Object Localization Using Crossbeam Sensing

Aaron S. Wallack [*] John F. Canny [†]
Computer Science Division
University of California
Berkeley, CA 94720

Dinesh Manocha [‡]
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract

Object localization is a fundamental problem in mechanical assembly. In this paper we present a localization technique using reliable, highly accurate, robust, inexpensive light beam sensors designed for mechanical assembly and manufacturing. The technique involves passing polyhedral objects through a set of crossed beams and storing the positions when any beam is broken or reconnects. The geometrical constraints simplify the correspondence problem so that it is solvable in linear time, or can be solved using a precomputed hash table. The global minimum least squared error pose is exactly computed using methods from algebra and numerical analysis. The limitations of this technique result from the beam sensor's limitations.

1 Introduction

Current robotic hardware systems such as a sawyer motor robot [1] combine high speed and high accuracy. In order to fully utilize this hardware the top level robotics control software needs very accurate data about the workspace ($\approx \frac{1}{1000}''$). A recognition and localization technique should match the capabilities of the underlying system to exploit its capabilities. Generic machine vision is not quick enough or precise enough to suit high speed manufacturing. In general, the problem of recognizing and localizing 3D objects in general position is too complex and too difficult to be solved in a reasonable amount of time. By extracting necessary and sufficient data using tailored sensors, recognition and localization can be done as fast and as accurately as high speed manufacturing requires (≈ 0.1 seconds).

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 sensed data. The model based localization problem is to determine the pose p for a given object O which best explains the sensed data.

In this paper we present a recognition and localization technique which combines speed, precision, and robustness by specializing in a specific type of objects: polyhedral objects. This method utilizes cheap, reliable, precise binary light beam sensors which we believe make this technique suitable in industry. The important features of this technique are its high speed and high precision. Given the critical points, the object's identification and pose are estimated in 5 microseconds using a hash table. The resultant method finds the optimal pose even in the presence of initial orientational error and local minima. The optimal pose, accurate to 1 mil, is computed in 0.1 seconds on a SPARC 1 .

1.1 Previous Work

J. Canny and K. Goldberg use the term RISC to describe Reduced Intricacy Sensing and Control as a focusing point for research in simple sensing techniques [3]. Instead of using general sensors, RISC prefers using many different "specialized" sensors in the workspace because custom sensors and algorithms are faster and more "robust" than general algorithms.

In this paper, we connect finger gap sensing with the larger field of object recognition. Goldberg and Mason [6] introduced the *diameter function* (see [15]) to model parallel jaw diameters as part of a probabilistic parts orienter, and they also [7] used finger gap measurements to stochastically generate plans to orient parts. Rao and Goldberg [16], [5] automatically oriented parts by generating plans consisting of a series of oriented grasps.

This is the first general and efficient solution to the non-linear least squares estimation problem such as occurs in computer vision. In partially solving the problem of object recognition and localization, this technique is related to the larger field of model based machine vision [8]. Machine vision has not exactly solved for the global minimum least squared error be-

[*]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).
[‡]Supported in part by a grant from Mitsubishi Electronic Research Lab

tween points and features in a general methodology. [10], [17] overcame the problems associated with 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, but cannot guarantee that the image sequence is in phase with the feature sequence. [4] presented an algorithm for localizing 3D objects using pointwise positional information and exactly solved for the minimum least squared error for each individual feature, rather than all of the features at once. [9] recognized 3D objects from images assuming orthographic projections and computed the exact transformation from three sample points, but did not compute the optimal transformation for all of the data. [11] improved [10] 2D object recognition by precomputing geometric hash tables containing all of the *interest* points, and computed the best match in linear time (by checking each of the *interest* points). .

1.2 Overview

This localization method is intended for use in automated manufacturing by quickly and precisely identifying and localizing polyhedral objects. Objects are identified and localized by passing them through a set of crossbeams, which are light beams oriented in different directions shown in figure 1 (the detecting distance L_1, 95 millimeters, is limited by the sensors). When an object passes through the apparatus, the cross beams perceive a two dimensional cross-section of a three dimensional object; a cross-section is defined as the intersection of the object and a horizontal plane. The robot's position is recorded at the critical points which occur when the beam is first broken, or the beam reconnects, as shown in figure 2. The

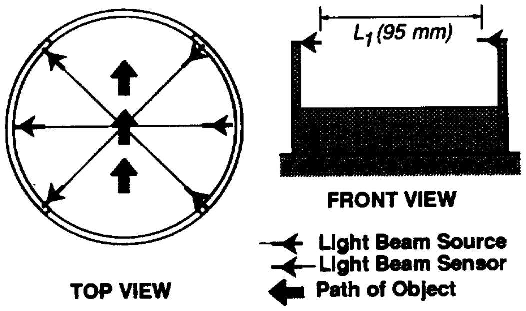

Figure 1: The configuration of the Light Beams

sensors return $2 \times \#$ *of beams* critical points. The technique determines which features of the object correspond to the critical points, (termed the correspondence problem), using either a precomputed hash table, or the object's *diameter function*. Each beam's diameter specifies a set of possible orientations for the object, and these sets are intersected to determine a set of the object's possible orientations. The diameter of the object normal to the direction of the beam depends upon the distance traveled by the robot while the object occludes the beam and upon the angle between the beam and the robot's motion.

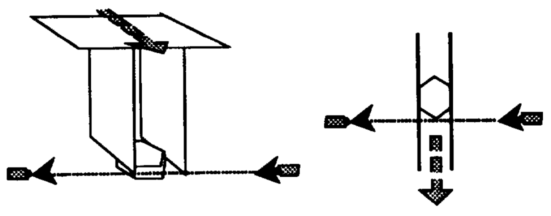

Figure 2: A *critical* point occurs when the object first breaks the beam of light, or when the beam first reconnects

In section 2, we present the theoretical background, and geometrical framework for the correspondence algorithm. In section 3, we present the correspondence algorithm, and section 4 details using hash tables to find the correspondence in constant time. In section 5, we explain the resultant formulation for computing the optimal pose. Section 6 contains the experiment and results. We conclude by highlighting the results and advantages of this technique.

2 Theoretical Framework

This section describes the notation and the theoretical background for the correspondence algorithm. This localization technique is useful for the following class of objects:

1. The object to be identified cannot be flat or translucent (ie., it must be able to break the light beams)
2. The object must initially be lying on a flat surface of its convex hull
3. Items with cross-section (at the pertinent height) with the same convex hull cannot be distinguished
4. The object's cross-section must be a polygon

The correspondence algorithm uses the diameter function which is defined, and diagrammed in this section. The correspondence algorithm efficiently intersects the geometrical constraints with sensed data using the diameter function which maps orientations to diameters.

2.1 Notation and Definitions

- An object's *cross-section* at a particular height is the intersection of that object and a horizontal plane at that height; the cross-section is a polygon.

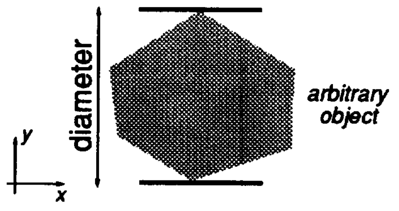

Figure 3: The diameter of a polygon

- The *diameter* of an object is the width of the projection of the object along an axis. In figure 3 the diameter is measured along the y-axis. Since the orientation of the unlocalized object is unknown, the original choice of axis is arbitrary.

  Given the set of critical points, call the robot's direction of travel $\vec{A}$, and let $\vec{B}$ be a perpendicular to the light beam (both unit vectors). The diameter is the travel between corresponding critical points multiplied by the cosine between $\vec{A}$ and $\vec{B}$, or rather *distance*$(\vec{A} \cdot \vec{B})$.

  - $\mathcal{D}_{object}(\theta, \phi)$ is defined as the diameter function of an *object* rotated by θ and ϕ (where θ is rotation about the z-axis and ϕ is rotation about an axis in the x-y plane).
  - $\mathcal{D}_f(\theta)$ is defined as the diameter function for *object* assuming it is resting on face f. (ϕ is fixed)
  - $\mathcal{D}_f^{beam}(\theta)$ is defined as the diameter function relative to a particular *beam* for *object* assuming it is resting on face f. (ϕ is fixed)

- $\Delta(sensed, uncertainty)$ is the range of values $[sensed - uncertainty, sensed + uncertainty]$.

- $\mathcal{D}^{-1}_{object}(\Delta(sensed, uncertainty))$ is defined as the set of orientations consistent with $\Delta(sensed, uncertainty)$.

- The *inset distance* of an edge in an n sided parallelepiped is the distance from the edge to the lowest intersection point of any edges, measured normal to that edge (see figure 4).

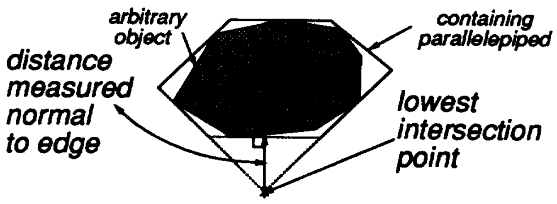
arbitrary object
containing parallelepiped
distance measured normal to edge
lowest intersection point

Figure 4: The inset distance of an edge

2.2 Simplifying Assumptions for $\mathcal{D}$

We make two assumptions: the crossbeam sensor perceives only the convex hull of a cross-section at a particular height, and that there is a finite set of cross-sections for any object assuming that the object was initially stably resting on a horizontal surface. The face upon which the object was initially resting corresponds to the cross-section with the best matching pose. For polygons, such as the cross-sections of the objects, $\mathcal{D}$ consists entirely of sinusoids.

2.3 Computing the Diameter Function ($\mathcal{D}_f$)

The diameter function is the maximum distance between any pair of vertices projected along an axis of angle θ. $\mathcal{D}_f$ can be defined as the pairs of extremal vertices (of the cross-section) for each direction. The diameter of a pair of vertices (of the cross-section) as a function of θ is composed of sinusoidal arcs. An example is shown in figure 5.

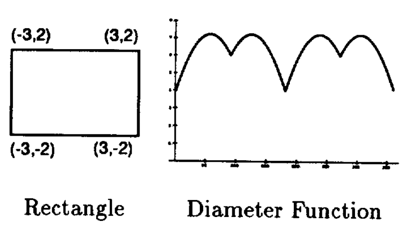

Figure 5: Diameter Function of a 2:3 Rectangle

2.3.1 Vertex-Vertex Pairs

Let L be the vector between a pair of *extremal* vertices in the cross-section. $\mid L \mid$, and $\angle L$ are referred to as: $peak_{distance}$ and $peak_{angle}$ (see Figure 6, equations 1, 2). The diameter function attains its peak value when $\theta = peak_{angle}$. $peak_{distance}$ equals the maximal projected distance between the pair of vertices.

$$local \ \ diameter(\theta) = \mid L \mid \cos(\theta - \angle L) \quad (1)$$

$$\mathcal{D}_f(\theta) = peak_{distance} \cos(\theta - peak_{angle}) \quad (2)$$

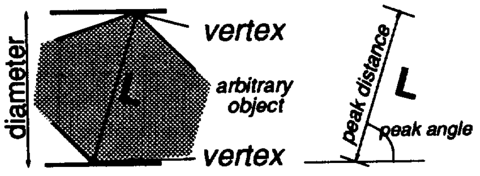

Figure 6: Vertex-vertex contact of a polygon

2.3.2 Collecting Sinusoidal Components into the Diameter Function

The diameter function, which is the maximum of all of those sinusoids, consists of $2n$ portions of sinusoidal arcs (see figure 5), where n is the number of vertices of the object. The intersection points of these sinusoidal components are found by solving equation 3. The diameter function is constructed by collecting all of the extremal pairs of vertices (some of which may be redundant), and intersecting them and retaining the maximal components.

$$\begin{aligned} peak_{dist_1}\cos(peak_{ang_1}-\theta) &= \\ peak_{dist_2}\cos(peak_{ang_2}-\theta) \end{aligned} \tag{3}$$

3 Correspondence Algorithm

In this section, we detail an algorithm for determining the points in the model which correspond to the sensed critical points. The correspondence algorithm determines which vertices might (under uncertainty) correspond to a set of critical points. The correspondence is independent of cartesian coordinates, so the critical points only contain $2n-2$ relevant values. The first step determines an object's set of candidate orientations using the diameter function by intersecting the consistent angle ranges from all of the beams (this utilizes n relevant values). The other $n-2$ relevant values are used to verify each candidate angle range by comparing the expected inset distances with the sensed inset distances. The algorithm is given below:

1. Compute $\mathcal{D}_f^{-1}(\Delta(sensor, uncertainty))$ by scanning through the sinusoid components of $\mathcal{D}_f$ (see figure 7, equation 4)

2. Intersect the sets by scanning down θ (offsetting θ by each beam's direction)

3. Verify the candidate angle range by comparing the expected inset distance for each beam with the sensed inset distances

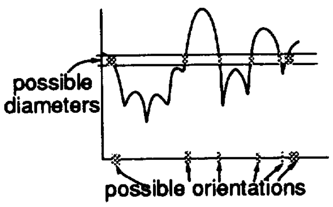

Figure 7: $\mathcal{D}_f^{-1}(\Delta(sensor, uncertainty))$ is the set of possible orientations consistent with the sensed diameter

3.1 Computing $\mathcal{D}^{-1}$

We compute $\mathcal{D}^{-1}$ by scanning through all of the sinusoidal arc components in $\mathcal{D}$ (equation 4). The range of possible diameters (Δ, which incorporates sensor uncertainty) are tested with each sinusoidal component. Since the diameters are discretized, the components corresponding to particular values are cached.

$$\begin{aligned} \mathcal{D}_f^{-1}(\Delta) = \bigvee_{vertex\_pairs} \\ peak_{angle} \pm \arccos\left(\frac{\Delta}{peak_{distance}}\right) \end{aligned} \tag{4}$$

4 Localizing Objects In Constant Time

In this section, we describe a second approach to solving the correspondence problem. We can also solve the correspondence problem in constant time by utilizing precomputed hash tables. The hash entries are indexed by the diameters measured normal to each beam, and the inset distances, and contain the orientation and contact points which generates those coordinates. The orientations are found by looking up the entries with those four coordinates in the hash table; the coordinates are positionally independent and depend solely upon the object's orientation θ. As a function of θ, the coordinates trace out a one-dimensional line in a four dimensional space (see figure 8).

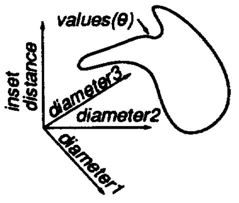

Figure 8: The values $diameter_1$, $diameter_2$, $diameter_3$, $inset\_distance$ depend solely upon θ

4.1 Constructing the Hash Table

We used an adaptively sized hash table and a linear hashing function with random prime scaling coefficients. The hash table entries are filled in by simulating the crossbeam sensors for various orientations, θ, and computing the associated coordinates (given the discretization λ). The values of θ should not be stepped through using fixed size steps $d\theta$ because it may not fill in all of the valid coordinates, $d\theta$ should be a function of $\{value_i(\theta)\}$ ($v_i(\theta)$ in equation 7) and the derivatives of $\{value_i(\theta)\}$ (pt_i refers to the point on the object which contacts $beam_i$). The sign of the derivative predicts which discretization boundary will be crossed as θ increases, and $dist_i$ (equation 7) is the distance between the current $value_i(\theta)$ and the predicted discretization boundary. The functions $value_i(\theta)$ are: the diameter functions and the inset distance function (refer figure 4).

$$\frac{d\mathcal{D}_f^{beam}(\theta)}{d\theta} = -peak_{distance}\sin(\theta - peak_{angle}) \quad (5)$$

$$\frac{d\,inset(\theta)}{d\theta} = \frac{d\,pt_0(\theta)}{d\theta} - \vec{beam}_1 \cdot \vec{beam}_0 \frac{d\,pt_1(\theta)}{d\theta} - \vec{beam}_2 \cdot \vec{beam}_0 \frac{d\,pt_2(\theta)}{d\theta} \quad (6)$$

$$dist_i = \begin{cases} v_i(\theta) \bmod \lambda & \text{if } \frac{dv_i(\theta)}{d\theta} > 0 \\ \lambda - v_i(\theta) \bmod \lambda & \text{otherwise} \end{cases} \quad (7)$$

$$d\theta\_to\_collision = \min_i \frac{dist_i}{\frac{dv_i}{d\theta}} \quad (8)$$

The step size of $d\theta = \frac{d\theta\_to\_collision}{2}$ is a conservative estimate because $\frac{dvalue_i}{d\theta}$ varies slowly. $d\theta$ should always be increased by at least a infinitesmal amount at each iteration in order to overcome the problem of decreasing derivatives close to a discretization step. The hash

object	disc.	# $d\theta$ steps	# entries
$\frac{3}{4}''$ hexagon	0.01 cm	264	81
$\frac{3}{4}''$ hexagon	0.001 cm	2138	793
$1''$ hexagon	0.01 cm	324	98
$1''$ hexagon	0.001 cm	2563	961
$\frac{5}{8}''$ square	0.01 cm	757	306
$\frac{5}{8}''$ square	0.001 cm	6710	2686

Figure 9: The efficiency of adaptively constructing the hash table

table is constructed by incrementing θ in this manner and adding entries into the hash table until $\theta = 2\pi$. For symmetric objects, θ only needs achieve $\frac{2\pi}{symmetry}$. This construction builds the table in linear time with respect to the size of the table. If the discretization resolution is larger than the anticipated sensor error, then noisy sensor data will often be discretized into the same hash table entry as exact sensor data.

4.2 Sensor and Object Error

The hash table approach must handle imperfect data point values: the sensors are susceptible to noise, and the objects may differ slightly from the model. There are two methods for handling such error: assume a large sensor while constructing the hash table, or search over a number of neighboring values in the minimal hash table.

Incorporating a large sensor error into the hash table has the effect of growing the number of hash table entries , which could make it too large to fit in main memory. If the hash table needs to be paged in and out, then accessing an entry is as slow as a disk access (on the order of tens of milliseconds).

Instead of growing the table, the program needs to search all of the data points within a conservative error range. With four coordinates axes, an error range of ± 1 requires $3^4 = 81$ hash table lookups. Each hash table lookup takes 5 microseconds.

5 Resultant Formulation

This section describes the method for computing the optimal pose given the set of critical points and the corresponding points on the model. The goal of this section is to compute the minimum sum squared error between all the points and their associated features after the points have been transformed by the matrix $\mathcal{T}(X,Y,\theta)$. The correspondence algorithm determined which points in the model were associated with each critical point (each critical point denotes a line parallel to its beam).

$$\mathcal{T}(X,Y,\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & X \\ \sin(\theta) & \cos(\theta) & Y \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$Mat(X,Y,t) \overset{t=\tan(\frac{\theta}{2})}{=} \begin{bmatrix} 1-t^2 & -2t & X(1+t^2) \\ 2t & 1-t^2 & Y(1+t^2) \\ 0 & 0 & 1+t^2 \end{bmatrix} \quad (10)$$

Given a point $\mathtt{X},\mathtt{Y}$ and an associated linear feature $\mathtt{a}X+\mathtt{b}Y=\mathtt{c}$, the error is: $\mathtt{aX} + \mathtt{bY} - \mathtt{c}$, or $(\mathtt{a},\mathtt{b},-\mathtt{c})\cdot(\mathtt{X},\mathtt{Y},1)$ (Let $\vec{a}$ be $\mathtt{a}$, $\mathtt{b}$, $-\mathtt{c}$ and $\vec{x}$ be $\mathtt{X}$, $\mathtt{Y}$, 1). The squared error, as a function of the transformation $\mathcal{T}$ is given in equation 11. The total least squared error for all of the points and associated features is given in equation 12.

$$\|(\mathtt{a},\mathtt{b},-\mathtt{c})\cdot(\mathcal{T}(X,Y,\theta)\cdot(\mathtt{X},\mathtt{Y},1))\|^2 = \frac{\|(\mathtt{a},\mathtt{b},-\mathtt{c})\cdot(Mat(X,Y,t)\cdot(\mathtt{X},\mathtt{Y},1))\|^2}{(1+t^2)^2} \quad (11)$$

$$Error(X,Y,\theta) = \sum \|\vec{a}\cdot(\mathcal{T}(X,Y,\theta)\cdot\vec{x})\|^2 = \quad (12)$$

$$\frac{\sum \|\vec{a}\cdot(Mat(X,Y,t)\cdot\vec{x})\|^2}{(1+t^2)^2}$$

For each point and associated linear feature (beam), the coefficients ($\mathbf{a}, \mathbf{ax_1}, \mathbf{ax_1y_1}, \ldots$) for each exponent, are computed and summed. 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 $\mathbf{ax_iy_jt_k}$ which depend upon the model points and beams. Given the set of points and associated lines, the global error function (equation 12) is transformed into an algebraic equation using the substitution $t = \frac{\tan(\theta)}{2}$ (equation 13).

$$\begin{aligned}FF(X,Y,t) = \sum \|\vec{a}\cdot(Mat(X,Y,t)\cdot\vec{x})\|^2 = \quad (13)\\ \mathbf{a} + \mathbf{ax_1}X + \mathbf{ax_1y_1}XY + \mathbf{ax_2}X^2 + \mathbf{ay_1}Y +\\ \mathbf{ay_2}Y^2 + \mathbf{at_1}t + \mathbf{ax_1t_1}Xt + \mathbf{ay_1t_1}Yt + \mathbf{at_2}t^2\\ +\mathbf{ax_1t_2}Xt^2 + 2\mathbf{ax_1y_1}XYt^2 + 2\mathbf{ax_2}X^2t^2\\ +\mathbf{ay_1t_2}Yt^2 + 2\mathbf{ay_2}Y^2t^2 + \mathbf{at_3}t^3 + \mathbf{ax_1t_3}Xt^3\\ +\mathbf{ay_1t_3}Yt^3 + \mathbf{at_4}t^4 + \mathbf{ax_1t_4}Xt^4 + \mathbf{ax_1y_1}XYt^4\\ +\mathbf{ax_2}X^2t^4 + \mathbf{ay_1t_4}Yt^4 + \mathbf{ay_2}Y^2t^4\end{aligned}$$

The global minimum of FF is determined by solving for all of the candidate global minima which are common roots to equations 15. This system of equations is rewritten algebraically; the partials of $Error$ with respect to X, Y are 0 when the partials of FF with respect to X, Y are 0, but $\frac{\partial Error}{t}$ is more complicated (equation 14)

$$\frac{\partial Error(X,Y,\theta)}{\partial t} = \frac{\partial \frac{FF(X,Y,t)}{(1+t^2)^2}}{\partial t} = \quad (14)$$
$$\frac{((1+t^2)\frac{\partial FF(X,Y,t)}{\partial t} - 4tFF(X,Y,t))}{(1+t^2)^3}$$

$$\nabla\cdot Error(X,Y,\theta) = \vec{0} \quad (15)$$

We determine the solutions to this system of equations by using a resultant [12] formulation (equation 16) which eliminates X and Y from the system of equations (eqn. 15). $M(t)$, a matrix polynomial, (see Appendix) is non-invertible ($\det M(t) = 0$) for all values of t which are common roots of the system of equations. M is separated into constant matrices of different exponents in t (equation 17).

$$\begin{aligned}M(t) &=\\ Resultant(&\frac{\partial Error(t)}{\partial X}, \frac{\partial Error(t)}{\partial Y}, \frac{\partial Error(t)}{\partial t}) \quad (16)\\ &= M_4t^4 + M_3t^3 + M_2t^2 + M_1t + M_0 \quad (17)\end{aligned}$$

$$E = \left[\begin{array}{cccc} 0 & I & 0 & 0\\ 0 & 0 & I & 0\\ 0 & 0 & 0 & I\\ M_4^{-1}M_0 & M_4^{-1}M_1 & M_4^{-1}M_2 & M_4^{-1}M_3\end{array}\right] \quad (18)$$

Solving $\det(M(t)) = 0$ is reduced to an eigenvalue problem by constructing the larger matrix E from degree matrices of t (equation 18). The eigenvalues of E are exactly the values of t which solve $\det M(t) = 0$, and are determined using numerical linear algebra routines (eigendecomposition, gaussian elimination, singular value decomposition) in LAPACK [2]. $X(t)$ and $Y(t)$ are solved in constant time since $\frac{\partial Error(X,Y,t)}{\partial X}$ and $\frac{\partial Error(X,Y,t)}{\partial Y}$ are linear in X and Y.

5.1 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
- 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

6 Experiment and Results

In this section, we explain our tests to determine the technique's ability to identify unknown objects and the precision in estimating poses. The experiments measured the accuracy of the localization technique by passing objects in known relative poses through the crossbeam sensor. The positional accuracy was tested by determining the position of an an object (in this case, the $\frac{5}{8}''$ square) passed through the crossbeam sensor along different paths. The orientational accuracy was tested in a similar manner. The technique's recognition performance was tested using a set of four possible objects (as shown in figure 10). The objects moved through the crossbeam (or vice versa) at a rate of 0.1 inches per second in order to achieve precise measurements; after upgrading the RobotWorld (RobotWorld is a registered trademark of Motoman Corporation) controller hardware (from multiprocessing Motorola 68020s [13] to a SPARC 1 [14]), we should be able to accurately scan objects at the rate of 2 inches per second.

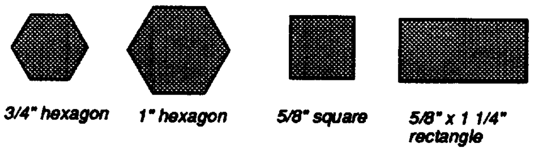

Figure 10: The set of possible objects

6.1 Positional Accuracy Results

The positional accuracy of this localization technique was measured by passing the $\frac{5}{8}''$ square through the crossbeam sensor along paths with different x positions (the object travels through the crossbeam sensor parallel to the y axis). The initial location of the square was unknown, but its relative position should remain constant since it is rigidly held by the gripper. The differences between the robot's commanded position and the square's computed position vary maximally only by $\pm\frac{1}{1000}''$, as shown in figure 11.

x-pos. (mm)	computed x-pos.	diff.	Error
71.497	71.728	0.231	0.01
73.997	74.210	0.213	0.01
76.498	76.718	0.220	0.01
78.998	79.214	0.216	0.01
81.498	81.716	0.218	0.01
81.998	82.206	0.208	0.02
82.498	83.715	0.217	0.02
82.998	83.213	0.215	0.01
83.499	83.710	0.211	0.01
83.999	84.209	0.210	0.01
84.499	84.723	0.224	0.02
84.999	85.209	0.210	0.01
85.499	85.721	0.212	0.02
85.999	86.217	0.218	0.02
86.499	86.712	0.213	0.01
88.999	89.191	0.191	0.02
91.500	91.710	0.210	0.02
94.000	94.207	0.207	0.02
96.496	96.690	0.194	0.02

Figure 11: The actual positions, computed positions, differences in positions (mm) and computed least squared error , from passing a $\frac{5}{8}''$ square through the crossbeam sensor

6.2 Orientational Accuracy Results

Similarly to the previous experiment, the orientational accuracy of the localization technique was tested by passing the $\frac{5}{8}''$ square through the crossbeam sensor at different orientations. Again, the initial orientation of the square object inside the gripper was not accurately measured. The differences between the robot's commanded orientation and the square's computed orientation varies only by at most half of a degree as shown in figure 12.

$\theta(^\circ)$	computed $\theta(^\circ)$	diff. ($^\circ$)	Error
90.076	5.525	5.449	0.01
0.076	5.452	5.376	0.02
-89.962	5.362	5.324	0.01
120.046	35.903	5.857	0.09
30.045	35.850	5.805	0.12
-59.930	35.777	5.847	0.09
-149.930	35.895	5.965	0.11
150.076	66.041	5.965	0.09
60.064	65.893	5.829	0.06
-29.956	65.972	5.928	0.09
-119.936	65.913	5.849	0.06

Figure 12: The actual orientations, computed orientations, differences, in degrees, and computed least squared error , from passing a $\frac{5}{8}''$ square through the crossbeam sensor

6.3 Recognition Experiment

The hash table technique's recognition performance was measured by identifying objects from the set of candidate objects shown in figure 10. We repeatedly identified each object by passing a mobile crossbeam sensor passed over it. The technique returned the model with the minimum squared error due to the observed data. In all 100 trials, the technique correctly recognized the object.

7 Future Work

In the future, we will expand the technique to deal with non-polyhedral objects. The hash table lookup correspondence approach and the algebraic resultant approach to the least squared error problem are both generic solutions to polygons and extendible to non-polygonal models. The technique will be broadened to include the class of generalized polygons, where each edge is linear or a circular arc.

8 Conclusion

In this paper we showed the feasibility and potential for RISC robotics which tries to solve general problems by introducing tailored sensing devices and efficient algorithms. This method localizes a specific subclass of objects, partially solving the harder problem of general localization. This technique uses robust, cost-effective sensors which are already found in manufacturing applications, and the theoretical framework can be used with existing light beam sensors.

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 upgrading the RobotWorld environment, Justus Grimalda for building the prototype crossbeam sensor, Steve Burgett and Joe Gavazza for designing the crossbeam sensor apparatus, Dr. Richard Murray for his assistance, and Brian Mirtich for helpful criticisms and suggestions on this report.

Appendix

The resultant M is given in equation 19:

$$\begin{aligned} M = \{\{&2ax_2 + 4ax_2t^2 + 2ax_2t^4, ax_1y_1 + 2ax_1y_1t^2 \\ &+ax_1y_1t^4, ax_1t_1 - 4ax_1t + 2ax_1t_2t - 3ax_1t_1t^2 \\ &+3ax_1t_3t^2 - 2ax_1t_2t^3 + 4ax_1t_4t^3 - ax_1t_3t^4\}, \\ &\{ax_1y_1 + 2ax_1y_1t^2 + ax_1y_1t^4, 2ay_2 + 4ay_2t^2 \\ &+2ay_2t^4, ay_1t_1 - 4ay_1t + 2ay_1t_2t - 3ay_1t_1t^2 \\ &+3ay_1t_3t^2 - 2ay_1t_2t^3 + 4ay_1t_4t^3 - ay_1t_3t^4\}, \\ &\{ax_1 + ax_1t_1t + ax_1t_2t^2 + ax_1t_3t^3 + ax_1t_4t^4, \\ &ay_1 + ay_1t_1t + ay_1t_2t^2 + ay_1t_3t^3 + ay_1t_4t^4, \\ &at_1 - 4at + 2at_2t - 3at_1t^2 + 3at_3t^2 - 2at_2t^3 \\ &+4at_4t^3 - at_3t^4\}\} \end{aligned} \tag{19}$$

References

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

[2] 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.

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

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

[5] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 1992.

[6] Kenneth Y. Goldberg and Matthew T. Mason. Bayesian grasping. In *IEEE Journal on Robotics and Automation*, pages 1264-1269, 1990.

[7] Kenneth Y. Goldberg and Matthew T. Mason. Generating stochastic parts for a programmable parts feeder. In *IEEE Journal on Robotics and Automation*, pages 352-359, 1991.

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

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

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

[11] 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.

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

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

[14] Ed Nicolson. *Talisker Reference Manual.* Berkeley Robotics Laboratory, 1992. Version 2.0.

[15] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[16] Anil S. Rao and Kenneth Y. Goldberg. Orienting generalized polygonal parts. In *IEEE International Conference on Robotics and Automation*, pages 2263-2268, 1992.

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

[18] A. Wallack and J. Canny. Object localization using light beam sensing. Technical Report 92-14, Engineering Systems Research Center, 1992.