Three-Dimensional Sensing and Interpretation

Gerald J. Agin, Martin J. Uram, and Peter T. Highnam

CMU-RI-TR-85-1

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

January 1985

Sponsored by Etefense Advanced Research Projects Agency (DoD); ARPA Order No. 3597, Amendment No. 18; and monitored by Air Force Wright Aeronautical Laboratories, Avionics Laboratory, under contract number F33615-83-C-1023.

Table of Contents

List of Figures

List of Tables

Abstract

The main goal of this project is to produce a representational schema that can be used to compare shape descriptions of solid three-dimensional objects. We are working toward a demonstration to show recognition and determination of position and orientation of solid objects on a tabletop, using a sensor consisting of a light-stripe projector and camera mounted in the hand of a robot manipulator. This report presents the results of a number of activities directed toward that goal: calibration and use of the range finder, a tactile sensor for exploring a three-dimensional domain, a system for identifying pipes in a bin, a method for assessing the similarity of cross-section curves, and pose cluster matching, a new technique for shape matching.

1. Introduction

This report summarizes progress in three-dimensional sensing and interpretation at Carnegie-Mellon University from July, 1982 through June, 1984.

An annotated bibliography of references relevant to this work has been assembled. This bibliography is available as a companion to this report.

1.1 Project goals

The main goal of this project is to produce a representational schema that can be used to compare shape descriptions of solid three-dimensional objects. To demonstrate whether or not we have reached this goal, we perform experiments in recognition and pose determination. We have a variety of objects to be recognized on a tabletop, and a range sensor for scanning them. The data from the sensor are processed and matched against stored prototypes.

We want recognition to be robust. Objects must be recognized in any pose, regardless of the point of view. Recognition should not depend on complete data; if only a portion of the object is visible, recognition should proceed with the portions that are available. Minor errors in sensing or low-level processing routines should be tolerated! To achieve this kind of robustness we believe matching must be done in a way that considers all the available data simultaneously and comes up with one or more interpretations that agree with the preponderance of the evidence. We rule out methods that focus on one or two key features, because if these features are missing, occluded, or erroneous, matching cannot proceed.

Recognition must be model-based. There must be a means of describing new prototypes to the system without reprogramming. These new models can come from "training by showing" or from links to a computer-aided design data base. However, CAD models are not necessarily the kind of models we want to use for recognition. The requirements of computer vision are very different from those of design, and some very different sorts of properties need to be represented.

How may different descriptions of the same object be compared? The strategy we use is twofold:

- choosing local features that are invariant to viewing angle, and
- incorporating as many local features as possible in our prototype models, without regard to redumlancy or overlap of multiple feature descriptions.

Matching can proceed on the basis of whatever subset of features correspond between image **and** prototype* without regard to '"completeness[1]* of any description or set of features,

Local features are an important part of our approach. A local feature denotes a significant region of an image or a three-dimensional scene. Exmaplies of local features in two-dimensional images include corners, edges, holes, or special markings. In three dimensions thay can include flat planes, edges between surfaces, high points and indentations, cylindrical regions, regions of elongation, and the like. To be useful for the purposes of recognition, local features must be discoverable by processing routines operating on visual data, and they must be independent of the viewing angle or the object's pose. Two kinds of information can be associated with a local feature: classification information such as size and type of feature, and pose information that describes the position and orientation of the feature.

We choose to acquire range data in a random-access fashion using a sensor mounted on the hand of a robot. We believe there are two main advantages to this approach from a research point of view, as will be described below.

The first advantage of our approach stems from flexibility of viewpoint: it forces us to think in three dimensions. All previous approaches to range data interpretation have dealt with densely-scanned scenes observed from a fixed point of view. This usually leads to a practice of storing the range data in a two-dimensional array as a depth map. We feel this leads to a kind of two-and-a-half-dimensional analysis that is heavily dependent on camera coordinates and that considers depth as a single-valued function of the raster scan coordinates. By using all six degrees of freedom of the robot we are able to move around and view the scene from different points of view. This forces our representation to be viewpoint-independent.

The other research advantage of our approach comes from random-access data gathering. It focuses attention on the high-level representation issues. After a small number of exploratory scans, further data gathering must be in response to specific recognition strategies that depend on the the data gathered so far. We must form one or more hypotheses about the contents of the scene and act to verify one or disambiguate among several hypotheses. Thus the program must contain some knowledge about its own knowledge; it must take note of what is known and what is unknown and use this do guide further data acquisition.

No claim is made that our strategy is optimal for any particular task. We are aware that physical motion of the robot to reposition the range scanner is time-consuming. Considerable speedup could be made by altering the scanning method. The issue is not speed, but whether we can accomplish the task at all. We need shape models that are useful for analyzing three-dimensional scenes. No one else has demonstrated a system that can recognize a broad variety of objects by their shape based on prototype descriptions. Once we demonstrate this goal, we can study how to speed the system up.

A convincing demonstration of the system would be to implement a bin-picking algorithm. The "acid test" of our methods would be to pile a number of different objects into a bin and ask the system to recognize them. The system should identify at least one object at the top of the heap and indicate where it should be grasped to remove it.

However, one should remember that bin picking is not the major goal of this project. The major goal is shape understanding: to develop a schema of computer models of three-dimensional objects and a set of routines for dealing with them that can compare shapes. We can expect the result to be applicable to a host of vision tasks, not limited to our scanning methods, perhaps not even limited to direct three-dimensional measurement. The understanding we gain can contribute to all of artificial intelligence.

1.2 Sensing

Our approach has been to use a simple light-stripe sensor for obtaining range by triangulation. The sensor is mounted in the hand of a PUMA robot, so that it can maneuver around objects lying on a tabletop. The main problems to be solved have included developing procedures for rapid calibration of the system, measuring and improving its accuracy, and enhancing the speed of its operation.

Appendix I is a reprint of a paper presented at a meeting in August 1982. Since then we have modified the calibration procedure somewhat, and discovered adjustments to the PUMA hardware that can improve accuracy. The measured accuracy is now in the range of 1 millimeter. Chapter 2 below details these improvements.

We are now concerned with improving the speed of operation of the range-finding system and adapting our procedures for more rapid calibration. We realize that our approach is inherently slow, and for this reason will probably never see actual application, but would like for research purposes to be able to complete our experiments in less time than it now takes. Chapter 2 also mentions our speedup efforts.

A unrelated development has been exploration of the use of a tactile sense to obtain three-dimensional information about a robot's physical environment. This is based on the use of a flexible probe to feel for objects, much like a blind man's cane or a cat whisker. Some work on assembling a device for tactile sensing and on characterizing the device and its response was done in the summer of 1983, and that work is summarized in Chapter 3.

1.3 Modeling

The models used for this project are for the sole purpose of identifying and determining the position and orientation of objects. There is no requirement for graphical rendering of objects, links to a computer-aided manufacturing system, calculating physical properties such as volume and strength, nor for any of the other traditional functions of solid modeling systems. We choose our models only on the basis of their relevance to the vision task at hand.

The primitives of the models are those structures in solid objects that are discoverable by a direct ranging system. As the system explores its environment, it builds up a description of the scene in terms of the primitives it has found. If the same object is scanned at different times in different positions and orientations, we cannot expect exactly the same primitives to be discovered each time, but if we have built our system properly, some common elements will recur. We can describe prototypes in terms of these recurring elements, and compare an unknown object with the prototype. Although these elements do not constitute a complete representation in the sense that a polyhedral or spline-based modeling system does, it is sufficient and appropriate for visual identification and matching.

We restrict ourselves here to considering flat planes and partial generalized cylinders. These are defined in terms of our light-stripe sensing system. A flat plane is a region of a three-dimensional scene in which the images of all light stripes are straight lines, regardless of orientation. A partial generalized cylinder is a region where the stripe images obtained from adjacent, parallel scans are congruent. Connecting parallel scans in the scene gives a direction for the axis of the cylinder.

We are now reliably able to locate circular cylinders in a scene. To demonstrate this capability, Martin J. Uram has put together a demonstration system to solve the visual part of the bin-picking problem using lengths of plastic pipe in a wire basket. That system is described in Chapter 4.

Our demonstrations so far have used objects that may be characterized as right circular cylinders. Now we are working towards dealing with more complex shapes. A crucial issue will be assessment of the similarity between cross-sectional shapes. Peter T. Highnam has done some preliminary work in this area. Cross-section curves, obtained from the light stripe apparatus, are fit with high-order polynomials to smooth them and to eliminate digitization noise. Then, having an analytic formulation for two or more curves, their similarity can be assessed as a function of horizontal and vertical shifts of one curve with respect to the other. A description of the techniques involved is contained in Chapter 5.

1.4 Matching

Our matching effort is centered around a new technique known as "pose cluster matching". In general, local evidence, such as planes and partial generalized cylinders found in range data, can be matched to various portions of a prototype object. Some of these matches will be correct and others will be incorrect. These matches, singly or in various combinations, can be used to infer the position and orientation of an instance of the prototype in the scene. Where the object actually exists in the scene, many different matches will point to the same position and orientation.

This matching technique is elegant and general. It can apply to a broad range of vision problems, including both two- and three-dimensional domains.

To test this technique while waiting for usable three-dimensional data from the light-stripe system, we have applied it to two-dimensional binary images obtained with our Machine Intelligence Corporation VS-100 vision module. The vision module software has been modified to approximate the outlines of blobs with polygons consisting of straight line segments. Using these segments as input to a pose cluster matching program, we have been able to recognize and classify objects from their silhouettes, including partial and overlapping objects.

Chapter 6 explains the pose cluster matching technique in detail, and gives some results applied to the two dimensional domain.

2. Calibration and Use of the Light-Stripe Range Finder

Appendix I is a reprint of a paper entitled "A Movable Light-Stripe Sensor for Obtaining Three-Dimensional Coordinate Measurements" by Gerald J. Agin and Peter T. Highnam, presented at the SPIE International Technical Symposium, August 21-27,1982, in San Diego, California. It represents the state of our efforts when work on this contract started.

The paper of Appendix I gives specific details of the construction of the range finding system, its principles of operation, and procedures for calibrating the system. The following discussion mentions improvements and enhancements to that basic capability, and presumes a familiarity with the material of that appendix.

We intend the range-finding system to be used mainly in a random-access fashion. However, it is sometimes useful for display purposes, or for evaluating the accuracy of the system, to cover an entire scene with a regular grid of parallel scans. An interactive procedure is used for specifying the initial and final positions of the stripe plane, and the number of steps to be taken. All points detected in the light stripe are converted to three dimensional coordinates, and these coordinates are written onto a disk file for later display.

A three-dimensional display capability has been very useful for this project. It is based on the Evans and Sutherland PS300 display system in the Legged Locomotion Laboratory in the Robotics Institute of C-MU, headed by Marc Raibert. The system consists of several processors, storage for vectors and programs, a display screen, a set of glasses for stereo viewing, and interactive devices for user inpyf and output, including a keyboard, eight dials, and ten function keys, each with a programmable LED label. Points to be displayed are downloaded from our Vax through a serial line to the PS3Q0. The *system* has a quite sophisticated programming language for specifying options for display [4]. We have used this programming language to create an interactive display system for examining our data. These programs are also downloaded from the Vax.

The user sees cm the screen a display, in perspective, of the vectors in the light stripe data Be My aitsr this display by rotating the dials: three correspoml to rotation, three to translation, and one to acale. As the user rotates tie dial, me display software automatically updates the perspeci» transform in real time* so that the user perceives the scene rotating at a speed .proportional to the nte at which he turns fie dial. This creates a vivid illusion of three-dimensionality of the data. For a *mm mM* display, stereo viewing is available, *using* special goggles.

Also displayed with the data are a triple of axes labeled *X*, *Y*, and *Z*. These axes serve two purposes: They may be used as a "cursor" to measure distances, and they serve as a center of rotation. One of the function keys is used to toggle between "world" mode and "cursor" mode. In world mode, manipulations of the dials move both the data points and the axes, but in cursor mode, only the axes move. The center about which manipulations of the rotation dials will rotate the points can be altered by moving the displayed axes to the desired center of rotation.

A digital LED readout is constantly updated to present numerically the absolute position of the axes and their length (measured tip-to-tip). Thus there are two ways of using the cursor to measure the distance between two points. One way is to place the cursor midway between the two points, use the scaling dial to alter the length of the axes until it appears to span the distance, and read off the length from the display. A more accurate method is to position the cursor precisely at each of the two points, reading off the exact position for each, and use a hand calculator to obtain the Euclidian distance between the two.

We use this interactive display to evaluate the accuracy and repeatability of the ranging system. There are several kinds of accuracies we might wish to evaluate. The first kind we call "random error", which is mainly digitization noise. This can be estimated by scanning a flat surface and looking at the scatter of points. The second kind we call "directional-dependent error." We believe that this error arises from errors in calibration. If an object is scanned twice with the ranger approaching it from different directions and we notice a shift in the apparent position of the object, the amount of that shift is position-dependent error. A third kind of error might be called "absolute accuracy", but measurement of absolute accuracy is difficult enough that we have made no attempt to quantify it.

We use a white-painted cube for a target, and make repeated scans of it. To measure random error, we rotate the display so that we are viewing one plane of the cube edge-on. This creates a band of vectors on the display, the thickness of which we can measure with the cursor.

We measure direction-dependent error by measuring the amount of apparent motion of the position of one corner of the cube when it is scanned in different directions or from different points of view. This can be measured by placing the cursor at a corner of the cube as seen in one view, reading the actual coordinates, doing the same with the other view's data, and obtaining the Euclidian distance between the two.

When we first were able to measure direction-dependent error, we found misregistrations of up to

one-half inch. We were able to bring these down by some mechanical adjustments to the PUMA arm. In particular, we made a great improvement by adjusting joint 5. The joint 5 actuator motor is connected through a flexible coupling and a long rod to the wrist assembly. This coupling is located in the PUMA forearm, near the elbow, and is accessible by removing a cover plate. The coupling may be loosened, and the rod rotated. Under program control we moved joint 5 to a nominal setting of zero degrees. Then we rotated the rod relative to its coupling to make the axis of joint 4 rotation and the axis of joint 6 rotation coincident. We may verify that these two axes are coincident if a 180 degree rotation of joint 4 together with a minus 180 degree rotation of joint 6 does not produce any change in the orientation of the end effector.

At present, the random errors in our system are about 2 millimeters peak-to-peak, and the direction-dependent errors are about 5 millimeters.

One minor annoyance in dealing with this range-measurement system is that it operates slowly: many seconds are required for each move of the arm. We are not (yet) trying to create a system that operates at factory speeds. However, the experiments we wish to perform require many scans of the environment, and slow response of the system limits the number of experiments we can perform within a comfortable laboratory session.

We can identify several factors which account for this slowness. One is that we deliberately slow down the mechanical speed of the arm to avoid accidental collisions. As we gain confidence with the system, we can bring the speed up, but whenever we try out new motions of the arm, it would be prudent to keep the motions as slow as possible.

Another factor is the verbose communication format dictated by VAL's interaction. In order to specify a new position and orientation to the PUMA, about 200 characters of communication are required over the serial line between our LSI-11 and the PUMA controller. We are examining the interaction and modifying our procedures to reduce the communication requirements. The long-term solution to this aspect is to upgrade our PUMA to use VAL-II.

3. A Tactile Sensor for Exploring a Three-Dimensional Domain

3.1 A review of past and current research

More research effort has gone into the design of touch sensors than into their use for robot guidance. The sensors generally fall into two categories: *force* sensors that measure the overall forces and torques being exerted on an object, and *tactile* sensors to measure local pressure patterns between a finger and an object being grasped. A recent survey [6] provides a good overview. This survey, together with a follow-on [7], contains a thorough bibliography of the field.

Most force sensors are designed to be placed between a manipulator's wrist and its gripper. They work by measuring the deflection of load-bearing elements in response to applied forces and torques. Usually the deflection is sensed by strain gauges, so the amount of actual deflection is small. These sensors can be designed to be rugged and reliable, and a number of commercial units are on the market. An alternative approach to force sensing involves monitoring actuator torques in the manipulator drive.

Effective use has been made of force sensing in certain kinds of assembly operations. Robots have been programmed to slide an object across a tabletop or to turn a crank using force feedback. Insertion is accomplished by nulling contact forces. By transforming sensed forces and torques through a compliance matrix and feeding them back to the manipulator control, a variety of compliant behaviors can be demonstrated. The remote center compliance concept was first developed using this technique before it became a hardware device.

Tactile sensors consist of pressure-sensitive elements, either singly or in regular arrays. Although a variety of sensing mechanisms have been used, the most promising candidates for implementing an "artificial skin" make use of changes in the resistance of a sheet of conductive rubber. High spatial resolution can be achieved by fabricating the sensors directly on a silicon VLSI chip. Problems with tactile sensors include hysteresis, nonlinearity, overload protection, reliability and processing techniques.

Very little use of tactile sensors has been made for control of robots. Several experiments have been performed to recognize objects on the basis of their pressure patterns on a tactile array [2]. In many respects the problem is similar to pattern recognition using vision, except that only the portion of the object in actual contact with the sensor may be sensed. The results of these efforts have not been encouraging.

A special case of tactile recognition involves grasping an object with a multi-fingered hand [3, 9]. The angles of each joint can constitute a vector that can be used to distinguish one object from another, using some rather sophisticated pattern-recognition techniques. Although methods like these have succeeded in some simple cases, they do not appear to be generalizable to real-world objects.

3.2 Active Touch: Integrating Sensing and Manipulation

Humans and animals obtain tactile information about their environment by a process of active exploration. Tactile perception is intimately associated with concepts of three-dimensional shape. Models of objects must be built up by delimiting their spatial extent and determining the spatial disposition of significant portions of the object.

We believe it is wrong to expect much in the way of meaningful results from tactile recognition experiments that seek to classify objects from a single impression on a receptive surface. Very little relevant information can be obtained in this way. A blindfolded or blind human would find it very difficult to identify an object statically pressed into his fingertip or palm. This is not to say that research on tactile array sensors is useless. These devices are potentially capable of identifying edges, projections, and depressions, and of determining local curvature. But higher-level analysis of all but the simplest objects will require higher-level information.

Informal experiments show that a blindfolded human subject, when handed an unknown and unfamiliar object, will run his fingers over the entire object, turning it over repeatedly in his hand, until the shape is, in some sense, "understood." The cues available seem to be numerous, including overall size and mass, surface roughness, thermal conductivity, as well as shape. Subjects tend to focus their attention on indentations and depressions and on edges and ridges.

We would like to emulate this kind of exploratory behavior using robots and touch sensors in our laboratory.

3.3 A Cat-Whisker Sensor

The sensor we use should be appropriate for probing an unknown environment. It should be capable of coming in contact with objects without damage to either the object or the sensor. And it should be capable of giving us as much information as possible about the nature of the contact. It will be mounted as the "end effector" of an industrial robot manipulator.

The ability to contact without damage implies that the system must possess a high degree **of** compliance. In principle, compliance can be achieved through tight closed-loop control of the manipulator using force sensing feedback. But it is not likely that a closed-loop system can be made to operate at adequate speed without a substantial development effort. It is better to rely on mechanical compliance, or "springiness," in the contacting element itself. A possible objection to high mechanical compliance is that it precludes fine control of position. But in our case this is a benefit rather than a drawback because positions are unknown to begin with, and once contact is established the precise position can be solved **for.**

We can achieve our goals with the "cat-whisker" sensor shown in Figure 3-1. The whisker itself is a flexible steel wire. It is anchored to a 6-axis force and torque sensor made by Astek Engineering, Inc. The force sensor is designed to be attached to the end of our Puma arm.

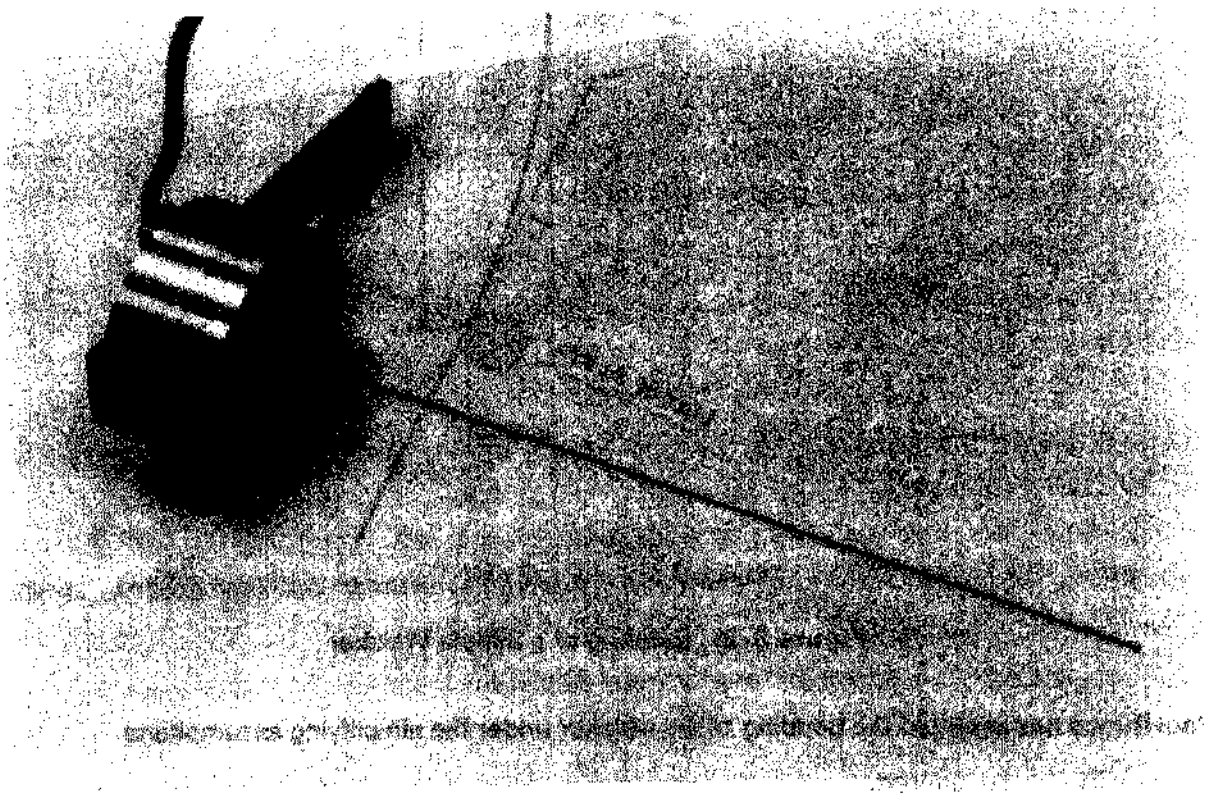


Figure 3-1: A Cat-Whisker Tactite **Sensor**

When the cat whisker comes in contact with an obstacle, the output of the wrist force and torque sensor can be used to determine the point of contact on the whisker. The three force measurements combine to give a vector direction for the total force applied to the whisker. Dividing the magnitude of

the applied force by the magnitude of the applied torque gives the moment arm of the force, which in turn gives a "line of action:" a line in space along which the force is acting. What remains is to find the point of application of that force along the line of action, by taking into account the bending of the whisker. The principle is illustrated in Figure 3-2.

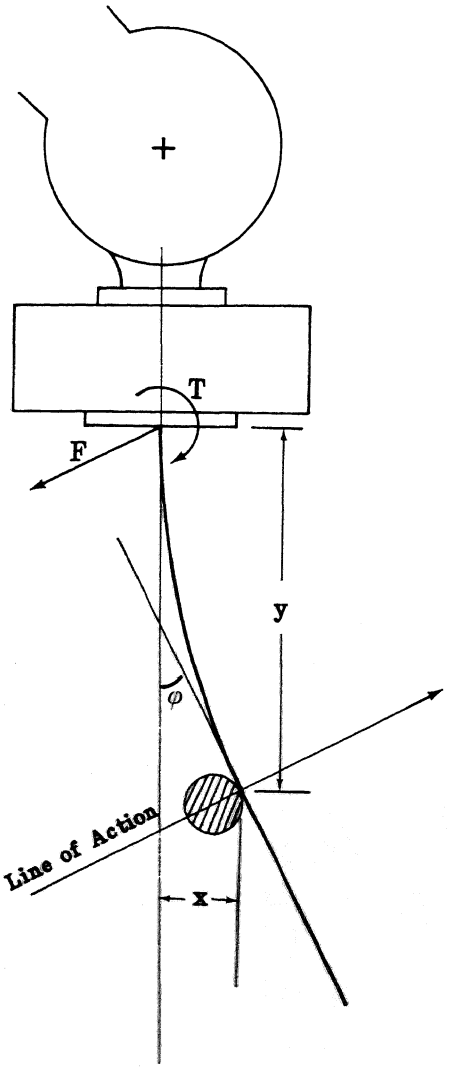


Figure 3-2: Bending of a Simple Whisker

Paul Runco has analyzed the bending of the whisker under the simplifying assumptions

1. gravity may be ignored,
2. the whisker is cantilevered at one end,
3. there is no friction, and
4. the bending radius of the whisker is much larger than its diameter.

The third assumption implies that the tangent to the whisker at the point of application is perpendicular to the direction of force. Let this tangent angle be called φ. Then it may be shown that

$$y = \sqrt{\frac{2EI}{F}}\sqrt{\sin\varphi}$$

$$x = V \wedge \int_0^{\varphi} \sqrt{\sin\varphi}\, d\varphi$$

The Astek force sensor has been interfaced to the LSI-11 controller that also manages communication with the Puma and the Vision Module.

4. ROPIS: Randomly Oriented Pipe Identification System

A difficult problem in manufacturing automation is "bin-picking," where a robot arm reaches into a bin of jumbled three-dimensional parts such as pipes, picks out just one, and presents it to the next operation in the correct orientation.

"Preserving the orientation of oriented parts is the most obvious way to avoid the problem of dealing with a bin of randomly oriented parts. This is particularly the case since commercially available industrial robots can load machines which are supplied in magazines or pallets" [8]. However, the preservation of orientation is often impractical or expensive. "Parts which come from vendors, long term storage or distant warehouses are usually shipped unoriented. The cost of storing or shipping oriented parts, due to low packing density, is usually prohibitive" [8].

The only other solution is to develop a vision system capable of handling randomly-oriented parts. One such system uses two-dimensional vision analysis and a parallel jaw gripper to acquire randomly oriented cylindrical workpieces piled in bins and deliver them to a V-chute which discharges oriented cylinders [8]. It represents an initial step towards giving robots the capabilities needed for future automation applications, especially in small batch production. Cycle times to acquire a cylinder and deliver it to a receiving chute ranged from 8 to 10 seconds when a single supply of one size of cylinders was used. "Unfortunately, the application of the above techniques to robot control has been limited because of the large amounts of computing time and memory space required. In order to be economically feasible for most industrial applications the entire system should cost from $5,000 to $8,000" [12]. ROPIS is a step in this direction.

ROPIS (*R*andomly *O*riented *P*ipe *I*dentification *S*ystem) is a vision system written in the C programming language and designed to run in a UNIX environment on a Digital Equipment Corporation (DEC) VAX 11/750. It uses the triangulation system of G. J. Agin and three-dimensional image analysis to identify cylindrical objects (i.e., pipes) which have been heaped randomly in a wire bin, as shown in Figure 4-1.

In actual operation, the immediate goal of ROPIS is to identify the top pipe of the heap -- the system's "target." This is assumed to be the pipe which can be picked up with the least amount of effort and the least amount of disturbance to the rest of the pile. Since a random-access range-measuring device is being used, ROPIS acquires data as it is needed, rather than scanning the entire environment (i.e., bin) in a predetermined sequence. There are a large number of degrees of freedom

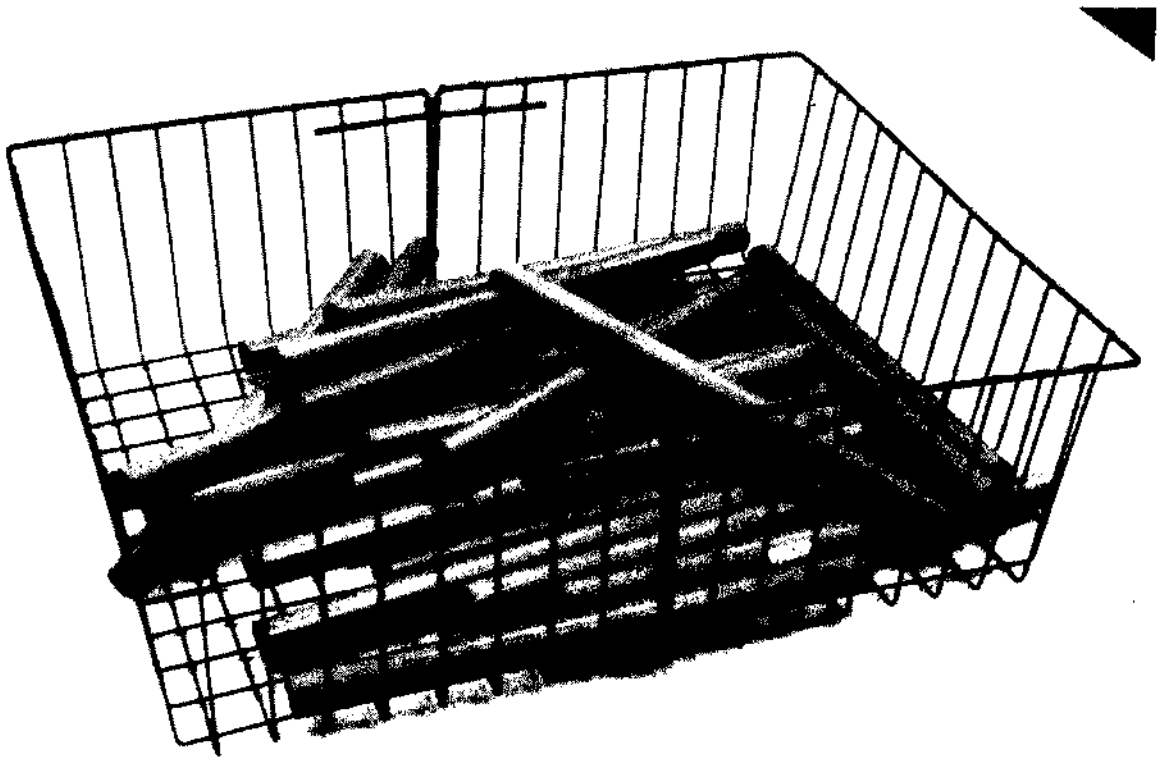


Figure 4-1: Pipes in a Wire Bin

in the placement of a single stripe (i.e., a scan). Since, at first, ROPIS knows little about the environment, it probes according to an initial search pattern. But as it learns more, ROPIS uses selective scanning, including the spacing of stripes and the direction of scans, combined with low- and high-level interpretation, to verify or reject particular hypotheses and thereby adapt to new knowledge as it is acquired. The decisions as to which scans to produce are ultimately left to ROPIS, with the decisions arising in a natural way when the upper levels of interpretation wish more information about an inadequately sensed or hypothesized surface.

The sequence of events which occur as ROPIS identifies its target is as follows: First, two scans are made in the robot tool x-direction to identify the immediate top pipe in ROPIS' field of view and to determine its orientation. This is done via determining the midpoint of the stripe segment in each scan which corresponds to the top pipe, and using these two points to determine the axis of rotation of the pipe via least squares curve fitting. The robot then "locks on" to this axis and aligns the light stripe along its length. Another scan is taken, from which the robot determines its next movement pose by finding one of the endpolnts of the stripe in view. Next, the robot moves to that point and orients the striper to take a cross-wise scan of the pipe. The midpoint of this segment is determined and checked against the values ROPIS believes the midpoint should possess. If they are within

tolerance, they are used as additional information to calculate a new axis of rotation. In this way, newly acquired information is used with old information to improve and refine a model -- ROPIS learns about the environment. A length-wise scan is performed next, followed by a cross-wise scan, etc., until one end of the pipe is identified. When ROPIS finds this, it performs the above-stated algorithm in the opposite direction to locate the other end of the pipe, after which it asks the user to remove the pipe. This is done since only one Puma robot is presently being used, and it is already holding the triangulation system. After the pipe is removed, ROPIS continues to identify the top-most pipe until it detects that the bin is empty.

If at any time a segment or midpoint is acquired which is not within tolerance of what ROPIS thinks it should be, ROPIS assumes that another pipe is on top of the pipe presently being examined. In this case, ROPIS forgets about the pipe presently being examined and investigates the new target. In this way, the system need not worry about predicting which other parts will move when the target is removed from the bin -- it begins anew after each new target start or pick operation to discover its next target.

ROPIS operates under a crucial basic assumption: it knows the cross-section and general form of the object it is to identify *a priori*. It expects to identify cylinders, and only cylinders. This makes the system somewhat limited in its use, but the techniques and algorithms used to accomplish the task of identifying a target are adaptable to other basic geometric forms with slight modifications.

The extraction of cylinder information from range data is subject to a number of kinds of error. Noise in the input data, both random and systematic, is a problem. Therefore, any system which determines these object parameters must be tolerant of occasional bad input. ROPIS has been designed with this in mind. In fact, rather than assuming perfect data, it expects to encounter noisy data with every stripe it sees. Thus, it is a system designed for real world applications.

ROPIS has been demonstrated successfully in the laboratory and is now in its final testing stage. A graphical display of the axis-of-rotation identification procedure is being developed. Recommendations will be made as to what can be done to speed up the system in real time. (Actual investigation into this is not possible because of delays in communication between the VAX and the Puma's LSI controller.) Accuracy of the entire system, including target identification errors, target position errors and target orientation errors are being explored. Limitations of the system will also be delineated.

5. Curve Fitting of Light Stripes for Similarity Assessment

A partial generalized cylinder is a region on the surface of a three-dimensional object where adjacent parallel cross section "cuts" of the light striper have a similar shape. Since we intend to rely heavily on these primitives for modeling and matching solid objects, it is important that we be able to assess the similarity in shape of two cross sections. This chapter describes a method for doing so.

The problem is compounded by the fact that we may not know the exact axis of the generalized cylinder. That is, even if two curves are identical in shape, one may need to be shifted in the plane in order for it to overlay the other. One useful characteristic of our method of curve matching is that it measures the actual amount of shift required to maximize the similarity of the two curves.

Although the cross section planes may be oriented arbitrarily in space, there is a preferred coordinate system for describing them: namely the coordinate system of the stripe projector. If we think of the curve's j>-axis as being in the direction the protector is aimed, and its x-axis in the plane of the stripe, then all points visible to the ranging system lie in the *(xy)* plane. Furthermore, vertical lines and overlaps, are very unlikely, hence we may regard *y* as a single-valued function of *x.* (In most scenes there are gaps and discontinuities in the light stripe, but we are dealing with only a single, continuous segment of the stripe at any given time.)

It is tempting to match curves in the camera's image plane. This has some of the desirable properties of the light-stripe plane, particularly single-valuedness, but it suffers from the fact that we would have to account for perspective distortion. The scaling from pixels in the image to millimeters in the scene is dependent on position in the image.

The first step is to fit the curve with a polynomial $y = /\{*\}$. Th*s ha s two effects: it smooths out tfie digitization noise in the curves, and it provides an analytic representation that is easy to deal with. Our computer programs will deal with polynomials of arbitrary degree. We have found that third-degree polynomials are adequate for simple cylindrical shapes. The fitting of the polynomial is by the method of Singular Value Decomposition [5].

Each curve is represented by a polynomial of fixed degree, and a range of *x* over which the curve is defined, Le., for which data points exist To match two curves, *f{x)* and $g(x)_t$ The second curve is shifted relative to the first to make their centrosds coincide. (Throughout the matching procedure shifts are made on *g(x):* we keep track of the cumulative amount of shift.) We now estimate tie mean-square error of the match

$$E = \frac{1}{L2 - L1} \int_{L1}^{12}$$

where LI and *12* delimit the range of overlap of the two curves. The integral is easy to calculate directly from the coefficients of the polynomials.

The remainder of the matching procedure involves hill-climbing, repeatedly shifting *g(x)* with respect to/(.x) to minimize *E.* We are aided by having analytic functions to evaluate: hence we can calculate the derivative of *E* with respect to shifts in *x* and *y.* If

$$E(\Delta x, \Delta y) = \frac{1}{L2 - L1} \int_{L1}^{12} [Ax) - (g(x + Ax) + Ay) f dx$$

Then

$$\frac{BE}{9AJC} = \frac{1}{\wedge 2 - 1 1 \wedge} \int_{}^{L_2} [f(x) - g(x)] \frac{3}{3x} [f(x) - g(x)] dx \qquad \textbf{(5.1)}$$

and

$$\frac{\partial E}{\partial \Delta y} = \frac{1}{L2 - L1} \int_{L1}^{12} [Ax) - g(x)] dx$$

The result for $\wedge \frac{\partial E}{\partial \Delta x}$ equation (5.1), does not include any effects due to changes in the limits of integration. If the range *L2—LI* is a constant interval, then an additional term is required: either *f{L2)-J{LI)* or g(L2)—#(11), depending on how the curves overlap. If the range is not constant, i.e., if the area of overlap is a function of *Ax,* then the effect on the partial derivative is more complicated. We have not attempted to quantify this effect. In any case, we ignore the change in limits of integration in evaluating the derivatives, and use only Equation (5.1).

If the shifting converges to a condition in which there is substantial overlap between the curves, we may assess ttie similarity in shape by examining the error *E.* If the average error (the total error divided by the length of overlap) is below a threshold, we may conclude the curves are "similar". This method is able to reliably identify anomalies in the cross sections of simple cylindrical shapes. We will next attempt to apply it to more complex shapes.

6. Pose Cluster Matching

6.1 Introduction

This chapter describes a technique for detecting and identifying objects in images, based on identifiable areas of the image we call "local features" and on the spatial relationships between those spatial features.

Images can be either two-dimensional or three-dimensional. We assume the existence of techniques that can locate specific features in these images such as edges, corners, holes, markings, flat surfaces, and the like. These are local features in the sense that each feature occupies only a local area of the image, and exists independently of other features which may or may not be in the picture.

The nomenclature of "local features" arises by contrast with "global features" such as area or perimeter. When global features are used to recognize an object, the entire object must be visible. Recognition by global features cannot deal with partially-visible objects arising from overlaps, occlusions, and objects partly outside the image frame. On the other hand, partial visibility will eliminate some but not all of the local features in an image. Thus, recognition based on local features has the potential of treating a much wider range of situations than that based on global features. The more tolerant these methods are of missing, extraneous, and incorrect features, the broader their applicability.

Recognizing objects is facilitated by having *models* of the objects to be recognized. These models contain descriptions of objects in terms of their features and the relationships among the features. Recognition is effected by matching the features in the image to the features of the model. If there are multiple distinct objects that need to be recognized, then they each have a model, and the features of the image must be matched against the features of each model in turn. Models are usually generated by actual analysis of an image containing the object under good viewing conditions ("training by showing"), but they also could be generated by CAD/CAM systems based on expected appearances.

Virtually all prior approaches to the matching of local features rely on two levels of match. At the low level is feature-to-feature matching. This level asks the question, "Could feature x from the image possibly correspond to feature y of the model?" If the low-level matches were always unique and unambiguous, there would be no need for a higher level of match, but in general many extraneous

matches will be made. The second level asks, "Given a set of low-level matches, what interpretation of the scene will maximize the number of valid interpretations?"

A number of researchers have used matching techniques in this spirit. Perkins [10] depended heavily on a feature-to-feature matching to find a distinctive sequence of bends and straight lines ("concurve") in the outline of an object. A great deal of effort went into finding the best concurve for this purpose. Only if the problem could not be solved with a single feature-to-feature match was a second concurve added to the analysis.

Belles used a method called "maximal cliques" [1]. All admissible low-level matches were made, then every pair of matches was considered for compatibility- Compatibility implies that both matches can conceivably arise from the same interpretation of the scene, and can be determined from the identity of the low-level features involved and the relative positions and orientations of the features. The high-level matching procedure involves finding cliques, or sets of matches that are mutually compatible.

Matching by clustering is discussed in an article by Stockman et al. [11]. Single feature-to-feature matches each suggest a particular position and orientation for an instance of a prototype within an image. Many such low-level matches that all suggest nearly similar positions and orientations are identified using a clustering procedure. In cases where the information from a single feature-to-feature match is insufficient to constrain the match position and orientation, an "abstract vector" is created which links two features, and high-level matching is carried out with these abstract vectors.

The method we will describe below is a synthesis of the ideas of Bolles [1] and Stockman [11]. The idea of mutual compatibility between two or more feature-to-feature matches, which comes from Bolles* work, is important. We replace Stockman's "abstract vectors" with the more general idea of constraining sets of feature matches. We express our method in a form that can be applied to arbitrary kinds of features, including those extractable from range data. The resultant general theory is applicable to both two- and three-dimensional images.

The method has many advantages that we desire of vision algorithms: It is robust and error-tolerant Decisions are made after digesting all the data available. Partial, occluded, or defective objects can be recognized from the portions that are visible and undamaged. And the method is relatively fast

6.2 The Basic Principle

Let us suppose we wish to be able to identify an object in a visual scene. The scene is represented by an *image* and the a description of the object we want to find is contained in a *prototype*.

We confine the following discussion to the case where we are interested in only a single prototype. If there are more than one distinct objects that might be in the scene, each will be represented by a prototype, and it will be necessary to match the features image sequentially against each prototype.

We first subject the image to a collection of image-processing routines that extract a number of *local image features*, which may correspond to physical features of objects, or which may be spurious. Each feature can be characterized by two types of data: *classification data* and *location data*. Classification data is that information useful in distinguishing one feature from another, such as dimensions (size, curvature, etc.), type of feature (edge, corner, hole, etc.), goodness of fit, etc., Location data includes position and orientation with respect to a coordinate system embedded in the object. Let the number of features extracted be n; we have the set of image features $\{f_i, i=1, \ldots, n\}$.

The prototype consists of a set of similar features, either obtained from a clean image of an example of the part, or generated automatically from a geometric data base. Let there be p model features $\{g_j, j=1, \ldots, p\}$.

A *high-level match* between the prototype and the image is characterized by a *pose* or spatial transform such that when the transform is applied to the model, some of the model features g_j correspond spatially to image features f_i. We may compute a quality measure for each high-level match from the number of features that correspond and the closeness with which they match in position, orientation, dimensions, and other classification data.

Now, the vision problem may be stated as a search to discover all the high-quality high-level matches between the prototype and the image. If several prototypes are compared, the match with the highest quality represents the most plausible interpretation of the scene. If it is known that the scene may contain more than one object, additional inferences may be made based on additional matches.

We propose the following three-step procedure for solving this problem:

1. Make tentative assignments between model features and image features based on classification data, on a feature-by-feature basis.

2. Consider m-tuples of model-to-image feature assignments such that

a. the assignments are mutually compatible, and

b. the assignments as group are sufficient to uniquely determine the pose of any high-level match that puts these features in correspondence.

Compute this pose for the next step.

3. Use a clustering technique to find which poses are most frequently suggested. The densest clusters will correspond to high-quality matches.

These three steps are elaborated in the following more detailed descriptions.

6.2.1 Single feature assignments

A single-feature assignment (or single-feature match) is a triple *(i,j,c)* that indicates that image feature^, is a possible match to model feature g- The parameter *c* is a confidence measure between 0 and 1 such that c = l indicates that the match is good on the basis of local evidence, and e=0 indicates a very poor match, *c* can be a likelyhood estimate.

Each of the *n* image features *{fi=\. . . . .n}* potentially matches each of the model features {&J=l....,p}. If all physical features were similar in appearance, or if our image processing techniques were incapable of distinguishing between types of image features, then we would have no choice but to accept all *nxp* potential matches. However, there is usually more information derivable from an image, and we can use that information to rule out impossible or unlikely matches.

The most obvious way to rule out matches is on the basis of feature type. An edge can match only an edge and a vertex only a vertex, etc.

Next there are quantitative sorts of measures that may be applied such as the length of a line segment, the size of an angle, or the dimension of a hole. If the measure of the image feature is not within a reasonable threshold of that of the corresponding model feature, the assignment may be ruled out as a bad match. If the measures are within tolerance of each other, the confidence measure ccan be made a function of the difference between the feature measures.

Finally ttiere are measures of similarity returnable by low-level pattern matching. If, for example, features are identified by correlating *small* patches over the image, then the degree of correlation should exceed a threshold for the assignment to be accepted* The confidence will be a function of fie degree of correlation.

At the conclusion of the sitigfe-feature assignment step we should have a collection of high-quality

matches between image features and model features. Each image feature may match zero, one, or more than one model feature. Similarly each model feature may match zero, one, or more than one image feature. In general, we expect to find many more single-feature matches than are necessary; a lot of assignments will be erroneous. Call the assignments $\{A^{\wedge}\ fc=1,\ldots,g\}$, where q is the number of assignments made and accepted, and each A_k is a triple $(i_{kj}|\!>\ c^{\wedge}$ as discussed above.

6.2.2 Mutually compatible sets of features that constrain the match pose

A *pose* is a transform from image space to object space. If scaling factors are known, then a pose may be thought of simply as a position and orientation. In two dimensions, it takes two degrees of freedom to specify position and one to specify orientation, for a total of three degrees of freedom. In three dimensions six degrees of freedom are needed: three of position and three of orientation. When the scale factor is presumed or allowed to vary between images, then the pose must include that factor. Then we need four degrees of freedom in two dimensions, and seven degrees of freedom in three dimensions.

Acceptance of the hypothesis that any particular assignment A_k is correct constrains the pose of possible matches between image and prototype. This constraint causes the space of possible poses to lose as many degrees of freedom as are needed to specify the pose of the features entering into the. match. We may speak of the set of permissible poses under an assignment and the number of degrees of freedom it contains.

Any two or more assignments are *compatible* if the intersection of their sets of permissible poses is non-empty. Any number of assignments are *constraining* if the intersection of all their sets of permissible poses is a single pose. Of course, every constraining set of assignments must also be compatible. Whether or not a group of assignments are compatible, or constraining, depends on the number of degrees of freedom involved as well as the nature and relationship of the features being matched. We give some examples on a case-by-case basis.

Consider two-dimensional images with known scale where all features are point-like, i.e., they are characterized only by a position in x and y. Matching to a single point constrains two degrees of freedom, leaving one degree of freedom in the set of permissible poses. Imagine overlaying two images so that they correspond at a single point The permissible degree of freedom in this match is represented by rotating the images about the point of the match.

A pair of point-feature assignments represents a match at two different points. Two points constrain four degrees of freedom—more than are needed to specify a pose in two dimensions. Therefore there

is a possibility that the intersection of the permissible poses for the two assignments will be empty. But if the distance between the two points in the image is the same as the distance between the corresponding two points in the model, then the assignments are compatible *and* constraining.

If we consider two-dimensional images with unknown scale, then two non-coincident point-feature assignments will *always* be compatible and constraining.

In three dimensions, features that are a portion of a flat plane surface without regard to its boundaries will constrain one degree of freedom in position and two in orientation. (Additional information about the boundaries of the plane might constrain additional degrees of freedom.) Two assignments, each involving planes, use six degrees of freedom. They will be compatible only if the angle between the planes is the same in the image as in the model. One degree of freedom is lost in satisfying compatibility--only five are available to constrain the pose of the match. The sixth degree of freedom represents an indeterminacy of position parallel to the line of intersection of the two planes.

We define a set of assignments to be *minimally constraining* if that set is constraining, but would not be constraining if any assignment were removed from the set.

The second step of our high-level matching procedure is to find sets of assignments that are constraining, using the assignments generated in the first step. Each set so found will have a pose associated with it. All poses generated in this step are passed on to the next step for cluster analysis.

There are several strategies for generating sets of compatible, constraining assignments. The most straightforward is to find all minimally constraining sets. Other strategies involve constructing sets that are larger than minimal, or considering only sets of assignments that obey certain criteria. The reasons for choosing one strategy over another involve tradeoffs among low-level processing, speed of search, and accuracy of pose determination. We will show several different strategies in later sections of this paper.

6.2.3 Clustering

If we assume the image contains a single instance of the object, and further assume perfect imaging and feature extraction, then there should be one high-level match that corresponds to a correct interpretation. One assignment should be made for each each feature of the model. If additional assignments are made because of similarity of features, then we will regard these as incorrect assignments. All the correct assignments will be mutually compatible, and any constraining subset of correct assignments will generate the same pose as every other constraining subset. On the other

hand, incorrect assignments may or may not be compatible with other assignments.. When compatible constraining subsets of assignments can be found, they will yield poses that vary widely. Some incorrect assignment groups might yield the same pose--this would indicate a degree of similarity between different portions of the model. But unless there are exact symmetries in the model, the correct pose will be suggested by more assignment groups than any other pose. (We can account for symmetries if they are known in advance. See below.)

When conditions are less than ideal, the above analysis gets fuzzy. Various factors conspire to cause incorrect assignments and to cause the correct assignments not to be made. Even when correct assignments are made, noise and digitization error cause variation in the positions and orientations of features, which translate to variation in the poses output by step 2 of the matching procedure described on page 1 (finding m-tuples of low-level matches). Instead of counting which poses are most frequently suggested, we must resort to a clustering technique to find a group of poses near to each other.

There are many clustering algorithms and heuristics available. We have found the following procedure to be simple and effective.

For the purposes of clustering, we need to be able to assign a measure of similarity to any two poses. We do this by independently assessing similarity in position, orientation, and scale, and taking a weighted sum of the three. The weights must be developed empirically to normalize the actual amount of variation observed in clusters that represent a correct interpretation.

It is to be understood that in decomposing a pose transform into independent descriptors of position, orientation, and scale, these operations must be with respect to a point near the object's center, usually the centroid or center of gravity. If rotation is about some other point such as a corner of the image, then position and orientation lose their independence, and a change in orientation or scale will affect the numerical position of the object.

Also, we need to define the "average" of any number of poses. This is done by independently averaging the position, orientation, and scale components of the pose.

With these preliminaries out of the way, then, we describe out clustering procedure. We define a cluster descriptor to contain two items: a count of the number of poses represented in that cluster, and a pose representing the average of all those poses. A list of all clusters is initialized to the empty list, then all the poses generated by Step 2 are considered one at a time as follows. If this new pose is

similar, within a threshold, to the average pose of any cluster, then it is added to the cluster it is closest to, and that cluster's statistics are updated. Otherwise the new pose becomes the first and only pose in a newly created cluster.

Using this clustering method, there is no need to save all the results of Step 2 before starting on Step 3 (clustering). The two steps can run concurrently, with each pose generated by Step 2 being immediately passed on for consideration by Step 3.

This clustering method suffers from the deficiency that its results might depend on the order in which poses are presented. But for most cases that should not be a problem. If the operation of the high-level match turns out to depend critically on the minor differences between one order of presentation and another, or even on the differences between one clustering technique and another, then it is likely that the scene itself is ambiguous and additional information will be needed to make further progress in interpreting it

The cluster containing the largest number of poses represents the most likely interpretation of the scene. The size of this cluster is related to confidence that the interpretation is correct. A cutoff threshold on size may be established, as a function of the size of cluster to be expected under ideal viewing conditions. If no cluster exceeds the threshold, then there is insufficient evidence to conclude that the model exists in the scene. If more than one cluster exceeds the threshold, then either there are two objects in the scene, or the scene is ambiguous.

One possible reason for ambiguity in a scene is symmetry. When a model possesses /i-fold rotational symmetry, then *n* different orientations of the object are equivalent Each cluster of poses matching image to model will be replicated *n* times. If the model is known to be rotationally symmetric, and if only a single axis of rotational symmetry exists, then we can modify our clustering procedure to avoid the ambiguity and the extra overhead. In two dimensions, then, we represent angles modulo *2m fn.* AH *n* clusters will fold into one. In three dimensions, we must choose the orientation of the coordinate system in which the model is represented so that its axis of symmetry lies on the z-axts. We then represent ttie angle ^, the tttird Euler angle$_t$ which represents a rotation about z_9 modulo 2 v / n . A further savings in efficiency can result if the search strategy used in Step 2 can exploit symmetry to reduce the number of assignments generated.

6.3 Complexity Analysis

It is useful to examine the computational complexity of pose cluster matching. In any given situation, actual run times of an algorithm are more useful than any theoretical analysis. The computational complexity then gives a rule for extrapolating these times to problems involving more elements.

We will analyze the complexity of each of the three steps described on page 1 separately. The complexity of each step depends on a number of assumptions about the nature of the objects and features being matched--we will state how these assumptions affect the complexity, and give best- and worst-case bounds on the complexity.

Step 1 is single-feature matching. If there are n features in the image and p features in the model, then at most $n \times p$ comparisons will be required for this step. A reduction in this complexity can be achieved if the features are of different types. For example, if features include both edges and vertices, it is necessary only to compare edges with edges and vertices with vertices.

Call the number of assignments generated in step 1 q. In the worst case, if every local feature is like every other local feature, q will equal $n \times p$. In the best case, if sufficient classification data are available to avoid making any incorrect assignments at all, q will be equal to n.

The complexity of step 2 depends on q, and also on the number of assignments to be grouped in a constraining match. Call that number k. (For simplicity, assume all constraining groups have the same number of assignments.) Then the strategy of examining all possible k-tuples of assignments for compatibility will require $\binom{q}{k}$ combinations, which is $O(q^k)$.

Let G be the number of groups generated in step 2. At worst this number will be $O(q^k)$, if step 2 is unsuccessful in eliminating incompatible assignments. At best, if no incorrect groupings are made, G will be $O(n^k)$.

The complexity of step 3 is $O(GC)$, where C is the number of clusters found. At worst, C is equal to G, making step 3 $O(G^2)$. At best, there will be only one cluster, and step 3 is $O(G)$.

Combining all the above results, it turns out that our method of high-level matching is between $O(n^k)$ and $O((np)^{2k})$. Thus pose clustering appears to be better in this respect than the method of maximal cliques, which has an exponential computational complexity [1].

It should be pointed out that the worst-case performance can actually occur if the model consists of n identical features arranged at regular spacing around a circle, or arranged at all the vertices of a three-dimensional regular solid, and if no account is taken of symmetry in search or clustering.

The computational complexity of this method of matching by pose clustering depends strongly on the number of false matches generated. Therefore, the use of additional processing to extract better low-information about low-level local features can be well compensated for by greater efficiency in later stages of processing.

6.4 Results

Pose cluster matching has been successfully applied to recognition of two-dimensional objects from their silhouettes.

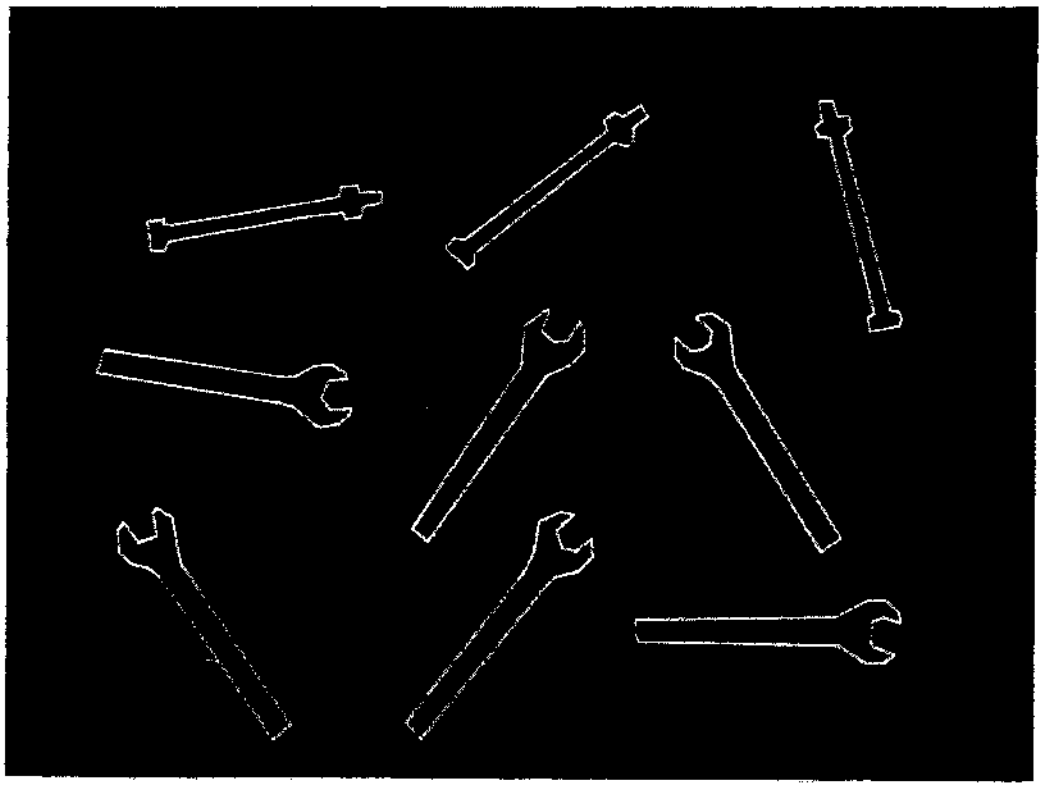
Objects are placed on a backlit table, and imaged by a TV camera attached to a Machine Intelligence Corporation VS-100 vision module. The vision module binarizes the image, and extracts the perimeter points of each connected "blob." Software developed especially for this project, residing in the vision module, segments the ordered list of perimeter points and fits straight line segments to the points. The segmentation points and the line equations are uploaded from the vision module to our Vax for analysis.

The line-fitting process is not particularly reliable. When the silhouettes have sharp corners and straight edges, the results are very repeatable; but when there are curves, the number of segmentation points and their location can be quite variable.

Tests have been carried using a number of different parts and part sets. For illustration purposes, we show here a collection of shapes that represent wrenches and bolts. Figure 6-1 shows nine instances of these shapes: three bolt images and six wrench images. Three of the wrenches have their heads pointing to the right and three to the left.

Two kinds of features were considered for low-level matches: corners and edges. Corners are characterized by their position, the orientation of their interior bisector, and the included angle. Edges are characterized by the position of their midpoint, their orientation, and their length.

According to this characterization, corners and edges have three degrees of freedom and are, therefore, sufficient to constrain the pose of a high-level match. In step 2 of our basic procedure, $m=1$ is sufficient to perform matching.



Figu re 6-1: Nine Prototype Outlines

We tested our matching method by matching each of the shapes in Figure 6-1 to all of the other eight shapes in the Figure. That is$_t$ we chose one shape to be the "unknown/^{1} designated the other eight shapes as prototypes, and asked the program to perform recognition. The program would match the unknown to each prototype in *turn*, and sort the list of all pose clusters thus obtained by the number of matches in the cluster, tf the method works 'perfectly, the two highest-ranking clusters should correspond to the other examples of a similar shape. Otherwise the number of mismatches in the two top-ranking clusters will provide an accuracy measure.

It turns 'Out that this matching procedure is able to distinguish bolts from wrenches, but net wrenches In a right-handed configuration from those in a left-handed configuration. In each case$_9$ the two highest-ranking pose clusters correspond to the correct part, but the performance in telling right from left is slightly worse than chance, The average time to match an image and a prototype is 1.1 seconds, or the average time to perform "recognition" with eight prototypes is 8.8 seconds. (Times reported are CPU time on a tisrte-shared Vax 750, with programs written in the C language. They do not include any allowance for extracting perimeters or fitting straight lines; analysis starts with the remits of the straight in© fitting.)

A more telling task is to find parts in scenes consisting of overlapping part outlines. Figures 6-2 through 6-4 show three such scenes. Each scene was matched against three different prototype sets. Each prototype set consisted of one of the three columns of Figure 6-1. A given scene was matched against each of the three prototypes in a given set, and the resulting list of pose clusters was sorted by the number of matches in the cluster. Here the desired result is to identify the two objects making up the scene. The highest-ranking cluster was taken as identifying one of the objects. Some judgement was involved in identifying the second object. If the second-highest-ranking cluster clearly applied to the other portion of the scene, then its identification was used. But frequently it was found that the highest and second-highest clusters "explained" the same portion of the scene, in which case the third-ranking cluster was used, or even the fourth or fifth, if higher-ranking clusters all pertained to the same comstellation of lines and corners. To automate this procedure, it would be possible to delete from the scene all features participating in the highest-ranking cluster, then perform the match again to locate the second object.

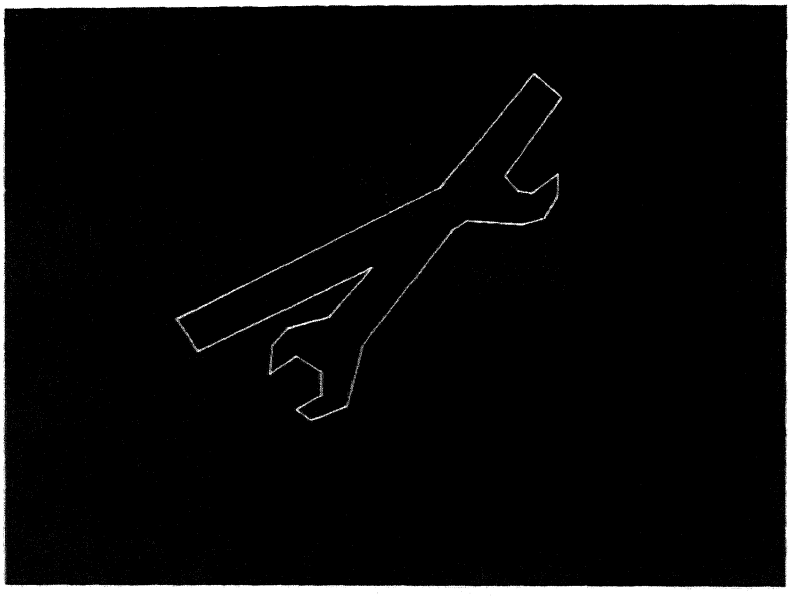


Figure 6-2: Two Overlapping Wrenches

The results were that when the handedness of the wrenches is ignored, the system was able to come up with a correct identification of a part in the correct orientation 70 percent of the time. Figure 6-5 shows an example of a correct analysis: the outline of the composite silhouette is overlaid with the outlines of the corresponding prototypes in their match position. Figure 6-6 shows a misidentification: in this case the correct prototype was chosen, but the inferred position and orientation are wrong. The average time to match each scene against the three prototypes was 11.0 seconds.

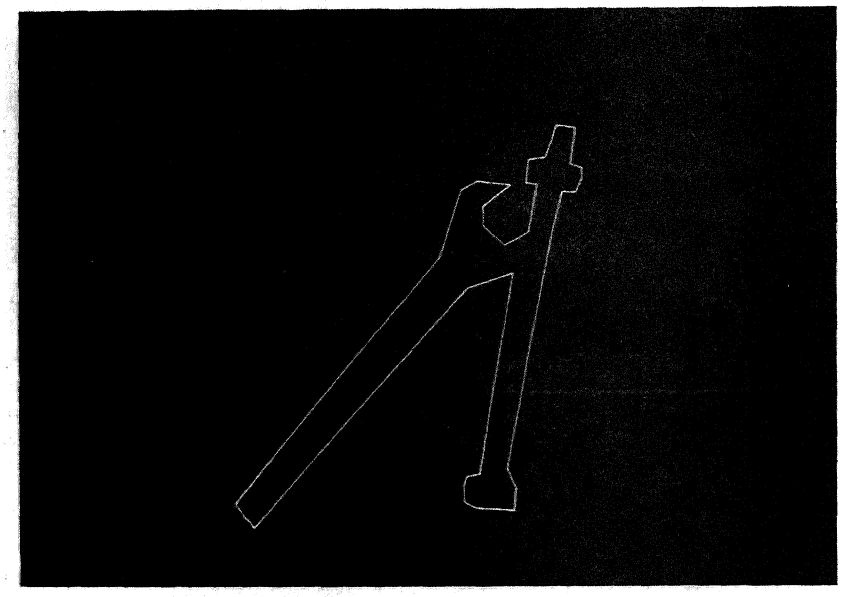


Figure 6-3: Overlapping Wrench and Bolt

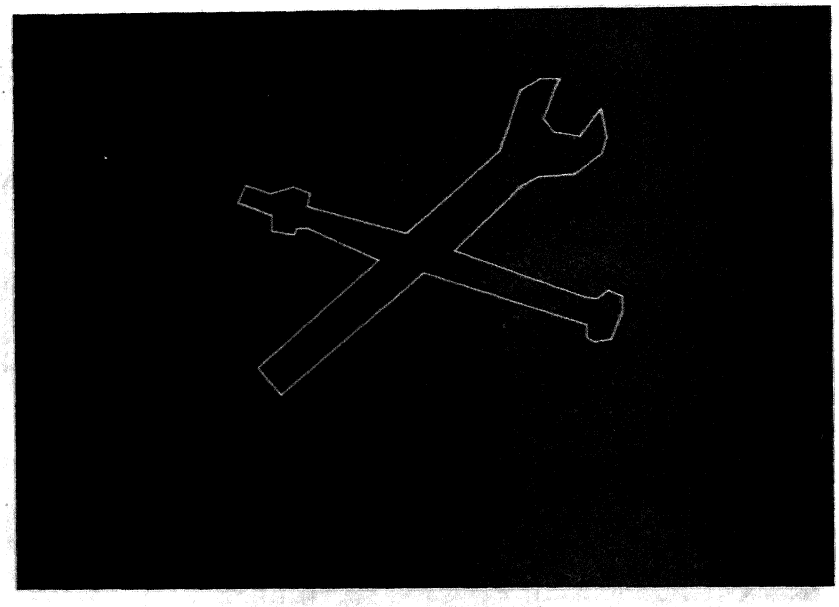


Figure 6-4: Overlapping Wrench and Bolt

We can improve the performance of the matcher by letting $m=2$, that is, considering *pairs* of feature-to-feature assignments for pose generation. Since each feature (corner or edge) has three degrees of freedom, each pair of features has six. Three degrees are needed to specify the pose; the other three are used to assure compatability between the feature matches. The features are checked for the proper distance: that is, the distance between the two features in the image must be equal, to within a threshold, to the distance between the corresponding distance in the prototype. Also, the angle of each feature with respect to the line joining the two features must match in the image and the prototype. For any pair of low-level matches to be considered compatible, it must meet all three tests (one of distance, two of angle).

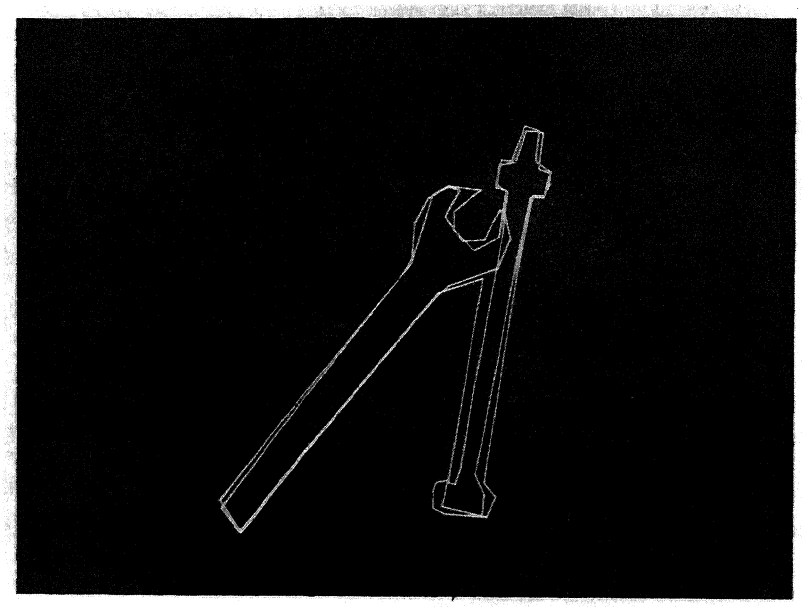


Figure 6-5: Two Correct Identifications

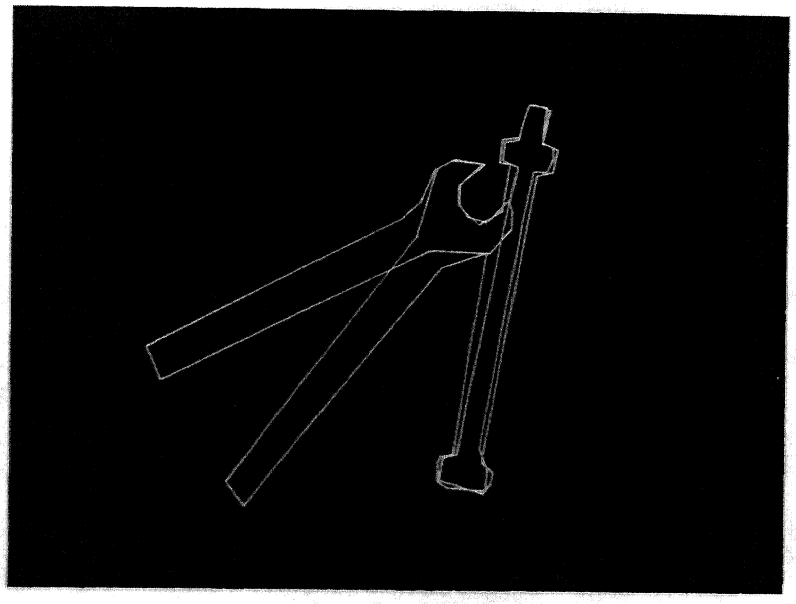


Figure 6-6: An Incorrect Identification

Two low-level matches that specify the pose of a high-level match are called "abstract vectors" by Stockman [11]. They were used in the cited work because the low-level features that were being used had insufficient degrees of freedom to completely specify the pose of a high-level match. We choose to use pairs of features because they give us a richer set of primitives. Also, we obtain better precision in specifying the orientation of poses, which gives us tighter clusters and less ambiguity. The more precise orientation comes from the fact that two features separated in space can specify an orientation better than local analysis of a portion of a line segment.

We repeated the two experiments using a value of #=2. In the case of matching each of the nine outlines against the other eight, the results were similar: the system was again able to distinguish reliably between wrenches and bolts. Recognition time increased from 8.8 seconds to 11.1 seconds, or from 1.1 second to 1.4 per individual match.

With the composite outlines there was a marked improvement in recognition accuracy: from 70 percent to 95 percent. Recognition time increased from 9.1 second to 12.2 second.

The system was still unable to reliably distinguish right-handed wrenches from left-handed ones, but the percentage of correct right-left discriminations increased slightly as *k* was changed from 1 to 2 in both experiments. It is difficult to assess the statistical significance of this result.

Although the above results are impressive, the time performance leaves a lot to be desired. Since execution time is a polynomial function of the number of low-level features found, a lot can be gained by reducing the number of features. We do this by combining short segments into "arcs." The intent is to identify sequences of short segments that could belong to the same curving cross section.

This is implemented by first comparing the exterior angles of each corner of the scene with a threshold that depends on the length of the edges on either side of the corner. Angles that are below this threshold are classified as "gently curving." Two or more adjacent gently curving corners ttiat curve in the same direction are eliminated: they and their surrounding edges are combined into one arc. An arc has the same properties as an edge.

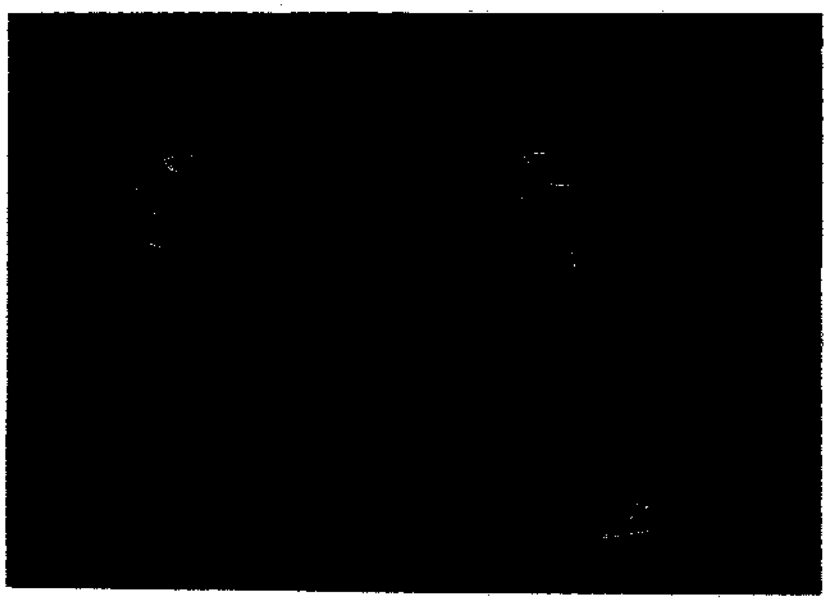


Figure 6-7: Wrench and' Bolt Outlines with Arcs

Figure 6-7 shows some part outlines where arcs are indicated by straight line segments "short-cutting" the sequence of lines they replace. The process of searching for arcs introduces some additional uncertainty into the data upon which matching is based, but the method is robust enough that it is not significantly affected by this amount of noise. In fact, for the prototype set we are using as an example here, recognition accuracy increased slightly.

Introducing arcs into the process reduced the number of edges and corners by about half. Recognition times fell to about one tenth their previous values, both for the case $k=1$ and for $k=2$. This result is somewhat unexpected. For when $k=1$, the time for low-level matching (step 1) and pose determination (step 2) should be related to the square of the number of features. Although clustering (step 3) is of a higher order of complexity, in practice it is small enough not to dominate the computation. The ten-to-one reduction in matching time must result from the elimination of many spurious matches between short line segments in the gently-curving portions of outlines.

The performance of our matching procedure in all experiments involving the wrench and bolt outlines is summarized in Table 6-1.

Table 6-1: Recognition Results

Experiment	k	arcs	**Accuracy 1** (percent)	**Accuracy 2** (percent)	time (seconds)
Match 9 outlines against each other	1	no	100	25	8.83
	2	no	100	33	11.08
	1	yes	100	25	1.39
	2	yes	100	50	1.62
Composite scenes of overlapping outlines	1	no	65	42	9.09
	2	no	95	58	12.24
	1	yes	100	67	1.00
	2	yes	100	75	1.16

Accuracy 1: Success in identifying and locating bolts and wrenches irrespective of handedness.

Accuracy 2: Success in distinguishing right-handed from left-handed outlines.

I. A Movable Light-Stripe Sensor for Obtaining Three-Dimensional Coordinate Measurements

« Presented at the SPIE International Technical Symposium. Conference 360 (Robotics and Industrial Inspection), August 21-27, 1982, San Diego, California »

A Movable Light-Stripe Sensor for Obtaining Three-Dimensional Coordinate Measurements

Gerald J. Agin and Peter T. Highnam

The Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pa 15213

Abstract

We describe an apparatus for obtaining three-dimensional surface information that may be used to recognize objects and determine their position and orientation. A lightweight camera and a light-stripe projector using an infrared laser diode are mounted in the hand of a robot manipulator. Image-processing routines locate the stripe in the camera image, and homogeneous coordinate transform techniques are then applied to solve for the three-dimensional coordinates of points illuminated by the stripe. We describe the hardware, the equations for coordinate measurement, and the procedures for accurately calibrating the apparatus.

Introduction

This paper describes a means for obtaining three-dimensional coordinates of surface points. The primary motivation for developing this apparatus is for research in the representation and recognition of three-dimensional objects.

There are two main reasons for studying three-dimensional vision. The first reason is practical: if useful hardware can be developed that directly measures three-dimensional shape, then it should be easier to analyze range images than the light images that are used for conventional computer vision. Success in this area can lead to an excellent system for robotic manipulation, inspection, bin picking, and assembly. The second reason is for the insight it can give into the human vision system and how human intelligence represents and reasons about shape and spatial relationships.

Much work has been done in the past in the representation of three-dimensional objects, principally for purposes of computer-aided design and manufacturing[6]. Sophisticated techniques have been developed for representing curved shapes such as automobile fenders and aircraft surfaces to a high degree of accuracy[3]. However, these techniques require a fixed coordinate system in which to describe the shapes. For recognition of an object in an unknown pose (position and orientation) no fixed coordinate system is available. Recognition must be based on *structural* features of objects, which should be rather more qualitative than quantitative, and independent of any coordinate system.

We have built our range-measuring device to obtain surface coordinates on which to base structural three-dimensional vision. Because objects will be lying in unknown poses, the ability to randomly access various parts of the scene is important. The flexibility to view objects from different viewpoints is also useful.

On the other hand, high accuracy is not a requirement. If we deal with objects in a size range of 5 to 20 centimeters, a resolution on the order of a millimeter or two should suffice for our purposes.

The approach we have taken is to mount a triangulation range finder in the hand of a robot manipulator. The range finder consists of a light-stripe projector and TV camera mounted in a rigid frame. The manipulator functions as a "programmable tripod," a controllable mount that may be used to place the range finder in an arbitrary pose.

Light striping for three-dimensional measurement has been demonstrated by a number of investigators [1, 9, 7]. The essential idea is the use of triangulation for ranging. Given the relative poses of the camera and projector, the three-dimensional coordinates of any spot illuminated by the projector may be found by solving some simple trigonometric relationships. From the shape of the stripe as viewed from the camera we may locate edges and discontinuities, deduce whether a surface is flat or curved, convex or concave, etc.

Placing a light-stripe projector and camera in the hand of a manipulator was first demonstrated by Agin[2]. This arrangement was used for visual feedback, to place the robot's end-effector in a given relationship to a target, for the purpose of seam tracking. It has been copied in a number of commercial systems for welding. Vanderbrug also placed a stripe projector and camera in the hand of a robot[10] which he used to locate objects on a tabletop. This is the first effort we are aware of in which careful attention has been paid to the measurement of coordinates within an absolute coordinate system.

Mechanical and optical

The camera and projector are mounted in an aluminum housing, as shown in Figure 1. The housing was machined from 4-inch tube stock. The housing gives some protection against accidental collisions, as well as providing a mounting place. The camera and projector are each held in place from beneath by 1/4-inch camera mount screws. Four holes drilled in the back face provide for connection to the mounting flange of the manipulator. These holes are closer to the camera end of the housing to provide better balance, since the camera is heavier than the projector. The entire assembly, including camera, projector, and housing, weighs approximately 37 ounces (1.05 kg).

The projector consists of a laser diode and a cylindrical lens in an aluminum mount. A closeup view of the projector mount is shown in Figure 2. The diode, a CW-operated Aluminum Gallium Arsenide Injection Laser, emits 15 milliwatts of infrared radiation at a wavelength of 820 nm. (A visible wavelength would have been preferrable, but diodes that had all the other desirable characteristics were only available in infrared wavelengths.) The cylindrical lens focuses the emitting area into a stripe of light. The length of the stripe is determined by the spread of the original beam, and the width is determined by the size of the emitting area and the magnification of the lens. The size of the emitting area is small: only about 40 μm x 15 μm. This configuration casts a stripe of light about 20 cm long and 1 mm wide at a distance of 30 cm from the lens.

The camera is a CID solid-state video camera, with 256 x 256 resolution elements. The lens is 25 mm, f1.4. The silicon photosensitive elements have a peak spectral sensitivity in the near infrared, so the laser stripe shows up well in the camera image. When an infrared spectral filter is placed in front of the lens to block the ambient illumination, only the stripe appears in the image. (Our laboratory is illuminated by fluorescent lamps. Incandescent lamps can not be used, since their infrared emissions would swamp the laser diode output.)

We must consider the depth of field of the sensor. The stripe itself is in focus over a wide range; however the camera focus creates something of a problem. To obtain sufficient light we must operate with the lens at its widest opening. Vanderbrug has dealt with this problem by inclining the camera's retina parallel with the plane of light[10]. A drawback of our system is that we do not do this. We just make the assumption that the location of the centerline of the stripe will unaffected by focus. If this assumption turns out to be false and it becomes necessary to improve the accuracy of particular measurements, we can always take one approximate measurement, and use that to position the camera so that it will be in focus.

Manipulation and vision subsystems

Figure 3 shows the major components of the range-measuring system. Overall control and intelligence resides in a VAX-750 computer. The VAX communicates over a serial line with a dedicated LSI-11 that handles the real-time requirements of communication with the manipulation and vision subsystems.

The manipulator is a Unimation PUMA 600 robot. The robot's control system includes a microcomputer that can operate the robot in any Cartesian coordinate system. In normal operation a terminal connected to the robot's controller is used to issue commands to the robot and to receive information about the robot's position and status. We have replaced that terminal with another computer, our LSI-11. The program running in the LSI-11 together with a local terminal allows the user to interact with the robot's controller as if the terminal were connected directly to the controller.

The LSI-11 can also be instructed from the VAX to send commands to or obtain information from the PUMA controller. A library of subroutines allows the VAX programmer to specify poses, tool transforms, and motions, and to receive feedback about the manipulator's current pose and status. In the VAX, all poses (positions and orientations) are represented as 4 x 4 homogeneous transform matrices. The LSI-11 monitors the output from the PUMA controller for error messages, so that the operator or programmer can be alerted to various common error conditions and take appropriate action.

A Machine Intelligence Corporation VS-100 Vision System serves as a preprocessor for output from the TV camera. This system contains a dedicated microprocessor for control and image processing. Normally the vision module is controlled by the operator's pointing with a light pen at items on a menu on the display screen. The module can also be controlled externally through a DRV-11 parallel interface. Our LSI-11 can place commands on and receive information from this interface, and another library of subroutines on the VAX allows the programmer access to all essential functions of the vision module.

A programmable threshold converts the TV image to binary. Under good viewing conditions, the light stripe in the thresholded image appears white against a black background. The connectivity algorithm in the vision module has been specially modified for use with stripes that run in a generally top-to-bottom orientation[8]. It finds the longest stripe in the image in terms of the number of rows it touches, throwing away spurs and other noise. The image coordinates of points at the centerline of the stripe, and the horizontal width of the stripe at each point, are uploaded to the VAX via the LSI-11.

Homogeneous Coordinate Equations for Depth Measurement

The triangulation equations for depth measurement may be formulated in terms of homogeneous coordinate notation. (The reader not familiar with homogeneous coordinates and their use in manipulation and vision may wish to consult one of the textbooks given in the references[4, 5]. The problem may be broken into two parts:

- measuring the parameters of the camera and the poses of camera and projector, and
- using that information to derive the three-dimensional coordinates of points illuminated by the light stripe.

The first of these parts we call *calibration*. The second part is the easier part, and we will describe it first.

Let there be a coordinate system embedded in the camera as shown in Figure 4. The lens points in the z-direction, the x-axis points to the right, and the y-axis points downward. The origin of camera coordinates is at the "lens center" of the camera. We can use the the symbol C to denote that coordinate system, as well as the 4 x 4 homogeneous transform matrix that represents the relationship of that coordinate system to global coordinates. Given x_c, y_c, and z_c, the coordinates of any point with respect to C, the global coordinates x, y, and z of that point may be found by the matrix equation

$$\mathbf{X} = \mathbf{C}\,\mathbf{X}_c, \tag{1}$$

where $\mathbf{X}_c = [\,x_c\ y_c\ z_c\ 1\,]^T$, and $\mathbf{X} = [\,x\ y\ z\ 1\,]^T$.

The perspective transformation of the camera may be represented by the equation

$$\mathbf{X}_i = \mathbf{P}\,\mathbf{X}_c, \tag{2}$$

or in expanded form,

$$\begin{matrix} x_i \\ y_i \\ z_i \\ h_i \end{matrix} = \begin{matrix} F_x & 0 & 0 & 0 \\ 0 & F_y & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix} \begin{matrix} x_c \\ y_c \\ z_c \\ 1 \end{matrix}, \tag{3}$$

F_x and F_y represent scaling parameters that take into account the focal length of the lens and the spacing of the photosensitive elements on the retina. $\mathbf{X}_i$ represents the image coordinates of the point at $\mathbf{X}_c$. The actual screen coordinates of the point are x_i / h_i and y_i / h_i.

From Equations (1) and (2) we have the relationship

$$\mathbf{X}_i = \mathbf{P}\,\mathbf{C}^{-1}\,\mathbf{X}. \tag{4}$$

This may be inverted to yield

$$\mathbf{X} = \mathbf{C}\,\mathbf{P}^{-1}\,\mathbf{X}_i. \tag{5}$$

Although this equation suggests a way of deriving the three-dimensional coordinates of a point from its image coordinates, it is not particularly useful as it stands. The depth in image coordinates, z_i, cannot be directly measured. But if the real-world point is known to lie on a plane whose equation is known, z_i can be determined from x_i and y_i.

Let the symbol S stand for the pose of the light-stripe projector, as in Figure 4. Analogous to Equation (1), we may write

$$\mathbf{X} = \mathbf{S}\,\mathbf{X}_s. \tag{6}$$

S's z-axis points in the direction the projector is aimed, and the plane of projected light lies in S's y-z plane. If we let $\mathbf{V}_x$ stand for the vector [1 0 0 0], then any point $\mathbf{X}_s$ (relative to the projector) lying in the plane of light must obey the relationship

$$\mathbf{V}_x\,\mathbf{X}_s = 0. \tag{7}$$

Substituting the inverse of Equation (6) into Equation (7) gives the relationship

$$\mathbf{V}_x\,\mathbf{S}^{-1}\,\mathbf{X} = 0, \tag{8}$$

which must hold for any point in the plane of the light stripe.

Considering Equations (5) and (8), we may write

$$\mathbf{V}_x\,\mathbf{S}^{-1}\,\mathbf{C}\,\mathbf{P}^{-1}\,\mathbf{X}_i = 0. \tag{9}$$

If we let Q be the matrix product $\mathbf{S}^{-1}\,\mathbf{C}\,\mathbf{P}^{-1}$, and let q_{ij} denote the elements of Q, then Equation (9) may be expanded to

$$q_{11}\,x_i + q_{12}\,y_i + q_{13}\,z_i + q_{14}\,h_i = 0. \tag{10}$$

Arbitrarily setting h_i to 1, this equation may be solved for z_i to yield

$$z_i = -(q_{11}/q_{13})\,x_i - (q_{12}/q_{13})\,y_i - (q_{14}/q_{13}). \tag{11}$$

This result may be applied directly to the calculation of X by explicitly solving for z_i, substituting the result in Equation (5), and normalizing to make the fourth element of the vector X equal to 1. However, it is possible to use Equation (11) to produce a 4 x 3 *collineation matrix* K such that

$$X = K [\, x_i \, y_i \, 1 \,]^T. \tag{12}$$

If we let r_{ij} denote the elements of $R = C P^{-1}$, then

$$K = \begin{matrix} r_{11} - r_{13}\, q_{11}/q_{13} & r_{12} - r_{13}\, q_{12}/q_{13} & r_{14} - r_{13}\, q_{14}/q_{13} \\ r_{21} - r_{23}\, q_{11}/q_{13} & r_{22} - r_{23}\, q_{12}/q_{13} & r_{24} - r_{23}\, q_{14}/q_{13} \\ r_{31} - r_{33}\, q_{11}/q_{13} & r_{32} - r_{33}\, q_{12}/q_{13} & r_{34} - r_{33}\, q_{14}/q_{13} \\ r_{41} - r_{43}\, q_{11}/q_{13} & r_{42} - r_{43}\, q_{12}/q_{13} & r_{44} - r_{43}\, q_{14}/q_{13} \end{matrix} \tag{13}$$

Equation (12) may be used to calculate the three-dimensional coordinates of any point on the centerline of the stripe seen in the TV image. However, calculation of the collineation matrix K requires knowledge of the matrices P, C, and S. The next section describes how these may be determined.

Calibration

Fourteen numeric parameters are needed for calibration: two for P (F_x and F_y), and six each for C and S (as Euler descriptors, three each of position and three each of orientation). We find it convenient to add a fifteenth parameter, a wrist rotation, as will be described below. These fifteen numbers are kept in a calibration file. Some of these can be measured with a ruler and stored permanently in the file. Others are more prone to change and require rather more elaborate procedures.

The camera pose C and the projector pose S both depend on the pose of the manipulator that supports them. The PUMA may be interrogated at any time to obtain the pose matrix T holding the position and orientation of the mounting flange to which the range-finder housing is bolted. If we let C_T and S_T denote the *relative* positions of the camera and projector with respect to the mounting, then we have the two relationships

$$C = T\, C_T \tag{1}$$
$$S = T\, S_T \tag{2}$$

A wrist rotation parameter w is useful for defining a coordinate system aligned with the principal axes of the mounting flange, as indicated by the coordinate system H in Figure 4. To measure this rotation parameter we place the housing in a horizontal position (as indicated by a spirit level), then interrogate the manipulator as to the current matrix value of T. We set w to the amount of rotation about the z-axis that must be applied to T to make its x-axis correspond with that of H. Let W be a homogeneous transform matrix embodying that rotation, so that we may write

$$C_T = W\, C_H \tag{3}$$
$$S_T = W\, S_H \tag{4}$$

C_H and S_H are the poses of the camera and projector, described in the coordinate system of the housing.

The camera is held in place by a 1/4-inch camera mount screw. When the screw is loosened, the camera can rotate about an axis passing through the mounting hole. Figure 5 shows this axis. We assume that the camera's principal ray intersects that rotation axis, and we define that intersection as the *camera center*. However, it is the *lens center* whose pose C_{ll} describes. We will derive C_{ll} as the product of three transformations: a translation from the housing mounting center to the camera center, a rotation to orient the principal ray, and a translation along that ray to the lens center.

The relationship between the camera center and the housing center can be measured accurately with a ruler. The distance of the retina behind the camera canter can be approximately measured, and the distance of the lens in front of the retina can be calculated from the focal length of the lens and the focus of the camera. These numbers can be coded as constants at setup. But the direction of the principal ray is variable and must be corrected each time the mounting screw is disturbed.

To determine the principal ray of the camera we take pictures of the same object from two different viewpoints. The camera is placed pointing downward, and a small target such as a coin is placed on the tabletop in front of the camera. (The laser diode must be turned off for this part of the calibration.) By interrogating the robot as to its current pose, and using any previously-estimated matrix value for $\mathbf{C}_H$, the position of the lens center is calculated. (It does not matter if the assumed matrix value for $\mathbf{C}_H$ is off by some small amount.) A picture of the target is taken. Based on an assumed perspective matrix P and the known height of the tabletop, we can calculate the distance from the target to the principal ray, i.e., to the point on the tabletop at the center of the TV screen. We use this distance to calculate a new, hypothetical lens center that the camera would have to move to if it were to center the image of the target without altering the camera's orientation.

Now the manipulator moves the camera away from the target, keeping the same orientation but increasing the distance between the camera and target, and the procedure above is repeated to find a second hypothetical lens center that will center the target's image. A line joining the two hypothetical centers defines the principal ray. If the initial assumption as to $\mathbf{C}_H$ was correct, that line will be perfectly parallel to the camera's z-axis, but in general, some correction will be necessary to make it so.

The above procedure can account for rotation about the camera's y-axis (due mainly to rotation about the camera mounting screw) and about its x-axis (perhaps because of the CID chip not being directly in line with the lens). Rotation about the camera's z-axis (due to errors in the placement of the CID chip), which we call ψ, must be measured in another way.

ψ and the two perspective projection scaling parameters F_x and F_y are measured together in the following procedure: The camera is aimed straight down with its lens center at some known height h above the tabletop. If the image coordinates of a contrasting target are determined, the camera moved parallel to the tabletop a known distance, and the coordinates of the target measured again, the size of the apparent motion of the target image should be predictable from the perspective scaling parameters. Actually, we assume the relationship between camera position (x_c, y_c) measured in the coordinate system of the camera's initial placement, and the image coordinates (x_i, y_i) of a fixed target is in the form of the following:

$$\begin{aligned} x_i &= \alpha x_c + \beta y_c + x_0 \\ y_i &= \gamma x_c + \delta y_c + y_0 \end{aligned} \tag{5}$$

where

$$\begin{aligned} \alpha &= -(F_x / h) \cos\psi \\ \beta &= -(F_x / h) \sin\psi \\ \gamma &= (F_y / h) \sin\psi \\ \delta &= -(F_y / h) \cos\psi \end{aligned}$$

and x_0 and y_0 are the image coordinates of the target in the camera's initial position. Pictures of the target are taken from several positions (usually with the image in the four corners of the screen), and a least-squares procedure is used to estimate α, β, γ, and δ. These, in turn, are used to solve for F_x, F_y, and ψ.

The most accurate way to calibrate the projector pose is to use the camera. With the light-stripe projector turned on, the camera and projector are positioned above the table top. The image of the stripe will be a straight line. Given the camera calibration parameters and the known height of the tabletop, the locations of two points at the ends of the stripe may be determined. (The procedure depends on deriving a collineation matrix, but using the known plane of the tabletop instead of the plane of the projected light.) These locations can be expressed in coordinate system H, that of the camera-and-projector housing. If this procedure is repeated at more than one height above the tabletop, several point locations will be determined, all relative to H, and these will all lie in the plane of light, which remains in a constant relation to the housing. A least-squares procedure can be used to solve for the coordinates of that plane.

Any point that lies in the plane of light may be chosen as the origin of coordinates of S_{If}, the the projector pose, without making any difference as far as the camera/projector collineation matrix is concerned. We choose as the origin the point in the plane closest to the the location of the projector aperture, as measured with a ruler. The orientation of S_H is set to make its Jt-axis perpendicular to the plane of light and its z-axis pointing outward, i.e., perpendicular to the yaxis of the housing coordinate system H.

Evaluation

As of the deadline for submission of this report, all hardware and software components of the system are working, but we have not yet been able to make systematic measurements of the system's accuracy. We can, however discuss some factors that we believe will limit the resolution and accuracy, and place some limits on the accuracy to be expected.

The depth resolution of any triangulation system is equal to the pixel resolution at the surface beieg measured, divided by the tangent of the angle of convergence between the camera and projector lines of sight In a typical experiment the camera might be 30 cm from the work, where 1 pixel corresponds to 0.4 mm. The convergence angle for that height is 36 degrees, which gives a depth resolution of 0.5 mm. At a range of 50 cm, the pixel resolution is 0.66 mm, the convergence angle is 23 degrees, and the resulting depth resolution is 1.6 mm.

The sort of errors we call *aiming errors* will probably detract more from overall accuracy than the limitations of depth resolution. Experience has shown that we can measure angles to about oee-half degree. For example, without loosening the camera mounting screw, repeated calibration of the camera pose yielded i set of angular corrections that had a standard deviation of 0.43 degrees. If the camera angle is mlscalibrated, the accuracy of depth (measurement in 2) is not likely to suffer, because the way we calibrate the projector will compensate for any errors in the camera. But measurement of *x* and *y* will be affected. One half degree at 30 cm range corresponds to an error of 2.6 mm. The same angular error at 50 cm ^ves rise to a 44 *mm* error.

The accuracy of three-dimensional measurement will be dependent *on* the distance from the range finder to the surface being measured. It is probably fair to say that fee overall accuracy at a working distance of 30 cm will be in the neighborhood of 3 mm.

References

1. G. J. Agin & T. O. Binford. "Computer Description of Curved Objects/" *IEEE* Tranx *Computers C-25** 4 (April 1976), 439-449.

1 G. J. Agin. Real-Time Control of a Robot with a Mobile Camera. 9th International Symposium oa Industrial Robots, Society of Manufacturing Engineers, Washington, D. C March, 1979, pp. 233-246,

3. R. Barahill and R. Riesenfeld. *Computer Aided Geometric Design.* Academic Press, New York, 1974

4. R. O. Duda & P. E Hart *Pattern Classification and Scene Analysis.* Wiley fotcrscicnce, 1973.

5. W. M. Newman & R. R Sproull. *Principles of Interactive Computer Graphic** Mcgraw-Hill, 1973,

6. *Proa NSF Workshop on Representation of Three-Dimensional Objects,* Philadelphia, May 1-2,1919.

7. R. J- Popplcstone, et aL Forming Models of PIanc-and-Cy!indcr Faceted liudica *fnm* Light Stripes. Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia. USSR, August 1975, pp-664-668.

8w C Rosen ct al. Machine Intelligence Research Applied 10 Industrial Automation. Ktghih Repot a cefc. RcpL Grant APR75-13074, SRI Project 4391, SRI International, Mcnlo Park, California. Augttf, J978.^

9. Y. Shirat & M. Suwa. Recognition of Polyhedrons with a Rangcfindcr. Pftic Second International j:nnt Conference on Artificial Intelligence, London, 1971, pp. 80-87«

10. G. J. Vandcrbmg et at. A Vision System for Real Time Control of Robots, 9th ffifemjtrruJ S:Tf.p on Industrial Robots, Washington, D. G, March, 1979, pp$_w$ 213-231

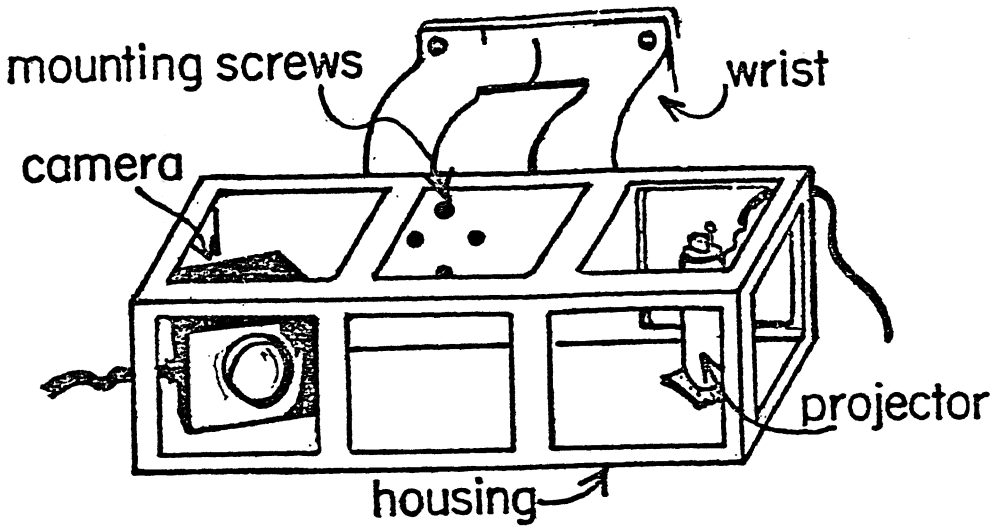


Figure 1: Camera and Projector in Housing

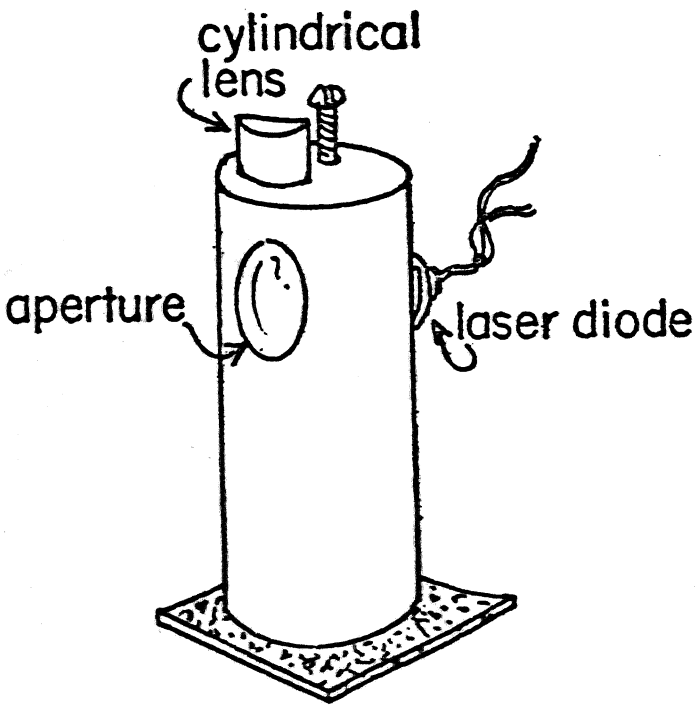


Figure 2: Close-up of Light-Stripe Projector

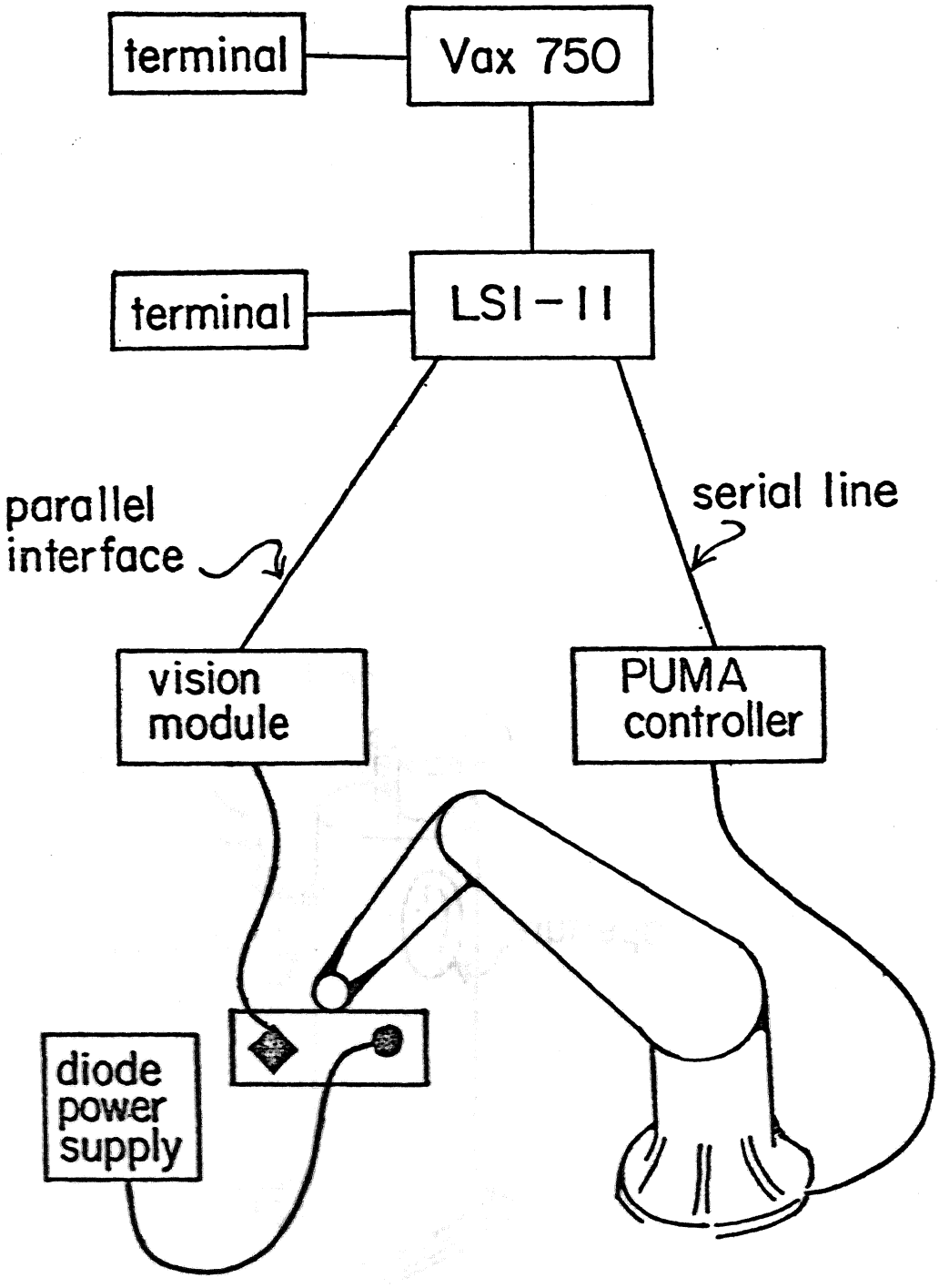


Figure 3: System Block Diagram

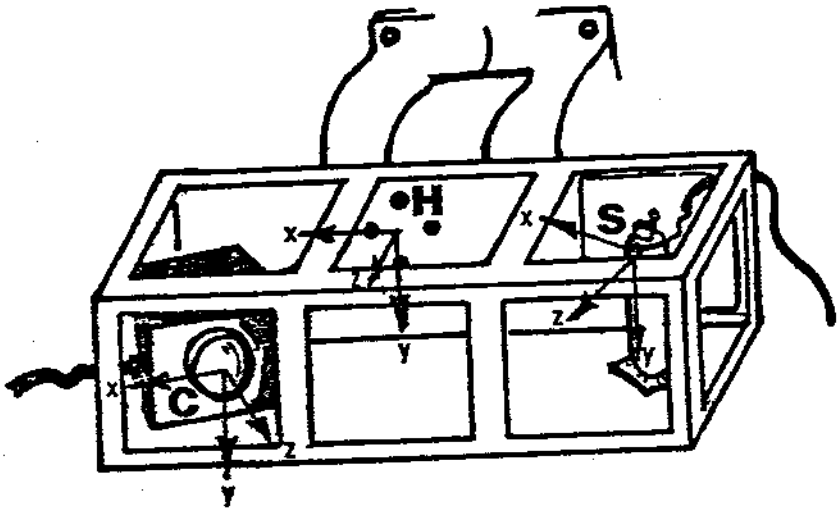


Figure 4; CoordinateSystemsforHousing.Camera.andProjector

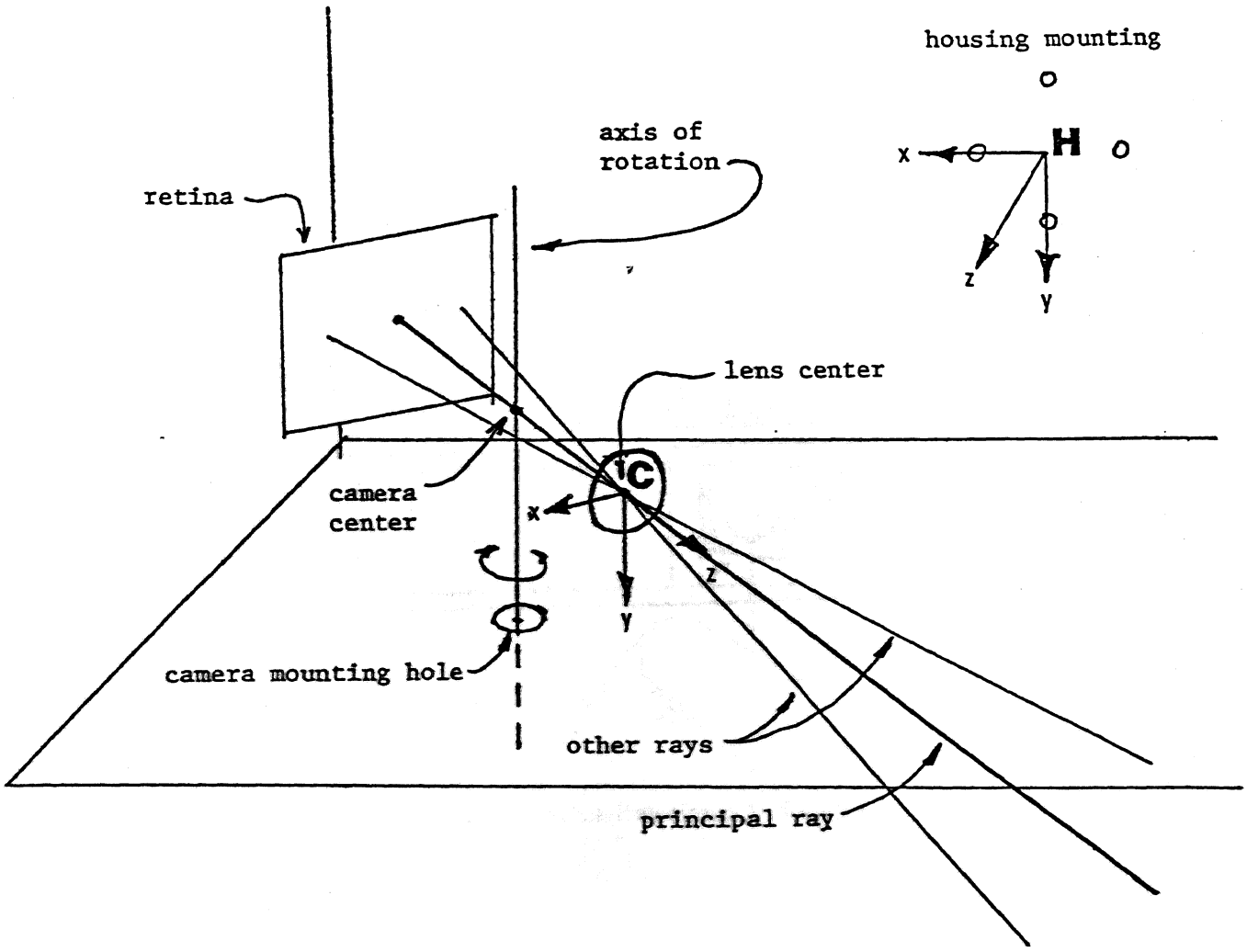


Figure 5: Camera Calibration

References

[I] Robert C. Bolles.
Robust Feature Matching through Maximal Cliques.
In *Imaging Applications for Automated Industrial Inspection and Assembly,* pages 140-149. Society of Photo-Optical Instrumentation Engineers, Washington, D.C., April, 1979.

[2] M. Briot.
The Utilization of an 'Artificial Skin' Sensor for the Identification of Solid Objects.
In *Ninth International Symposuim on Industrial Robots,* pages 529-548. Society of Manufacturing Engineers, Washington, D. C, March, 1979.

[3] M. Briot, M. Renaud, and Z. Stojiljkovic.
An Approach to Spatial Pattern Recognition of Solid Objects.
IEEE Trans. Systems, Man, and Cybernetics SMC-8(9):690-694, Sept, 1978.

[41 *PS 300 User's-Manual*
Version P4.V01 edition, Evans & Sutherland Computer Corporation, P. O. Box 8700, Salt Lake City, Utah 84108,1982.

[5] G.H. Golub and C. Reinsch.
Singular Value Decomposition and Least Squares Solutions.
In J.H. Wilkinson and C. Reinsch (editors), *Handbook for automatic computation.* Springer-Verlag,1971.

[6] Leon D. Harmon.
Touch-Sensing Technology: A Review.
Technical Report MSR80-03, Society of Manufacturing Engineers, Dearborn, Michigan, 1960.

[7] Leon D. Harmon.
Automated Tactile Sensing.
International J. *Robotics Research* 1(2):3-32, Summer, 82.

[8] Robert B. Kelley, et al.
A Robot System Which Acquires Cylindrical Workpieces from Bins.
IEEE Trans, Systems, Man, and Cybernetics SMC-12(2):204-213, Mar/Apr, 1982.

[9] T. Okada and S. Tsuchiya.
Object Recognition by Grasping.
Pattern Recognition 9(3):111-119, October, 1977*

[10] Walton A. Perkins.
A Model-Based Vision System for Industrial Parts.
/£££ Trans. Computers :126-143, February,, 1978.

[II] George Stockman, Steven Kopstein and Sanford Benett
Matching Images to Models for Registration and Object Detection via Clustering.
/£££ Trans. Pattern Analysis and Machine intelligence PAMt-4{3):229-241, May, 1982.

[12] G. J. Vanderbrug, J. S. Albus, and, E. Barkmeyen
A Vision System for Real Time Control of Robots.
In *Ninth international Symposium on Industrial Robots,* pages 213-232. Washington, D. C, March, 1979.