[5] A. Wallack and J. Canny. Object localization using finger gap sensing. Technical Report 92-3ESRC, Engineering Systems Research Center, 1992.

having the same convex hull. The light beam scanning algorithm will be able to localize flat and non-convex parts.

The same apparatus (fig 1) was used in combination with hole sniffing to accomplish the informed-peg-in-hole task. The beam sensors were used to localize the hole sniffer, which was then used to accurately find the hole. Then, the peg localized relative to the sniffer by running it through the beams. Thereby software had accurately found the relative position of the hole relative to the peg.

We will also be adding light beam sensors onto grippers of robot modules. This serves two purposes. First, the grippers can pass over objects and then be reoriented to induce a good grasp. Secondly, the light beams can be used to sense extruding parts in the environment, such as pins.

This project is a partial solution to object localization, which is an important subproblem in the areas of automated placement and assembly. This report describes a framework for intersecting the constraints from light beam-breaking sensors. This project localized parts efficiently using robust, cost-effective sensors which are already found in manufacturing applications. Finally, this project demonstrated the feasibility and potential for RISC robotics. RISC robotics tries to solve general problems by introducing tailored sensing devices, and efficient algorithms.

By taking advantage of the geometric constraints, only a few data points are neecessary. The software algorithm identifies and localizes parts in time linear with the number of possible objects, beams, and edges on each object. It is divisible into two subprocedures: using the diameter function to compute a set of consistent orientations, and using the critical points for finding the position of the part, or deciding that the orientation was inconsistent. Localization can even be accomplished in constant time by preccomputing a hash table.

References

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

[2] 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.

[3] W. Eric L. Grimson. On the recognition of curved objects. Technical Report 983, Massachusetts Institute of Technology, 1987.

[4] R. M. Murray, D. C. Deno, K. S. J. Pister, and S. S. Sastry. Control primitives for robot systems. *IEEE Transactions on Systems, Man and Cybernetics*, (to appear).

For lambertian surfaces, the radiance on C and D is θ^2 and θ'^2 respectively. For specular surfaces, we can conservatively assume that all of the light on the patch reflects into the receiver; therefore the radiance on C and D is θ and θ' respectively. For lambertian surfaces, the reflected light falls off as $\frac{1}{length^2}$ and for specular surfaces it falls off as $\frac{1}{length}$.

4.3.3 Cross Reflections

Cross reflections might become a problem if the beams are not modulated at different frequencies. In this case, it is possible for one beam to be reflected into a different beams sensor.

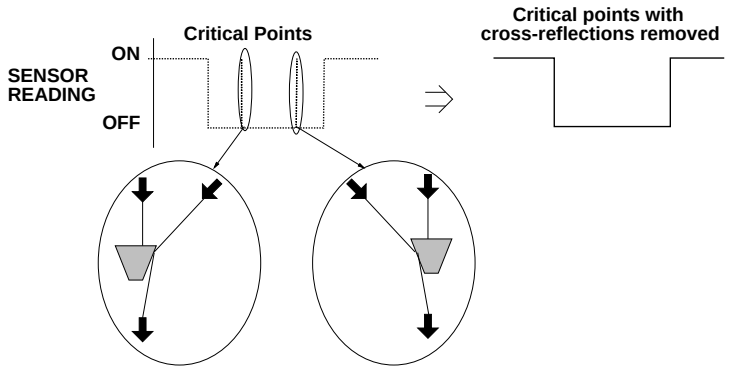


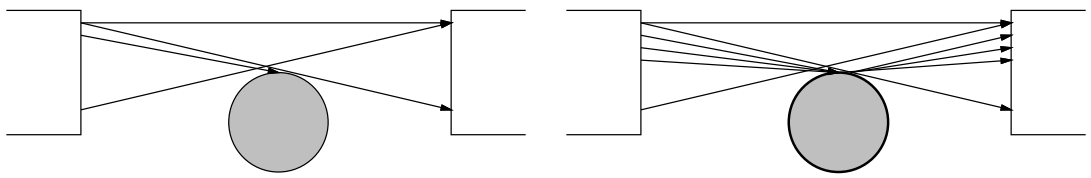
Figure 16: Cross reflections might occur if the beams are not modulated at different frequencies

5 Discussion

The central focus of this line of research is to enumerate some general problems in automated manufacturing, and to solve them using the same strategy as this project. The RISC philosophy is to design a specific sensing apparatus, and a tailored algorithm suited to a particular set of tasks. The goal is to produce a sufficient set of small, efficient, accurate systems which cover many different problems. But the most important idea is that the sensing devices will be cost-effective in industry. We are currently working on other RISC robotics, to identify and localize parts as well as to accomplish informed-peg-in-hole insertion.

We have already developed an algorithm which will scan objects with light beams, but we are waiting for a tailored sensor. The technique involves scanning an object along a few parallel lines, but the sensed data points will be very precise. The reason we are developing a different algorithm/sensor is that this project's light beam breaking approach is not suited to flat parts, nor will it distinguish between objects

the beams were broken. Figure 14 depicts the extreme cases: completely lambertian or completely specular.



Lambertian Object Specular Object

Figure 14: The observed intensity depends upon the reflectivity of the object

The amount of noise is related to the widths of the beam emitter and the beam detector. Figure 14 shows that the detector will receive 50 percent more light in the specular case than in the lambertian case. One example of "noise" is that the specular object will "break" the beam at a later point than the lambertian object since the emitter's output depends upon thresholding the detected intensity. One method of handling this "albedo" noise is to seperate the emitter and detector, ie. increase the span of the beam. Albedo noise drops off linearly with the length of the span (see figure 15).

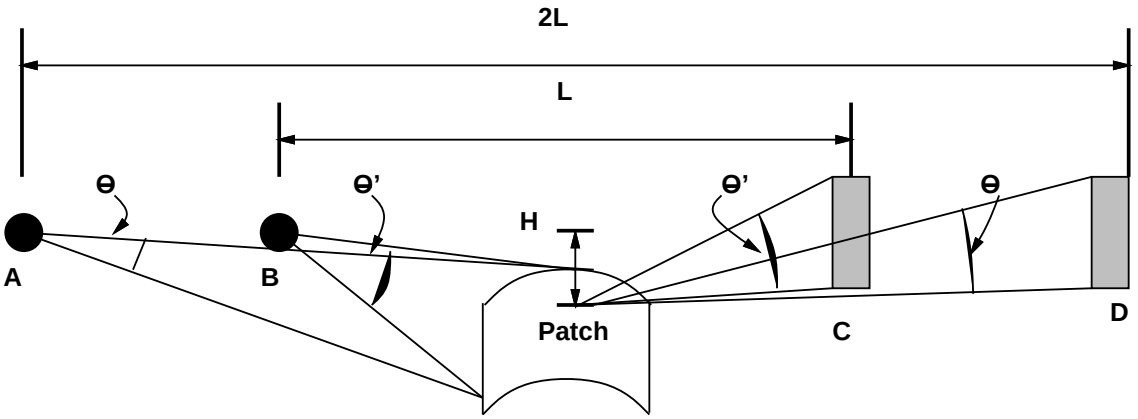


Figure 15: A, D represent a configuration where the emitter and receiver are $2L$ apart, B, C are L apart. A and B are points on different emitters, and C and D represent different receiver sensors. The irradiance of the patch goes inversely as the length, and for specular surfaces the radiance goes off inversely as the length, and for lambertian surfaces the radiance goes off inversely as the square of the length

$$\theta = \arctan \frac{H}{L} \rightarrow \theta = \frac{H}{L} \tag{7}$$

$$\theta' = \arctan \frac{2H}{L} \rightarrow \theta' = \frac{2H}{L} \tag{8}$$

$$Irradiance(From(A)) \approx \frac{H}{L} \qquad Irradiance(From(B)) \approx \frac{2H}{L} \tag{9}$$

4.2 Results

This experiment was run twice for each height, and the technique did indeed compute positions which outlined a circle. The center of that circle, ie. the average of the measured positions, was less than 0.001 centimeters from the origin ($\frac{1}{2000}$") for each of the heights. The standard deviations of the radii were smaller than 0.001 centimeters for each of the heights. The two observations for each height and orientation differed at most by 0.002 cm.

The standard deviations due to error would fall off as $\sqrt{N}$. Another estimate for the error for each observation would be to multiply the standard deviations of the radii by $\sqrt{8}$. This calculates the error for each observation to be between 1 and 2 mil.

4.3 Limitations of the Actual Sensors

An ideal beam sensor would use an infinitely small beam and respond immediately when the beam is crossed. The actual beam sensor differed from this ideal because its beam had a finite size, because the output was decided by thresholding the amount of light incident on the receiver, and because there was a delay associated with the sensor. The finite size of the emitter and receiver imply that various orientations are represented, but a longer beam span restricts the set of orientations incident on the receiver. The thresholding indirectly affected the beam sensors. These non-ideal characteristics combined to affect the sensor values in the following ways: crosstalk, "albedo" noise, and cross reflections.

The best solution to all of these problems is to increase the beam span and to gange the beams out of phase. Placing the emitter and receiver farther apart improves albedo noise, but this is limited by the straightness of the beam. Ganging the beams out of phase would solve any problems associated with using multiple apparati close together.

4.3.1 Crosstalk

The beam sensors, which we used, only came in three different modulated frequencies, but one type of sensor only worked with short spans, so we were only able to use two different frequencies. The beam breakings were more difficult to detect when the beam receiver was collecting photons from other sources. We tried to eliminate "crosstalk" by using a different frequency for the middle beam.

4.3.2 "Albedo" Noise

The object's reflectivity affects the sensed diameter because a specular surface can reflect light towards the receiver whereas a lambertian surface cannot. Since the algorithm is trying to identify the object, it does not know the object's albedo, therefore, the critical points do not exactly correspond to the points where the exact centers of

respond to accurate measurements. This section describes the experiment and the results. The effects of the sensors' limitations are also discussed in this section.

The technique identifies and localizes objects will not succeed if two different objects induce similar beam sensor readings. The probability of such an event should be relatively small for three beams, because each object has three degrees of freedom, and each critical point contributes two constraints. Therefore, the probability of inducing similar readings should decrease by a factor of $(\frac{sensor\ \ error}{object\ \ size})^2$ for each additional beam.

4.1 Experiment

This experiment was designed to measure the repeatability of the output of the algorithm. The repeatability of the algorithm is more stable than the repeatability of any single sensor because the algorithm combines the constraints from all of the sensors.

The peg in figure 13 was attached to the end effector of a RobotWorld module, directly aligned with the centering pin. In order to check all directions and heights and albedoes, the peg was passed through the beam sensor apparatus at different orientations and different heights.

The end-effector's centering pin is not directly under and aligned with the module's center of rotation; the technique should find that the peg's circular cross-section should move around the module's center of rotation. The different orientations evenly spread over the range of possible orientations (ie, $0^o, 60^o, 120^o, 180^o, 240^o, 300^o, 360^o$). Therefore, the technique should measure positions that lie on a circle with respect to the center of the module. Furthermore, if the beams were measured using that module, the center of the measured positions should be the origin. The center of these measured positions, and the standard deviations of the radii, were computed for ten heights and eight orientations.

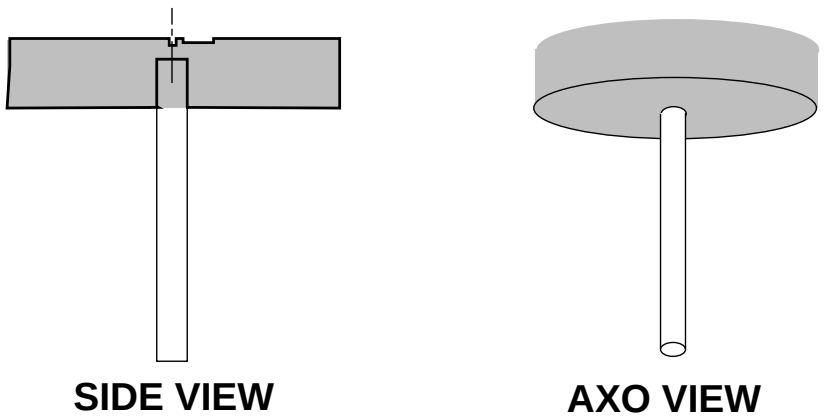


Figure 13: A $\frac{1}{4}''$ diameter peg such that the peg is directly in line with the end-effector's centering pin

3.9.2 Recovering the Pose from the Normalized Pose

The algorithm computes the *relative* position from the normalized data points. The relative position is the distance between the actual x, y position and the direction of travel. The actual x, y position is constructed by adding the offset back to the *normalized* position. The reason for normalization is that the dimension of the set of possible positions was reduced from 3 (x, y, θ) to 2 $(distance\ \ from\ \ direction\ \ of\ \ travel, \theta)$.

3.9.3 Constructing the Hash Table

The localization technique is used to generate the hash table using a brush-fire algorithm. Start with a seed "data point" which corresponds to an actual pose. Perform the localization algorithm on its neighbors and all of their neighbors, until the whole set has been covered. The hash table should assume an *epsilon* amount of sensor error to handle roundoff errors. This approach will take $O(size\ \ set)$.

3.9.4 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. This choice impacts on concurrently using hash tables for different objects.

Incorporating a large sensor error into the hash table has the effect of enlarging the table, which might make the table too big to fit in main memory. If the hash table needs to be paged, then the constant time lookup has to wait for a disk access, which may be a thousand times slower than a lookup. Alternatively, the program would need to search all of the data points within an error box. With six axes, an error box of ± 1 would require 729 table lookups.

4 Experiment and Results

The technique correctly identified an object from a library of three different objects (a $1''$ hexnut, a $1\frac{1}{4}''$ hexnut, and a $\frac{3}{8}'' \times 1''$ rectangle). The correctness of the algorithm could be exhaustively tested with a large library of parts, but that would not present any insight whatsoever. It is already known that the technique cannot distinguish between different objects which happen to have similar critical points, and that it could identify and localize parts in linear time with respect to the size of the vocabulary of objects.

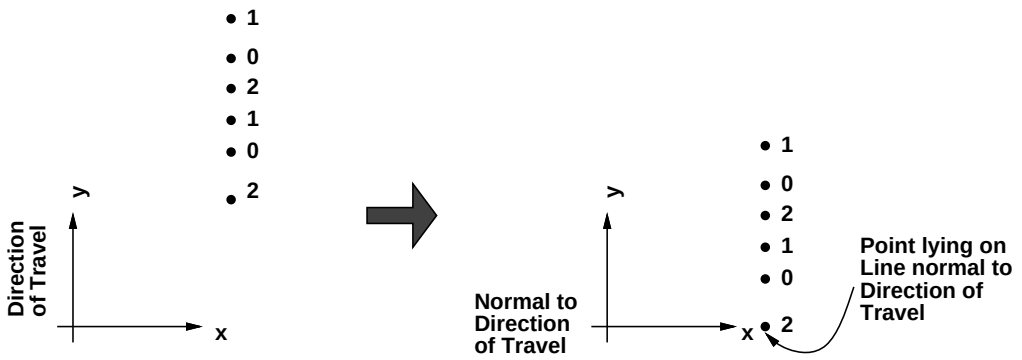
The algorithm's correctness and validity depend upon the light beam sensor's accuracy. Therefore, the purpose of the experiment was to determine the repeatability of the beam sensor readings. In this case, we can assume that repeated values cor-

3.9 Localizing Objects In Constant Time

Objects can be localized in constant time using a precomputed hash table. Data points are normalized to reduce the number of hash table entries. The hash table emulates a $n-1$ dimensional table indexed by discretized measurements; it contains a two dimensional sub-surface of non-zero entries. The table is constructed off-line by using a brush-fire algorithm and the forward algorithm discussed. The normalized data points are used as indices into the hash table, and the position is recovered from the hash table entry. The advantage of creating this hash table is that it enables constant time localization; the disadvantage is that the table becomes useless if any of the beams change slightly.

3.9.1 Normalizing the Data Points

The data points are normalized in order to reduce the size of the hash table by combining *pseudo-redundant* entries into a single hash table entry. The absolute positions of the data points along the direction of travel are *pseudo-redundant* because the algorithm only uses the relative positions to identify an object. The data points are normalized by shifting them along the direction of travel until the dot products of all of the points with the direction of travel is positive and one of the dot products is zero (it lies on the line normal to the direction of travel). For example, the points in figure 12.



$$\begin{bmatrix} (7.559 \;\; 49.125) & (7.559 \;\; 50.432) \\ (7.559 \;\; 49.625) & (7.559 \;\; 50.925) \\ (7.559 \;\; 48.152) & (7.559 \;\; 49.824) \end{bmatrix} \rightarrow \begin{bmatrix} (7.559 \;\; 0.973) & (7.559 \;\; 2.280) \\ (7.559 \;\; 1.473) & (7.559 \;\; 2.773) \\ (7.559 \;\; 0.000) & (7.559 \;\; 1.672) \end{bmatrix}$$

Figure 12: Normalizing a set of data points in the Y-direction

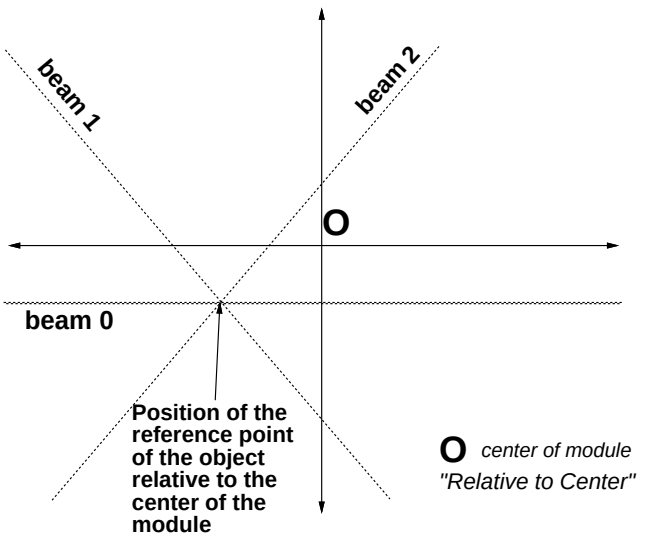


Figure 11: Lines in RCOM space intersect at the object's reference point

3.7 The Relationship Between the Sensor Accuracy and the Resulting Accuracy

The diameter function is a collection of sinusoidal components. When a sinusoidal components is locally maximum, its derivative is very small. In order to compensate for the sensor inaccuracy, the angle range consists of any angles which are consistent with any diameter in Θ . The sinusoids are cosines at their maxima, and $\cos = 1 + \frac{x^2}{2} + \ldots$, the uncertainty of the angle range is related to the sensor accuracy by a factor of a **square-root**.

3.8 Dealing with Uncertainty

This project dealt with uncertainty by using a conservative estimate of the sum of all the uncertainties, called δ. Uncertainty was incorporated into determining the set of consistent angle ranges, and verifying if the set of lines (in the dual) intersected at a single point.

The uncertainty arises because the sensing system is imperfect. This is due both to the hardware and the software. The beam sensing hardware has a number of undesirable features: delay, hysteresis, non-rigid construction, a non-infinitesimally small beam. The sensing software also has problems: the latency associated with polling, the asynchronous communication between the module controller and the polling software. The aim of this project is to overcome such uncertainties, but first they must be recognized.

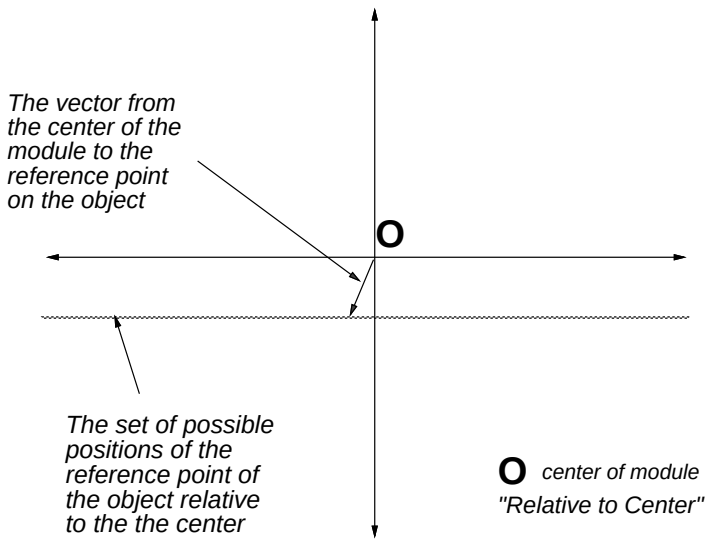


Figure 10: The set of consistent positions of the object's reference point are viewed in RTC-space (relatve to center) because they can be easily intersected

Using the critical point, in relation to the position of the beams relative to the center of the module, compute the line of possible offset to the reference point of the object (as shown in figure 9). For each of the critical points, generate the line of possible offsets to the reference point of the object. These define lines in a space which is relative to the center of the module (RTC - Relative to center space, see figure 10). If all of the lines in RTC space intersect at a particular point (given some slack for sensing uncertainty), then that intersection point is the true relative position of the reference point of the object (figure 11). Lines intersection is done using the least squares algorithm for the dual.

3.6 Locating the Beams Very Precisely

If the beams are not calibrated very precisely, then this procedure is correspondingly just as inaccurate. Therefore, it is very important to try to calibrate the beams as accurately as possible. And furthermore, since this technique provides such high accuracy, it makes sense to determine the relative off axis pitches of the RobotWorld modules because the RobotWorld control system is accurate to 10^{-3} cm (this calibration technique has been implemented but experiments have not been done yet).

The calibration tool was attached to the end effector of the RobotWorld module. The tool was passed through the sensor in both directions and the critical points were averaged, in order to average out the hysteresis in the beam sensors. Then, the line of the beam was computed by using a least squares algorithm on all of the critical points for the beams.

This procedure efficiently determines the object's position, relative to the center of the module, and it determines whether the orientation is consistent with the detected critical points. This heuristic assumes that the positions of the beams have been determined to high precision. It takes all of the critical points, the orientation of the object, and the set of extremal pairs of vertices for each diameter.

First, the pairs of vertices are rotated to determine which vertices were touching the beams at the critical points. The point on the object which contacted a beam, and the position of the module at that critical point simultaneously constrain the object's position relative to the center of the module. Consider a reference point, say the origin of the coordinate frame that the object was defined in; each beam critical point constrains that reference point to lie in some line parallel to the beam (as shown in figure 8).

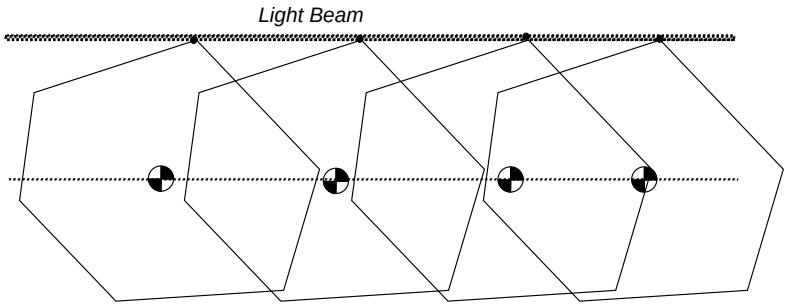


Figure 8: The module's position at the time of the critical point constraints the position of the reference point of the object to lie along a straight line parallel to the beam

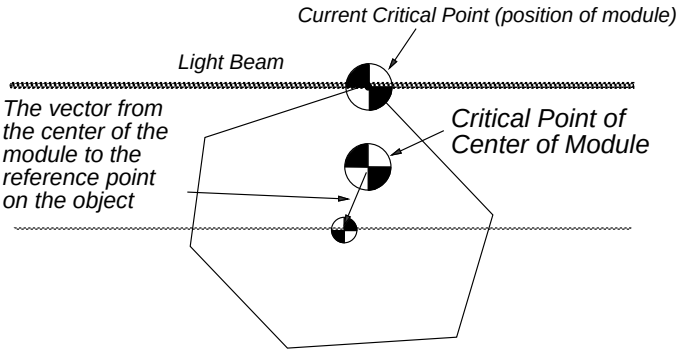


Figure 9: The critical point is viewed relative to the position of the module when the light beam is directly below the module's center

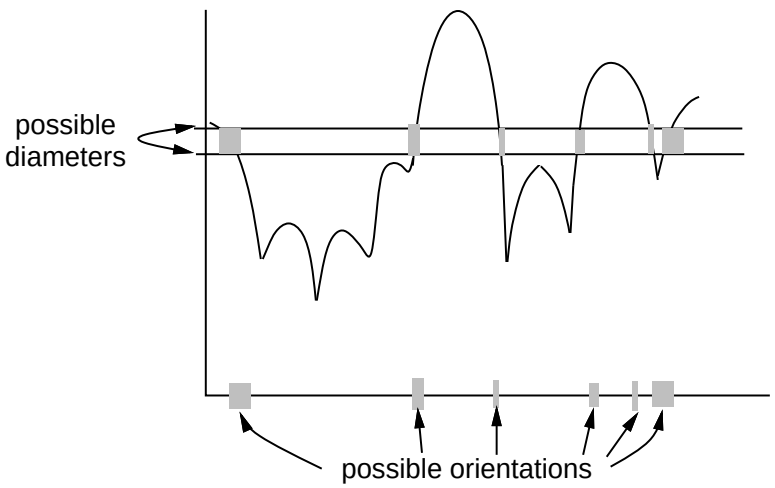


Figure 7: $\mathcal{D}_f^{-1}(\Theta(sensor, uncertainty))$ is the set of possible orientations consistent with the sensed diameter

$$\mathcal{D}_f^{-1}(\Theta) = \bigvee_{vertex_pairs} peak_angle \pm \arccos(\frac{\Theta - peak_plateau}{peak_distance}) \tag{6}$$

3.3 Intersecting $\mathcal{D}^{-1}$ from Each Beam

The angle ranges for each beam can be intersected efficiently because the angle ranges are all ordered in θ. Simply scan down θ in each set ($\mathcal{D}^{-1}$), and intersect the ranges from all of the beams *the different light beam directions are taken into account by offsetting the angle ranges $\mathcal{D}^{-1}$ for each beam.*

3.4 Using Lookup Tables for $\mathcal{D}^{-1}$

A lookup table could be used instead of scanning through $\mathcal{D}$ for every diameter It would need to map: $(diameter) \rightarrow (\mathcal{D}^{-1}(diameter))$. In the average case, this will be a constant time lookup, instead of a linear search. The only cost is to compute the table once and store it.

3.5 Determining the Position and Validating the Angles

This subsection will explain the need for a complementary constraint heuristic, as well as describe the heuristic. Consider our experimental apparatus consisting of three beams; they generate six constraints, ie. six data points. Since the diameter function only uses the relative positions, it only accounts for three of the data points. Furthermore, consistent diameters are not definite proof of the object's orientation, because the same diameters correspond to many different polygonal objects.

Computing the intersection points is equivalent to solving the following formula for each adjacent pair of sinusoidal components:

$$plateau_1 + peakDist_1 \cos(peakAng_1 - \theta) = plateau_2 + peakDist_2 \cos(peakAng_2 - \theta) \quad (5)$$

3 The Algorithm

The algorithm takes as input the critical points that result from passing the object through a set of beams, or passing the beams past an object. The critical points are the positions of the robot at the instant the object begins to break the beam and the instant the object no longer breaks the beam (the points when the beam sensor's output changes).

The algorithm for determining the object's pose from the critical points is divisible into two subprocedures. The first procedure generates the object's set of possible orientations by using the diameter function. Then, each plausible range of angles is verified in the second procedure, which also determines the relative position of the object. The reason for two seperate heuristics is that the diameter function is an efficient tool, but it's two major drawbacks are that it only utilizes half of the data (the difference between the critical points), and it does not generate the relative position.

The algorithm is given below:

1. Compute $\mathcal{D}_f^{-1}(\Theta(sensor, uncertainty))$ by scanning through the sinusoid components of $\mathcal{D}_f$ (see figure 7)
2. Intersect the sets by scanning down θ (offsetting θ for each beam)
3. Determine the position of the reference point for each of the angle ranges from the previous step. This will also validate whether the orientation was valid

3.1 Computing the Diameters from the Critical Points

The diameters can be straightforwardly extracted from the critical points. Call the robot's direction of travel $\vec{A}$, and the light beams direction $\vec{B}$. The diameter is the distance divided by the cosine between $\vec{A}$ and $\vec{B}$, or rather $\frac{distance}{\vec{A} \circ \vec{B}}$.

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

The simplest method for computing $\mathcal{D}^{-1}$ is to scan through all of the sinusoidal arc components in $\mathcal{D}$. The range of possible diameters (Θ, which incorporates sensor uncertainty) are checked against each sinusoidal component:

$$\mathcal{D}_f(\theta) = peak_distance \cos(\theta - peak_angle) \tag{2}$$

2.4.2 Handling Rounded Corners

The algorithm can also handle polyhedral parts with rounded corners. The diameter function is still related to the "extremal" vertices when those corners are rounded, by the same argument. Consider the object with rounded vertices in figure 6, the relationship differs only slightly.

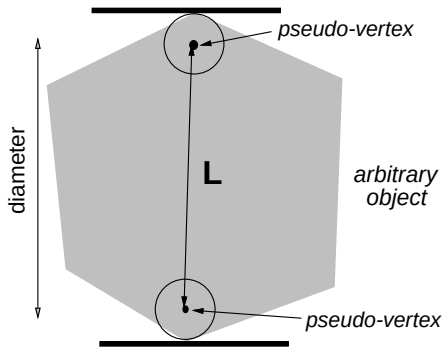


Figure 6: A *rounded* vertex vertex contact

The *rounded* component is analogous to a constant offset, and the remaining L vector still contributes a sinusoidal component. In this case, L refers to the vector between the centers of the two *vertices*. The equation for vertices can be extended by providing a space for rounded corners; vertices are equivalent to rounded corners with radius 0.

$$\mathcal{D}_f(\theta) = peak_plateau + peak_distance \cos(\theta - peak_angle) \tag{3}$$

2.4.3 Collecting Sinusoidal Components into the Diameter Function

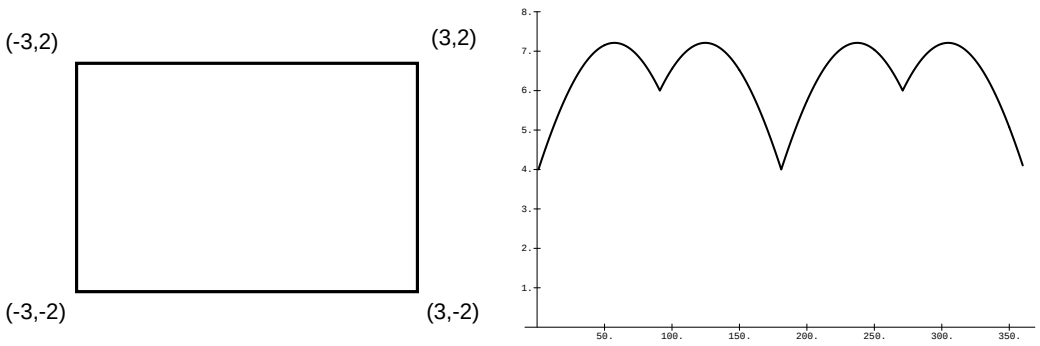
The diameter function, which is the maximum of all of those sinusoids, consists of portions of sinusiodal arcs (as shown for a rectangle in figure 4). It can be constructively shown that there are only $2n$ pairs of vertices (sinusoidal arc components). The diameter is a continuous function, and its derivative is piecewise continuous. The only problem left is to determine the intersection points of all of these sinusoidal components.

Given an approximate construction of the extremal pairs of vertices (in which redundant pairs may be included), we only need to compute the intersection points of the sinusoidal components with *orientationally* adjacent peak angles. Furthermore, the structure of all of the sinusoidal components is:

$$plateauDistance + peakDistance \cos(peakAngle - \theta) \tag{4}$$

points in any direction are its vertices. And equivalently, the diameter function can be thought of as the maximal distance between any pair of vertices along any projection direction. Each pair of vertices corresponds to a vector, and the magnitude of its projection upon an axis is the cosine of the angle between the vector and the projection axis. The method for transforming pairs of vertices into their equivalent sinusoidal representation in the diameter function domain is as follows:

2.4.1 Handling Sharp Corners



Rectangle Diameter Function

Figure 4: Diameter Function of a 2:3 Rectangle

Let L be the vector from one of the *extremal* vertices in the object's $cross-section$. $\mid L \mid$, and $\angle L$ are alternatively referred to as: $peak_distance$ and $peak_angle$ (see Figure 5) because the diameter function has its peak (maximal) value when θ is equal to $peak_angle$. The peak distance is the maximal distance between the two furthest points on the object (in some direction), and the peak angle is the angle of that line.

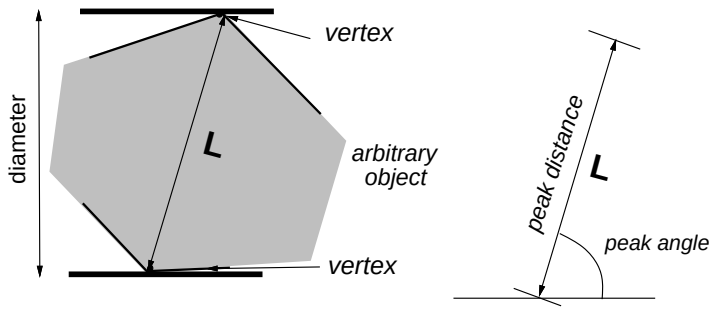


Figure 5: The vertex vertex contact of a polygonal object

$$diameter(\theta) = \mid L \mid \cos(\theta - \angle L) \tag{1}$$

orientation of the unlocalized object is unknown, the original choice of axis is arbitrary and irrelevant.

  - $\mathcal{D}_{object}(\theta, \phi)$ 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. (Therefore, Φ is fixed)

- $\Theta(sensed, uncertainty)$ is the range of values $[sensed - uncertainty, sensed + uncertainty]$, which are consistent with the sensor reading.

- $\mathcal{D}^{-1}_{object}(\Theta(sensed, uncertainty))$ is defined as the set of orientations consistent with $\Theta(sensed, uncertainty)$. This set is an ordered list of sinusoid components (with respect to some origin).

2.2 Reason for Using the Diameter Function

The algorithm, which is the original content of this project, is designed to efficiently intersect geometrical constraints with sensed data. The diameter function is a good representation because it is a one-to-one "invertible" mapping from the orientation to a value that can be found experimentally. The diameter function is computable in linear time, and leads to the task of intersecting ranges of orientations which is easily done.

2.3 Simplifying Assumptions for Diameter Function $\mathcal{D}$

Two basic assumptions will simplify the task of computing the diameter function. First, the light beams can only register an object's convex hull, because they cannot "see" concavities in the object. Secondly, the object will initially be at rest, therefore resting on a planar face of its convex hull.

The diameter function does not have to be exhaustively computed for all orientations (θ, ϕ) because of the fact that the object is initially at rest, stably on the table. The object's only other degrees of freedom are the face it was resting on, and its position and orientation. Since the beams are at a constant height above the table, only a specific cross-section will be registered. The pose of the object is equal to the pose of the cross-section. Therefore, we only need to determine the position and orientation for all of the cross-sections corresponding to each face the object was resting upon.

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

The diameter function is intrinsically equivalent to the set of extremal points for each projection direction. Similar to linear programming, a polygonal object's extremal

1. The object to be identified cannot be flat
2. The object to be identified cannot be translucent
3. The object must initially be lying on a flat surface of its convex hull
4. Items with the same convex hull cannot be discriminated by this technique
5. Items with the same cross-section at the sensed height cannot be discriminated

The next section will give the theoretical, and geometrical framework for the algorithm. The third section will describe the forward algorithm, its susceptibility to uncertainty, and the hash-table approach. The fourth section will discuss the limitations due to the actual beam sensors, describe the experiment, and give the results. We will discuss this method's benefits and put the results in perspective in the conclusion.

2 Theoretical Framework

This section is divided into three parts: describing the notation, stating the assumptions about the diameter function, and diagramming the diameter function.

2.1 Notation and Definition

- The *cross-section* of an object is a slice through an object at a particular height (the exact height will be explained later). Cross-sections are used because that is the only part of the object which will matter when it is passed through a set of light beams at a particular height.

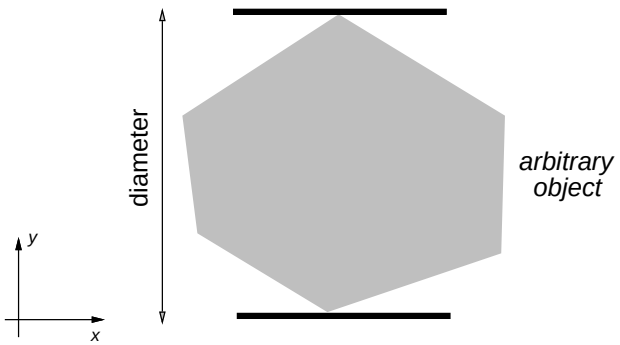


Figure 3: The diameter of a polygon

- The *diameter* of an object is the width of the projection of the object along an axis. In figure 3 the diameter is measured in terms of the y-axis. Since the

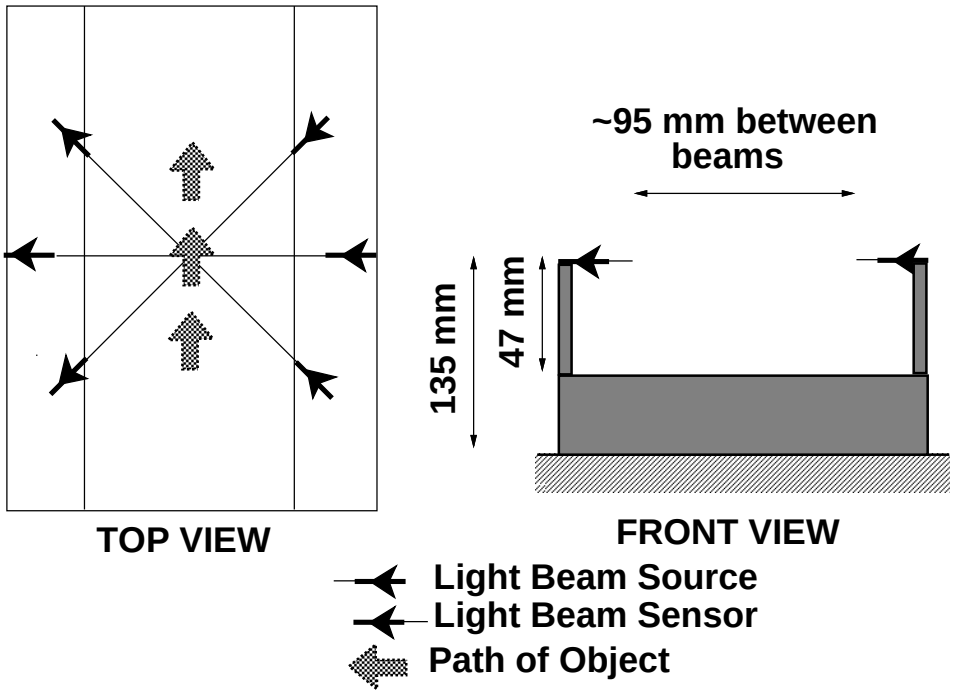


Figure 1: The configuration of the Light Beams

the data is post-processed to determine the object's position, thereby validating each orientation. This technique takes advantage of the high precision of the RobotWorld system in the U.C. Berkeley robotics laboratory, which can be controlled to around one mil (one thousandth of an inch)[4].

The philosophy of RISC robotics is to make very effective specialized techniques for different classes of problems, and then solve problems using one or a combination of these techniques. This technique was designed for a specific class of objects, and performs very well on that class; its basic constraints are:

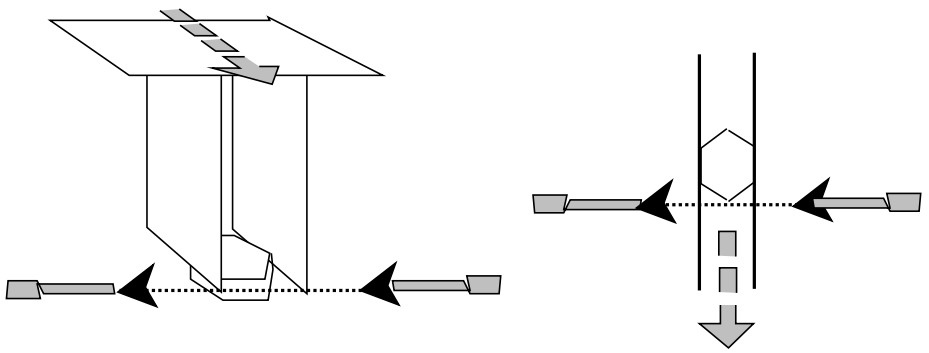


Figure 2: A *critical* point occurs when the object first breaks the beam of light, or when the beam first reconnects

Furthermore, the light beams also serve to calibrate the RobotWorld environment to 1 mil.

1.2 Previous Work

Goldberg laid the foundations for this technique when he invented the notion of *diameter function* which is central to this algorithm. It was introduced in [1] to model parallel jaw grasps in his probabilistic parts orienter. [2] stochastically generated plans for orienting polygonal parts. Another reference was [3] which used interpretation trees to overcome uncertainty and account for sensor and model error. This technique also stems out of previous work by Canny and Wallack [5] which determined the pose of an object and computed an optimal regrasp using the finger gap.

This project bridges a gap between automatic part orientation and object localization, in that it effectively solves the problem of orientation using localization. The major reason why researchers sought a sensorless automatic part orienter was that they felt uncomfortable using expensive and fragile equipment in a manufacturing environment. The task is made much simpler if we are allowed simple, robust sensing devices and corresponding algorithms.

1.3 Overview

The main problem that we are attempting to solve is assembly tasks, and the strategy is to very quickly and precisely determine the position and orientations of objects. As per the RISC philosophy outlined above, this technique only applies to a certain subclass of objects–non-flat three dimensional objects. This technique utilizes cheap, reliable, very precise binary sensors which have already been proven in industry. An experiment was performed using an array of binary light beam sensors in a RobotWorld environment after the sensors were used to calibrate the robot.

An object is identified and localized after passing it through a number of light beams. The light beams are oriented in different directions (as shown in figure 1, and the robot's position is recorded at the critical points (a critical point occurs when the object first breaks a light beam, or the beam first reconnects (as shown in figure 2)). The algorithm computes the object's pose by intersecting geometric constraints with the information in the critical points. The forward algorithm runs in $O(km)$ average linear time and $O(kmn)$ worst case where k is the number of beams, m is the number of objects, and n is the number of vertices on each object. The fastest localization scheme would run in constant time by utilizing a hash table: the computations could be stored in a precomputed hash table which maps critical points to object, pose.

The first idea is that the critical points are related to the object's diameter in the direction of the beam. Each diameter gives rise to a set of possible orientations. These sets are intersected to determine a set of possible orientations for the object. Then,

1 Introduction

This localization method can be used to reduce pose uncertainty in automated manufacturing and assembly. This method quickly and cost-effectively identifies and localizes non-flat parts. The focus of this paper is to significantly reduce the uncertainty with a "better" localization technique. In order to impact the manufacturing sector, we believe that we must produce systems using cost-effective, durable, and accurate components.

When an object is run through the beam sensor, the beam sensor apparatus observes a two dimensional cross-section of three dimensional objects. This reduction in dimension has advantages and disadvantages corresponding to a complexity/generality trade-off. The advantage is that the algorithms are simpler for two dimensions than for three dimensions because there are only three degrees of freedom in two dimensions and six degrees of freedom in three dimensions. The disadvantage is that this technique cannot distinguish between objects with the same convex hull.

Although general vision algorithms can already localizeing objects, these methods are not cost-effective in a manufacturing environment: expensive video cameras are expensive and relatively fragile. Furthermore, general vision algorithms identify and localize objects less precisely than the method outlined in this report.

1.1 Simple Sensing Techniques (RISC Robotics)

J. Canny and K. Goldberg invented the notion of RISC Robotics (Reduced Intricacy Sensing and Control) as a focusing point for research in simple sensing techniques. Instead of developing sophisticated "general" algorithms, we believe in devoting a section of the workspace to "tailored" algorithms and sensors. Compared with complex algorithms, such as general anthropomorphic vision, simple algorithms are cleaner and faster, and the instruments are more robust. The current aim of RISC robotics is to improve and automate manufacturing by designing elegant solutions to general problems in manufacturing. This project is an example of, and demonstrates the feasibility of this approach.

The RISC philosophy is to develop "tailored" algorithms and sensors instead of developing sophisticated "general" algorithms. Simple algorithms, like this one, are cleaner, faster, and use more "robust" sensors than complex "general" algorithms (stereoscopic vision). RISC also emphasizes using intelligent algorithms and "dumb" (cheap, robust, unbreakable, precise) sensors. The current aim of RISC robotics is to focus on automated manufacturing and introduce elegant solutions. In summation, the RISC philosophy is to intelligently blend sensing and algorithms together.

The notion that simpler sensors are inferior is unfounded. In some respects, beam sensors are superior to video cameras: they are economically superior, more rugged, more accurate (precise to less than one mil), and less likely to fail. This project *identifies an object and determines its pose using only six data points* from the beam sensor.

Object Localization Using Light Beam Sensing

Aaron S. Wallack[1] *John F. Canny*[2]

Computer Science Division
University of California
Berkeley, CA 94720

We implemented a localization method which partially solves the problems of placement and assembly. The idea was to localize objects precisely (1 mil) so that uncertainty was negligible and dead reckoned positions were sufficient. The method was geared towards industry by using reliable, accurate, robust, inexpensive, sensors; light beam sensors have already been used in manufacturing environments. This technique involved passing objects through a set of coplanar beams and recording the robot's position when a light beam was broken or a beam reconnected. The algorithm localized and identified objects with the assumption that they were rigidly held. The advantages of this approach were: the feasibility and precision of the sensors, constant time identification using precomputed hash tables, and worst-case linear time identification algorithm. The disadvantages were: it cannot distinguish objects with equivalent convex-hulls, and does not work on "thin" objects. The performance and limitations of the actual beam sensors affected the design of the apparatus.

[1]Supported by Fannie and John Hertz Fellowship,

[2]Supported in part by David and Lucile Packard Fellowship and National Science Foundation Presidential Young Investigator Award (# IRI-8958577).