4 and equation 2).

$$Normalize(ContactPoint_i.x(y,\theta)) = ContactPoint_i.x(y,\theta) - \min ContactPoint.x(y,\theta) \quad (1)$$

$$Coordinate_i(y,\theta) = Normalize(ContactPoint_i.x(y,\theta)) \quad (2)$$

The algorithm for solving non-linear equations using resultants makes use of matrix computations like matrix polynomials, eigendecomposition, singular value decomposition and solving linear systems using Gauss elimination [GL89]. Routines for these matrix computations are available as part of standard numerical linear algebra libraries. In particular, we used LAPACK routines in the implementation of our algorithm [ABB+92].

3 Localizing Objects In Constant Time

In a typical industrial environment, the localization technique is continuously performed for the same model. There are many geometric approaches known in the literature for solving the correspondence problem [WC92, AF86, KSSS86, HU90, CJ91, SS87, FH86], and their running time is at least linear or quadratic in terms of number of features. Most industrial applications demand that the correspondence problem is repeatedly solved online and that makes most of these geometric algorithms suboptimal. The approach highlighted in this section involves preprocessing, given the geometry of the optical sensor and the database of objects and constructing a hash table. The hash table records enough information so that the correspondence can be looked up from the data.

3.1 Hash Table

We solve the correspondence problem in constant time using precomputed hash tables. The hash table entries are indexed by discretized, normalized (see figure 4) scan line endpoints and contain the set of edges corresponding the the scan line endpoints. Normalization extracts positional information independent of the direction of travel.

The advantage of the hash table approach is that the forward problem: computing scan lines from a model, is simpler than the inverse problem: localizing objects given the scan lines. The hash table approach solves the correspondence problem for arbitrary 2D objects.

For completeness, all the valid hash table entries are inserted into the hash table; the entries are produced by simulating horizontal beam sensor readings for the object which is equivalent to intersecting horizontal lines with a rotated, translated copy of the object. The entries are parameterized by Y and θ because the object is rotated about the origin by θ and then translated in the Y-direction by Y (see figure 2). Each entry contains the coordinates: normalized intersection points, and the parameters: Y and θ.

The coordinate space is $n \times \mathbb{R}^{2n-1}$ (where n is the number of scanning beams) because the

Overview: The rest of the paper is organized in the following manner. In Section 2 we briefly describe the beam scanners and introduce multipolynomial resultants. In Section 3 we present the hash table construction for the correspondence problem, given the geometry of the problem and the data base of objects $\{O_1, O_2, \ldots, O_n\}$. Section 4 describes the algorithm to compute the pose of the object using resultants and matrix computations. In Section 5 we highlight the performance of the algorithms on some data from beam scanners and finally, in section 6, we conclude by summarizing the results and advantages of these techniques.

2 Background

2.1 Beam Scanners

In a binary beam scanners the objects are scanned by using a small number of very precise beam sensors, and the beams output constrains the identity and position of the object (figure 1). The geometric constraints of the beam are used by the breaking and unbreaking of the beams to efficiently compute the possible edges that obstruct the beam so that the object pose can be determined. More details on their construction and application are given in [WC92].

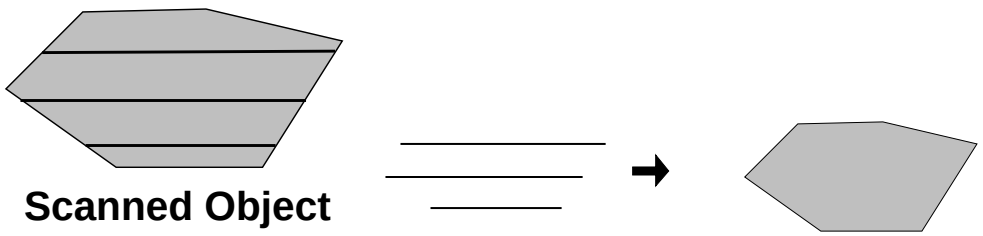

Figure 1: Given the scan lines determined from an arbitrary object, the algorithm determines the objects identity and pose

2.2 Resultants and Equation Solving

The algorithm for computing the global minima of an algebraic function involves solving a system of nonlinear algebraic equations. There are many techniques known in literature for solving algebraic equations. However, most of them suffer from practical problems. Local methods like Newton's method need good initial guesses, while continuation techniques are slow and ineffective for practical applications [Hor91]. We use new algorithms based on multipolynomial resultants and matrix computations described in [Man92, MD92]. Resultants are well known in classical algebraic geometry literature, [Sal85, Mac02] and have recently been effectively used in many problems in robotics, computer graphics and geometric modeling [Can88, Man92].

optimal if objects are modeled as features; sampling points from a feature discards the inherent information. The sampled image sequence could be offset from sampled model sequence, but this would not matter if the feature was left intact. Given a set of points and corresponding features, Ayache and Faugeras [AF86] attempted to compute the least squared error pose by iteration and Kalman filters, but this approach may return a local minimum least squared error.

A great deal of work has been done on matching features (points to points, lines to lines, points to features etc.) in the context of interpreting *shape from motion* [FM90, Hor91]. The problem involves computation of the corresponding features from two distinct 2D views of an an object followed by reducing the problem of motion computation to solving system of nonlinear algebraic equations. In practice, the resulting system is overconstrained and the coefficients of the equations are inaccurate due to noise. As a result the optimal solution is found using global least square minimization. Most of the local methods known in the literature do not work well in practice [JJ90] and an algorithm based on global minimization is greatly desired.

Faugeras and Hebert [FH86] presented an algorithm for localizing 3D objects using point-wise positional information and exactly solved for the minimum least squared error pose for each planar or quadratic feature, but did not solve for the pose with the minimum total error. Huttenlocher and Ullman [HU90] recognized 3D objects from images assuming orthographic projections and computed the exact transformation from three sample points, however using only three of the data points does not provide enough information to compute the optimal transformation.

Hashed indexed tables have been utilized to speed up the task of determining correspondences by matching image features to model features. Hash tables have been earlier used in vision for recognizing 3D objects from 2D images and in some cases from 3D sensor data [AF86, LW88, LSW88, Ols]. They are used for extracting salient features of the object or relative coordinates of the points. [CJ91] used hashed index tables to match quartets of image points to quartets of model points for 3D objects, but their approach requires $O(m^4\ \epsilon^4)$ time (where m refers to the object's complexity and ϵ refers to the the error bound). Furthermore [CJ91] constructed a hashed index table for each quartet of model points, and searched through all possible matches by looking up the sensed data in each of the index tables. By using an overconstrained amount of data to look up table entries, the feature entries are separated enough so that they can all be incorporated into one table. That way, only one entry would need to be looked up to perform recognition and to determine the correspondences using a precomputed table.

set of edges of the object corresponding to the scan line endpoints (the correspondence problem) followed by pose determination by algebraic methods. These edges may be linear or correspond to curved boundaries like circles or ellipses. We present a solution to the correspondence problem by looking up the scan line measurements in an indexed hash table. The major advantage of the hash table approach comes from the fact that the forward problem of computing scan lines from a model is simpler than the inverse problem of localizing objects given the scan line. Given the database of objects and the geometry of beam sensor, the hash table involves a considerable amount of preprocessing. However, at run time it involves only a table lookup and is therefore, very useful. The pose of the object is determined by mapping all the points with their associated features. Using matrix formulation of the rigid motion this problem is reduced to finding an optimal solution to a system of overconstrained algebraic equations. Furthermore, the optimal solution is computed by least square minimization algorithm. A *main characteristic* of our approach is that we are able to compute the *global minima* accurately using algebraic and numeric techniques. In particular, we reduce the minimization problem to solving algebraic equations and use resultants and matrix computations. The resulting algorithm is robust, accurate and efficient and does not require any initial guess like the local methods. We illustrate the recognition and localization algorithms on some examples of optical data.

The important features of these technique are high speed and high precision; given the critical points, objects with linear boundaries are recognized and localized to within $\frac{2}{1000}''$ in 0.1 seconds using a SPARC 1. The algorithm is directly applicable to curved objects as well with some increase in the running time. This is due to the increased complexity of the problem of optimal pose computation. We implemented the technique using Robotworld [ATT67], a high speed, high precision robotic system, to localize linear objects in almost constant time.

1.1 Previous Work

The techniques presented in the paper are related to the larger field of model based machine vision in that they partially solve the problem of object recognition and localization. Previous machine vision research has not solved exactly for the global minimum least squared error between points and features. Kalvin, Schonberg, Schwartz, and Shair [KSSS86], [SS87] overcame the problems of object occlusion and exactly solved for the global minimum least squared error match between sequences of feature points in an image and sequences of feature points in a modeled object. This approach is optimal if objects are modeled as sets of points, but not

1 Introduction

Model based recognition and localization are important components of systems capable of analyzing and understanding 3D scenes [Hor89]. There are many applications for these systems: inspection and quality control, mobile robot navigation, monitoring and surveillance etc. The input to such systems typically consist of two-dimensional images or sensed data like range sensors. The model based recognition problem is to determine which object O from a set of candidate objects $\{O_1, O_2, \ldots\}$ best explains the input data. It turns out that the object O can have any orientation or location and localization problem is to compute the pose p for a given object O which best explains the sensed data.

The problems of recognition and localization have been extensively studied in the vision literature [Hor89]. However, most of the effort has been directed towards recognition of a 3D object from one or multiple 2D views of the object [AF86, KSSS86, SS87, HU90]. In some cases complex sensors providing 3D positional information have been used as well[FH86].

Recently Canny and Goldberg have advocated the use of simple and specialized sensing techniques as opposed to general sensors for recognition applications [CG]. These "specialized" sensors are faster and more robust than general algorithms/sensors and therefore, more useful in the workspace. Two particular sensors based on optical mechanisms are crossbeam sensing and beam scanners are described in [WCM93, WC92]. The crossbeam sensor senses convex hull of objects [WCM93] and the correspondence algorithm presented in [WC92] uses geometric properties of the object and the sensor. The algorithms presented in these papers do not perform any precomputation and are based on online geometric computations. The overall benefit of using simple scalar measurements is the high precision coupled with high speed. For example, using scanning sensors, objects are repeatably localized with high precision (a few thousandths of an inch).

In this paper we present recognition and localization techniques and demonstrate their use on data obtained from optical sensors like beam scanners. Overall these techniques are useful for general machine vision applications. We describe two methods based on *hash table lookups* and *optimal solution* to the problem of matching points to features. The optical scanner based recognition approaches are based on a vertical silhouette of a 3D object and the resulting problem reduces to a 2D recognition and localization problem. The techniques proceed by determining the

Object Recognition and Localization from Optical Sensors *

Dinesh Manocha †
Department of Computer Science
Sitterson Hall, CB #3175
University of North Carolina
Chapel Hill, NC 27599-3175
USA

Aaron S. Wallack ‡ John F. Canny §
Computer Science Division
571 Evans Hall
University of California
Berkeley, CA 94720
USA

Abstract: The problems of object recognition and localization are fundamental to machine vision applications. While the recognition problem deals with finding the identity of an object from a given set to match the sensed data, localization is concerned with pose determination for the matching process. In this paper we present efficient and robust techniques for these problems and demonstrate their application on data obtained from robust, inexpensive and precise optical sensors. Objects are recognized and localized after being scanned by a small number of light beams. The correspondence problem is solved in constant time using precomputed hash tables. After establishing the correspondence, the pose of the object is determined by matching points to features and the problem reduces to solving a system of nonlinear algebraic equations. The resulting system is overconstrained and we find the optimal solution using global least square minimization. The global minima is computed by solving a system of equations using resultants and matrix computations. The overall algorithm for recognition and localization takes about 0.1 seconds on a SPARC I and its accuracy is within 0.002 inches for objects with linear boundaries. The technique is applicable to curved objects as well with some increase in the running time.

*Submitted to the 1993 ICCV International Conference on Computer Vision
†Supported in part by a grant from Mitsubishi Electric Research Lab
‡Supported by Fannie and John Hertz Fellowship
§Supported in part by David and Lucile Packard Fellowship and National Science Foundation Presidential Young Investigator Award (# IRI-8958577).