A Tactile Sensing Strategy for Model-Based Object Recognition

R. E. Ellis

COINS Technical Report 87-96

Laboratory for Perceptual Robotics
Department of Computer and Information Science
University of Massachusetts, Amherst MA 01003

ABSTRACT

An outstanding problem in model-based recognition of objects by robot systems is how the system should proceed when the acquired data are insufficient to identify the model instance and model pose that best interpret the object. Such a situation can arise when there are multiple model instances that could be interpretations of the object, or when there are ambiguous poses of a given model instance.

This work proposes a generic method for automatically finding a path along which the robot could move a tactile sensor, so that the robot system can uniquely and efficiently identify the object. The problem framework is defined, a methodology for finding paths is proposed, and an evaluation of the costs and benefits of sensing paths is presented, all of which must be done in the presence of geometric uncertainty about the possible locations and orientations of the object.

The two-dimensional problem is solved by a projection-space approach, in which the optimal sensing path is found by efficiently searching through the sets of paths passing through each object face. A path is sought which distinguishes as many distinct interpretations as possible, subject to design constraints. It is shown that employing realistic assumptions the problem is tractable, and that for the two-dimensional case the solution time is comparable to the robot motion time.

For the three-dimensional problem, an analysis of the structure of the path parameter space shows why the problem is inherently difficult. Several alternative solutions are examined, and a taxonomy of approaches classifies related work into a more general hierarchy of problem decompositions.

Originally presented as a Ph.D. dissertation to the Graduate School of the University of Massachusetts at Amherst, and supported in part by the Office of Naval Research under Contract N00014-84-K-0564.

Contents

Acknowledgements

I would like to thank all of the LPR and VISIONS members who have helped me along the way: Ken Overton and Steve Begej, who first piqued my interest in tactile sensing; Miles Lane, who assisted with hardware in many ways; Judy Franklin, Benny Murah, and T.V. Subramanian, who patiently helped with controlling the seemingly uncontrollable; P. Anandan, Les Kitchen, George Reynolds, and Rich Weiss, who listened to more higher-dimensional geometry questions than anyone should have to; and Ron Arkin, for just being around when I needed to talk.

I would also like to recognize the contributions of my committee members: Ted, for control; Robin and Al, for their invaluable general guidance and checking the math; but most of all, thanks, Ed, for support through the tough times.

This dissertation has been brought to you in part by grants from the Natural Sciences and Engineering Research Council of Canada, the National Science Foundation, and the Office of Naval Research.

Chapter 1

The Problem and the Approach

"...that palter with us in a double sense"

Contemporary industrial robots are limited in the tasks they can perform. One of the more important reasons for their inflexibility is that they are deficient in their ability to sense the world, and in their ability to process and respond to sensory data.

This work addresses the question of how a robot, equipped with a tactile sensor, can recognize and locate an object in its workspace. Specifically, we consider the situation in which some tactile data about the object are already available, but the data do not uniquely determine the object and its pose. The problem is to acquire and process new tactile data in a sequential and efficient manner, so that the object can be recognized and its location and orientation identified.

The object recognition method used here is *model-based* – there is a set of known object models, and for a given object the goal is to determine which model instance best describes the object. The data for this system are the three-dimensional positions of surface points, and the normals to the surface at these points (all surfaces are modelled as planar patches). An *interpretation* of a set of data is an assignment of these data to faces of a particular model. If the interpretation is valid, the assignment of data to faces yields the rotational and translational parameters that determine the *pose* of the object in space.

Object recognition in this system is performed as a sequential, data-driven process. The sensor data are processed by applying local geometric constraints to pairs of data and possible assignments of model elements (faces, edges, or

vertices) to these data in order to generate *feasible* interpretations. New data reduce the number of feasible interpretations; if there are enough data, and they provide the right information, then the object can be recognized and its pose uniquely determined in a model-testing step.

Suppose, however, that the data either fail to identify the object model instance, or for a given model instance there are multiple poses which are consistent with the data. The problem, stated in the large, is how the robot is to acquire new sensor data so there are fewer interpretations – ideally, so that there is only one interpretation. More specifically, the problem addressed here is how to determine the path of a tactile sensor so that as it moves along the path, the data will provide the recognition system with information about model faces that have not yet been sensed, and how to evaluate the costs and benefits of this path.

The problem of acquiring new tactile data occurs in the context of the more general problem of object recognition. Our system for the recognition of objects from tactile data has the following overall structure:

1. Acquire the initial set of tactile data.

2. Interpret these data by sequentially applying local and global geometric constraints between the data and the object models, i.e., find the possible translations and rotations of each model that are consistent with the data.

3. Repeatedly:

   - Find a path along which to move a sensor.
   - Execute the path, stopping when the sensor comes into contact with an object.
   - Interpret the acquired datum: either it identifies the object, or it reduces the set of interpretations to a new, smaller set.

The approach used to solve the problem of intelligently acquiring data will be a *generic* one, in that model-specific strategies will be eschewed. The set of

model instances are considered to be a set of model faces scattered in space; those faces which provide no novel information or which can obstruct execution of the sensor path are to be avoided, and the remainders are candidates for being sensed. Solution of this problem, phrased in terms of computational geometry, requires representing physical properties of tactile sensors in geometric terms and finding a path with respect to these constraints. For example, all tactile sensors have a lower limit on their spatial resolution (they cannot provide perfect information), they occupy a certain volume and thus cannot be moved through arbitrarily tight obstacle freeways, they skid off surfaces when striking them at too oblique an angle, and it takes some time to move them and extract the requisite features. All of these must be accounted for in a system which directs a robot to acquire new tactile data.

Furthermore, all of the analysis must be conducted in the presence of uncertainty. The data are presumably gathered by real devices, which results in *data error*, that is, a difference between the true object position or local surface normal and the data that are actually sensed. This data error leads to *pose uncertainty*, which is the possible deviation of the true position from the nominal calculated position. Throughout, we attempt to incorporate these uncertainties into our approach, yielding a system that is robust.

In our research paradigm we suppose that there is a single object in the robot's workspace, and that some initial data-acquisition strategy, e.g., regular or random sensing, has been used to gather tactile data. These tactile data are contact points on the object's surface; each datum is a pair of vectors, representing the approximate location and local surface normal of that part of the object. A number of object models can fit these initial data, and the problem we seek to solve is how efficiently to acquire new tactile data to determine uniquely the model type and location that best describe the object.

Briefly, our acquisition methodology is to examine unsensed portions of the object that is in the workspace. When there are multiple interpretations of the

initial data, e.g., several different models could fit the data, there are a number of faces from different models that have not yet been sensed. If we imagine the interpretations to be superposed, then some of the unsensed faces "line up", i.e., if a sensor placed on the tip of a long rod were moved along a special line then it would pass through (or *pierce*) these faces. Since only one of these model interpretations can really occur, we can tell which one is the correct one by determining which face was hit, i.e., which position and local surface normal were actually detected by the sensor. Figure 1.1 shows a simple two-dimensional object, and several superposed interpretations of some tactile data; the thick line indicates an automatically planned linear path that would contact each interpretation. If executed, this path would uniquely determine which interpretation of the original data was the correct one.

Identification of an object from ambiguous data can be accomplished if a line can be found that passes through an unassigned face of each valid model interpretation (*modulo* sensor limitations). Our method for finding these lines involves changing the representation of the problem, and asking what sheaf of lines can possibly pass through each unassigned face of each model. The intersection of the sheaves of a set of faces is the sheaf of lines that pass through *all* of the faces. We will show below that it is possible to find an element of this intersection (and thus find a sensing path for the robot) in an efficient and general manner for two-dimensional objects, and indicate feasible approaches for three-dimensional objects.

1.1 Paradigmatic Assumptions

The problem – automatically determining how to acquire new tactile data, when the current data cannot be uniquely interpreted – is somewhat vague and ill-defined. In order to focus on a manageable subset of the problem, four restricting assumptions will be made.

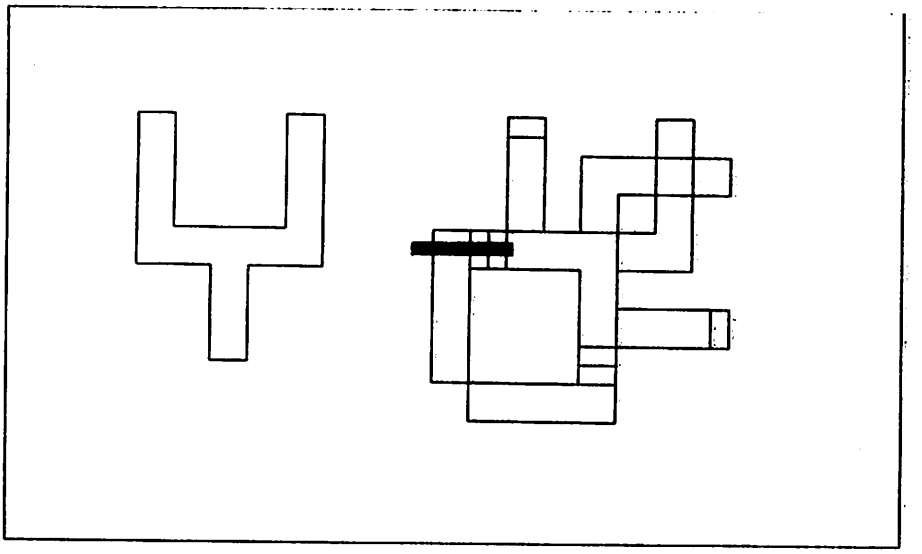

Figure 1.1: An object model, four interpretations of data, and a path that identifies the true interpretation of the original data.

The first assumption is that we wish to be intelligent about data acquisition, asserting that simply waving a tactile sensor about in the workspace is not a particularly effective strategy for object recognition. Hence, we will try to employ what we know about the initial data, the limitations of the sensor and manipulator, and the geometry of the instantiated models to better plan a sensor path. This assumption has an important side effect: the methodology cannot be selected independently of the general properties of the sensor.[1] Our assumptions about the sensor are that it is relatively small and is capable of moderately accurate parameter estimation; thus, our methodology is to examine faces, which are easiest to contact with a small sensor moving in an uncertain environment. Another effect of intelligent acquisition is elimination of random, or workspace-scanning, algorithms for acquisition of any but the initial data.

The second restricting assumption is that we are concerned solely with the acquisition of new data, and *not* with how best to use the existing data. These

[1] Nor should a sensor be selected without some idea of how it is to be used, as a general rule of engineering robotic systems.

are two quite different problems, and one should not automatically expect that one can solve both problems with the same approach. For example, the problem of optimizing the motion of a sensor in a robot workspace requires knowledge of the obstacles, arm configuration, and sensor capabilities; on the other hand, optimizing use of existing data can be cast as a search, where one is restricted to a fixed set of data and must choose (or order) to best accomplish the recognition process.

The third restricting assumption is that the new data are to be acquired, wherever possible, from faces that are not currently assigned. Unsensed portions of previously sensed faces can provide crucial distinguishing information at times, but in general we will suppose that whenever possible we seek to pierce at least one unknown face for each path execution. This is not a particularly critical assumption, but is useful in speeding the search for sensing paths.

The final restricting assumption is that sensing will occur only along a *linear* path. This assumption is justified by the physical problems that arise when one attempts to construct a sensor that is arbitrarily small, and that can be moved along an arbitrary three-dimensional path in a workspace that contains an unknown object. Thus, any sensing path is restricted to be a straight line, must begin outside the object, and must not be such that the sensor comes into contact with a face that has already been assigned in an interpretation, for all interpretations.

1.2 Related Work

There is relatively little work, old or recent, on ways of intelligently and automatically acquiring tactile data for the purposes of object identification. One class of related work is exemplified by [Allen & Bajcsy,1985] and [Luo *et al.*,1984a]. The former work used vision to reduce the number of possible models, and used surface-following to verify the model instance; this approach, while effective in

the experiments they describe, is very time-consuming. The latter work also used vision initially, and then simple tactile features, to search through a decision tree; the sensing strategy is very simple, consisting of repeated rotation of the sensor about the object. Although it is effective in simple cases, the authors point out its shortcomings in dealing with smooth or highly symmetric objects.

There is also some recent work that is closely related to ours. This other class of related work is within the same research paradigm as the present work, and is represented by [Grimson,1986] and [Schneiter,1986]. Both authors attack the same problem presented here, but in different ways.

Grimson's approach is very similar to the present one, in that he uses projections of faces onto starting lines (or in three dimensions, starting planes), and examines overlaps to determine how many could be pierced by a given path; however, he does not attempt to optimize simultaneously over the direction and positional parameters. Schneiter uses a very different approach, in which one seeks regions in which a face from each interpretation is represented; this can be implemented in a very fast scheme, but occasionally fails to find paths which can identify the object (where the present scheme can find such paths). Below, in Section 5.4, we will contrast their work to ours in more detail.

We might also note some research which might appear, at first glance, to be related but which address different problems. One such group of work is that of Cole and Yap [Cole & Yap,1983], where there is an ambiguity in the word 'identify' which can lead to some confusion. In their terminology, identification consists in the determination of the location and orientation of object faces in the plane; we, by contrast, use the word to mean 'uniquely distinguish among several interpretations', which is a very different sense.

Another group of work to be distinguished from our is that of Stansfield [Stansfield,1987], in which a tactile sensor is guided by vision and force to probe an object. Stansfield, however, is concerned with the acquisition of surface description primitives which may then be employed in recognition and manipulation; there

is no object model present in her system, and thus the path planning strategies are very different. Our work supposes a set of object models, and reasons about their relationships to determine sensing paths.

In other domains, such as visual object recognition, model-specific methods for improved recognition performance have been used with some success. These methods typically preprocess the database of known models to find features, or groups of features, that are useful in the recognition process. Examples of this approach are the *Local-Feature-Focus* method of [Bolles & Cain,1982], and the Bayesian signal-detection method of [Turney *et al.*,1985].

One drawback of feature-based approaches is that the addition of a model to the database usually requires recomputation of the model-specific strategies, especially if the feature-based approach uses statistics of model features (which can change significantly with the addition of a single model). Another drawback is that it is not clear that such methods degrade gracefully as uncertainty in the initial data increases, or as the number of known models rises. Our approach, which is generic, seems to exhibit both generality and graceful degradation. None the less, feature-based approaches can be a powerful and speedy tool in appropriate applications.

1.3 Nomenclature

The work described here employs analytic geometry of two-, three-, and four-dimensional manifolds, and in addition uses abbreviations for various values. The standard representation for geometric objects will be as follows:

- **Points and Direction Vectors** are simply n-dimensional vectors representing a displacement from the origin of the Cartesian coordinate system, and a direction, respectively. All vectors that are accented with a circumflex are of unit length; so a is simply a vector, while $\hat{a}$ is a vector that must be of unit length.

- **Line Segments** are represented as an ordered pair, the first element being a point on the line and the second being a direction *which is not necessarily of unit length*. A line is given parametrically by the equation $L(\lambda) = P + \lambda \cdot D$ where P is the point on the line, D is the direction, and λ is the line parameter. For convenience in computing distances and intersections, we restrict the value of the line parameter to lie between 0 and 1 for a line segment.

- **Polygons** are represented as a list of points which constitute the vertices of the polygon. It is always assumed that the list of vertices constitute a simple polygon, i.e., the points are coplanar, no three consecutive points are colinear, and the polygonal edges intersect only at vertices. To find the normal of the plane in which the polygon lies, the right-hand rule is employed: if the first three (non-colinear) points are V_1, V_2, and V_3, then the plane normal lies in the direction of $(V_2 - V_1) \times (V_3 - V_1)$.

- **Planes** are represented as a list consisting of a point that lies in the plane, and the unit normal of the plane. If the point is P and the normal is $\hat{Z}$, the point X lies on the plane if it satisfies the plane equation is $\hat{Z} \cdot (X - P) = 0$. Note that because of the procedure used to match sensor data and planar faces, there is no ambiguity in the direction of the plane normal: the two equations $\hat{Z} \cdot (X - P) = 0$ and $\hat{Z} \cdot (X + P) = 0$ represent different planes.

- **Transformations** are expressed as separated rotations and translations. The transformation of a vector a to a vector a' by the formula $a' = R \cdot a + T$ is the rotation of the original vector about the origin by the orthonormal rotation matrix R, followed by a translation of T.

Table 1.1 summarizes this nomenclature, and gives the standard usage of other variables which will be described below.

Table 1.1: Notation

a, b, c, d, X, P_i	:	Position vectors; points
$\|\|a\|\|$	:	The length of the vector a
p_i	:	Distance from a plane to the origin
V_i	:	Vertex of a polygon
$\hat{N}_i$	:	Unit normal vector of surface, as sensed
$\hat{U}_i$	:	True unit normal of surface
$\hat{Z}_i$	:	A unit normal vector (of a model face), in model coordinates
ϵ_i	:	The maximum distance error of point P_i
δ_i	:	The maximum angular error of normal $\hat{N}_i$
F_i	:	A model face
ς, η, D_i	:	Distance, difference, or direction vectors
R	:	A rotation matrix
r	:	An axis of rotation
$\theta, \phi, \psi, \omega$	:	An angle, in radians
T	:	Translation vector, or offset
s, t	:	Scalar parameters
$L_i(\lambda)$	:	Parametric form of a line in space
Q_P, Q_L	:	Parameter of a point or line in Projection space
θ, ϕ	:	Angles in the direction of the **X** and **Y** axes, respectively
α, β	:	Tangents, or slopes; $\alpha = \tan\theta$ and $\beta = \tan\phi$
ϵ_{sensor}	:	The minimum spatial distance resolvable by a sensor
δ_{sensor}	:	The cosine of the minimum angle resolvable by a sensor
Ω	:	The maximum acceptable angle between a sensor and a surface

1.4 Tactile Sensors and Tactile Sensing

For the purposes of this research, a **tactile sensor** is defined as a device that provides three-dimensional information about a surface by direct physical contact. A sensor is typically attached to an end-effector of a robot arm, so that it can be moved through the environment under computer control.

The object recognition and localization scheme we use here requires relatively simple, sparse data about the surface of an object. It is assumed that the data are not perfect, and that there is an explicit model of the kinds of error that may be present in the data. The scheme requires that the sensor return the three-dimensional position of a point of contact and the local surface normal at that point. These two values have error bounds associated with them, and the errors are presented to the recognition scheme as parameters. A tactile sensor can be used to acquire data for recognition provided that:

- A point on the sensor can be identified as contacting a surface.
- An error bound on the location of this point can be calculated.
- The local surface normal at this point can be deduced from the raw sensor data.
- An error bound on angle of this local surface normal can be calculated.

There are many transduction methods, sensor geometries, and feature extraction algorithms that can be used in the process of inferring three-dimensional surface information from direct contact. At the present stage of technological development, tactile sensors may be grouped into two major classes: those that provide information about some components of the net force, and those that provide information about the distribution of strains in the sensing medium as it is brought into contact with the external surface. Henceforth, these will be referred to as *force sensors* and *tactile array sensors* respectively, the latter because they

usually report some regular array of strain information. Both classes of sensor are capable, in principle, of providing the data required by the present system.

1.4.1 Force Sensors

Force sensors provide information about the forces acting between the supporting member of the sensor and the sensing member of the sensor. The geometries, number of force components, and force resolution of this class of sensor vary considerably.

Theoretically, in three-dimensional space, there are at most six independent *wrenches* – forces or torques – that can be sensed. If a local Cartesian coordinate system is erected within the sensor, these wrenches can be expressed as the forces directed along the axes, and the torques about the axes.

The most complex kind of force sensor provides all six wrenches. Typically, these are relatively large devices, and are mounted between the end of the robot arm, and the end effector or gripper, acting as wrist sensors to provide feedback for manipulation. It is possible to miniaturize a six-wrench sensor and fit it to the tip of a robotic finger; this requires solving some interesting engineering problems, but results in a sensor capable of use within our paradigm.

Much of the work below will describe recognition and planning sensing paths in two dimensions. In order to sense in two dimensions, a very simple but accurate construction is to instrument a beam, of square cross-section, with strain gauges. Equipped with a circular tip, the beam's bending directly encodes the net direction of contact. Because of the known geometry, the position of contact can also be deduced (assuming that there is a single point of contact). This sensor design has been used in our experiments, and is more fully described in Appendix A.

We may thus conclude that force sensors are capable of providing the position and local normal information needed to conduct object recognition in the

proposed manner. They have the advantage of sensitivity, but may be difficult to manufacture when the goal is sensing small objects with high accuracy.

1.4.2 Tactile Array Sensors

Tactile array sensors provide information about the distribution of strains in some sensing medium when it is brought into contact with an external surface. As noted in [Fearing & Hollerbach,1984], the strained medium is usually a homogeneous, isotropic, noncompressible, elastic, layer which either abuts a sensing layer or has strain transducers as an integral component of the medium. Deducing a point of contact, and the local surface normal at that point, requires analysis of an array of low-level data; this is a more involved operation than is required by a force sensor, which provided the necessary information more directly. A tactile array sensor, however, can provide a great deal more information about the nature of the contact with the object's surface than can a simple force sensor.

There is a great variety in the nature of transduction methods that have been employed in tactile array sensors. The first transducers sensed changes in electrical resistance as a material is strained; [Hillis, 1984] and [Overton,1984] employed conductive elastomers and resistive bridges, while [Raibert,1984] used an elastomer directly contacting a VLSI transducer/processor chip. Electrical capacitance was sensed in [Boie,1984], [Seigel *et al.*,1987], and [Fearing,1987], providing greater sensitivity but with increased susceptibility to noise induced by external electric fields. Magnetic properties, such as magnetostriction [Luo *et al.*,1984b], can be used to increase sensitivity and reduce noise susceptibility, but at a cost of array element density. Electrical charge produced by piezoelectric polymers [Dario *et al.*,1984] rates highly as a transduction technique, but can only be used in an active mode since piezoelectricity is generated only by changes in strain, i.e., there is no DC response.

Various optical phenomena have also been used in tactile array sensor technology. Perhaps the simplest is optical beam eclipsing [Rebman & Morris,1983].

Optical phase shifting [Jacobsen *et al.*,1984], produced by reflecting a light beam off a birefringent material, is extremely sensitive, but is not as rugged as might be desired. One technology that produces very compact and sensitive arrays relies on the frustration of total internal reflection [Begej,1985], in which lateral internal reflection in a transparent plate is frustrated by the presence of a microtextured material; its principal drawback is large hysteresis caused by adherence of the rubbery asperities to the transparent plate.

The data required by our object recognition system is provided in a direct manner by a force sensor, but requires considerable inference from the raw data of a tactile array sensor. Some difficulties with using the inferred contacts are that rapid processing of the data is required to achieve servo control of the robot, and it is difficult to deduce the error bounds on the inferred position and local surface normal.

The amount of information provided by a tactile array sensor can, however, be used to considerable advantage. Aside from being able to infer multi-point contact, an array sensor can be used to infer the presence and location of many kinds of tactile features. Some features that can be deduced are: the deformation of an object; its local texture; whether the contact is best described as point, line, arc, or surface; and the radius of curvature of an edge or surface (cf. [Ellis,1984], [Ellis,1986] for examples of tactile feature extraction).[2]

Texture and local contact geometry can be used in our system with a simple modification of the recognition scheme. However, all that we require here is a point of contact with the surface of an object, and the local surface normal at this point. We will now examine how such simple data may be used to recognize objects for which polyhedral models are available.

---

[2]These are all *local* features, and suppose that the tactile is much smaller than the object that is being sensed. See [Overton,1984] or [Togai *et al.*,1984] for approaches that can be used when the sensor and object are of comparable size.

Chapter 2

Model-Based Recognition and Localization

"Is this a dagger which I see before me?"

Model-based recognition may be defined as the recognition of an object as an instance of a known model. The task that such a recognition system must perform, then, is to accept sensor data and determine from which object the data have been derived. The response of the recognition system is the model class, if the object is one of those already known.

The above description considerably simplifies the issues. A recognition system must often deal with data that are erroneous, that come from more than one object, or that do not adequately determine of which model the object is an instance. Furthermore, the recognition system should also indicate the location and orientation – the *pose* – of the object.

In this research the data are assumed to be derived from some sort of tactile sensor, which senses the surface of an object by physical contact. The method used here requires relatively simple data, and assumes that these data are imperfect; thus, there is an explicit model of error incorporated into the recognition and localization procedures. A tactile sensing system is assumed to provide the following information about a contact:

- A point on the sensor can be identified as contacting a surface.
- An error bound on the location of this point can be calculated.
- The local surface normal can be deduced from the low-level sensor data.
- An error bound on this local surface normal can be calculated.

Since the goal of the present research is to examine sensing strategies, some simplifications of the general object recognition problem may be made in order to focus on the issues of interest. It will be assumed that all of the data come from the surface of a single object, and that the error associated with the position and surface normal of the data *completely* describe the errors[1], i.e., there are no spurious data.

The remainder of this chapter is devoted to a discussion of the object representation, recognition, and localization methods, and an approach to the estimation and management of pose uncertainty. The basic methods used in the present work build upon the foundation laid in [Grimson & Lozano-Pérez,1984], which describes a method for model-based recognition and localization from sparse range or tactile data; that work is in turn an extension of the approach used in [Gaston and Lozano-Pérez,1984]. To this, we will add an analysis of how sense data error leads to uncertainty in determining the location and orientation of an object, because tight bounds on the pose uncertainty are critical to the subsequent problem of planning distinguishing paths.

2.1 Object Representation

Objects in the present system are represented as polygons in two-dimensional space, and as polyhedra in three-dimensional space. Because the data are of points on the surface of an object, the most natural representation is to model the object as a set of polygonal faces (for 3–D objects), edges (called faces in 2–D), and vertices.

Each model has associated with it a local coordinate system. A vertex is represented as a vector point, offset from the origin; an edge is a pair of points,

[1] Among the sources of error are physical inaccuracies of the sensor, problems in inference of the position or local normal, passive compliance of the manipulator/sensor system, and approximation of a curved surface as a linear model face.

and a polygonal face is a list of points. The right-hand rule is employed to deduce the normal of the plane in which a polygon lies, i.e., the cross product of the first polygonal boundary (the difference between the second point and the first) and the second polygonal boundary (the difference between the third point and the second) points *outward.* Another way to think of this convention is that if the first vertex point of the polygon is the origin, then the cross product of the vectors given by the next two vertices must point outward.

The object representation is not complete, in that the set of faces, edges, and vertices need not form a proper closed volume. This is partly a matter of convenience, and partly a matter of parsimony. It is often convenient to represent a complicated polygonal face as the union of simpler faces, in which case the abutting edges are artifactual, and could not actually be sensed. Regarding parsimony: if a surface can not physically be sensed, e.g., it is an interior face of a convoluted bottle, then there is no *need* to represent it; indeed, representing a face that can never be matched to a datum increases computation time unnecessarily.

The representation used in [Grimson & Lozano-Pérez,1984] contains only faces. Our representation adds edges and vertices which, although they are less likely to be sensed, provide very powerful constraints on possible matches. Faces, however, remain the most important part of the object representation, and most of the discussion will concentrate upon them (for brevity, as well as for their importance).

There are, of course, many other object representations that might have been used. Aside from representations used in computer graphics, e.g., many of those cited in [Requicha,1980], there are a number which have been employed in model-based recognition. Extended Gaussian images [Horn, 1983] (sometimes called needle maps) have been described in [Ikeuchi,1981] and [Horn & Ikeuchi,1983]; their principal drawback is that a given needle map can represent a large class of polyhedra and curved objects that are geometrically distinct.

Some approaches have used direct volumetric representations of objects. Gen-

eralized cylinders, used in [Brooks,1981] and [Kuan & Drazovitch,1983], can represent a large variety of objects and can be used in a very general image understanding framework; such models, however, tend to be computationally expensive to process and manipulate. Another representation, employed in [Bolles *et al.*,1983], is an extension to constructive solid geometry that includes direct representation of features that are useful for recognizing the object.

Other approaches in two or three dimensions combine simple linear surfaces with low-order curves, e.g., [Bolles & Cain,1982], [Perkins, 1978], and [Turney *et al.*,1985]. These can represent objects with more precision and more concisely than a purely planar model, with a slight increase in computational cost. Recognition strategies with such models tend to be feature-based, rather than being general geometric strategies.

Most three-dimensional recognition systems, though, use some sort of planar approximation to the object surface: [Bhanu, 1984], [Boisonnat,1984], [Faugeras & Hebert,1983], [Faugeras *et al.*,1984], [Grimson & Lozano-Pérez,1985], and [Oshima & Shirai, 1983] are among the many systems that have been described. Rather than using feature-based recognition strategies, most of these systems match small planar patches extracted from the data to the planar (or sometimes higher-order) patches of the model by either geometric or statistical decision procedures.

The present approach concentrates upon a simple representation of objects as a set of bounding polygons. These can be rapidly manipulated and can be used in an object recognition strategy that is powerful, general, and fast.

2.2 Interpretation of Sensor Data

Grimson and Lozano-Pérez have, in several works, developed and expounded a framework for the model-based recognition of sparse, local data. This section provides a detailed description of our implementation of their approach, includ-

ing explicit descriptions of our techniques for calculation (especially for those calculations which may become ill-behaved). The interpretation process begins with searching all possible interpretations by pruning the interpretation tree with powerful local geometric constraints, and then finding the valid interpretations by calculating the nominal rotation and translation, then globally verifying the consistency of the solution. We will use vector algebra to express the relationships and constraints, keeping in mind that although intended for recognition of three-dimensional objects it can readily be simplified for two-dimensional objects.

Recognition proceeds by attempting to find an *interpretation* of the data, which is defined as an assignment of a model face, edge, or vertex to each data point. For a given model, if there are k faces and n points, then there are potentially k^n interpretations of the data. This is clearly impractical for complex objects or large amounts of data; some other approach is needed. Figure 2.1 shows an interpretation tree for two data and an object model with four structural elements.

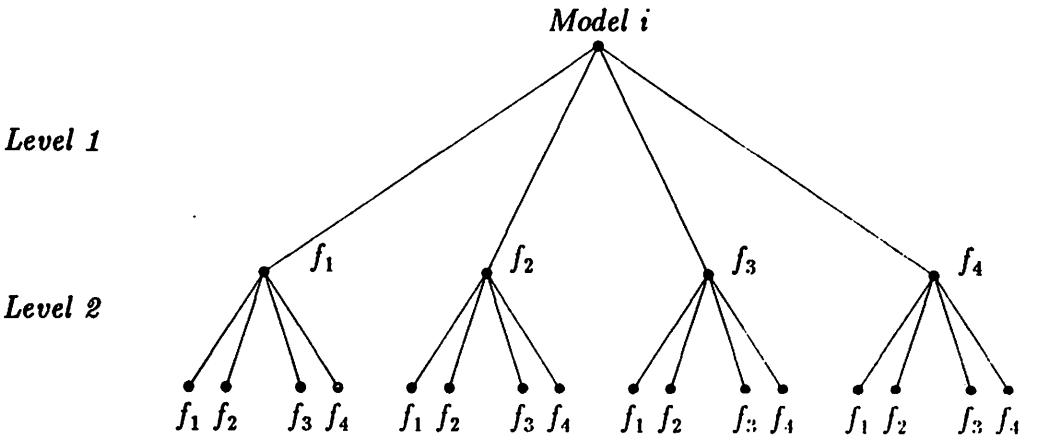

Figure 2.1: Two levels of an interpretation tree for a four-sided object.

The present system interprets the tactile data by applying simple geometric

constraints between pairs of assignments, thus generating interpretations which are feasible (they are tested for consistency at a later stage). The interpretations are built in a tree; if a set of assignments is unfeasible, then so are all supersets, and the interpretation tree may be pruned at the node where an assignment failed to meet the local geometric constraints. This is an elegant, and surprisingly powerful, method of reducing the set of possible interpretations of the data; [Grimson,1984] gives an analysis of the combinatorial power of these constraints.

The geometric constraints are relatively simple. Since a single point can be assigned to any face, it is necessary to examine at least a pair of points and a pair of model faces. Disregarding error for now, a pair of data points presents four pieces of information: two positions, and two normals. If the positions are represented as C_1 and C_2 (with the difference vector being $D_{12} \stackrel{\text{def}}{=} C_1 - C_2$), and the associated sensed normals be $\hat{N}_1$ and $\hat{N}_2$, then there are four scalar values which are independent of the global coordinate system, representing local information only:

- The (squared) distance between the points: $\|D_{12}\|^2 = D_{12} \cdot D_{12}$.
- The cosine of the angle between the normals: $\hat{N}_1 \cdot \hat{N}_2$.
- The *direction* component, in the direction of the first normal, of the vector going from the first point to the second: $\hat{N}_1 \cdot D_{12}$.
- The direction component from the second point to the first: $\hat{N}_2 \cdot -D_{12}$.

These last two are a bit unusual, in that they combine distance and angular information; but all four of these scalars are independent, and form a basis in which the original data may be expressed in a purely local manner. See Figure 2.2 for a diagram of the construction.

Each model is processed offline to find the minimum and maximum values of each of these constraints between all possible pairs of faces. Because these

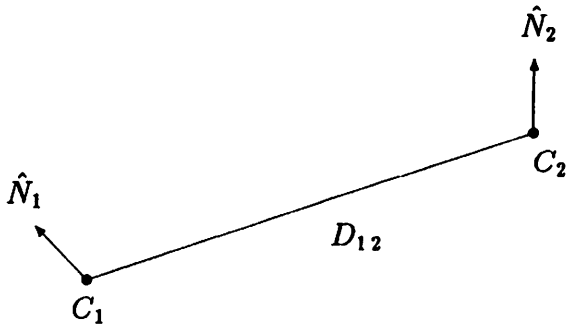

Figure 2.2: Local geometric construction for two data points.

constraint bounds are between pairs of faces, they can be represented as two-dimensional tables, which are indexed by the faces. One [symmetric] table is built indicating the range of distances between two faces; one [symmetric] for the angles (which is just a single value); and one [non-symmetric] for the directions. This pre-processing is done only once, and is reasonably efficient in that the distance and direction tables can be built using only perimeter information (interior points, in general, need not be examined). These tables now form part of the representation of the object.

Each step of generating a feasible interpretation involves taking a new datum, and finding the local constraints between it and each of the previously interpreted points. The new datum is then tentatively assigned to each face of the model. For each tentative assignment, the local constraints can be compared to the range of constraints already computed for the faces to which each point has been assigned. If the local constraints on distance, normal angle, and direction all fall within the ranges for the faces, then the new assignment is consistent with the previous one. This test is conducted between the new point and *all* previously assigned points in the current interpretation tree.

If the point–face assignment is consistent with all previous assignments, then

this assignment creates a new feasible interpretation. If it is inconsistent with *any* of the previous assignments, then the interpretation tree may be pruned at this point, and all subtrees deriving from the new assignment may be ignored. When all of the data have been processed, the leaves of the tree represent all feasible interpretations of the sensor data.

The method used in this research is an extension of the above method. In addition to applying constraints between faces, constraints including edges and vertices are added. Edges add angular and distance constraints; vertices add only distance constraints. However, the distance constraints provided by vertices are much more powerful than those provided by edges, which in turn are more powerful than those provided by faces.

The above description is slightly more complicated in implementation because error must be accounted for. Each sensed position is known only within a small sphere, and the sensed normals within a small cone. The constraints must be applied with these error bounds in mind; when the appropriate mathematical formalism is employed, the addition of error amounts principally to extending the constraint tables of the models by the indicated error radii.

2.2.1 Calculating Local Constraints from a Model

The present implementation of model-based recognition derives from the description given in [Grimson & Lozano-Pérez,1984], where details of proofs, etc. may be found. The model constraints are those of distance, angle, and direction; each of these is now described in more detail.

2.2.1.1 Distance Constraints

The distance constraints are the lower and upper bounds on the distance between two model elements, e.g., faces. The bounds may be found by examining all edge-edge pairs, vertex-vertex pairs, and edge-vertex pairs of the model

elements. The distance between two points is straightforward; but to find the distance between a point and a line, or between two lines, some extra checking is necessary.

The distance between a point X and a line $L(\lambda) = P + \lambda \cdot D$ is derived by finding the parameter of the location on the line nearest to the given point. If the given point is X, then the line parameter of the nearest point is

$$t = (X - P) \cdot \frac{D}{\|D\|}$$

where P is a point on the line and D the line's (possibly non-unit) direction. If the parameter value t is between 0 and 1 then the nearest point lies within the segment, and the distance may be found directly; otherwise, the nearest distance is given by the endpoint of the segment that is closest to the point determined by the calculated parameter.

The derivation of the distance between two lines is somewhat more involved. Let the two lines be

$$L_1(t) = a + t \cdot \hat{\varsigma}$$

and

$$L_2(s) = b + s \cdot \hat{\eta}$$

where $\hat{\varsigma}$ and $\hat{\eta}$ are of unit length. If the abbreviation $c = (a - b)$ is used, the length of the shortest segment between the two lines is

$$\|d\| = \|L_1(t) - L_2(s)\| = \|c + t \cdot \hat{\varsigma} - s \cdot \hat{\eta}\|$$

under the conditions that the shortest line is perpendicular to both L_1 and L_2, i.e.,

$$d \cdot \hat{\varsigma} = d \cdot \hat{\eta} = 0$$

From these conditions, the two line parameters can be derived as

$$t = \frac{(\hat{\varsigma} \cdot \hat{\eta})(c \cdot \hat{\eta}) - (c \cdot \hat{\varsigma})}{1 - (\hat{\varsigma} \cdot \hat{\eta})^2}$$

and

$$s = c \cdot \hat{\eta} + t(\hat{\varsigma} \cdot \hat{\eta})$$

Once again, these parameters are tested to ensure that the determined points lie on the segment. If the two lines are parallel, then $\hat{\varsigma} \cdot \hat{\eta} = 1$ and the equation specifying t is indeterminate, so for parallel lines an endpoint is used to find the point on the other line nearest to the endpoint. (It is necessary to check only the endpoints of one segment to ensure that the two parallel segments have a perpendicular joining line that satisfies the constraint of passing within both line segments).

For efficiency, it may be noted that full processing is needed only to find the *minimum* distance. The maximum distance between two polygons is always between a pair of vertices from the convex hulls of the respective faces, and thus we can restrict our computation of maximum distance to the vertices of the faces.

2.2.1.2 Angle Constraint

The angle constraint between two faces is very simple. The unit normal of each polygon is calculated from the cross product of the first two bounding edges; the constraint used is simply the dot product of the unit normals.

As noted above, the direction is unambiguous. During construction of the model, care must be taken that the polygons are specified with the vertices in the correct order.

2.2.1.3 Direction Constraints

The direction constraint is the dot product of a difference vector and a direction vector. For a pair of faces, it can be shown that the direction bounds can be found by examining only differences between *vertices* of the faces.[2]

[2]Proof of this can be found in [Grimson & Lozano-Pérez,1984].

For two faces i and j, let V_i be a vertex from face i, V_j be from face j, and the unit normal of face i be $\hat{Z}_i$. Then the direction constraints are the bounds on the value

$$\hat{Z}_i \cdot (V_j - V_i)$$

as V_i and V_j range over the set of vertices of the respective faces.

The constraints on distance, angle, and direction may be pre-calculated for each model and saved in tables. For the purposes of recognition, these tables are a very important part of the model, as they represent the local geometric information that is used to find feasible interpretations of sensor data.

2.2.2 Model Constraints and Feasible Interpretations

Feasible interpretations can be found by examining local geometric properties of pairs of sensor data, with constraints provided by a given model. The current system calculates the minimum and maximum values of the properties for all pairs of data, and uses these to prune the interpretation tree in a depth-first search. The properties tested, in order, are distance, angle, and direction; a new assignment of a datum to a face must be consistent with *all* predecessors in the interpretation tree.

For the following discussion, the sensor data will be referred to and subscripted with i and j, while the faces to which they are tentatively assigned will be m and n.

2.2.2.1 Distance Pruning

For a pair of sensor position data C_i and C_j, with positional error radii ϵ_i and ϵ_j respectively, the minimum and maximum distances between the points are

$$\max(0, \|C_i - C_j\| - \epsilon_i - \epsilon_j)$$

and

$$\|C_i - C_j\| + \epsilon_i + \epsilon_j$$

The data may be assigned to a pair of faces F_m and F_n if the range of distances overlap. Formally, this means that the maximum data distance must be no less than the minimum face-face distance *and* the minimum data distance must be no greater than the maximum face-face distance. If this condition is true, then the assignment is consistent with the distance constraint.

2.2.2.2 Angle Pruning

For a pair of sensed normals $\hat{N}_i$ and $\hat{N}_j$, the angle errors are δ_i and δ_j. Defining $\hat{U}_i$ as the true surface normal, the angle error provides a bound on the dot product between the sensed normal and the true normal:

$$\hat{N}_i \cdot \hat{U}_i \geq \delta_i$$

Letting $\cos\varsigma_i \stackrel{\text{def}}{=} \delta_i$, $\cos\varsigma_j \stackrel{\text{def}}{=} \delta_j$, and $\cos\gamma_{ij} \stackrel{\text{def}}{=} \hat{N}_i \cdot \hat{N}_j$, then the minimum dot product achievable with sensor error is

$$\cos[\min(\pi, \gamma_{ij} + \varsigma_i + \varsigma_j)]$$

and the maximum is

$$\cos[\max(0, \gamma_{ij} - \varsigma_i - \varsigma_j)]$$

The points satisfy the angle constraint if the dot product of the normals of the faces to which they are assigned falls within these ranges.[3]

2.2.2.3 Direction Pruning

The final constraints are those on direction, which is the dot product of a unit normal and a difference vector. Let D_{ij} be the difference vector between positions C_i and C_j. $\hat{N}_i$, δ_i, and ς_i will be as defined above.

[3]See [Grimson & Lozano-Pérez,1984] for a derivation of these bounds.

The value of the direction from point i to point j, disregarding sensor error, is

$$\hat{N}_i \cdot (C_i - C_j) = \hat{N}_i \cdot D_{ij}$$

Where $\|D_{ij}\|$ is length of D_{ij}, and $\hat{D}_{ij}$ is the unit vector in the same direction as D_{ij}, then the value can be expressed as

$$\|D_{ij}\|(\hat{N}_i \cdot \hat{D}_{ij})$$

When expressed this way, the error in sensing positions and local surface normals can be inserted into the formulation. As with the angle constraints, what is sought are bounds on the product of the local normal and the unit direction. Once these are found, they can be multiplied by the length of the difference vector and the bounds on the value of the direction property have been found. It is expected that the positional error provides much less uncertainty than the angular error, so its contribution is neglected.

The formulation of error in the dot product of the sensed normal and the unit direction is analogous to the formulation used in the angle constraint. Let $\cos\omega_{ij} = \hat{N}_i \cdot d_{ij}$. The minimum direction value is thus

$$\|D_{ij}\| \cdot \cos(\zeta_i + \omega_{ij})$$

and the maximum is

$$\|D_{ij}\| \cdot \cos(\zeta_i - \omega_{ij})$$

These ranges are compared with the face-face direction constraints in a manner similar to the comparison for distance constraints, since a non-null intersection of the ranges is being sought. Unlike the distance constraint, the direction constraint is not symmetric, so the order of the data must be reversed and the test performed again.

The direction constraint is a reasonably powerful one, but because it requires much more calculation it is usually performed after distance and angle testing. If

all pairs of assignments satisfy the direction constraint, then the interpretation is said to be feasible.

2.2.3 Finding Valid Interpretations

An interpretation of the data that is feasible is not necessarily valid: a feasible interpretation is based upon pairwise constraints, whereas a valid interpretation must take into account global n-ary constraints, where n is the number of faces that have data points assigned to them. Thus, the model must be tested, to ensure that all of the sensed normals are consistent with the model normals, and that all of the sensed points lie within the model faces. This involves finding the three translational parameters and three rotational parameters that transform the model to the data, and then checking the match between data and model.

The first step in testing an interpretation is to find the rotational component. This can be done with two independent assignments in the interpretation, finding first the direction of the rotation and then then the angle of rotation about this direction vector. Once the rotational component is found, the translational component can be computed from three positions. These computations are for perfect data; there are a number of techniques, varying in reliability and cost, which might be used to increase reliability given potentially erroneous data [Grimson & Lozano-Pérez,1984], [Faugeras & Hebert,1983].

Once the transformation from model instance to data is found, the final model test can be done. This involves checking each sensed normal to ensure that it matches, within its error cone, to the transformed normal of the face to which it has been assigned. Similarly, each point must not only lie on or near the plane of the face to which it is assigned; it must also lie within or near the polygon that constitutes the boundary of the face.

2.2.3.1 Calculating The Model Pose

The *pose* of the model is the rotation and translation that brings it from its local defining coordinate system (which is rather arbitrary) to the coordinate system of the sensor data and robot system. There are six parameters of the pose – three for the rotation, and three for the translation. The present system finds them by first determining the rotational component, and subsequently determining the translational component.

Finding the rotation component proceeds by first determining the direction of the axis of rotation, and then the amount of rotation about that axis. If datum i with sensed normal $\hat{N}_i$ is assigned to face m with plane normal $\hat{Z}_m$, then any valid rotation component must preserve the angle between the new, transformed vector and the direction of rotation. This can happen only when the direction of rotation is perpendicular to the difference between the sensed normal and the face normal, i.e., if r_{im} is an axis of rotation that takes $\hat{N}_i$ to $\hat{Z}_m$, then

$$r_{ij} \cdot (\hat{Z}_m - \hat{N}_i)$$

If datum j is assigned to face n, with $\hat{Z}_n \neq \hat{Z}_m$, the same condition must hold; the joint conditions can be met only by the cross product of the difference vectors, i.e.,

$$r = (\hat{Z}_m - \hat{N}_i) \times (\hat{Z}_n - \hat{N}_j)$$

There is an ambiguity of 180° in the cross product, which can be removed by arbitrarily declaring that the angle of rotation about the direction be non-negative.

With the direction of rotation known, the angle of rotation θ can be found. The amount of rotation is given by

$$\cos\theta = 1 - \frac{1 - (\hat{N}_i \cdot \hat{Z}_m)}{1 - (r \cdot \hat{N}_i)(r \cdot \hat{Z}_m)}$$

$$\sin\theta = \frac{(r \cdot \hat{N}_i) \cdot \hat{Z}_m}{1 - (r \cdot \hat{N}_i)(r \cdot \hat{Z}_m)}$$

If $\sin\theta = 0$, the ambiguity is resolved by taking $\theta = \pi$.

From the axis of rotation and the angle of rotation, we can finally form the rotation matrix according to a standard formula. Where r_x, r_y, and r_z are the components of the direction of rotation, the rotation matrix R is

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1 - \cos\theta) \begin{bmatrix} r_x^2 & r_x r_y & r_x r_z \\ r_y r_x & r_y^2 & r_y r_z \\ r_z r_x & r_z r_y & r_z^2 \end{bmatrix} - \sin\theta \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

These calculations assume perfect sensor data. When there is error present, a number of alternatives may be used. [Grimson & Lozano-Pérez,1984] suggest calculating the rotation directions for all pairs of sensed normals and using an unspecified clustering algorithm. [Faugeras & Hebert,1983] use a least-squares approach, which is very expensive. The present system uses the very simple method of selecting the sensed normals with the smallest magnitude dot product, on the grounds that the greater angular separation provides greater constraint for constant sensor error. Clearly, more work on this topic is needed.

Once the rotation component is found, it is possible to solve for the translation component of the pose transformation. What is sought is the total transformation from a vector v_m in model coordinates to a vector v_s in sensor coordinates: $v_s = Rv_m + T$, where T is a translation vector. This can be found if there are three faces with independent normal vectors.

Consider datum i which is assigned to face m. As above, $\hat{Z}_m$ is the unit normal of the face; let the equation of the plane in which the face lies be

$$X \cdot \hat{Z}_m - p_m = 0$$

Working backwards, one can take the sensed position C_i and remove the effect of the translation and rotation; its value is then

$$R^{-1}(C_i - T)$$

Since it is assigned to lie on face m, it must satisfy the plane equation of the face, so

$$\hat{Z}_m \cdot \left(R^{-1}(C_i - T)\right) = p_m$$

Three faces with independent normals allow us to construct a system of three equations in three unknowns, after slight re-arrangement of terms:

$$\begin{aligned}
(R\,\hat{Z}_l\,)\cdot T &= (R\,\hat{Z}_l\,)\cdot C_i - p_l \\
(R\,\hat{Z}_m)\cdot T &= (R\,\hat{Z}_m)\cdot C_j - p_m \\
(R\,\hat{Z}_n\,)\cdot T &= (R\,\hat{Z}_n\,)\cdot C_k - p_n
\end{aligned}$$

These permit us to find a solution for the translation component as

$$\begin{aligned}
\left[\hat{Z}_l \cdot (\hat{Z}_m \times \hat{Z}_n)\right] T \;=\; & ((R\,\hat{Z}_l\,)\cdot C_i - p_l\,)\,((R\,\hat{Z}_m)\times(R\,\hat{Z}_n\,)) \\
& + ((R\,\hat{Z}_m)\cdot C_j - p_m)\,((R\,\hat{Z}_n\,)\times(R\,\hat{Z}_l\,)) \\
& + ((R\,\hat{Z}_n\,)\cdot C_k - p_n)\,((R\,\hat{Z}_l\,)\times(R\,\hat{Z}_m))
\end{aligned}$$

Once again, these calculations assume that the sensor data are perfect. [Grimson & Lozano-Pérez,1984] appear to reduce the effects of sensor error by performing the calculations over all triples whose face assignments meet the independence criteria, and taking the average of the resulting translations. The present system simply uses those data whose assigned face normals have a scalar triple product of a larger magnitude than any other triplet.

2.2.3.2 Testing the Model

Having found the complete pose transformation from model to data, it is necessary to ensure that the data points would lie within the assigned faces of the model. It is computationally faster to invert the process, transforming sensor coordinates to model coordinates, and then checking the faces.

The first step in this process is to transform each position point i by

$$C_i' = R^{-1}(C_i - T)$$

This transformed point must lie on the plane of face m. Since there is some error in the point location ϵ_i, one finds the point X that is on the plane, and nearest to the point, from

$$X = C_i^* + [p_m - \hat{Z}_m \cdot C_i^*]\hat{Z}_m$$

A test of the transformation is that $\|C_i^* - X\| \leq \epsilon_i$, if all has proceeded correctly.

The final validation of an assignment is that the transformed point must lie *within* the face, not just on the same plane as the face. Because the sensor error may be non-zero, this test should be done with a variable comparison tolerance. The algorithm employed here relies upon the geometrical observation that the sum of the angles between successive vertices, as viewed from the point, is 0 if the point is outside a polygon and 2π if the point is inside. Figure 2.3 shows the angles to be summed in two examples.

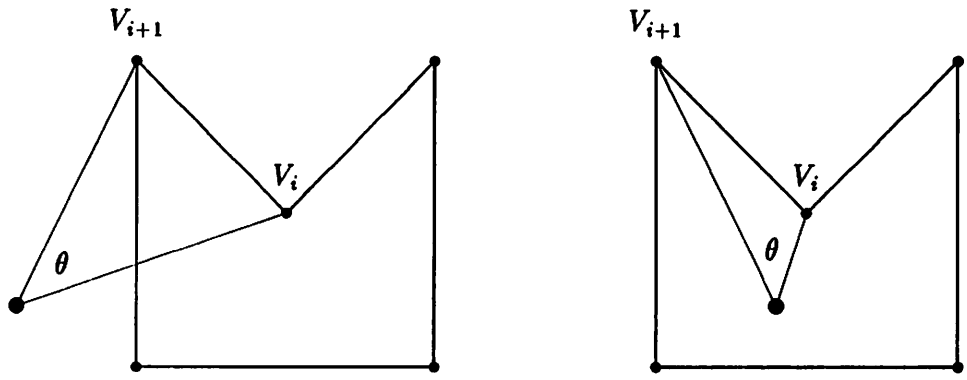

Figure 2.3: Determining if a point is within a polygon by summing angles.

The algorithm used in the present system translates the polygon–point system so that the point is at the origin. For each vertex V_i of the polygon, a virtual rotation of ς is performed to bring the vertex onto the **X** axis. The value of ω is

given by

$$\cos\omega = \frac{V_{i(x)}}{\|V_i\|}$$

$$\sin\omega = \frac{V_{i(y)}}{\|V_i\|}$$

where $V_{i(x)}$ is the component of V_i in the **X** direction. The angle θ between V_i and its successor V_{i+1} can be expressed as

$$\theta = \tan^{-1}\left(\frac{V_{i+1(x)}\sin\omega + V_{i+1(y)}\cos\omega}{V_{i+1(x)}\cos\omega + V_{i+1(y)}\sin\omega}\right)$$

This calculation is ill-behaved if the point is near a vertex or polygonal edge; the calculation is made robust by declaring the point to be inside the polygon if it is within ϵ_i of either a vertex or an edge.

This procedure can be made more efficient by noting that the property of a point being inside or outside a polygon is unchanged by affine transformations of the plane; in particular, an orthographic projection of the face and point onto one of the Cartesian coordinate planes has the same topology as the original face and point. Such a projection is very simple, amounting to no more than ignoring one of the vector elements by contracting the vectors from three dimensions to two. For increased numerical stability, the plane chosen for the projection should be one with a normal that produces the smallest absolute angle with the normal of the face, and the comparison tolerance should also be adjusted to reflect its new size on the coordinate plane. Since this test is conducted in model coordinates, the projections of faces onto coordinate planes can be done off-line and stored with the model.

If every assignment of point to face in a feasible interpretation passes all of these tests then it is *valid*, and the object can be said to have been recognized as being an instance of the model in a particular pose. The case of interest for this research is when there are multiple legitimate interpretations; the problem

is then to acquire new sensor data, in an efficient manner, so that the object can be uniquely identified.

2.3 Using the Calculated Interpretations

The above section described an approach to the recognition and localization of polygonal or polyhedral objects from sparse, local data. For some systems, recognition of all possibilities is acceptable, and further work need not be done. We, however, propose to identify the single correct interpretation by planning paths through the workspace. Because there can be errors in the low-level data, there can be considerable uncertainty in the orientation and location of the various interpretations of the object.

Such uncertainty is exhibited in Figure 2.4. For a given object and pair of tactile data, we may calculate the nominal pose, and the worst possible orientations that still permit a valid interpretation. We can see from this figure that some regions exhibit very little positional deviation, and others exhibit very considerable excursions.

The above section described the simplistic approach to management of uncertainty used by others; while providing some worst-case approximations, they do not characterize pose error sufficiently to provide truly useful constraints in planning a sensing path. We will now provide a more detailed analysis, first in two dimensions and then in three dimensions, so that we can find tight bounds on the potential deviation of an interpretation from its nominal pose.

2.4 Uncertainties in Two Dimensions

The two components of a pose are the rotation that brings model features into alignment with sensed normals, and the subsequent translation that brings the features into close proximity to the sensed positions. To a first approximation,

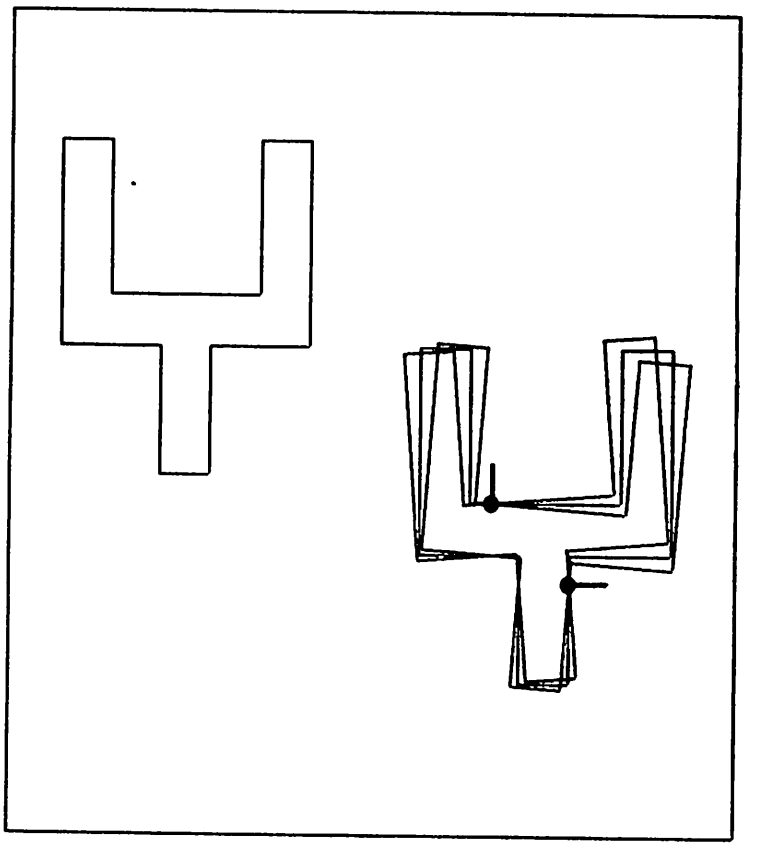

Figure 2.4: An object, and the range of uncertainty in interpreting it.

the rotational error depends only upon the error in sensed normals, whereas the translational error depends upon both sensed normal and position error.

The uncertainties in the determination of the sensed point's position and local surface normal lead to uncertainty in the pose of the instantiated model. In turn, this leads to uncertainties in the location and orientation of model faces; thus, in order to ensure that a planned path contacts the desired faces and misses those that are not targets, we must have some understanding of the nature of the pose uncertainty. Worst-case bounds for the three-dimensional case have been derived in [Grimson,1986]; we now will derive tighter bounds for the two-dimensional case, and discuss implications for three-dimensional pose uncertainty.

In the interpretation paradigm we follow, a two-dimensional model feature – a line segment – is transformed so that a sensed point lies on or near the matched, transformed segment. In two dimensions, the rotation component can be found from the match between one sensed normal $\hat{N}_i$ and one model normal $\hat{Z}_i$, simply by rotating $\hat{Z}_i$ by θ so that it equals $\hat{N}_i$. Let us denote this rotated face as

$$L_i(\lambda_i) = P_i + \lambda_i \hat{D}_i$$

We will call the rotated face normal $\hat{d}_i$, with the convention that

$$\hat{d}_i = \hat{D}_i \times \hat{k}$$

with $\hat{k}$ being the unit vector pointing in the direction of the **Z** axis, and $\hat{D}_i$ is the unit direction of the line segment.

Finding the translation component requires two sensed positions C_i and C_j, and two matched rotated lines $L_i(\lambda_i) = P_i + \lambda_i \hat{D}_i$ and $L_j(\lambda_j) = P_j + \lambda_j \hat{D}_j$. The translation component can be found by translating in the direction $\hat{D}_i$ by the perpendicular component of C_j with respect to L_j scaled by the sine of the angle between the two lines (this brings C_j onto L_j), and then translating along D_j

similarly; thus, the nominal translation component is

$$T_N = \frac{\hat{D}_i[\hat{d}_j \cdot C_j] + \hat{D}_j[\hat{d}_i \cdot C_i]}{\|\hat{D}_i \times \hat{D}_j\|} \tag{2.1}$$

and so we can represent the transformation of any point q from model coordinates to sensor coordinates as

$$\Theta(q) + T_N$$

where Θ is the planar rigid rotation about the origin by θ.

2.4.1 Rotational Uncertainty

From these preliminaries, we can find the uncertainty in the pose, i.e., the maximum possible difference between the calculated pose and the true pose of the object in question. Taking rotational uncertainty first, we can let the maximum error in a sensed normal be δ, i.e., the sensed normal $\hat{N}_i$ and the true local surface normal $\hat{U}_i$ must always lie within δ radians of each other. If the calculated rotation angle from model coordinates to sensor coordinates is θ, then we would expect that in the ideal case the angle between the sensed normal $\hat{N}_i$ and the model normal $\hat{Z}_i$ would be θ. In the presence of error, however, the angle will in general be different from θ (but never by more than δ). We will call the total amount that θ could be changed, and still have the model normals transformed to within δ of the sensed normals, the *residual rotation error.*

This residual is readily calculated. Observe that the range of deviation, centred about θ, is $\pm\delta$. Let the angular deviation of the angle between $\hat{d}_i$ and $\hat{N}_i$ be denoted Δ_i; we see that

$$-\delta \leq \Delta_i \leq \delta$$

for every sensed normal $\hat{N}_i$. Then θ could be changed in one direction until the smallest Δ_i became $-\delta$, or in the other direction until the largest Δ_i became δ;

thus, the residual error in θ is

$$\gamma = 2 \cdot \delta - (\max_i(\Delta_i) - \min_i(\Delta_i))$$

Hence, the residual error is largest when the sensed normals deviate from the rotated model normals the *least*; as the deviation increases yet stays within the error bounds, the residual error in the rotation component of the transformation decreases. Figure 2.5 shows two model vectors that have been rotated, and how the distance to the nearest error bound of the associated sensed normal reduces the uncertainty in the deduced rotational transformation.

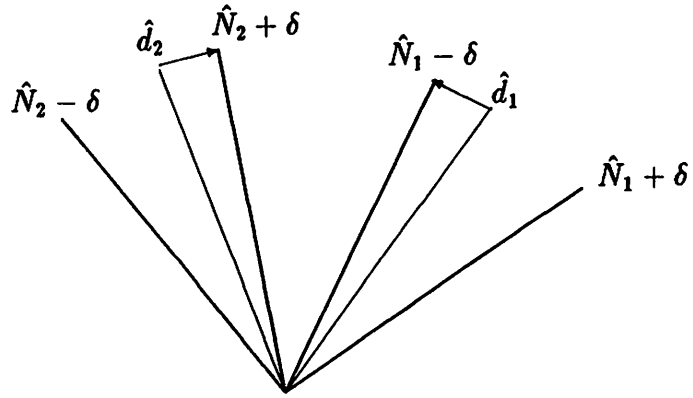

Figure 2.5: Rotational uncertainty bounds in two dimensions.

From the initial guess at the rotation component we can now form a better, tighter estimate: let the nominal rotation angle be

$$\phi = \theta + \frac{(\max_i(\Delta_i) - \min_i(\Delta_i))}{2} \tag{2.2}$$

and its associated rigid transform be Φ. The true rotation of the object must then be some θ', where $\phi - (\gamma/2) \leq \theta' \leq \phi + (\gamma/2)]$.

Assuming that each sensed point can be validly matched under any rotation within our estimate, we can find the maximum excursion E_q of a model point q by transforming it at the extremes Φ_H and Φ_L:

$$E_q = \|(\Phi_H(q) + T_H) - (\Phi_L(q) + T_L)\| \tag{2.3}$$

Note that the translation components will in general differ, and must be computed separately.

2.4.2 Translational Uncertainty

The uncertainty in computing the translation component is independent of the uncertainty in rotation (unless a sensed point lies very near an endpoint of a transformed line segment), but depends upon the angles of the matched faces. This is because a point may be translated arbitrarily in the same direction as a line and still lie within ϵ of the line; the limiting factor is the translation component perpendicular to the line. For the minimal case of two points, we can translate the model in the direction $\hat{d}_i$ by ϵ, and also in the direction $\hat{d}_j$ by ϵ, and still have the points lie near the lines. The translation component has an associated error vector that lies within a rhombus of size ϵ; the maximum error in computing translation is the major diagonal of this rhombus, or

$$E_T = \frac{\epsilon}{\hat{d}_i \cdot \hat{D}_j} \cdot \|\hat{D}_i + \hat{D}_j\|$$

Figure 2.6 gives an example of how two faces each generate a pair of constraint lines for the translation; the nominal translation T_N, which is the point lying within the parallelogram, is offset from the constraint lines by the distance that the sensed position points are offset from their respective matched, transformed faces.

The nominal translation is always at the centre of this rhombus. It is interesting to note that the uncertainty region can be decreased by the addition

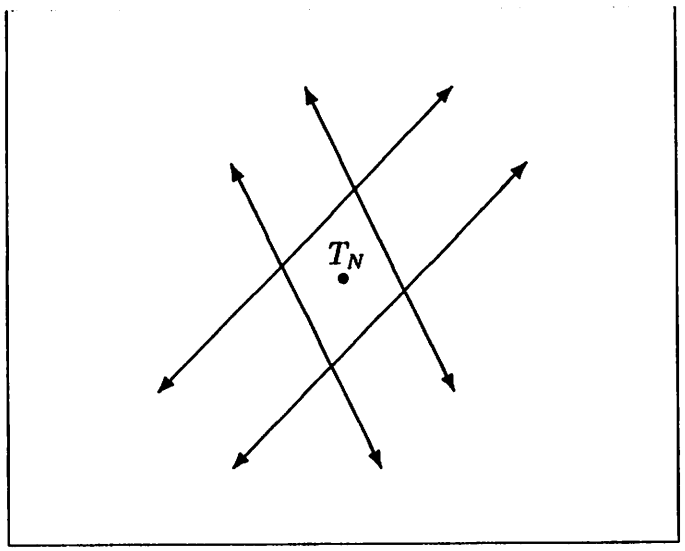

Figure 2.6: Translational uncertainty bounds in two dimensions.

of more positional data points. Taking the nominal translation as the origin, additional constraint lines can be added; importantly, these lines can be offset from the nominal translation by the distance from the data point to the rotated, translated model edge. The effect of adding points which are slightly in error is to *decrease* the total uncertainty in model position – essentially, in order to satisfy all of the constraints of being within ϵ of the model edges, the space of possible positions is successively reduced. It is, of course, possible that the original nominal translation is incorrect; in this successive approximation, then, the nominal translation will have to be updated, perhaps by always selecting the centroid of the constraint region.

We note also that the geometry of this constraint region is dependent solely on the geometry of the model and the positional data points. This means that its size and shape are independent of the rotation, although the nominal rotation parameter will induce a rigid rotation and translation of the constraint region as the rotation parameter is varied.

This picture becomes more complicated if we include endpoint conditions. The true constraint is that the positional datum lie within ϵ of the rotated model face; the actual uncertainty region is thus not an infinitely long swath but rather a rectangle with semi-circles of radius ϵ stuck onto the ends (see Figure 2.7 for a pictorial representation). The translational component is constrained to lie within the intersection of a number of such convex sets, one generated from each face to which the point is assigned.

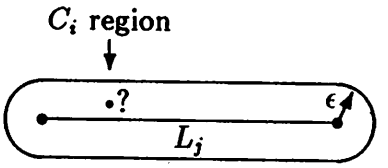

Figure 2.7: Positional uncertainty region for a 2–D face.

2.5 Uncertainties in Three Dimensions

The uncertainty in the pose of a three-dimensional object may also be broken into uncertainty regarding its orientation, and uncertainty in its position. The translational component may be analysed in a manner analogous to the two-dimensional case; the rotational component, however, is somewhat more complicated, and needs detailed discussion.

2.5.1 Rotational Uncertainty

The rotational component of the pose is defined as that member of $SO(3)$ which brings the model normals into correspondence with the sensed normals. Because there is error in a sensor reading, there is uncertainty in the sensed normal; thus, we aim to characterize set of $SO(3)$ rotations which brings all of the model normals to within some angle ψ of their matched sensed normals. We

may visualize the sensed normals as yielding a set of circles on the Gaussian sphere, and the problem is to keep the set of rigidly rotating model normal points within their respective spheres. What we seek is a bound on the maximum angle of rotation, about any axis, that still leaves the points within their appointed circles.

We will begin our analysis with the simplest case, where we have a pair of independent nominal sensed normals $\hat{N}_1$ and $\hat{N}_2$, a pair of model normals $\hat{Z}_1$ and $\hat{Z}_2$, and an uncertainty angle ψ. Initially, let us also suppose that the model normals match the nominal sensed normals perfectly, i.e., $\hat{N}_1 \cdot \hat{N}_2 = \hat{Z}_1 \cdot \hat{Z}_2$. In this case, we can describe the pose rotation as taking $\hat{Z}_1$ into alignment with $\hat{N}_1 = \hat{Z}_1^*$, and then a rotation about $\hat{N}_1$ until $\hat{Z}_2^* = \hat{N}_2$.

The crucial observation here is that we can bound the uncertainty by examining the constraints on the vectors orthogonal to $\hat{Z}_1^*$ and $\hat{Z}_2^*$, in particular examining $\hat{Z}_1^* \times \hat{Z}_2^*$. Let $\hat{K}_1$ be any unit vector that is orthogonal to $\hat{Z}_1^*$; since $\hat{Z}_1^* = \hat{N}_1$, we can bound $\hat{K}_1$ to an equatorial band of width 2ψ, with $\hat{N}_1$ as its pole. Similarly, we can construct an equatorial band using $\hat{N}_2$ as a pole, and bound the possible location of $\hat{K}_2 \perp \hat{Z}_2^*$. Such a construction is illustrated in Figure 2.8. We can see that no vector exhibits a greater net range of movement in rotation about $\hat{Z}_1^*$ than a vector that is orthogonal to it; so if we can bound the movement of an orthogonal vector, we bound also the maximum uncertainty in the net rotation of the system.

In finding the nominal rotation, what we have effectively done is observe that $\hat{C}$, which is parallel to $\hat{Z}_1^* \times \hat{Z}_2^*$, is constrained to lie on the equatorial great circles of both $\hat{N}_1$ and $\hat{N}_2$, and that there is only one such point that lies on both equators and respects the handedness of the coordinate system. In the presence of uncertainty, this constraint is weakened to having the cross product lie within both equatorial bands; thus, all we can say is that the cross product must lie within the curvilinear rhombus that is the intersection of the equatorial bands.

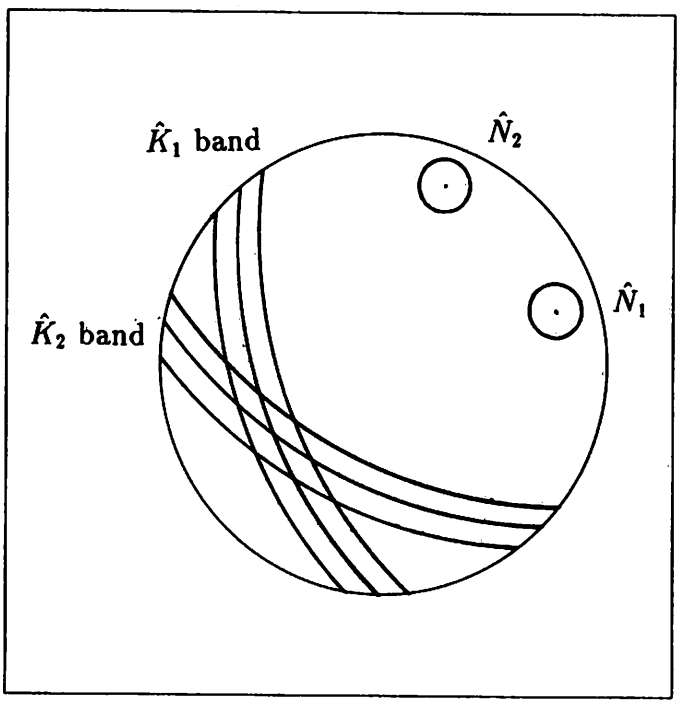

Figure 2.8: Constraint bands for a pair of matched normals.

A local planar approximation of this situation is given in Figure 2.9, where $\hat{C}$ is the normalized nominal cross product of the model normal vectors.

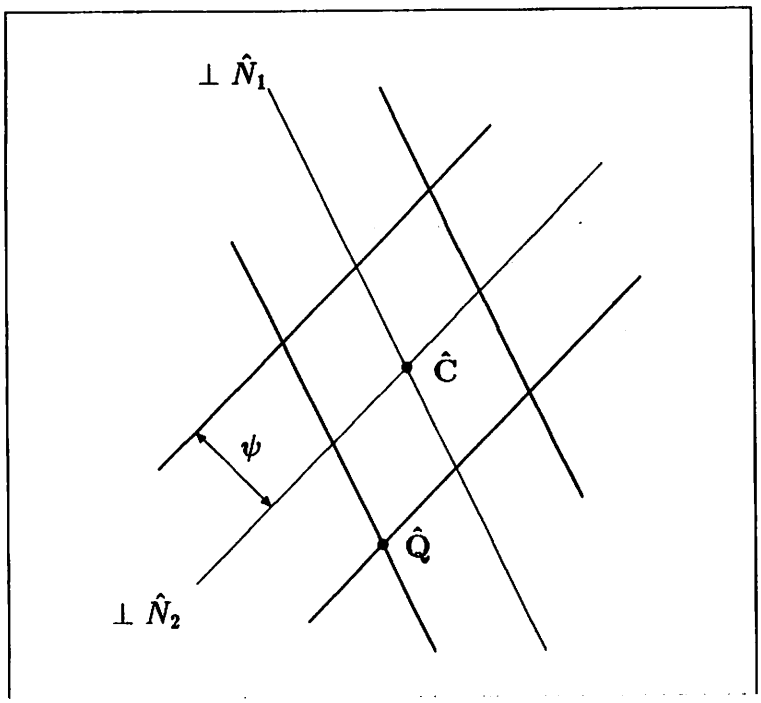

Figure 2.9: Constraint rhombus for the cross product of matched normals.

We observe immediately that the maximum excursion of the cross product vector is along the great circle that constitutes the major diagonal of the rhombus. We will now calculate this angle, and the axis of rotation for the worst-case deviation.

By definition,

$$\hat{C} = \frac{\hat{N}_1 \times \hat{N}_2}{\sin\theta}$$

Note that because we are dealing with poles and equators, we take θ to be the net angle between the poles, i.e., $0 \leq \theta \leq \pi/2$. Then one of the major diagonal endpoints $\hat{Q}$ on the constraint rhombus on the Gaussian sphere may be defined

implicitly as

$$\begin{aligned}\hat{Q}\cdot\hat{N}_1 &= \cos(\tfrac{\pi}{2}+\psi) &&= -\sin\psi\\ \hat{Q}\cdot\hat{N}_2 &= \cos(\tfrac{\pi}{2}-\psi) &&= \sin\psi\\ \hat{Q}\cdot\hat{Q} &= 1\end{aligned}$$

where ψ is the angle of uncertainty in the sensed normals. We note that because $\hat{Q}\cdot(\hat{N}_1+\hat{N}_2)=0$, $\hat{Q}$ must be orthogonal to the mean of the two sensed normals; this entails that the worst-case rotation axis is aligned with the normalized midpoint of the sensed normals on the Gaussian sphere.

We now need the angle ϕ between the points $\hat{Q}$ and $\hat{C}$. This can be derived by

$$\begin{aligned}\sin\theta(\hat{Q}\times\hat{C}) &= \hat{Q}\times(\hat{N}_1\times\hat{N}_2)\\ &= (\hat{Q}\cdot\hat{N}_2)\hat{N}_1-(\hat{Q}\cdot\hat{N}_1)\hat{N}_2\\ &= \sin\psi\,\hat{N}_1+\sin\psi\,\hat{N}_2\\ &= \sin\psi(\hat{N}_1+\hat{N}_2)\\ \Rightarrow \hat{Q}\times\hat{C} &= \frac{\sin\psi}{\sin\theta}(\hat{N}_1+\hat{N}_2)\end{aligned}$$

Because $\sin\phi=\|\hat{Q}\times\hat{C}\|$, we have

$$\begin{aligned}\sin^2\phi &= \|\hat{Q}\times\hat{C}\|^2\\ &= \frac{\sin^2\psi}{\sin^2\theta}(\|\hat{N}_1+\hat{N}_2\|^2)\\ &= \frac{\sin^2\psi}{\sin^2\theta}(\|\hat{N}_1\|^2+\|\hat{N}_2\|^2+2\hat{N}_1\cdot\hat{N}_2)\\ &= 2\frac{\sin^2\psi}{\sin^2\theta}(1+\hat{N}_1\cdot\hat{N}_2)\\ &= 2\frac{\sin^2\psi}{1-\cos^2\theta}(1+\cos\theta)\\ &= \sin^2\psi\frac{2}{1-\cos\theta}\end{aligned}$$

$$= \frac{\sin^2\psi}{\sin^2(\theta/2)}$$

so we can express the angle ϕ between $\hat{Q}$ and $\hat{C}$ as

$$\sin\phi = \frac{\sin\psi}{\sin(\theta/2)}$$

Finally, the total angular uncertainty in the location of the cross product of the two sensed normals is just 2ϕ. We note that the uncertainty is minimized, as might be expected, when the sensed normals are orthogonal; in such a case, the uncertainty angle is $\phi = \sin^{-1}(\sqrt{2}\sin\psi)$.

The above describes the worst-case uncertainty for a perfect match of two pairs of unit normals. When the match is imperfect, or when there are more than two pairs of matches, the analysis becomes more complicated.

In the general case, we will suppose that the Gaussian sphere has been rigidly rotated so that the nominal worst-case rotation axis – the normalized average of the two sensed normals – has been aligned with the **Z** axis, such that the two sensed normals lie in the **X-Z** plane. According to this construction, the maximum angular excursion takes a model normal to the limit of its corresponding uncertainty circle; since the rotational axis is **Z**, we can find the excursion if we can find the one or two points of intersection of each [tilted] uncertainty circle with the plane parallel to **X-Y** that passed through the model normal. That is, we must find the coordinates of the points on a tilted circle with some **Z** value of interest.

A construction that readily permits determination of these points is the representation of the circle of interest as that locus which maintains a constant angle with respect to the nominal sensed normal and is of unit length. We thus define the uncertainty circle of a sensed normal $\hat{N}$ to be

$$\begin{cases} \hat{X}\cdot\hat{N} = \delta = \cos\psi \\ \hat{X}\cdot\hat{X} = 1 \end{cases}$$

We seek the components X_1 and X_2 when $X_3 = Z_3$, i.e., when we cut the Gaussian sphere through some point $\hat{Z}$ parallel to the **X-Y** plane. Hence, from

$$\begin{cases} X_1N_1 + X_2N_2 + X_3N_3 &= \delta \\ X_1X_1 + X_2X_2 + X_3X_3 &= 1 \end{cases}$$

we have the constraints

$$\begin{cases} X_1N_1 + X_2N_2 &= \delta - Z_3N_3 \\ X_2^2 &= 1 - Z_3^2 - X_1^2 \end{cases}$$

Let $\alpha = \delta - Z_3N_3$, for brevity. Then by substitution, we have

$$\begin{aligned} X_2^2N_2^2 &= (\alpha - X_1N_1)^2 \\ \Rightarrow N_2^2(1 - Z_3^2) - N_2^2X_1^2 &= \alpha^2 + N_1^2X_1^2 - 2\alpha N_1X_1 \end{aligned}$$

$$\Rightarrow (N_1^2 + N_2^2)X_1^2 - (2\alpha N_1)X_1 + (\alpha^2 + N_2^2Z_3^2 - N_2^2) = 0$$

from which we conclude that

$$X_1 = \frac{2\alpha N_1 \pm \sqrt{4\alpha^2N_1^1 - 4(N_1^2N_2^2)(\alpha^2 + N_2^2Z_3^2 - N_2^2)}}{2(N_1^2 + N_2^2)} \tag{2.4}$$

Examination of the sign of the discriminant of this equation allows determination of the number of points of intersection. We can also see that there is a singularity when $N_1^2 + N_2^2 = 0$, which occurs when the circle's centre is coincident with the **Z** axis.

From Equation 2.4, we can formulate a function $\Phi = F(\hat{N}_i, \hat{Z}_i)$ that yields the maximum rotation about the **Z** axis of the point $\hat{Z}_i$ and yet keeps it in its matched circle about $\hat{N}_i$. Because there are three components to an $SO(3)$ rotation, and F gives us the yaw component about Z, we must in general also rotate the Gaussian sphere in a roll about X and a pitch about Y in order to fully describe the maximum rotation. The upper bound on the maximum rotation

angle for a set of j matches between model normals $\hat{Z}_i$ and sensed normals $\hat{N}_i$ is thus

$$\Phi_{max} = \max_{i=1}^{j} F(\hat{N}_i, \underset{\phi\in[-\psi,\psi]}{\text{Rot}(\mathbf{Y},\phi)} \underset{\theta\in[-\psi,\psi]}{\text{Rot}(\mathbf{X},\theta)} \hat{Z}_i) \tag{2.5}$$

Computationally, this can be estimated by discretizing $[-\psi, \psi]$ and successively rolling and pitching the model normals through the values, then finding the excursion angles in this new rotation. For all orientations which leave all model normals within the uncertainty bounds of their matches, the maximum of these excursion angles is the maximum uncertainty in the pose rotation; the axis of maximum uncertainty can be found by applying the reverse rotations to the **Z** axis, which takes it back into the coordinates of the sensor data.

2.5.2 Translational Uncertainty

Translational uncertainty is much easier to bound than rotational uncertainty, both analytically and approximately, because the transformation occurs in a Cartesian space. We may construct a space in which each point's position encodes the translation which brings the rotated model's faces within ϵ of the assigned positional data. About this point in translation space, for each sensed point, we may place two planes which represent the limits of translation for that datum, i.e., any point that lies between the planes encodes a model translation that brings the model face within ϵ of the point; these are analogous to the constraint lines generated by a face in the 2–D analysis. These constraint planes must be ϵ apart, and are oriented the same way as the nominally rotated model face. Their distance from the nominal translation point is the distance from the sensed position and the rotated, translated model face.

In this translational uncertainty space, we immediately see that it takes at least three mutually independent model faces to form a closed volume for the model translation. As more points are introduced, more constraint plane pairs

are added, and so probabilistically the bounds get tighter and tighter. Analytically, the translational uncertainty is completely specified by the volume that is composed from the intersection of the interplane boundaries formed from each position-point/model-face pair. For three model faces that are mutually perpendicular, with a positional uncertainty of ϵ, the worst-case translational uncertainty is the diagonal of a cube of size ϵ, and so of course must be $\sqrt{3}\epsilon$. When the model faces are askew, the maximum translational uncertainty is the major diagonal of a parallelepiped whose faces are ϵ apart, and so the length of this diagonal can be very large.

2.6 Summary

This chapter presented a description of how to perform the initial interpretation of sparse, local data, which is a crucial stage of the process of recognizing just what object is present in a robot's workspace. The object representation and basic interpretation process are due to Grimson and Lozano-Pérez, and we have provided a description of some significant parts of this process.

We have also presented, in detail, how to derive tight bounds on the rotational and translational uncertainty of an interpretation. This uncertainty is critical in the correct and reliable use of the nominal pose in any attempt to interact the object, whether the interaction be sensing, grasping, or manipulation. We will now present a method for planning sensing paths that distinguish among multiple interpretations of some initial data, in the presence of this uncertainty.

Chapter 3

Finding Sensing Paths in Two Dimensions

"Say from whence you owe this strange intelligence"

The principal conceptual problem in finding a path that will distinguish among ambiguous interpretations of two-dimensional data along which to move a tactile sensor is that of finding a line that pierces a number of line segments, or edges, that constitute the features of the object models. The line is the sensing path, and the segments are unassigned edges from the various interpretations of the data. There are a number of other non-trivial considerations, e.g., how to evaluate the path and how to find it efficiently, but the core problem is that of finding the parameters of a line that passes through a set of segments. An example of a path that passes through a set of edges is given in Figure 3.1.

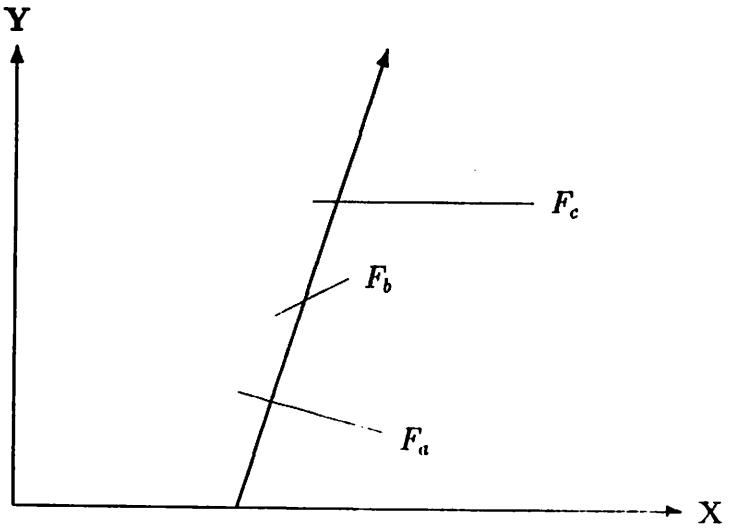

Figure 3.1: A path that pierces a set of edges in the plane.

We will solve this problem by inverting it, asking instead what set of lines pass through a given edge; the intersection of such sets derived from several edges is the set of lines passing through all of the edges.

The sheaf of lines passing through a two-dimensional line segment can be represented as a convex polygon in a new space, which we will refer to as **projection space**. This representation of the set of lines passing through a set of edges is powerful, and greatly facilitates the development of algorithms for finding sensing paths to distinguish planar objects.

3.1 Finding the Path Parameters

In the plane, a line has two degrees of freedom, one positional and one directional. It is possible to restrict the starting position of the sensing path to lie on some locus, e.g., a line or circle (which must satisfy such physical requirements as not intersecting the object to be identified); the two parameters are then the position on this start locus, and the direction of the path.

Without loss of generality, let us suppose that the starting locus is the **X** axis.[1] The endpoints of each edge can be expressed as a pair of points, and the edge is a line segment between these points. Thus, we seek the parameters of a line that intersects the **X** axis, and pierces each of a given set of line segments. Figure 3.2 shows an edge in two-dimensional real space, indicating also the convention for assigning the edge normal.

The path parameters can thus be expressed as the starting position of the path on the **X** axis, and the direction of the path. In order to make it clear that some later formulae have important linear forms, the path direction will be expressed as the slope α of the line, taken with respect to the **Y** axis. That is, if the angle between the path and the **Y** axis is θ, we will deal only with the

[1] We may rotate and translate an arbitrary starting line so that it coincides with the **X** axis, and transform the edges accordingly.

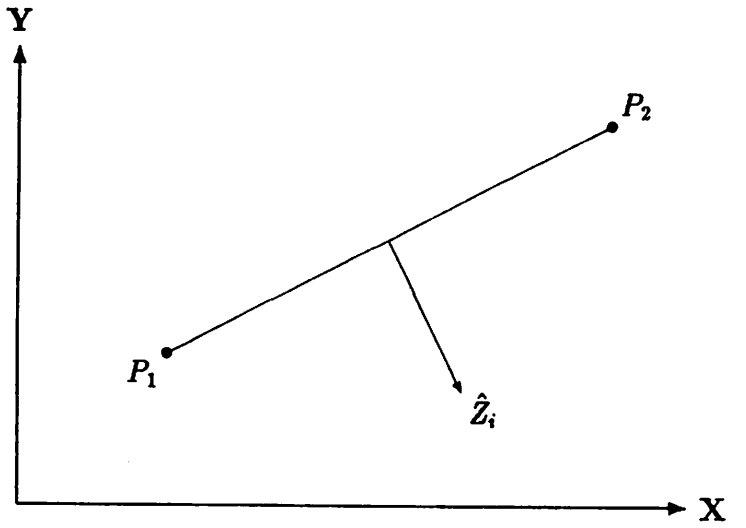

Figure 3.2: An edge in **X-Y** space.

slope of the line which is $\alpha = \tan\theta$. The line will thus be parallel to the vector $(\alpha, 1) = (\tan\theta, 1)$, where α is positive if the line is inclined towards the positive half of the **X** axis.

From these preliminaries the procedure for finding a sensing path can be developed. Beginning with the simplest possible case, suppose that we wish to find a path that intersects a particular point **P**. The bounds on the slopes of the lines passing through **P** will be referred to as α_{min} and α_{max}.

The crucial observation is that the problem can be inverted from finding a path that pierces the point, to finding the sheaf of lines going through the point that satisfy the constraints. For any given slope α, the line passing through the point **P** intersects the **X** axis at a single, determinable point that we will denote as Q. Let us call the intersection of this line with the **X** axis the *projection* of **P**. The projection function, which varies with the slope of the line, is

$$\mathrm{Q}(\alpha) = P_X \quad \alpha \cdot P_Y$$

This function yields the point on the **X** axis that pierces the point **P** with a path

whose slope is α. Figure 3.3 shows the geometry of the projection of a point **P** onto the **X** axis for a given value of α.

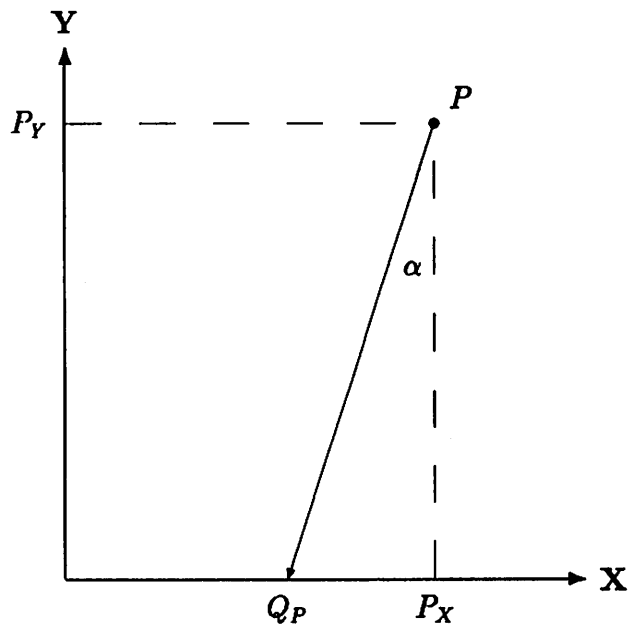

Figure 3.3: Projection of a point onto the **X** axis.

Now, we can represent this projection function as a *line* in a new Euclidean space, which we will call **projection space**. One axis of this space is the original **X** axis, and the other is the angular **A** axis. In this new space, the projection function may be represented as

$$\mathbf{Q}_P(\alpha) = (P_X - \alpha \cdot P_Y\ ,\ \alpha) \tag{3.1}$$

This function may be interpreted as giving the position of a point Q_P in projection space, derived by projecting the original point P (in **X-Y** space) onto the **X** axis in the direction α. Figure 3.4 shows the representation of a point P in projection space.

By virtue of the definition of this line, it has a very useful property: the coordinates of any point on this line directly encode the parameters of a path starting on the **X** axis that pierces the original point **P**. The line Q_P in projection space

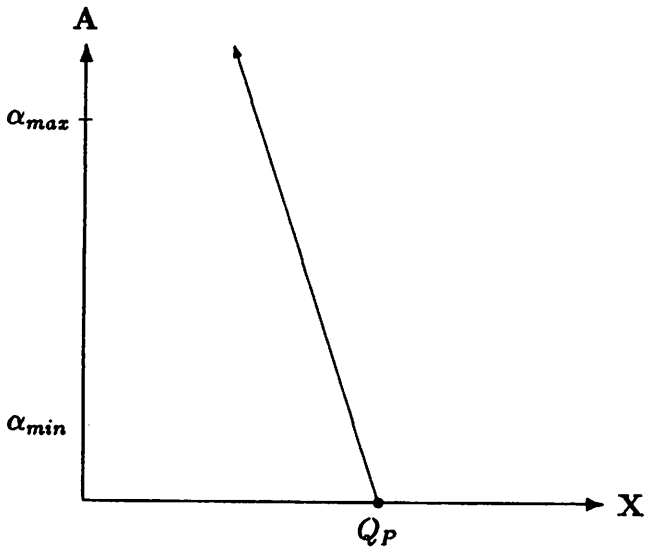

Figure 3.4: Representation of a point in projection space.

thus completely describes the sheaf of paths that pierce **P**, under the constraints we have set out above.

From this basis, we can derive more useful results. In two dimensions, we wish to pierce not points but line segments. This more complex problem can be solved by employing the parametric representation of a line, and using Formula 3.1 to determine how each point on the line segment would project.

A line $L(\lambda)$ can be represented parametrically as a point, and a displacement from this point along some direction D, that is, as

$$L(\lambda) = P + \lambda \cdot D$$

Choosing some particular value of λ gives some point that lies on the line. For line segments, it is customary to let P be one of the endpoints (say, P_1), and the direction vector $D = P_2 - P_1$, which in general is not a unit vector. The points on the segment are given by values of λ bounded by $0 \leq \lambda \leq 1$.

The projection formula for a point on a line is derived by substituting the line point into the projection Formula 3.1. The new formula, which is a function of

the path slope α and the line parameter λ, is

$$\begin{aligned} \mathbf{Q_L}(\lambda, \alpha) &= (L_X(\lambda) - \alpha \cdot L_Y(\lambda)\,,\ \alpha) \\ &= (P_X + \lambda \cdot D_X - \alpha \cdot P_Y - \lambda \cdot \alpha \cdot D_Y\,,\ \alpha) \end{aligned} \tag{3.2}$$

where the subscripts indicate components of the line, point, and direction. This formula is non-linear in α and λ. However, for fixed λ, it is linear in α; in particular, the endpoints of a segment become lines in projection space. If D_Y is nonzero, i.e., if the line segment is not parallel to the **X** axis, then by Formula 3.2 the projected lines of the endpoints will have different slopes. Figure 3.5 shows the representation of an edge in projection space; the boundary is not a parallelogram because the edge was originally tilted with respect to the **X** axis.

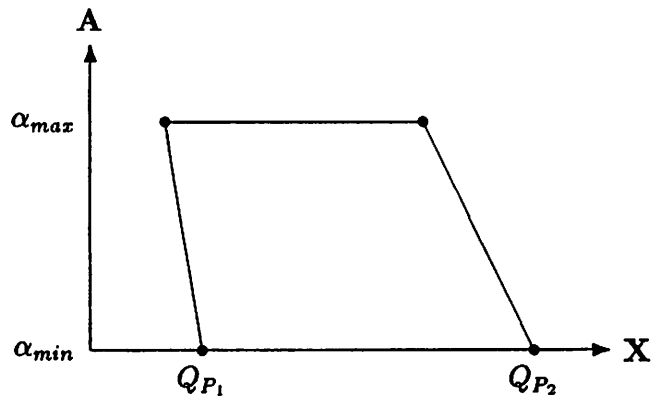

Figure 3.5: Representation of an edge in projection space.

Note that the locus of points in projection space that is described by this projection, when α and λ are bounded independently, is a trapezoid. In particular, the parallel segments of the trapezoid are parallel to the **X** axis, and the other segments have the slopes described above.

The points in the interior of this trapezoid have the property that their coordinates represent the parameters of a sensing path that pierces the desired line segment. If we take two distinct line segments, and represent each in projection

space, the result is two trapezoids. The intersection of these two trapezoids is a convex set of points, whose coordinates describe the parameters of the set of paths that intersect *both* of the original line segments.[2] Figure 3.6 shows the representation of two distinct edges in projection space; the area of intersection is the set of points that specifies the sheaf of paths that pierce both original edges.

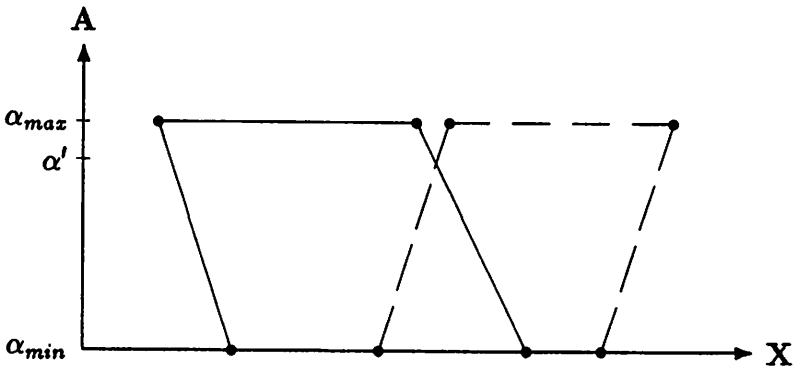

Figure 3.6: Two edges in projection space, and their intersection.

The general two-degree-of-freedom problem can thus be expressed, in these new terms, as finding a point that is in the interior of the intersection of a number of trapezoids in projection space. In either approach, all that is sought is a *single point* inside the polygon which is the intersection of the projection trapezoids. The coordinates of this point represent the position on the X axis and the direction α of a line that passes through the edges whose projection is part of this intersection. Once these parameter values have been found, the rotation and translation can be reversed, and the start point and path direction expressed in the natural coordinates of the X-Y plane.

Implementation of this sensor path planning system requires attending to the two difficulties of efficient computation of a path, and of ensuring that the path

[2] If the sets do not intersect, then there is no line that obeys the constraints and passes through both edges.

could actually be used by a robot system. These issues are closely related, because some of the constraints of realistic sensing can be used to reduce the complexity of finding the path parameters.

Our approach to calculating a sensing path involves examining each edge F_i in turn, and trying to find a path through it and as many other edges as possible. Assuming that the sensor can contact an edge at very oblique angles, we can form the set of edges $\{F_j \cdots F_k\}$ such that the angle between the normal $\hat{Z}_i$ of F_i and the normal $\hat{Z}_j$ of any edge F_j in the set is greater than 0. The set $\{F_i, F_j, \cdots F_k\}$ will be referred to as the **candidate set** of F_i, which will be called the **generating edge** (or generator, for short). Note that the candidates in this set must include the edges already sensed; it is possible that a path through the generator also contacts one or more of these assigned edges, though the contact will likely be at some distance from the previously sensed point if the path is a good one.

In outline, our algorithm for finding a sensing path is:

1. Calculate the candidate set of each edge.
2. Sort the generators according to how many interpretations are present in the candidate set.
3. Find and test a feasible path through the generator and its candidates:
   (a) Find the projection parameter α' which creates the maximum overlap of candidate edges with the generating edge.
   (b) Find an **X** value, in this projection, that is in the intersection of the projections.
   (c) Determine the ability of the path to distinguish amongst the current interpretations.

The two parts of this algorithm requiring the most explication are finding the value of the projection parameter that produces the best overlap, and evaluating the quality of the path.

3.2 Computing the Path Parameters

In projection space, the representation of a given edge is a convex polygon, and the coordinates of points in its interior and boundary represent parameters of the sheaf of lines passing through the edge. Thus, the parameters of the sheaf passing through a set of edges is represented by the intersection of the sheaves of each edge, which is also a convex polygon; let us call this the **sheaf polygon**. The problem we must solve, then, is finding a *single* point in the interior of the sheaf polygon that is produced by intersecting the projections of edges from as many different interpretations as possible. This search for a point must be done in the presence of uncertainty about the exact location of the edges; for clarity of exposition, we will discuss the problem as though it is geometrically exact, and later describe how to compensate for uncertainty in location of the edges.

The two closely related subproblems of determining what is the largest number of projections that intersect, and of finding a point in this intersection, can be solved efficiently by taking advantage of a simple geometric observation. Suppose that a given set of edges $\{F_i \cdots F_j\}$ can be pierced by some line; the intersections of the projections, then, forms a convex sheaf polygon. Our minimal problem is solved if we can find *any* point in this polygon. Let us concentrate our attention on the vertices.

A vertex of a sheaf polygon is produced when the representation of two or more edge endpoints intersect in projection space. For brevity, let us call the intersection of the projections of any two endpoints a **critical point** in projection space. Figure 3.7 shows the projections of three edges; the critical points that lie within the $[\alpha_{min}, \alpha_{max}]$ bounds are noted as dots at the intersections, and their α values are marked as dashes on the vertical **A** axis.[3] The region labelled **S** (which is bounded above by the line $\alpha = \alpha_{max}$) is the sheaf polygon for all three edges.

[3]Some of the critical points in this example have the same α value, so there are fewer dashes marked on the **A** axis than dots in the figure.

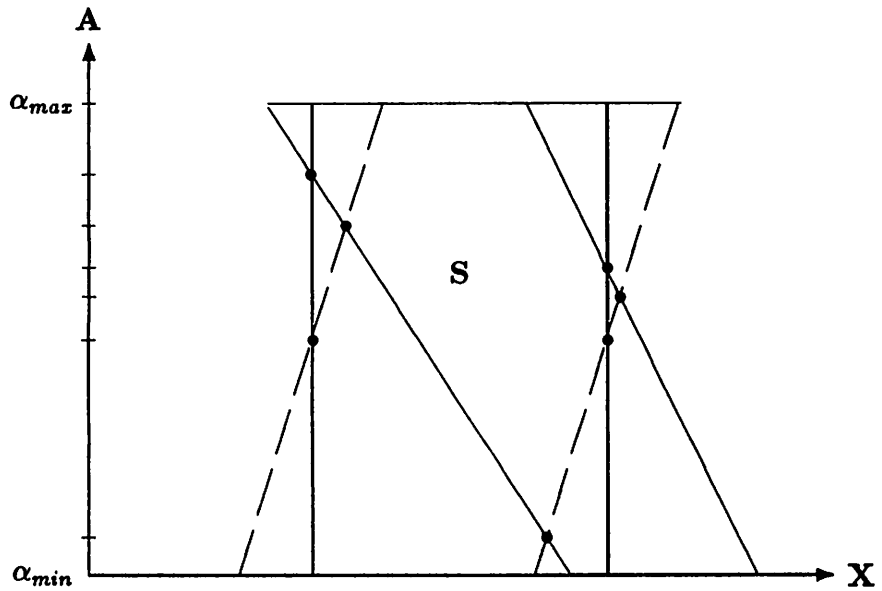

Figure 3.7: Three edges in projection space.

The critical points are indicated as dots, and their α values are marked on the **A** axis.

Observe that by definition, a critical point $\mathbf{c} = (\mathbf{x}', \alpha')$ is on the boundary of a sheaf polygon if and only if the *projections* of all of the edges overlap on the **X** axis when projected by the value α', i.e., when we project the real-space edges onto the real-space **X** axis, the intervals that are projected must overlap. Equivalently, the edge representations in projection space must overlap at the value $\alpha = \alpha'$ if **c** is a critical point; in particular, all of the edge projection intervals must overlap $\mathbf{x}'$, which can be tested efficiently.

This means that in order to find a point in the intersection region of the largest number of edges, we need only project the edges by those α values given by the critical points and then examine the resulting overlaps, rather than searching through all of the values in $[\alpha_{min}, \alpha_{max}]$ to find *both* the region of best overlap and some point in this region. The critical points of edge boundaries encode exactly

those regions of projection space where qualitative changes of edge overlaps occur.

Checking for intersection at a critical point is quite straightforward. For a pair of edge endpoints P and R, the critical point occurs when the projections are equal,which is when $c = Q_P = Q_R$. Solving for α', the critical point occurs at

$$\alpha' = \frac{P_X - R_X}{P_Y - R_Y}$$

which is undefined when $P_Y = R_Y$, i.e., when the points in **X-Y** space are the same height from the **X** axis there is no line that connects both of the points and also intersects the **X** axis. This value α' is used in the projection formula of each edge in turn; the critical point bounds a sheaf polygon including the edge if and only if the **X** value of the critical point lies between the **X** value of the edge endpoint projections at α'. Thus, we need only count the number of edges whose projections bracket the critical point to determine how many edges can be pierced by a sensing path that has a slope of α'.

The **X** value of the critical point can be used as the starting point of the sensing path, but it has properties which may render it undesirable. In particular, it represents the starting point of a path which just barely contacts two or more edges. This can be remedied, if necessary, by re-examining the edges under the projection at α', finding the extent of the overlap of the edge projections, and taking some other **X** value, e.g., the middle of the overlap. From linear programming theory, we know that the maximum width of a convex polygon (in any given direction) occurs at a vertex; thus, the critical points can be searched to find not only the largest *number* of overlapping edges from different interpretations, but also the largest *extent* of overlap. This second pass would occur only on a small subset of critical points, and would use projection values that were calculated previously and thus would already be available.

The present system sorts the critical values from the middle of $[\alpha_{min}, \alpha_{max}]$ outwards, and determines how many distinct interpretations have an edge whose projection contains the critical point; this is the overlap count for the critical

point. The largest overlap count is determined, and one of the points with the largest overlap count is selected for processing (this is currently done arbitrarily). The α' of the critical point represents the path direction, and the midpoint of the intersection region represents the starting point of the path. Once found, however, this path must be evaluated to determine how well it distinguishes amongst the various possible interpretations.

3.3 Verifying the Usefulness of a Sensing Path

That a path intersects several edges does not imply that a tactile sensor can determine which edge has been contacted. There are limits to the ability of sensors to discriminate depth and orientation, and regardless, it is possible for several edges to coincide exactly. These conditions must be examined to determine how good a path is at reducing the number of interpretations.

Three properties of tactile sensors are that they have a finite ability to discriminate depth and contact normals, and that if the contact angle is too oblique then the sensed normal value may be unreliable. Recalling that δ_{sensor} is the cosine of the expected maximum error in sensing the local surface normal, ϵ_{sensor} is the maximum expected positional error, and that Ω is the maximum desired contact angle between the path and the object edge, these practical constraints can be summarized as:

- The edge normals must be distinguishable by the tactile sensor, i.e.,
  - for any pair of normals $\hat{Z}_i$ and $\hat{Z}_j$, $\angle(\hat{Z}_i \hat{Z}_j) > 2 \cdot \delta_{sensor}$, **or**
  - The positions of the edges must be distinguishable by the tactile sensor, i.e., for any pair of edges F_i and F_j, the points of contact P_i and P_j must be such that $\|P_i - P_j\| > \epsilon_{sensor}$; **and**
- The angle between the sensing path direction D and the normal of any edge F_i must not exceed Ω, i.e., $\angle(D\hat{Z}_i) < \Omega$.

Once a path has been found, all pierced edges must be examined to determine how these constraints apply. It often happens that edges appear in similar configurations, and so they could not be distinguished by a tactile sensor.

3.3.1 Pose Uncertainty and Path Evaluation

As was noted in Section 2.4, uncertainty in the values of the initial data lead to uncertainty in the calculation of the object pose. This pose uncertainty leads in turn to uncertainty in the position of an object edge: a path that is intended to strike an edge must be evaluated to ensure that the edge is not missed entirely. This uncertainty can be incorporated directly into the search for the parameters of a sensing path.

The excursion of an edge due to pose uncertainty can be determined by finding the position of the edge in the two poses that are derived from the outer bounds upon the space of pose rotations, i.e., we calculate the maximum rotation Φ_H and the minimum rotation Φ_L that bring the model into correspondence with the data normals, and find the nominal translations for these rotations. The edge position is then computed for each of these values.[4]

Under a given projection, each of these edge positions will yield an image on the **X** axis. The intersection of the images gives the set of starting points which are guaranteed to contact the edge regardless of its true position; the union of the images represents the set of starting points which *might* contact the edge. Figure 3.8 gives an example of an edge's projection at its two pose limits, the images it forms under a projection, its outer left and right projection bounds O_L and O_R, and its inner projection bounds I_L and I_R. These represent, respectively, the worst *interference* that an edge can present to a path that is not intended to

[4]For typical sensor uncertainty values propagated through the interpretation process, rotational uncertainty vastly dominates translational uncertainty. Once rotational uncertainty is managed, translational uncertainty can usually be incorporated in a straightforward manner, by offsetting in the appropriate direction.

strike that edge, and the *guaranteed* region that is presented to the path in the presence of the uncertainty in calculated pose.

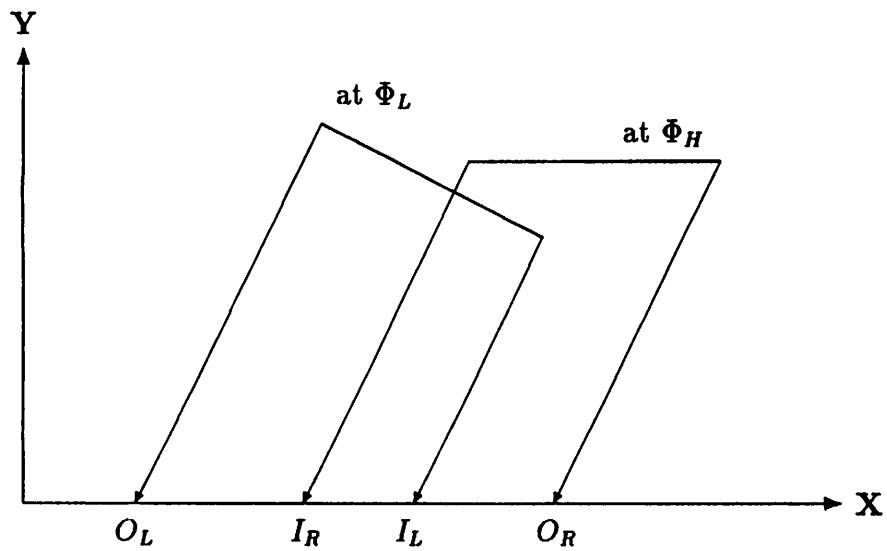

Figure 3.8: Projection of an edge at its pose limits.

This property may be used in both the original formation of the path and in its evaluation. In forming the path, a conservative strategy dictates that the uncertainty be incorporated by finding the nominal critical points, then taking only the inner projection values of each edge when testing for projection overlap; in this way, one can be very sure that the path will contact those edges that it is planned to contact, if the object is indeed in the robot's workspace.

In resolving the distinguishing power of the path, the outer limits should be used to determine which edges in the candidate set *might* be contacted by the path. A very good path will pierce only those edges it needs to pierce, and will avoid those that could provide spurious information.

This simple approach to incorporation of pose uncertainty will fail if the uncertainty is very large, because the projection of an edge can wander significantly from its nominal position. It should be noted, however, that modern sensors are sufficiently accurate that small angles can be accurately distinguished; and re-

gardless, in the presence of large uncertainty, it is quite difficult to make *any* firm geometric statements at all; in such circumstances, one must employ reasoning strategies which do not hope to describe the potential ambiguity of interpretation as carefully as we seek to here.

3.4 Evaluation of a Path: The Ambiguity Tree

Our recognition strategy provides a method for gathering new tactile data in order to reduce the ambiguity of interpretation of the current data. A path is not always perfect, i.e., it does not invariably reduce the number of possible interpretations to a unique one; in particular, as the number of interpretations increase, the chance of finding a perfect path becomes very small. When it is not perfect, though, the path can be viewed as *reducing* the ambiguity; as tactile data are (or are not) gathered, the amount and kind of information changes. We can express the effectiveness of the algorithm, in a given case, by the way in which the ambiguity is reduced if particular data are detected along the paths it finds. The reduction in ambiguity can be expressed as a tree, which we will call the **ambiguity tree** for the experiment.

Initially, there is a set of possible interpretations of the given tactile data. As a path is traversed by the sensor, a new tactile datum may be gathered. If there is only one possible edge that this datum could be assigned to, the datum uniquely identifies the interpretation and thus can be considered to be a terminal node in a tree. Sometimes, however, the datum might possibly be assigned to edges from more than one interpretation; in such a case the datum has reduced the ambiguity, but not eliminated it. We can say that such a datum constitutes a non-terminal node in a tree, and that if this datum was detected by the sensor then at least one additional path will be needed to uniquely determine the correct interpretation of the object. An important kind of non-terminal node is formed by the equivalence class of *unexpected* data. These unexpected data can arise when the object is unknown to the system, when the path-finding algorithm has been prematurely

stopped before it found the optimal path (the path that distinguishes the largest number of ambiguity classes) or when an edge is struck at too oblique an angle for the sensor to detect reliably the local surface normal or position. Suboptimal paths most commonly arise, though, because the path-finding process can be prematurely stopped according to various criteria, e.g., by reaching a limit on computation time, or by having found a path that identifies some given number of interpretations.

The path-finding algorithm, then, can be viewed as producing a tree of successive refinements of the description of the tactile data. At any level in the tree, the width represents how many equivalence classes of interpretations are formed by the path; the depth below a node in the tree represents the worst-case number of paths that must be traversed before the object is identified.

An example ambiguity tree is given in Figure 3.9. Suppose that there were five interpretations of the original data that were valid. In this instance, the first level gives the number of interpretations distinguishable by each sensing incident. For sense datum S_4 of level 1, there were two edges from distinct interpretations that were indistinguishable; hence, if the sensor contacted that edge we would know that none of the other three interpretations were possible, but could not distinguish between I_2 and I_4. An additional path would be required in the worst case, for a total of two movements.

An ambiguity tree can be used theoretically, when the path-finding process is stopped prematurely, to evaluate the algorithm's performance when the design parameters (computation time or identification thresholds) are varied. Practically, as each level of the tree is computed, it can be used to guide the computation of new paths, e.g., non-terminal nodes which have the greatest ambiguity may be expanded next, while the current path is executed by the robot manipulator.

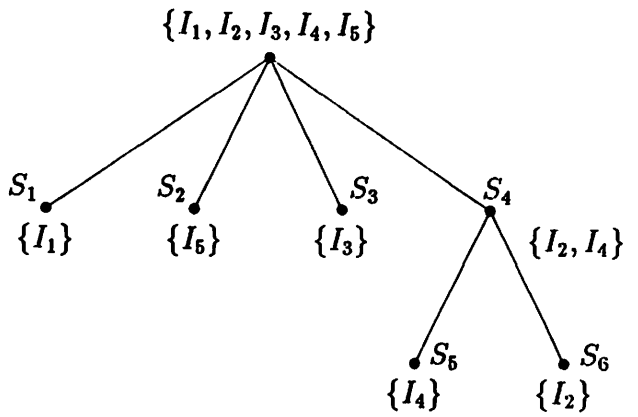

Figure 3.9: Ambiguity tree for a set of interpretations.

3.5 Combinatorics of Parameter Determination

There are several stages to the algorithm as described, each with different computational cost. The important stages are formation of candidate sets, finding the critical points, finding the regions of overlap, and resolving the sensing conflicts for the path.

Formation of the candidate sets is relatively simple, as it just entails determining which edges could conceivably have a path pass through both the generating edge F_i and the candidate edge F_j. This means that we need to determine whether the normal $\hat{Z}_j$ of F_j is not antiparallel to the generator normal $\hat{Z}_i$, so we can simply check to ensure that

$$\hat{Z}_i \cdot \hat{Z}_j > -1$$

This must be done for all pairs of edges, so if there are M unsensed edges from all possible interpretations, and K sensed edges, there are $M \cdot (K + M - 1)/2$ inner products to be computed.

Once the candidate sets are formed, the largest is selected and path parameters

are found. We will denote the size of each candidate set by N. Because the size is determined by how many edge normals are not antiparallel to the generator, in general N will be only slightly smaller than M.

The number of critical points is a linear function of the size of the candidate set. Each candidate endpoint must be tested against the generator endpoints, and to this we add the values $\alpha_{min}, \alpha_{max}$, and 0. Thus, there are $4N + 3$ to be examined; of these, we discard those critical values that are outside the bounds and remove redundancies. The critical values are then sorted by absolute value, which is typically an operation of complexity $O(N \log(N))$. The entire process of finding and preparing critical points is thus $O(N \log(N))$ in complexity, as the complexity of sorting the points dominates the complexity of finding and testing them. This stage usually proceeds quickly because of the low cost of the operations.

The next stage is finding the overlap regions of the edge projections. Each endpoint in the candidate set must be projected under each α' found, and then tested against the $\mathbf{x}'$ component of the critical point. Because there are $2N$ endpoints and, at worst, $4N + 3$ critical points, the complexity of this stage is $O(N^2)$. Once a path has been proposed, the pierce test is repeated on the edges, in case there are edges that overlap the generator at this α projection without overlapping the generator endpoints. The contact points are then found, and sorted according to their position along the path; since the test dominates, this stage has a net complexity of $O(N^2)$. There is a significant amount of work to be done for each α, so this stage tends to dominate the computation.

Once the set of edges pierced by the path has been found, we determine the first contact on each interpretation by linear search; there can only be as many of these as there are interpretations, so the number is quite small. Supposing that there are I interpretations, and all of them are contacted, we must then resolve the sensing conflicts. This entails determining the contact angle and contact distances for each pair of points, which requires I^2 tests. Forming the equivalence classes

of contacts is a linear operation, so resolution is of complexity $O(I^2)$.

We may thus conclude that finding and evaluating a sensing path through a given candidate set is of complexity $O(N^2)$. At worst, we must search through every candidate set to find the path, which would require $O(N^3)$ calculations. However, practice has shown that most good sensing paths can be found by examination of just a few of the most populous candidate sets, so finding a sensing path is a quadratic function of the number of unsensed edges. The dominant part of the calculation is determining and testing overlaps of candidates with the generator.

A final observation is that although this algorithm will almost certainly find a good path, if one exists, it will not necessarily do so in the most efficient manner. We employ some heuristics to improve the likelihood of finding a good path early in the search, but which path of the desired quality will be chosen (and how quickly it will be chosen) can be highly variable. In the next chapter, we will examine the mean performance of an implementation of this algorithm, in both simulation and in hardware tests, to better assess its qualities.

Chapter 4

Experimental Results

"The instruments of darkness tell us truths"

Our two-dimensional object identification approach has been experimentally verified, both in simulation and in implementation on a robot system. The simulations permit rapid exploration and evaluation; the implementation provides a demonstration that the approach is feasible in the presence of real errors produced by hardware exploration of an object in the robot workspace.

Our algorithm has been implemented on a VAX 11/750, in a combination of LISP and C. This combination provided a comfortable development environment that also permitted efficient implementation of those portions of the algorithm which were computationally intensive. We will now give a *précis* of how our system operates, and later note how details change between simulation and actual operation.

4.1 Overview of the Algorithm

The first phase of our algorithm is **acquisition** of the initial data. In simulation, this can be done either by the user selecting some data, or by selecting a model from which data are randomly generated. In operation, the robot system moves the tactile sensor through the workspace in a systematic manner, conducting a binary subdivision of the workspace until independent surface normals have been gathered.

Next, we must perform the **recognition** phase, which includes pruning the interpretation tree, finding the pose parameters, verifying global geometric con-

straints, and finding the uncertainty bounds on the pose parameters. The result of this phase is a list of valid interpretations, including the pose parameters. Every edge of every model is transformed into its interpreted position, and for each edge a candidate set is generated. These sets are then ordered according to the number of distinct interpretations present in the set.

In the **path planning** phase, each candidate set is examined in turn. First, the set is rotated and translated so that the generator lies on the **X** axis. Second, critical points are found, values very near each other are compacted into a single value, and the critical values are sorted from 0 outwards according to their absolute value, reasoning that contact nearly normal to the generator is most desirable. The result of these operations is a set of edges ready for piercing, and a set of α projection values.

For each α value, a candidate edge is projected onto the **X** axis. If it overlaps either endpoint of the generator edge, it is kept for further examination. When all of the candidates have been projected, a path is formed from the set of edges whose projection overlaps each endpoint generator; the starting value is the middle of the intersection area. This path is then re-projected through each candidate face as a line, the λ values are accumulated and sorted, and the first contact of each interpretation is found.

These contact values are then resolved to determine the distinguishing power of the path. The contacts are tested according to the distance of contact and local surface normal; indistinct contacts are grouped into ambiguity classes. The result of this operation is a list of ambiguity classes, representing how good this path is at determining which interpretation may be correct. The path is categorically rejected if it might possibly contact any face at an angle that is too oblique, so that only reliable information is gained.

At this juncture, the path planning ceases if either the path uniquely determines which interpretation is correct or if the number of unique contacts exceeds a provided threshold value. This latter check allows the user to prevent the sys-

tem, in very complex situations, from searching exhaustively for the best path and instead to accept a lesser path. A typical setting, which experience has shown works moderately well, will stop searching the present candidate set when 1/2 of the interpretations are uniquely contacted.

The final detail of the algorithm is dependent on whether the system is simulating or operating. In simulation we are interested in the full ambiguity tree; in operation, we want the information provided by the tactile sensor to further guide the processing.

In simulation, when an acceptable path is found, the system recursively decomposes the problem and solves for each ambiguity set. This is done by directing attention to only those interpretations which are ambiguous with respect to the present path, pruning the present candidate sets so that all other interpretations are eliminated. This collection of candidate sets and interpretations are then processed, and so on until the problem is completely solved.

In operation, the **execution** phase is when the sensor is moved about the perimeter of the workspace (which is assumed to be unoccupied) and the path is then followed. If no contact occurs, then the appropriate branch of the ambiguity tree is invoked and a new path is planned. If contact occurs, then the newly acquired datum is re-interpreted, resulting in a new set of valid interpretations. If this set of interpretations is not a unit set, a new path is planned.

4.2 Simulations

Testing of our approach proceeded by providing the values of some initial data and a set of models, interpreting the data, and planning distinguishing paths for a sensor. The purpose of the simulations was to determine the efficacy of our approach under various conditions; we thus produced the entire ambiguity tree for evaluation, and generated graphics displays to assist our understanding of the system performance.

The resultant ambiguity tree provides a measure of the efficacy of the algorithm at solving the given problem. For simple problems, or one in which the system has managed to find a perfect path, the tree is completely flat. For more complex problems, it can be quite deep; it is interesting to note, however, that in most such trees the most common ambiguity for our set of test objects is two-fold, i.e., it is relatively rare for a contact to result in a set of three or more ambiguous interpretations. By far the most common cause of large ambiguity sets is the path's failing to touch any of the interpretations: a lack of contact will obviously not provide any positive information, although it serves to eliminate from consideration those interpretations which should have provided contact but did not.

We have run several thousand simulations with our system, demonstrating the effectiveness of our approach at generating identifying paths for two-dimensional objects that are modelled as polygons assumed to be lying within the robot workspace at arbitrary orientations. The most extensive series of tests were conducted with the objects shown in Figure 4.2. The experiments involved using only two tactile data, which are shown in the midst of the objects – the dots are the positional information, and the spikes the direction of the local surface normal. These data were chosen because of the considerable ambiguity with which they can be interpreted. There was a very small amount of positional error associated with each datum ($\pm$0.5mm, which is less than the thickness of the lines in the drawing), and a normal direction error of about $\pm 4°$.

The simplest experiments attempted to distinguish various poses of each object. An example of such a run is given in Figure 4.2, which shows the four superposed interpretations of the stylized robot gripper. The path, starting at the perimeter, is indicated by the bar that ends at the innermost interpretation; this path is capable of determining exactly which pose is the correct one. In this experiment we assumed that the sensor could determine local surface normal and depth very well and had a sensor skid angle of 89°, i.e., any slight touch of the

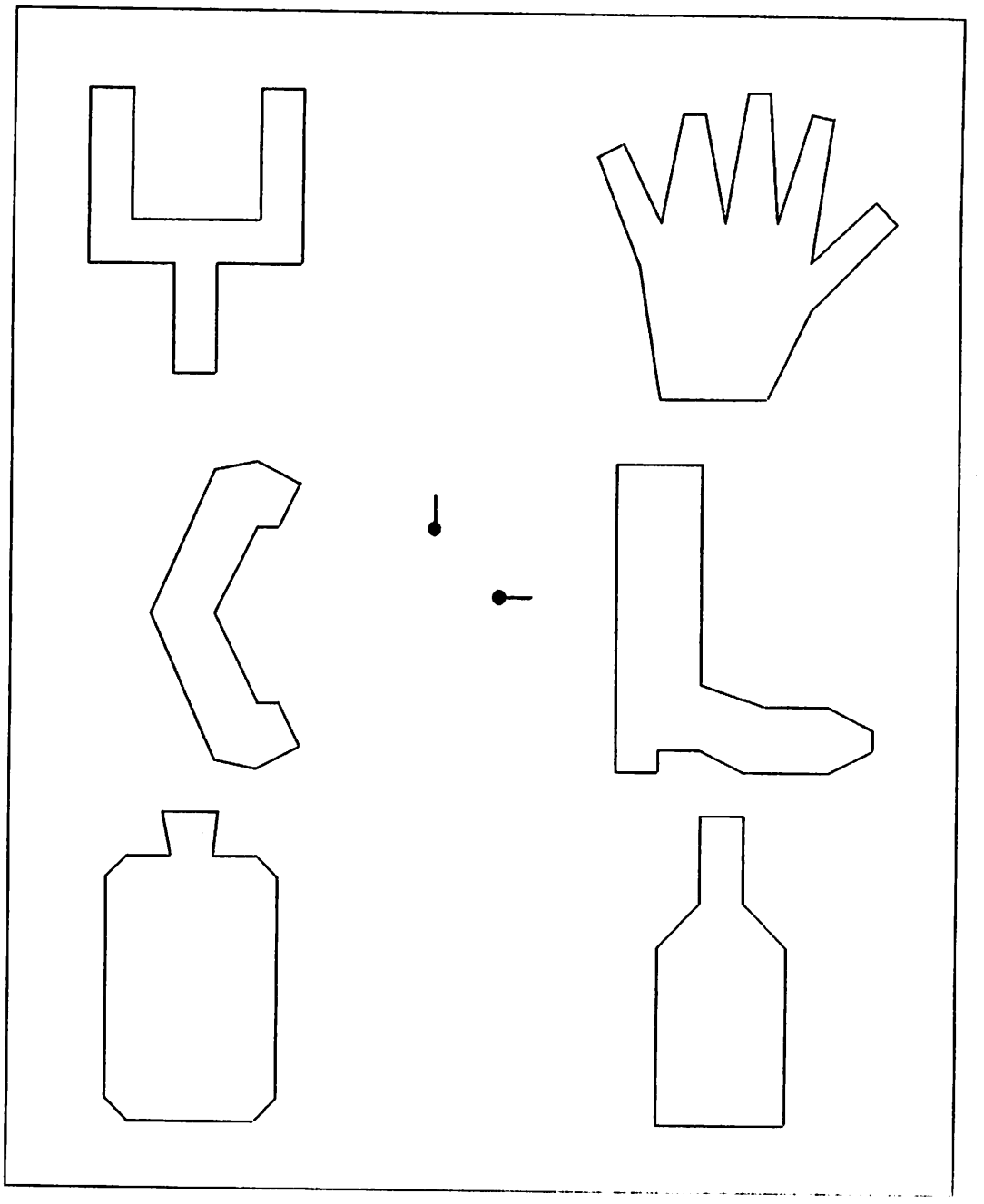

Figure 4.1: Object models and initial tactile data used in the experiments. The reader can find interpretations of these objects by copying the tactile data onto a transparent sheet, and moving the sheet about to find places where the position and local-surface-normal constraints are simultaneously satisfied.

surface would be sufficient to gather data. The path actually contacts each face parallel to its surface normal, so the latter design parameter could be tightened considerably without affecting this result.

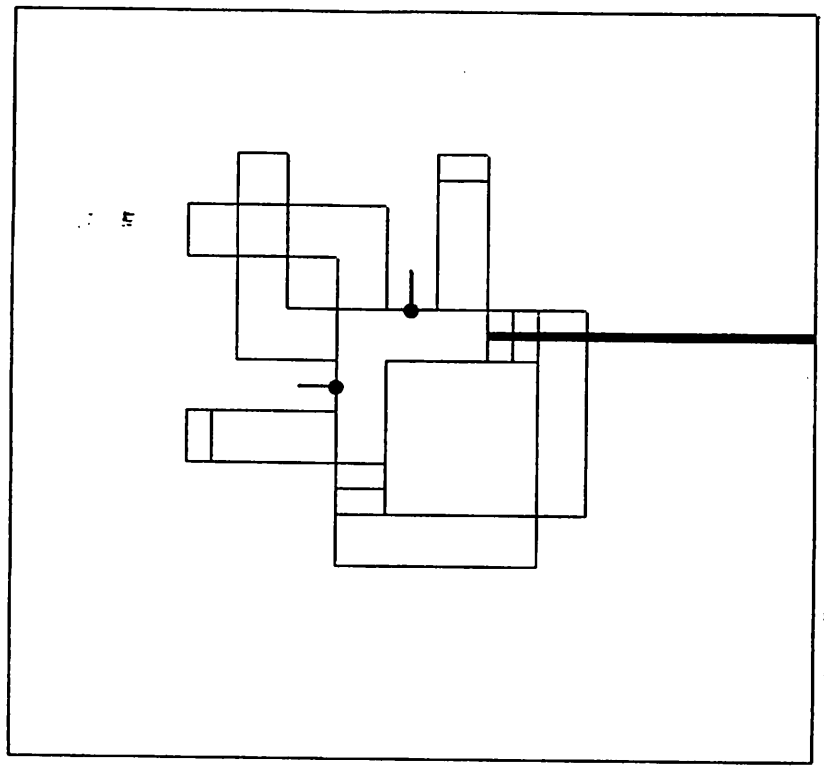

Figure 4.2: Path for ambiguous interpretations of the robot gripper.

The path for the most complicated object, the crude line drawing of the human hand, is shown in Figure 4.3. Although there are not very many interpretations of it, the number of faces resulted in candidate sets that were quite large. The path is capable of distinguishing the interpretations by contact with large external feature faces of the object.

An intriguing example of a path is that for the boot, shown in Figure 4.4. Observe that the path contacts one of the interpretations on a face *that has already been sensed*. However, this seemingly redundant contact is effective: if the first interpretation is *not* contacted, then there is no longer ambiguity in the

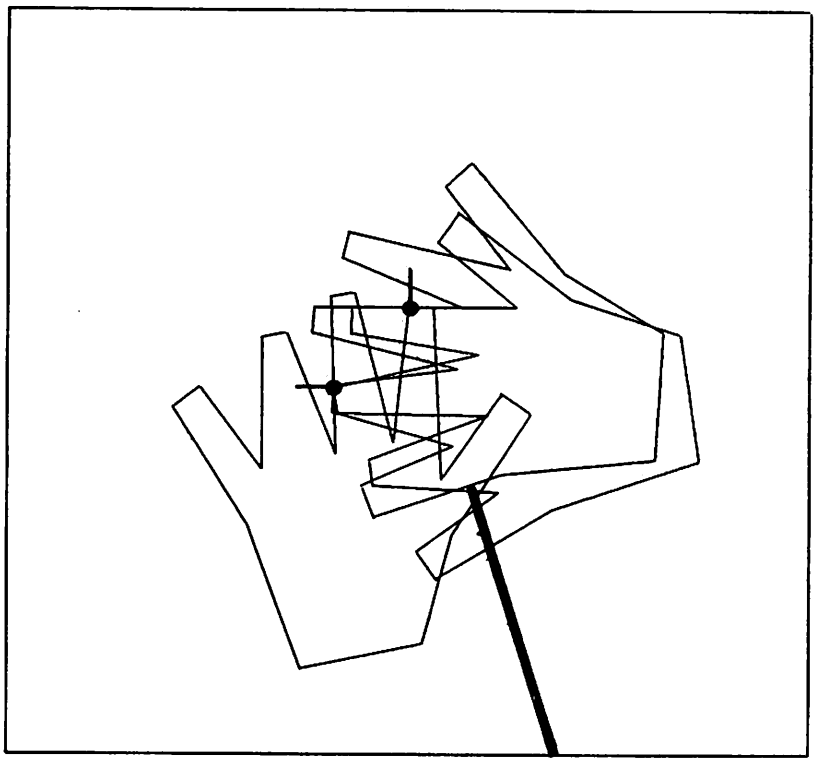

Figure 4.3: Path for ambiguous interpretations of the "hand".

second contact, and so a touch on a previously sensed face (but in a new and unsensed region) is effective.[1]

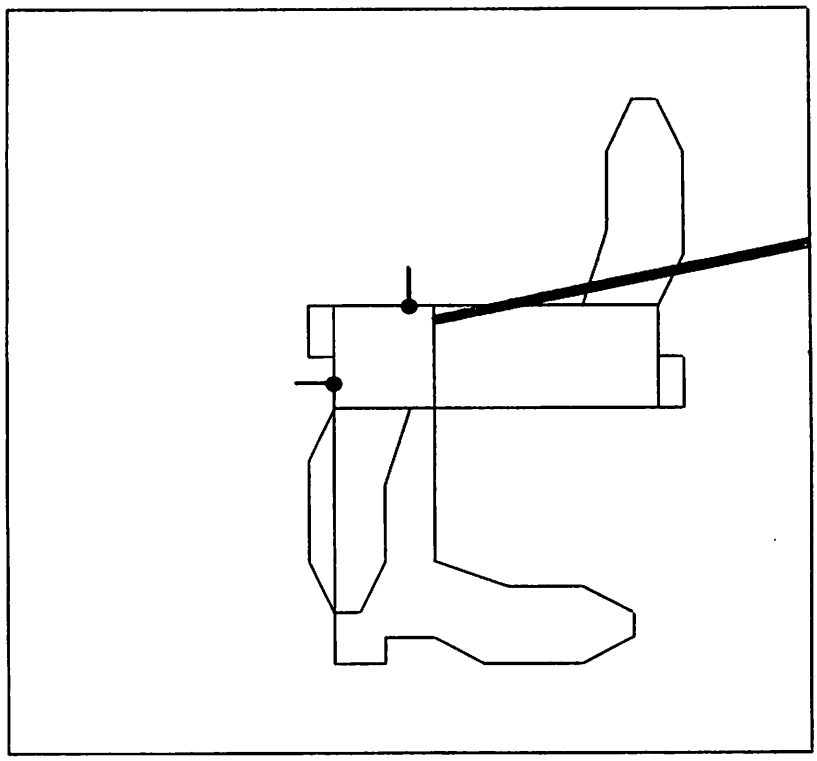

Figure 4.4: Path for three ambiguous interpretations of the boot.

The path that was most dependent on the amount of rotational uncertainty was that for the model of the video camera, shown in Figure 4.5. With a high accuracy in sensing the surface normal, the object's possible positions could be computed very accurately and thus the computed path would work. The path, however, is very sensitive to rotational uncertainty, as slight rotations would carry some of the faces far from the path, leaving an ambiguity. Indeed, only a small increase in the error cone of the sensor resulted in an ambiguity that always requires at least two paths, both aiming at the lens areas of the camera (which, along with its elongation, is the principal distinguishing feature of the camera).

[1] Here the system out-smarted the designer, who had not expected this situation.

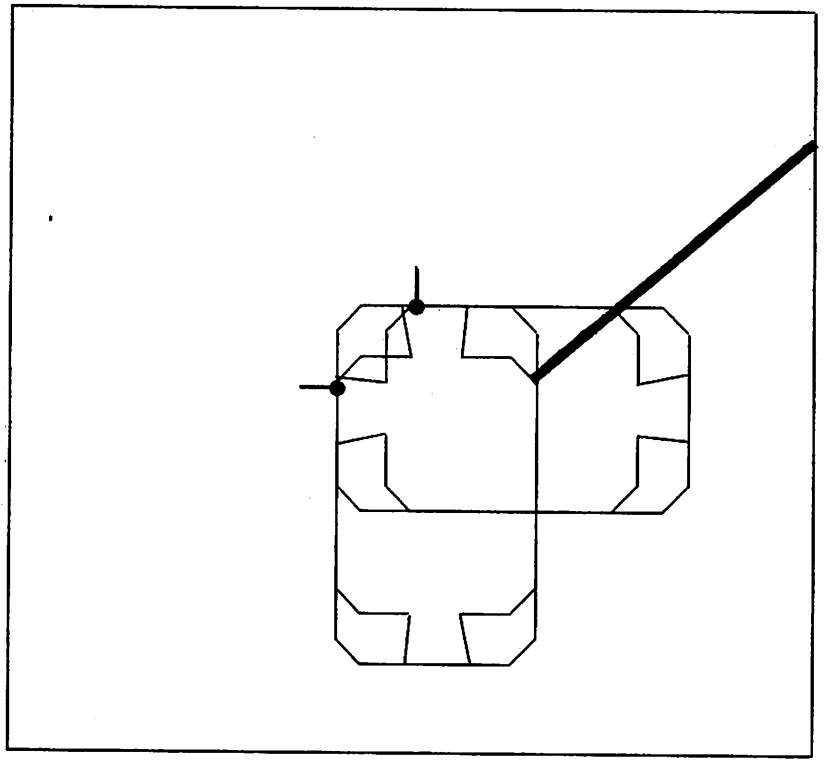

Figure 4.5: Path for ambiguous interpretations of the camera.

These simple experiments showed that even where there are a number of coincident faces, the algorithm can select paths that distinguish all of the interpretations. Table 4.1 gives the time required, on a dedicated VAX 11/750, to find all interpretations of each object and to find a perfect path for distinguishing among the interpretations. To be fair, we should point out that the recognition phase is written almost entirely in LISP, while the path-finding phase has the core modules written in C. Experience has shown that a factor-of-two speedup is not unreasonable in going from LISP to C (with the implementation of LISP we use, and our coding style), and the VAX 11/750 can no longer be considered a fast machine, so these times can be reduced by an order of magnitude or more by careful implementation on current dedicated hardware.

Table 4.1: Interpretations of Standard Points.

Object	Number of Faces	Number of Interp's	Time for Recognition (CPU sec)	Time for Perfect Path (CPU sec)
robot-hand	12	4	5.6	1.3
human-hand	18	3	8.3	2.5
telephone	12	2	4.2	0.4
boot	13	3	5.2	1.0
camera	12	4	6.5	1.9
beer-bottle	8	3	3.3	0.7

Although they are more difficult to represent, the cases in which more than one kind of object may be present are a tougher test of the system's ability to plan distinguishing paths. In particular, because we can more easily construct difficult and confusing cases, the multiple-object cases provide non-trivial ambiguity trees that permit us to better evaluate the system performance.

The results of some multiple-object runs are given in Table 4.2. The first column gives the object pairs; we omit the recognition time because it is simply the sum of the recognition times of all objects. The second column indicates the time

needed to search exhaustively for the first perfect path (if one exists). The third column indicates the time required when search was limited by stopping when a path identified at least two interpretations. The last column gives the number of identified interpretations at each level of the ambiguity tree for the limited-search paths of the test set, e.g., in row 2, the first path identified three interpretations and the second path identified the remaining two. The difference between the second and third columns indicates some of the tradeoffs between path planning time and path execution time; for instance, in the last row, computation is reduced by almost 7 seconds and the resulting path identifies 6 out of 7 possibilities. By applying various stopping criteria and examining the corresponding ambiguity trees, it is possible to tune the performance for a particular set of objects.

Table 4.2: Distinguishing Multiple Objects.

Objects	Interp's Found	Time for Perfect Path	2-Unique Path	
			Calculation Time	Interp's Identified
robot-hand human-hand	4 3	14.0	14.0	6, 1
telephone boot	2 3	36.1	5.1	3, 2
camera beer-bottle	4 3	12.2	5.5	6, 1

It is illuminating to examine in detail at least one of these runs; let us do so for the case in which either the telephone receiver or the boot may be present. There are a total of five interpretations by these two objects from the given tactile data; the limited search resulted in the ambiguity tree shown in Figure 4.6. The first path results potentially in three contacts, all of which are unique. If no contact occurs, it could be either of two remaining interpretations, and a single path exists which can distinguish these. Thus, although two paths have been

planned, it is most likely that only the first will need to be executed in order to determine the object type and its pose in the robot workspace.

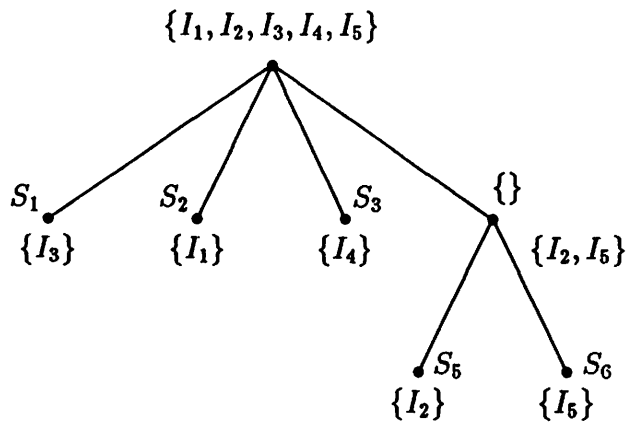

Figure 4.6: Ambiguity tree for two objects and limited search.

The first path for this test case is shown in Figure 4.7. The second path, which distinguishes the two remaining interpretations, is given in Figure 4.8; in order to express the known ambiguity at execution, the system omits display of the three interpretations which have been ruled out by the previous path.

The last issue which we examined in simulation is how often we need to plan a path and how expensive it is, i.e., how often do ambiguous interpretations occur and what is the average path count and solution time. This issue was addressed by choosing a model, randomly selecting a pair of initial data points on its perimeter which had surface normals that were distinguishable by the system, and running the full recognition and identification procedure on the points.

Table 4.3 summarizes the raw results of such a simulation for the same sets presented above. In this experiment, data were randomly selected from the first model in the set, then from the second model, and so on; in all, we selected 50 pairs from each model in each set. The issues we explored in this experiment were the quality of paths that resulted from searching for a moderate amount of time,

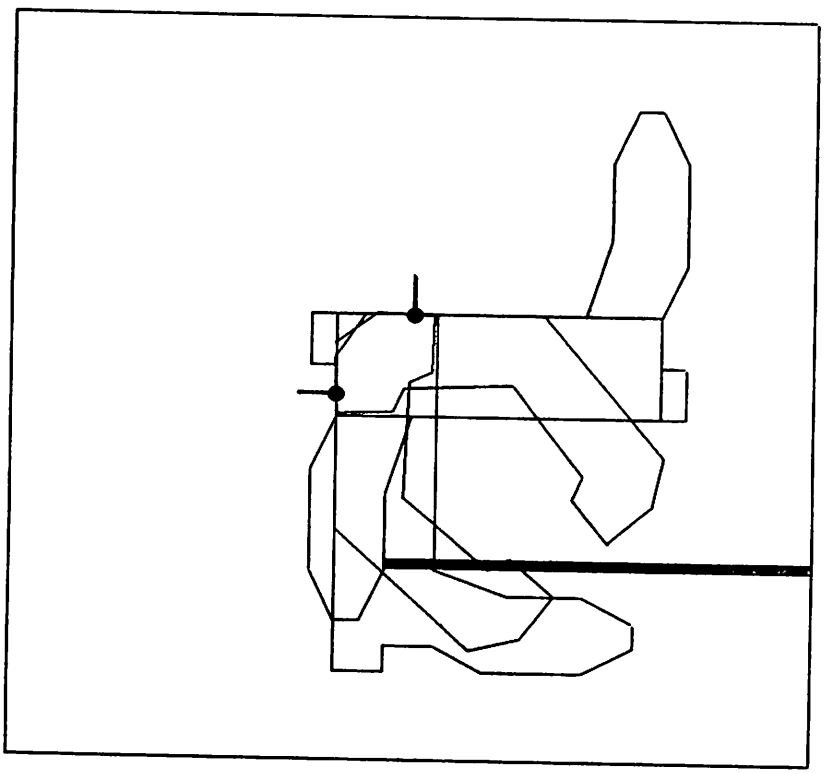

Figure 4.7: First path for ambiguous interpretations of two objects.

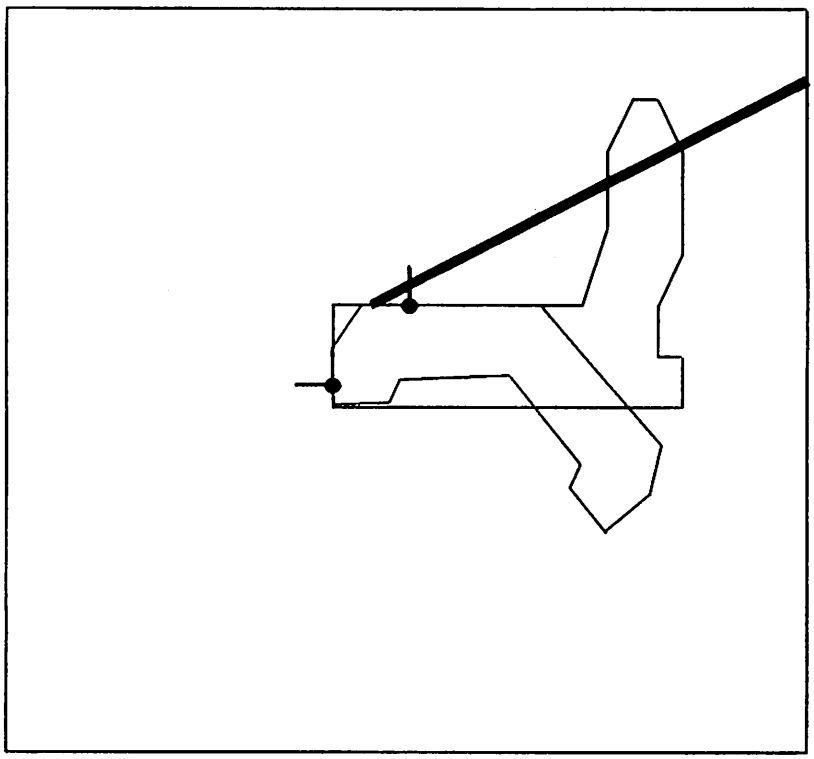

Figure 4.8: Second path for ambiguous interpretations of two objects.

and the paths that resulted from searching for a specified distinguishing power. These constraints represent the typical criteria of finding the best answer before a hard time deadline is met, *vs.* finding the first answer that reduces the problem to a more manageable state.

The first column of Table 4.3 indicates the mean ambiguity, i.e., the mean number of interpretations of the initial data. The second column is the mean time needed to conduct the entire recognition procedure. In the third column we have the mean time required to find the best path within 60 seconds, that is, we run until either a perfect path is found or 60 CPU seconds have elapsed; this column also notes the mean depth of the ambiguity tree that results from a time-limited search. The final column lists the times and depths when the stopping criterion was that at least three interpretations were distinguished by the path; since this measure makes sense only for those cases where there are more than three interpretations present, it represents the average of data somewhat fewer runs than the total.

In comparing Table 4.3 with Table 4.1, there are a few anomalies that require explanation. Beginning with the robot hand, we see that although the mean number of interpretations of random data is about the same as for the standard data, the random data appear to have taken much longer to interpret. This is because a few instances of extraordinary ambiguity skewed the statistics. The planning time for the "human" hand is unusually low, principally because there are so few ambiguities in interpreting random data (the face–face angles tend to be unique).

We can see from this table that for objects that provide a number of distinguishing features (i.e., all but the camera), the time needed to plan even a very good path is on average less than the time needed for recognition. The very good path sets tend to identify the correct pose with a single path; but even in moderately difficult cases, where there are about four interpretations of the initial data, we can plan one or two paths very quickly indeed. As the number of

Table 4.3: Interpretation and Path Planning from Random Data.

Objects	Mean # of Interp's	Time for Recognition	60 sec. Path		3–Unique Path	
			Time	Depth	Time	Depth
robot-hand	3.66	15.66	10.87	1.15	3.62	1.46
human-hand	1.22	7.52	1.12	1.00	0.0	0.0
telephone	1.37	4.16	0.80	1.00	4.02	1.50
boot	1.70	5.17	2.57	1.00	2.80	1.38
camera	2.12	5.09	25.85	1.37	15.40	1.78
beer-bottle	1.73	3.20	0.46	1.00	1.50	1.75
robot-hand human-hand	6.10	25.19	14.02	1.11	17.69	1.53
telephone boot	3.70	15.98	2.68	1.00	3.84	1.41
camera beer-bottle	4.90	13.67	22.38	1.27	10.30	1.93
All Models	22.6	30.65	53.36	1.47	43.03	2.11

interpretations increases, we begin to see the rise of the quadratic search curve for each candidate set: when all models are present, there are many possible interpretations and the path-planning times are definitely beginning to dominate. This suggests that for large databases of models, we might want to gather more initial data randomly to reduce the potential number of interpretations, or use some different approach to data acquisition. The tradeoffs are implementation-dependent, involving robot motion time, recognition time, planning time, and sensor accuracy, as all of these affect the result.

We can conclude that two-dimensional path planning for the intelligent acquisition of tactile data from a small set of known objects can be done quickly. In particular, the total cost of planning all of the paths, for moderately complicated cases, is about the cost of recognition, and is much less than the actual acquisition time of real robot systems. From our combinatorial analysis of the problem, however, we can also conclude that for large data sets the path planning time will eventually exceed the sum of the acquisition time and the recognition time, and for large data sets some alternative approach may be preferred.

4.3 Demonstrations

The above simulations indicate the theoretical potential of our approach to path planning in two dimensions. We have implemented this approach on a real robot system, using the hardware described in Appendix A. The objects for these tests were less whimsical and more prosaic, being parts from an electric hand drill. The parts, shown in Figure 4.9 and whose two-dimensional plans are shown in Figure 4.10, are respectively the left and right plastic housings of the drill and its motor stator armature.

These presented several challenges to the system, not the least being that they were truly three-dimensional curved objects that were modelled as two-dimensional polygons. Although the sensor was very accurate, being capable of

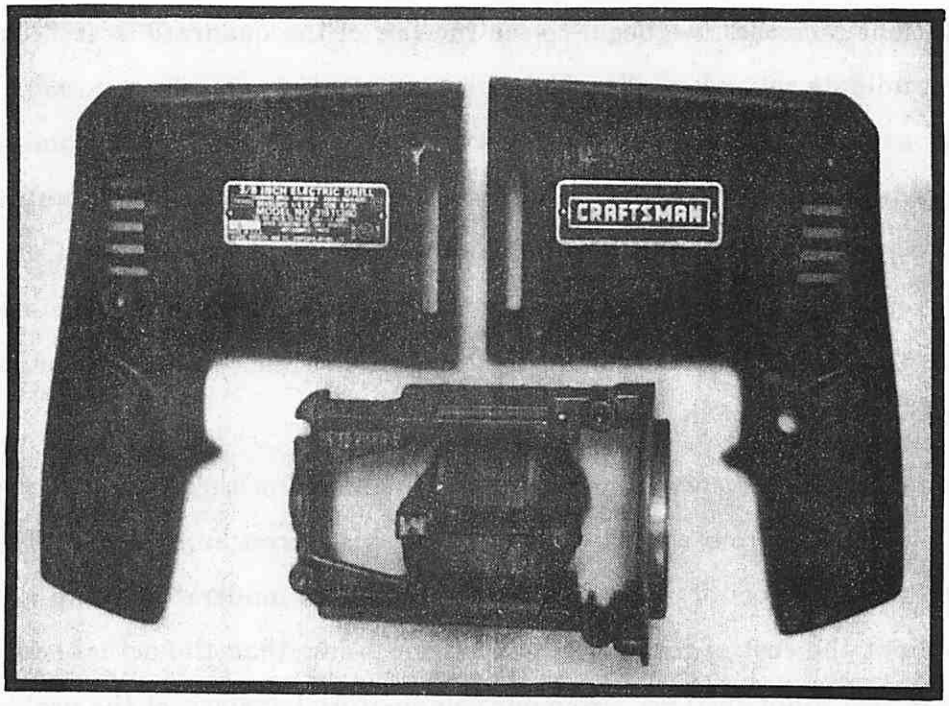

Figure 4.9: The objects used in the actual hardware experiments.

determining a planar surface normal within $\pm 1°$ and a position within 0.5mm over the entire workspace (290mm by 320mm), we were forced to broaden the uncertainty to $\pm 5°$ and 4.0mm respectively, to account for errors in modelling.

The results of the demonstration runs were in excellent accordance with the simulation results. Because it usually took about 25 seconds to acquire a tactile data pair, most efficient recognition took place when the time to plan a path was less than about 7 seconds. The only significant problem encountered was that the sensor could not reliably deduce the local surface normal of the motor housing, because in certain configurations the support beam of the sensor touched the object; thus, the housings was used principally as a source of further ambiguity and confusion, testing the system's ability to plan paths. This system has been run hundreds of times on a large variety of object configurations; we will present a coherent set of runs to summarize the results. Figure 4.11 shows the manipulator, holding a sensor, as it is contacting one of the objects.

A typical run is shown in Figure 4.12, where the only object presumed to be

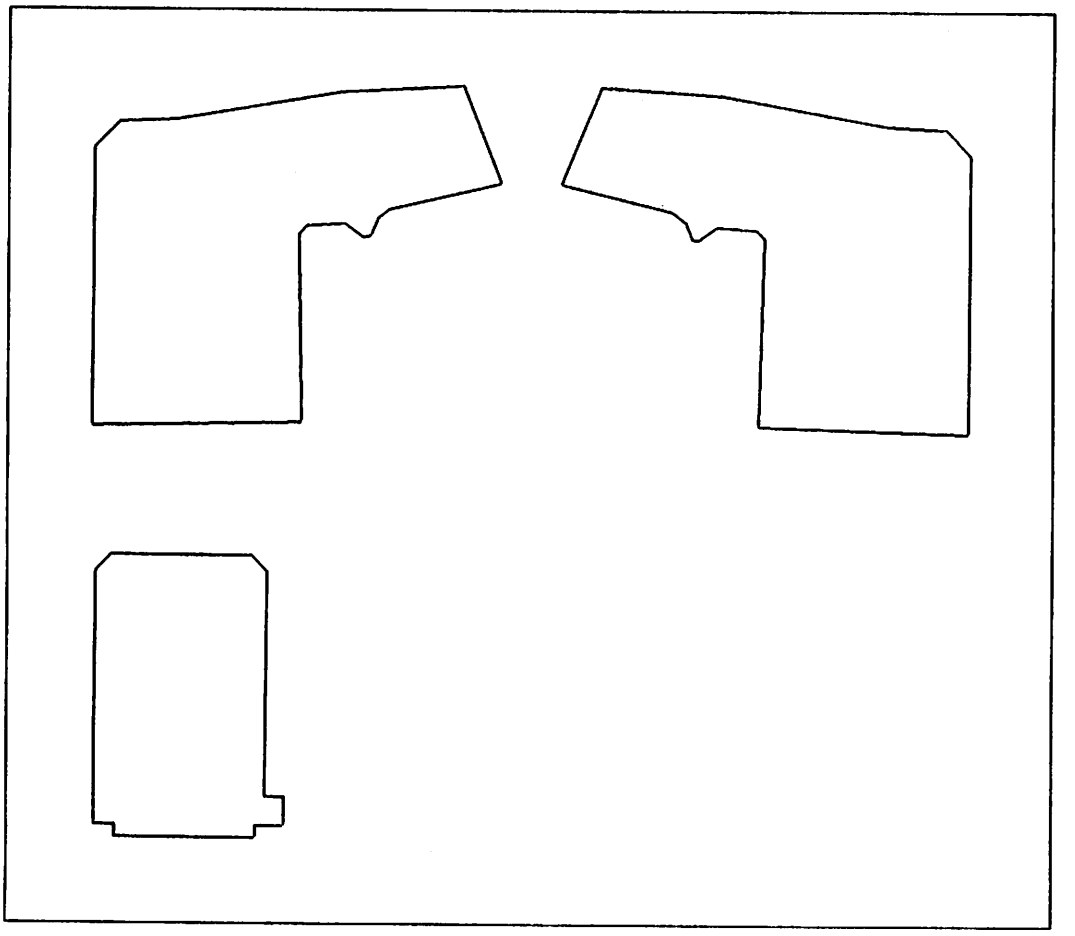

Figure 4.10: Object models used in the hardware experiments.

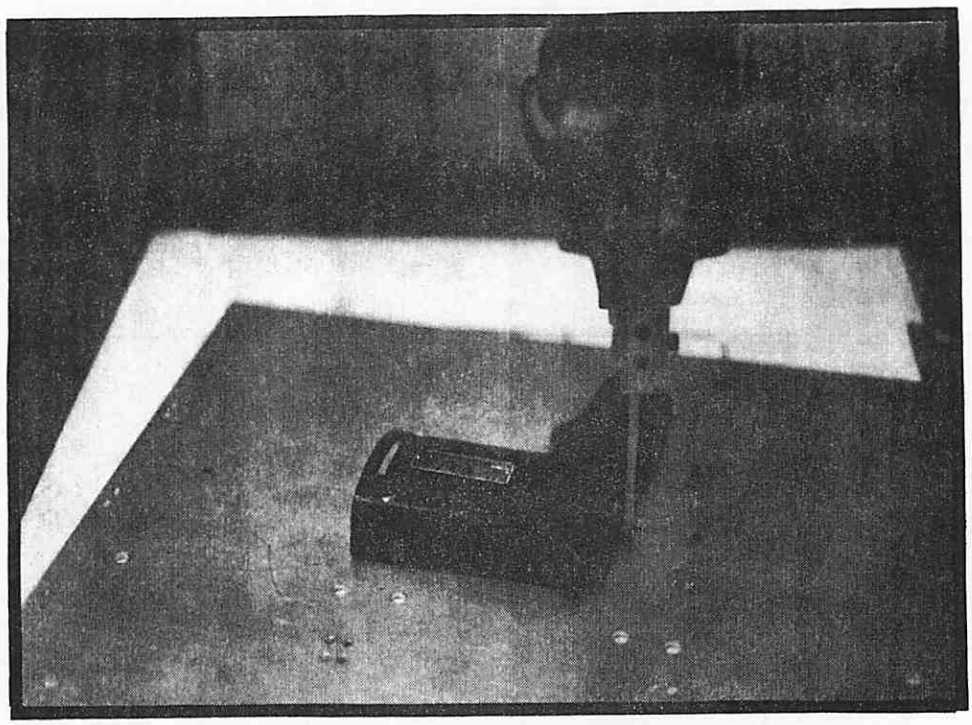

Figure 4.11: The system in operation.

in the workspace is the right section of the drill casing. The initial scans of the workspace bisect it horizontally and vertically, but because the drill was tilted the normals do not point along the directions of the initial paths. The system plans a path that will contact all three interpretations, one of which was the true one. Thus, with a single extra probe, it could uniquely determine the drill casing location and orientation.

A more complex example is when the right casing is in the workspace, but the system is told that either the left *or* right casing may be present. Figure 4.13 shows the first path for this confusing state, and Figure 4.14 shows the final state. Table 4.4 summarizes the total time taken for different phases of this experiment. Note that the times for recognition and planning are stated as total elapsed time, and thus include the times for various system overhead functions (including LISP garbage collection) as well as basic computation time. We can see from this table that the time spent in recognition and planning is considerably less than the time spent in acquisition of the data, indicating the efficacy of our

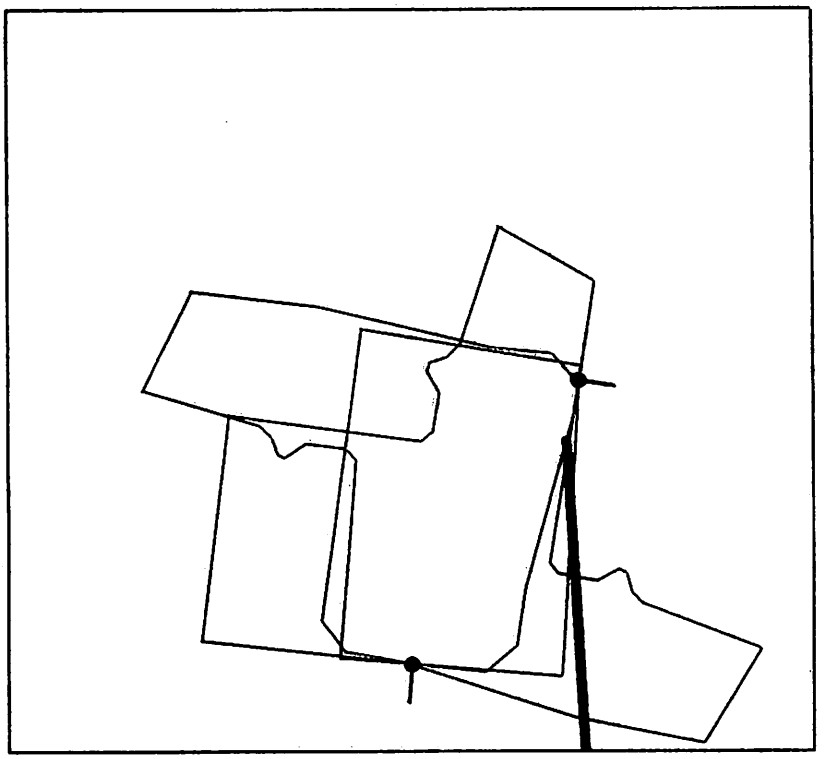

Figure 4.12: Path for the right section of the drill casing.

approach to intelligent acquisition of data for model-based object recognition in two dimensions.

Table 4.4: Elapsed Times for Complete Recognition Experiments.

Objects	Initial Data Points	Recognition Time	Path Planning Time	Path Execution Time
Drill-right	26	12	1.5	6
Drill-left Drill-right	26	17	5	10

(All times are measured in seconds.)

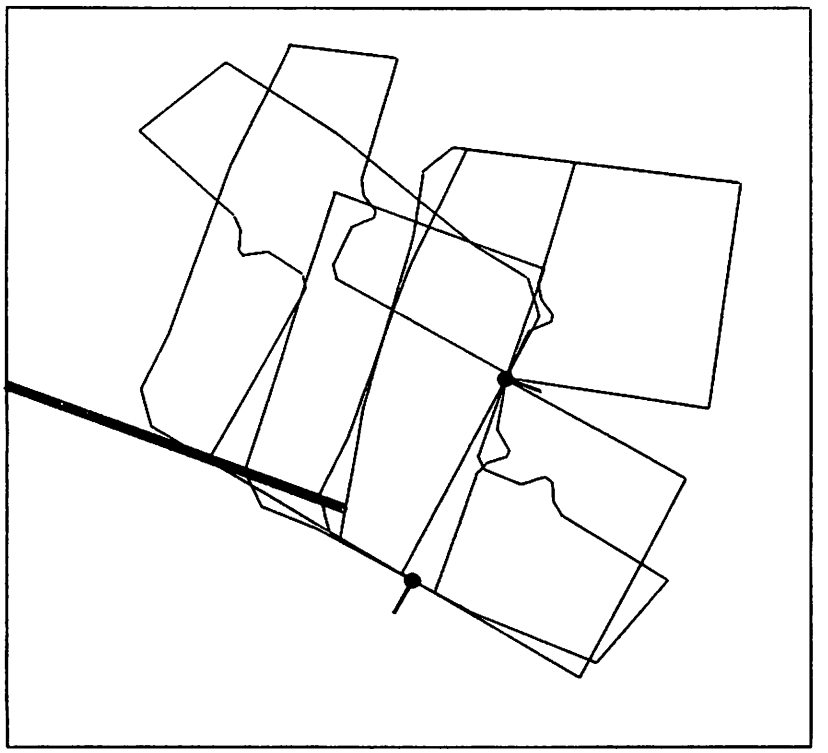

Figure 4.13: First path for the left and right sections of the drill casing.

We will conclude with a somewhat surprising example, that indicates the

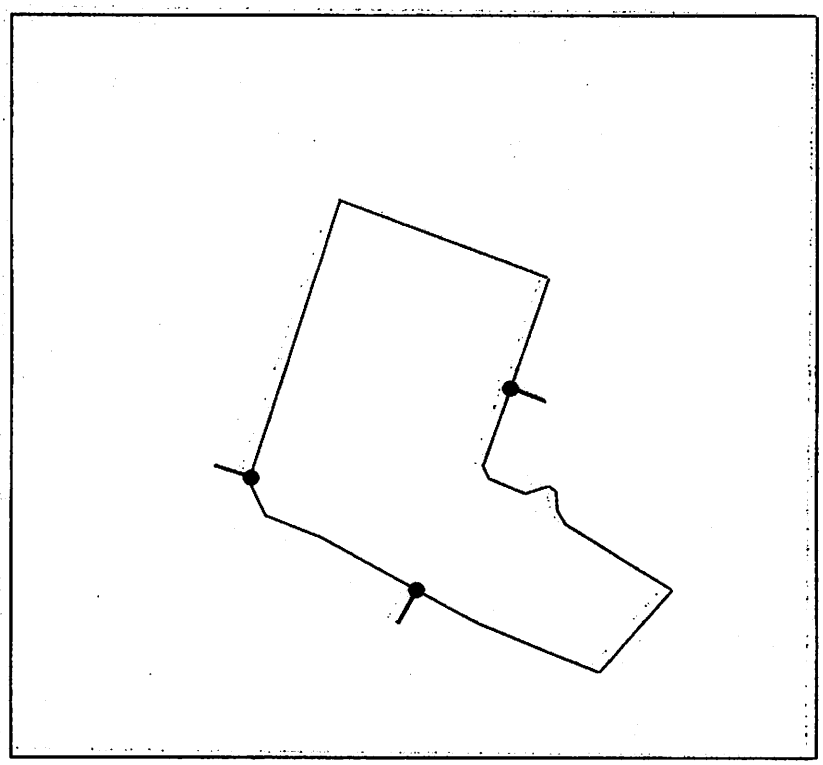

Figure 4.14: Final state for the left and right sections of the drill casing.

accuracy with which our sensor and system operate. Assume that there is only the right drill casing in the workspace, and that the data error bounds are the measured sensor values of ± 0.5mm and $\pm 1°$. In one such run, from the two initial contacts there were three interpretations; as we might expect, there is very little pose uncertainty and thus even small features are geometrically stable. The path planned in Figure 4.15 terminates on one such small feature, the small upper rear corner of the drill casing. When this path was executed, the sensor did indeed hit the feature it aimed at, resulting in the contact and deduced pose shown in Figure 4.16. With good hardware and careful object modelling, we see that high-quality paths can be planned and executed by existing robot systems.

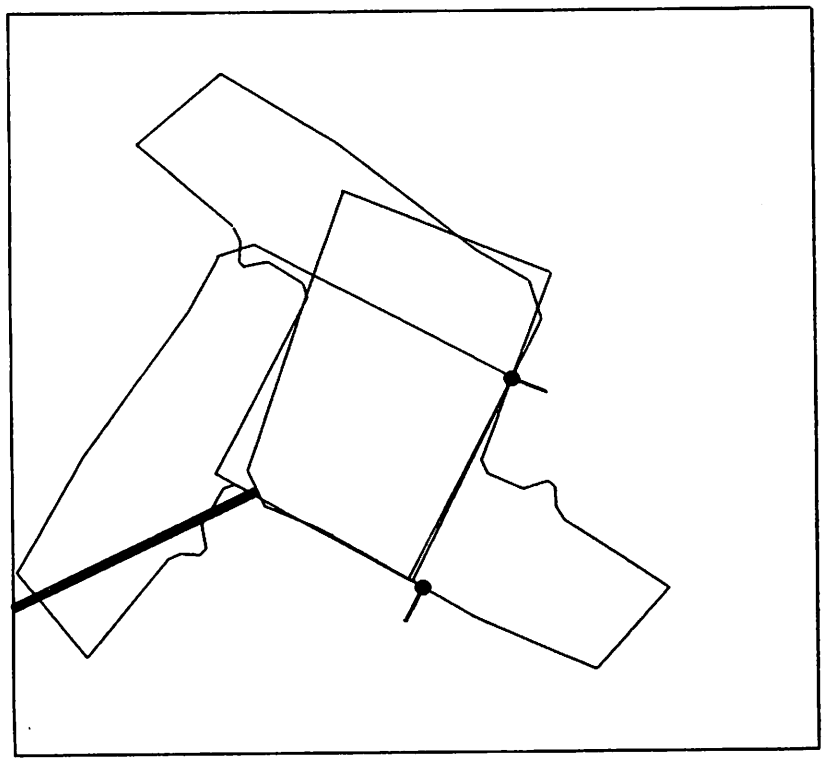

Figure 4.15: First path for the right drill casing, low uncertainty.

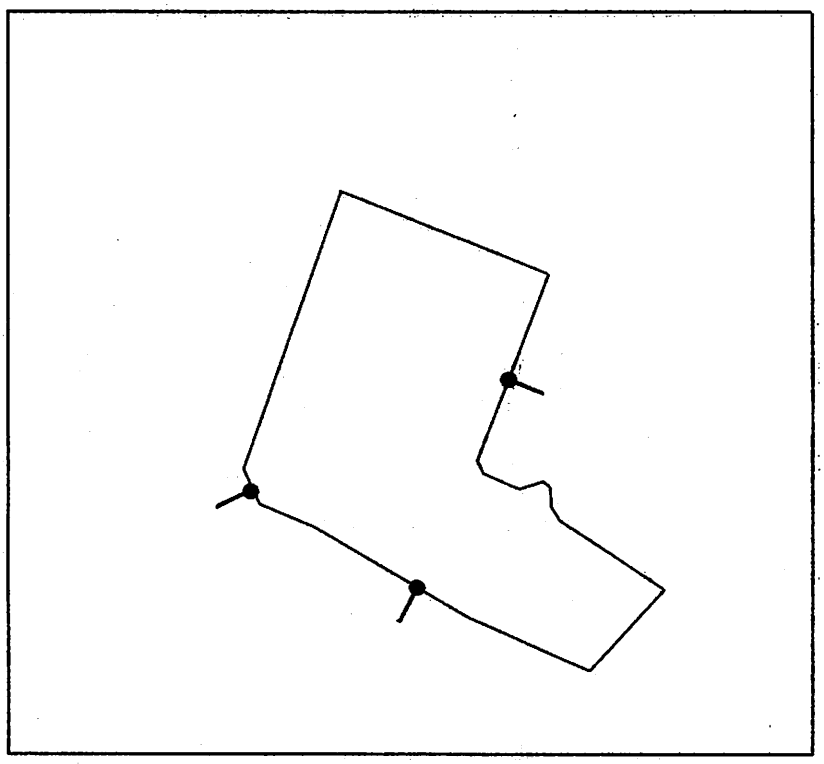

Figure 4.16: Final state for the right drill casing, low uncertainty.

4.4 Summary

We have tested our system for intelligent acquisition of sparse tactile data both in simulation and in actual hardware experiments. The simulation results were in accordance with our combinatorial predictions, and the hardware experiments were in accordance with the simulation predictions.

These results indicate that for moderately complicated situations, contemporary computers can plan paths for tactile sensors in an efficient manner. The planning time, when the ambiguity is less than about five possibilities, is on the order of the time to gather the data or to perform the object recognition phase.

There are two obvious shortcomings of the approach. The first is a combinatorial breakdown when there are large databases or when the objects have many faces; this might be addressed just by randomly gathering more data, since [Grimson,1984] indicates that this is likely to result in a reduction to a reasonable level of ambiguity with just a few probes. The other shortcoming is that this method works only for planar objects; the next chapter indicates ways in which paths may be planned for the intelligent recognition of polyhedral objects.

Chapter 5

Finding Sensing Paths in Three Dimensions

"That function is smathered in surmise"

We have described an approach to intelligent acquisition of tactile data in order to accomplish model-based object recognition of planar objects. The methodology we have developed, using a projection space to reduce search complexity, can also be used to analyse the problem of planning sensing paths in three dimensions. The overall approach would be the same: select sets of faces to pierce; find the parameters of a feasible path; verify that the path meets the distinguishing criteria; and execute the path, re-interpreting the acquired data.

The three-dimensional path-planning problem, however, is far more difficult to solve computationally than is the two-dimensional problem. The principal reason is that the non-linearities of projection, which we were able to circumvent in two dimensions, cannot be eliminated in a three-dimensional projection. We will now describe analytically the three-dimensional problem, and examine three general approaches to its solution: a direct approach of optimization of the path parameters; heuristic linearization; and heuristic decomposition of the problem into simpler forms.

5.1 Analytic Description

In three dimensions, a line (which is how a sensing path is represented in our paradigm) has four degrees of freedom. Two of these describe its orientation, the third orientational parameter being superfluous because a line exhibits infinite rotational symmetry about its direction; and two describe its position on a plane

perpendicular to the line, the third positional parameter being eliminated by a line's infinite translational symmetry along its direction.

Following our paradigm, we seek the representation of all of the lines passing through a face – which is now a polygon in real three-space – in a four-dimensional manifold whose points directly encode the lines' parameters. This is very complicated, because the projection of a polygon is a hypervolume bounded by curved hypersurfaces, and to find the intersection of an arbitrary number of such hypervolumes is, to say the least, non-trivial.

The fundamental reason for the increase in complexity is that the boundary of a polygonal face in three-space is a line, whereas the boundary of an edge face in two-space is a point. In both cases, the projection into the position–direction space is non-linear, but the boundary conditions in two-space permit the problem to be simplified by considering only the endpoints of a line segment. In three-space, the non-linear projection cannot be reduced in such a manner.

The nature of this complexity increase can be illustrated by considering an intermediate, three-degree-of-freedom problem in **X-Y-Z** space. Consider a three-space line segment $L(\lambda)$ that we wish to contact when starting from the **X-Y** plane; such a line is shown in Figure 5.1. The parameters defining the sensing path in this construction are the two coordinates of the starting point on the **X-Y** plane, the slope of the path from the **Z** axis in the **X** direction (which we will call α), and the slope of the path from the **Z** axis in the **Y** direction (which we will call β).

In this simple problem, let us hold one of the angular parameters constant, say, β. We thus wish to find the position of the starting point on the **X-Y** plane, and the inclination of the path in the **X** direction. This can be done by projecting the line $L(\lambda)$ into the three-dimensional manifold **X-Y-A**.

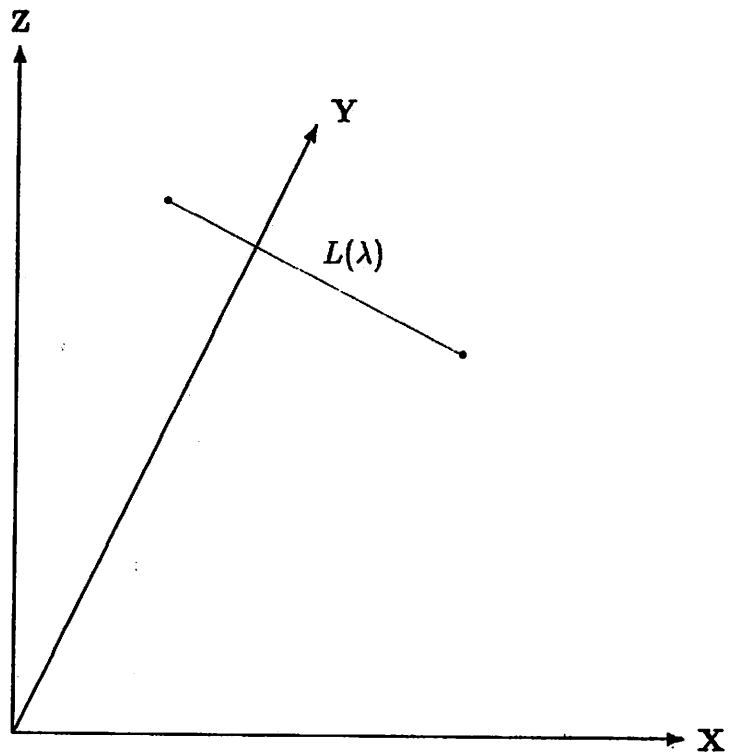

Figure 5.1: An edge in X-Y-Z space.

A line $L(\lambda) = P + \lambda \cdot D$ is projected into the **X-Y-A** manifold by the formula

$$\begin{aligned} Q_L(\lambda, \alpha) &= (L_X(\lambda) - \alpha \cdot L_Z(\lambda),\ L_Y - \beta \cdot L_Z(\lambda),\ \alpha) \qquad (5.1) \\ &= (P_X + \lambda \cdot D_X - \alpha \cdot P_Z - \lambda \cdot \alpha \cdot D_Z,\ P_Y + \lambda \cdot D_Y - \beta \cdot P_Z - \lambda \cdot \beta \cdot D_Z,\ \alpha) \end{aligned}$$

where as before the subscripts indicate the components in the three-space Cartesian coordinates. Observe that β is a fixed value, and is not a parameter of the projection.

If $D_Z \neq 0$, i.e., if the line is not parallel to the **X-Y** plane, then the locus in **X-Y-A** described by this projection is a ruled quadric surface, and so is non-planar. For any two values of α, the projection describes a pair of skew lines, and the ruled surface is the surface given by smoothly rotating one line into another according to the above formula. The coordinates of any point on this surface give the values of the parameters of a sensing path that starts from the **X-Y** plane, and intersects the line $L(\lambda)$. An illustration of the projection of an edge into **X-Y-A** space, for several values of α, is given in Figure 5.2. Each of these segments lies on a surface that twists as α increases.

So, even when we simply seek to intersect a line when holding one of the path directions constant, the problem of finding the path parameters is non-linear. This difficulty is compounded when we wish to pierce a polygon in **X-Y-Z** space, because a polygon is bounded by a set of line segments. Thus, the space that describes the locus of paths that pierces such a polygon is a three-dimensional volume in **X-Y-A** space that is bounded by ruled quadric surfaces. Piercing a candidate set of polygons requires finding the intersection of a number of such volumes, which is computationally very expensive.

In the full four-degree-of-freedom problem, a point projects onto a hyperplane, and a line projects under the two direction parameters α and β onto a doubly-ruled hypersurface. The formulae for these surfaces in the four-dimensional manifold **X-Y-A-B** are

$$Q_P(\alpha, \beta) = (P_X - \alpha \cdot P_Z,\ P_Y - \beta \cdot P_Z,\ \alpha, \beta)$$

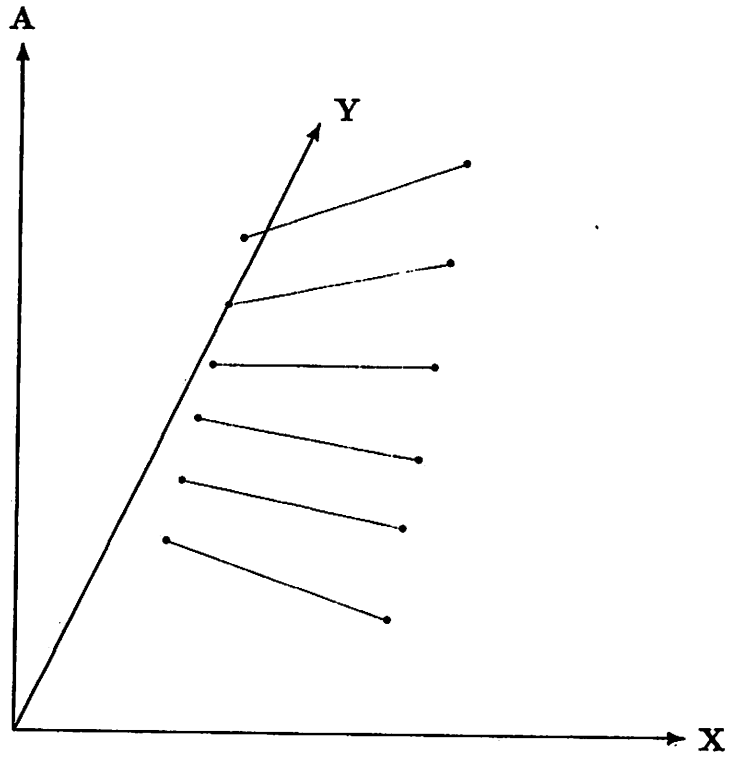

Figure 5.2: An edge projected into X-Y-A space, for several α.

and

$$\begin{aligned} Q_L(\lambda,\alpha,\beta) &= (L_X(\lambda) - \alpha\cdot L_Z(\lambda),\ L_Y - \beta\cdot L_Z(\lambda),\ \alpha,\beta) \qquad (5.2)\\ &= (P_X + \lambda\cdot D_X - \alpha\cdot P_Z - \lambda\cdot\alpha\cdot D_Z, P_Y + \lambda\cdot D_Y - \beta\cdot P_Z - \lambda\cdot\beta\cdot D_Z,\\ &\quad \alpha,\beta) \end{aligned}$$

where as before the subscripts indicate the components in the three-space Cartesian coordinates. A polygon projects to the hypervolume bounded by the ruled hypersurfaces formed by the projection of the polygon's bounding line segments into the four-dimensional manifold. The path parameters of all of the lines that pierces a set of polygons are the coordinates of the points that are within the hypervolume that is formed from the intersection of the hypervolumes of projection of all of the original three-space polygons.

Clearly, this intersection hypervolume is in general very complicated, and a full description of its boundary in terms of implicit parametric equations is not a computationally feasible approach to planning a sensing path. Remembering, however, that what we seek is a *single point* in the interior of this four-dimensional structure, there are several possible attacks we might mount. Also, the solution becomes simpler if we seek not always a perfect path or even an optimal path, but are content with just a good path, one that reduces the ambiguity considerably but can be found with reasonable computational expense.

5.2 Optimization

One approach to solving the problem of piercing a set of polygonal faces is to construct the projections of the faces, and then try to apply an optimization procedure over the projection parameters to find the point in projection space that provides the "best" path through the faces. Such an approach is possible, but it is also very costly and is not likely to be practical. We will examine here one possible construction of such a procedure, to illuminate some of the difficulties of a direct approach.

Let us begin by considering how to formulate the optimization for a set of polygons on the **X-Y** plane, disregarding for now the problem of projection. We will also make an important simplifying assumption: *all of the polygons are convex*. The optimization is thus a matter of finding an **(X,Y)** pair that is in the interior of as many polygons as possible.

For purposes of representation and calculation, we will adopt the convention that the normal of a line segment points towards the interior of the polygon; thus, if a line segment is given by

$$L(\lambda) = P + \lambda \cdot D$$

then the unit normal will be

$$\hat{N} = \frac{(-D_Y, D_X)}{\|D\|}$$

This convention is consistent with our specification of a polygon as a set of vertices that are sequenced counter-clockwise about the polygon.

For any point $R = (R_X, R_Y)$, the signed distance from R to a line $L(\lambda)$ is the dot product of the line normal with the difference between R and any point on $L(\lambda)$; for convenience, let us choose the point on the line to be P. Thus, the distance can be expressed as

$$d(R, L) = (R - P) \cdot \hat{N}$$

where $\hat{N}$ is that given above.

We observe now that R is in the interior of a convex polygon if and only if the distance from R to *all* of the bounding lines is non-negative, as shown in Figure 5.3. Hence, R is in the interior of a set of polygons if and only if the distance from R to all of the bounding lines of all of the polygons is non-negative. This implies that we could find some optimal position R_{opt} by attempting to maximize the sum of the distances to all of the lines. That is,

$$R_{opt} = \max_R \left(\sum_i d(R, L_i) \right)$$

subject to the constraints

$$d(R, L_i) \geq 0$$

defines the "best" point for the set of polygonal boundaries $\{L_i\}$.

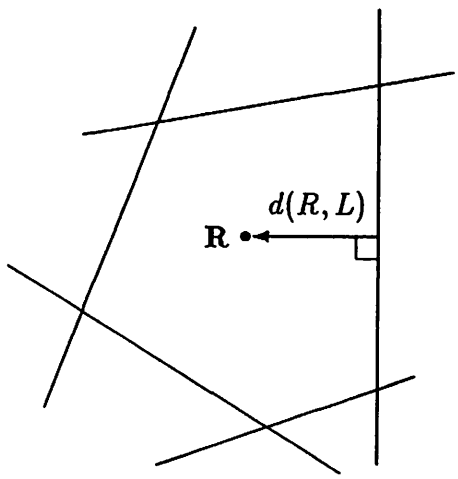

Figure 5.3: A convex polygon defined by constraint lines.

Phrased in this manner, R_{opt} can be found by a standard optimization procedure such as gradient descent, or can be solved by linear programming. Gradient descent, although slower in general, has the advantage of finding a solution when the constrains are inconsistent, which occurs when the polygons cannot all be pierced by a sensing line. More elaborate, but more expensive, procedures may also be used; see [Duda & Hart,1973] for a discussion of a variety of optimization methods.

This construction can be extended to provide the framework for the more general case of a number of polygons that are to be pierced from one of a variety of projections. A line $L(\lambda)$ projects into the **X-Y-A-B** manifold according to Equation 5.2. Elementary calculus shows that the direction from a point R to the nearest point on an unbounded parametric surface (with no singularities)

is parallel to the normal (or gradient); since the normal at any point on the projection of a line is in the direction

$$N = (\beta \cdot D_Z - D_Y\ ,\ D_X - \alpha \cdot D_Z\ ,\ 0\ ,\ 0)$$

we can construct the distance to the boundary hypersurfaces in projection space as the component, in the direction of the normal, of the difference between the point R and a point on the line $L(\lambda)$ under the projection:

$$d(R, Q_L(\alpha\ ,\ \beta)) = (X - Q_P(\alpha\ ,\ \beta)) \cdot \hat{N}$$

Thus, the optimal path has its parameters specified as

$$R_{opt} = \max_R \left(\sum_i d(R, Q_i) \right)$$

subject to the constraints

$$\begin{array}{rcl} d(R, L_i) & \geq & 0 \\ \alpha_{min} \leq \alpha \leq \alpha_{max} & & \\ \beta_{min} \leq \beta \leq \beta_{max} & & \end{array}$$

where R and R_{opt} have the four parameters $(R_X, R_Y, R_\alpha, r_\beta)$ which respectively encode the starting positions, X and Y, and the two projection values, α in the X direction and β in the Y direction.

This optimization must occur over four free variables, simultaneously, subject to a large number of linear constraints; while mathematically tractable - a convergent algorithm can be employed – it can be expected in general to be scot-texpensive, because there is one constraint for every polygonal edge boundary in the candidate set. Two further difficulties with this formulation are that it is expressed for convex faces, and that it assumes that the positions of the faces are known exactly.

Non-convexity can be managed in two ways. In the optimal but expensive formulation, a non-convex face can be represented as the union of a set of convex

faces; what results from this decomposition is that we must search for a path in disjoint sets of faces that are formed from the Cartesian product of the convex faces derived from each original face. Thus, if there are n_i convex elements for face F_i and there are j faces, we must:

- Form $M = \prod_{i=1}^{j}(n_i)$ sets;
- Try to optimize the four parameters for each of these M sets of convex faces;
- Evaluate these M paths, and choose the best as the distinguishing path for this candidate set of faces.

Clearly, this can be a very costly operation.

Another alternative to the non-convexity of a face is simply to form the convex hull of the face – preferably off-line – and use the hull as the face from which the constraint hypersurfaces are formed. Although not guaranteed to produce a piercing path, this is far less expensive than the formal decomposition of a face.

Uncertainty in the exact position of a face is more difficult to manage. One approach is to estimate the guaranteed volume in which part of the face must lie, approximate this volume with a polygonal slice, and employ the slice in forming the optimization constraints. Another, less exact alternative would be to seek paths that are as "central" to the faces as possible, and hope that the uncertainty is sufficiently small that the path is still adequate. Neither of these is a particularly palatable solution, and they reflect the difficulty of trying to formulate an optimization in the presence of geometric uncertainty.

We may summarize the optimization approach to path planning as computationally tractable but expensive, and one in which the uncertainty in the position of the object faces deeply affects the efficacy of the optimization.

5.3 Linearization

Another approach to efficiently finding sensing paths for multiple interpretations of three-dimensional objects is to try to eliminate the nonlinearities of the projection equation of a line segment. If the projection function can be made linear, then linear programming or some similar rapid search procedure might be applied to the problem.

As we see from Equation 5.2, the nonlinearities arise from the product of the projection direction, the line parameter, and the component of the line direction in the **Z** direction. It is not possible to eliminate either of the first two multipliers, but we might try to set D_Z to zero. Geometrically, this is equivalent to approximating a tilted polygon by one parallel to the **X-Y** plane at some arbitrary height. A reasonable choice of height for the approximating polygon is the mean height of the original polygon.

The approximation of one polygon by another introduces error. We will now derive the error in approximating a line segment by its linearization, and show how this value may be used in estimating the error of approximation for a tilted polygon. This latter error may then be used to measure the probable efficacy of linearization of the problem.

5.3.1 Error in Approximation of a Line

We will approximate a general parametric line $L = P + \lambda D$ by a corresponding nontilted line

$$L^* = P^* + \lambda D^*$$

where

$$\begin{aligned} P^* &= (P_X, P_Y, P) \\ D^* &= (D_X, D_Y, 0) \end{aligned}$$

with $\overline{P}$ chosen freely.

We now seek the difference of these lines under projection in the orthogonal directions of α and β. Any given value of λ determines a point Q_L in three-space that lies on $L(\lambda)$, and another point Q^* that lies on $L^*(\lambda)$. The difference between these points, projected onto the **X-Y** plane, is

$$\begin{aligned}\Delta Q &\stackrel{\text{def}}{=} Q - Q^* \\ &= ([L_X(\lambda) - \alpha L_Z(\lambda)] - [L_X^*(\lambda) - \alpha L_Z^*(\lambda)], \\ &\quad [L_Y(\lambda) - \beta L_Z(\lambda)] - [L_Y^*(\lambda) - \beta L_Z^*(\lambda)])\end{aligned}$$

By virtue of the definitions of the points and lines, elementary algebra gives

$$\Delta Q = \left(-\alpha[(P_Z - \overline{P}) + \lambda D_Z], -\beta[(P_Z - \overline{P}) + \lambda D_Z]\right)$$

and if we define as shorthand

$$\begin{aligned}\Delta P &\stackrel{\text{def}}{=} |P_Z - \overline{P}| \\ \Delta D &\stackrel{\text{def}}{=} |D_Z|\end{aligned}$$

we can conclude that

$$\begin{aligned}\|\Delta Q\|^2 &= \Delta Q \cdot \Delta Q \\ &= (\alpha^2 + \beta^2)(\Delta P + \lambda\, \Delta D)^2\end{aligned}$$

and so

$$\|\Delta Q\| = \sqrt{\alpha^2 + \beta^2}\,(\Delta P + \lambda\, \Delta D)$$

Under any given projection $\{\alpha, \beta\}$, the "error" between the line segments' projections, as λ varies from 0 at one segment endpoint to 1 at the other, is

$$\begin{aligned}E_L &= \int_0^1 \|\Delta Q\| d\lambda \\ &= \sqrt{\alpha^2 + \beta^2} \int_0^1 (\Delta P + \lambda \Delta D) d\lambda\end{aligned}$$

$$= \sqrt{\alpha^2+\beta^2}\left[\Delta P\lambda + \frac{\Delta D}{2}\lambda^2\right]_0^1$$

$$= \sqrt{\alpha^2+\beta^2}\left(\Delta P + \frac{\Delta D}{2}\right)$$

The total error in the projection is found by integrating this value of E_L over the entire range of acceptable $\{\alpha,\beta\}$ values.

Up to this point, we have represented projections not as angles but as tangents, to better show (especially in 2–D) the linear nature of the projection-space representation of a point. Continuing in this vein, we can represent the [non-unit] direction vector of a projection as

$$V_D \stackrel{\text{def}}{=} (\alpha,\beta,1)$$

If we seek path directions which are within some angle Ω of the normal of the generating face, i.e., within Ω of the **Z** axis, we can derive the constraint

$$\frac{V_D\cdot\vec{e}_Z}{\|V_D\|} \geq \cos\Omega$$

$$\Rightarrow \frac{1}{\sqrt{\alpha^2+\beta^2+1}} \geq \cos\Omega$$

$$\Rightarrow \frac{1}{\cos^2\Omega} \geq \alpha^2+\beta^2+1$$

$$\Rightarrow \tan^2\Omega \geq \alpha^2+\beta^2$$

so if $\rho = \tan\Omega$, we wish to integrate over the set

$$\{(\alpha,\beta)|\alpha^2+\beta^2 \leq \rho^2\}$$

which is a circle of radius ρ centred at the origin.

The total error in approximating a tilted line by a horizontal line is thus

$$E = \iint_{\alpha^2+\beta^2\leq\rho^2} E_L\, d\alpha\, d\beta$$

$$= \left(\Delta P + \frac{\Delta D}{2}\right)\iint_{\alpha^2+\beta^2\leq\rho^2}\sqrt{\alpha^2+\beta^2}\, d\alpha\, d\beta$$

This is not an easy integral to solve by direct attack. However, it has a simple geometric interpretation: the quantity under the square root represents the surface of a cone, which by the integral limits has its vertex at the origin and a maximum height of ρ. The integral is thus the difference in volume between

- a CYLINDER of height and radius ρ, and
- a CONE of height and radius ρ.

This volume is

$$(\pi\rho^2)\rho - 1/3(\pi\rho^2)\rho = \frac{2\pi\rho^3}{3}$$

and so

$$E = \left(\Delta P + \frac{\Delta D}{2}\right)\frac{2\pi\rho^3}{3} \tag{5.3}$$

is the total error in the linearization approximation of a tilted line segment by a segment parallel to the X-Y plane.

5.3.2 Error in Approximation of a Polygon

We can express the error of approximating a tilted polygon by an untilted polygon as the sum of the errors of the corresponding bounded line segments. Depending on the polygon, the approximation may in some places produce a projected polygon on the X-Y plane that is outside the projection of the original, and in some places is inside; what we seek is the total error, regardless of its sign.

Suppose that a polygon of interest has N vertices, and thus N bounding segments, and we wish to approximate it at some height $\bar{P}$, e.g., at the mean value of the height of the original polygon. Observing that $D_i = P_{i+1} - P_i$, and for convenience defining $P_{N+1} = P_1$, we can then represent the total error of linearization as

$$E_P = \sum_{i=1}^{N} E_i$$

$$= \frac{2\pi\rho^3}{3} \sum_{i=1}^{N} \left(\Delta P_i + \frac{\Delta D_i}{2} \right)$$

$$= \frac{2\pi\rho^3}{3} \sum_{i=1}^{N} \left(|P_i - \overline{P}| + \frac{|P_{i+1} - P_i|}{2} \right)$$

$$= \frac{2\pi\rho^3}{3} \left(\sum_{i=1}^{N} |P_i - \overline{P}| + \frac{1}{2} \sum_{i=1}^{N} |P_{i+1} - P_i| \right)$$

We will now decompose this expression, examine best-case and worst-case distributions of vertices, and form bounds on E_P. The best-case performance occurs when the original polygon is parallel to the **X-Y** plane; in this orientation $P_i = P$ for all i, and so the total error of approximation is 0.

The worst case for the approximation is when half of the vertices are at some **Z** value P_{min} and half at P_{max}, which represent the minimum and maximum heights above the **X-Y** plane of the polygon being approximated. When this occurs,

$$\sum_{i=1}^{N} |P_i - \overline{P}| = \frac{N}{2} \left(\overline{P} - P_{min} \right) + \frac{N}{2} \left(\overline{P} - P_{min} \right)$$

$$= \frac{N}{2} \left(\frac{P_{max} - P_{min}}{2} \right) + \frac{N}{2} \left(\frac{P_{max} - P_{min}}{2} \right)$$

$$= \frac{N}{2} \left(P_{max} - P_{min} \right)$$

and so we have the bounds

$$0 \leq \sum_{i=1}^{N} |P_i - \overline{P}| \leq \frac{N}{2} (P_{max} - P_{min})$$

For the second half of the expression for E_P, we see that the worst arrangement of the vertices is when all of the odd ones are at one extreme and all of the even ones are at the other; when this occurs,

$$\sum_{i=1}^{N} |P_{i+1} - P_i| = N(P_{max} - P_{min})$$

and so

$$0 \leq \sum_{i=1}^{N} |P_{i+1} - P_i| \leq N(P_{max} - P_{min})$$

Combining these bounds, we have

$$0 \leq E_P \leq \frac{2\pi N \rho^3}{3} (P_{max} - P_{min})$$

Since P_{max} and P_{min} are the vertices of a polygon on a plane, we can express the approximation error in terms of the inclination angle and the maximum vertex-vertex distance W of the polygon. A simple construction, shown in Figure 5.4, shows that

$$\begin{aligned} \sin\theta &= \frac{P_{max} - P_{min}}{W} \\ \Rightarrow W \cos\theta &= P_{max} - P_{min} \end{aligned}$$

From the constraint on the angle between a path and a face to pierced, we know that $\tan\theta \leq \rho$, and so the absolute outer bounds of the error in approximating a polygon by its untilted mean projection is

$$0 \leq E_P \leq \frac{2\pi NW \tan^3\Omega \sin\Omega}{3}$$

although in general the approximation error will be much less than the worst-case upper bound.

Finally, we can give a more intuitive measure of this error hypervolume. Consider an *untilted* polygon, which we desire to approximate by a circumscribed polygon, e.g., by its bounding rectangle. The error hypervolume E_R of this approximation is just the differential area ΔA integrated over the bounds of the projection, so

$$\begin{aligned} E_R &= \iint_{\alpha^2+\beta^2 \leq \rho^2} \Delta A \, d\alpha \, d\beta \qquad (5.4) \\ &= \pi\rho^2 \Delta A \\ &= \pi \Delta A \tan^2\Omega \end{aligned}$$

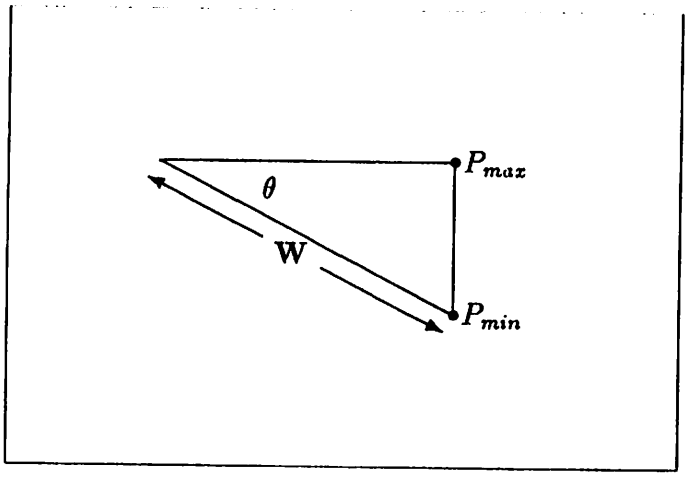

Figure 5.4: Bounds on a tilted polygon.

Setting $E_R = E_P$ to determine when a tilted approximation produces the same error as a rectangular approximation yields, after suitable algebra,

$$\tan\Omega \sin\Omega = \frac{3\Delta A}{2NW}$$

If we use $W = 1$ as a scale factor, and $\Delta A = W^2/2$ for a rectangle that is twice the size of the polygon it approximates, we can express the tilt angle as a function of the number of vertices of the polygon. For example, if $N = 9$, then in the very worst case we observe that $\Omega \approx 16.5°$, i.e., a zig-zag polygon can only be tilted up to 16.5° before it produces the same error, for a 3–D path-planning algorithm, as a polygon that was approximated by a rectangle twice its size. For convex polygons the error is much lower, and for a given desired error we can tilt the polygon much more, which increases the likelihood that a successful path through a candidate set will be found.

Although we cannot fully evaluate the effectiveness of linearization without an implementation, we may surmise from the error bounds that it is likely to be effective only for small values of Ω. Our experience with planning sensing paths in two dimensions indicates that values of Ω up to 60° are often needed; if this

experience is at all applicable to planning in three dimensions, linearization is not likely to prve an effective approach.

5.4 Decomposition

The three-dimensional sensing-path problem can, as we have seen, be attacked by a costly direct optimization or by a linearizing approximation. If we are willing to trade optimal information-gathering power for rapidity of calculation, another attack can be mounted: we might try to decompose the four-degree-of-freedom problem into simpler linear problems.

5.4.1 A Taxonomy of Decompositions

A simple, exhaustive taxonomy of decompositions can easily be formed by considering all of the possible subsets of the four path parameters. Because the projections of lines and polygons are non-linear in three–space under even one projection direction, we can concentrate exclusively on decompositions into a pair of two-degree-of-freedom problems.

Of the four path parameters, two are directional – α and β, measured from the **Z** axis towards the **X** and **Y** axes respectively – and two are positional, representing the starting point on the **X**-**Y** plane of the line.[1]

Rather than attempting to optimize all four parameters simultaneously, we may instead optimize a *pair* of parameters, and then on that basis optimize the other pair. In such a paradigm there are only a few combinations of pairs; we will represent the decomposition as $(a, b) \times (c, d)$ to denote that we first optimize (a, b) and then optimize (c, d). In addition to the four parameters, we will also note that there is a variant representation of a line segment as a pair of endpoints,

[1] The reader is reminded that the coordinate system is located and oriented for convenience of calculation.

i.e., as $\{P, P'\}$. We can now categorize a decomposition as belonging to one of the following classes:

1. $(X,\beta) \times (Y,\alpha)$ or $(Y,\alpha) \times (X,\beta)$ or
   $(X,Y') \times (Y,X')$ or $(Y,X') \times (X,Y')$.
2. $(X,Y) \times (\alpha,\beta)$.
3. $(\alpha,\beta) \times (X,Y)$.
4. $(X',Y') \times (X,Y)$ or $(X,Y) \times (X',Y')$.
5. $(X,\alpha) \times (Y,\beta)$ or $(Y,\beta) \times (X,\alpha)$ or
   $(X,X') \times (Y,Y')$ or $(Y,Y') \times (X,X')$.

This can be seen to exhaust the combinations of parameter pairs.

Before examining these decompositions further, it may be instructive to give some cursory comments about each. The decomposition classes may be characterized thusly:

1. Attempt to find, e.g., the **Y** direction and the **X** position simultaneously, and then find the remaining two parameters. This is a poor breakdown of the problem, in that there is no interaction between the parameters in a pair.
2. Attempt to find a good location, and optimize the path direction. This is one of two intepretations of the Grimson approach described below.
3. Attempt to find a good direction, and optimize the starting position. This is the other interpretation of the Grimson approach.
4. Attempt to find a starting point and a point to aim at. This is discussed below as the Schneiter approach.
5. Attempt to find the **X** parameters and then the **Y** parameters, or *vice versa*. This takes advantage of the orthogonality of the projection equations, and will be examined below as the **decoupling** approach.

We can see that there are relatively few ways of attacking the decomposition of the four-degree-of-freedom problem into simpler problems. Of the five general ways, one is a poor approach, two may be subsumed under Grimson's methodology, one under Schneiter's methodology, and one has not been discussed to our knowledge. We will now describe these decompositions in turn, and briefly examine their salient characteristics.

5.4.2 The Grimson Approach

One approach, reported in detail in [Grimson,1986], seeks regions in range images that will provide maximal distinguishing information. We will now attempt to show that this is equivalent to one of two decompositions of the general problem, the difference deriving from the assumptions one makes about the sensor.

Grimson's approach is to select a sensing direction, choose a starting plane normal to the direction and outside of all interpretations, and in effect employ an orthographic rendering of all poses onto this sensing plane. By discretizing the starting plane, it is possible to use computer graphics techniques to render the distance and local surface normal of the projected region, and efficiently search the buckets for regions exhibiting the desired distinguishability and resistance to pose uncertainty.

As stated, this approach subsumes the decomposition in which the direction is selected, followed by selection of the starting position. We note, however, that if one were to use a perspective transformation onto an imaging plane, then application of this framework would amount to selection first of the position of the focal point of the transformation, followed by selection of the direction of gaze that would best provide distinguishing data.

Grimson has implemented his approach, and it seems computationally tractable although timings are not given. The search procedure he describes is a divide-and-conquer algorithm, which rapidly focusses attention on regions of interest; for complicated interpretation sets, it is likely that the graphics rendering

will dominate computationally. A further advantage of his approach is careful inclusion of uncertainty analysis into the computation, resulting in a very robust sensing technique.

The principal drawback of Grimson's approach is that he provides no suggestion as to how to choose the sensing direction in the first place. In his paradigm, he is provided with a small and fixed set of directions, corresponding to fixed locations of range sensors in the workspace; although one might be able to *view* the entire workspace with them, it by no means follows that one can *optimally sense* from the given sensor arrangement. Optimally selecting view directions independent of selection of positional parameters remains an open question.

5.4.3 The Schneiter Approach

Another approach, described in [Schneiter,1986], is to compute simultaneously a start point and an end point of a path. This computation is based on some observations on necessary conditions for the endpoints, combined with path characteristics such as its distinguishing power, that permits a reasonably efficient 2–D search to be implemented.

Schneiter's approach derives from the observation that for closed polygonal objects, the boundary of the area of union of the overlapping polygons and the boundary of the area of intersection are both well defined; hence, one may use these boundaries as the sets of possible path start points and end points, respectively. From this result, some straightforward but powerful existence theorems for paths may be derived.

The search for paths is guided by the principle that, whenever it is feasible to do so, a path should not pass through a point that could be validly assigned to a face from more than one interpretation. These regions of multiple overlap – and hence multiple assignment – he calls *blocking boundaries*, because they act to block the search procedure. What is sought, then, are paths with a start point

on the union boundary, an endpoint on the intersection boundary, and such that they pass through as few blocking boundaries as possible.

Schneiter's system finds such paths by concentrating attention on the blocking boundary. If the blocking boundary, extended from a given point, is an *open* contour then the region immediately beyond the blocking boundary endpoint is a region that provides a channel for a path that is not blocked, i.e., for a path that has some significant distinguishing power. If, for example, there was a single blocking boundary that was not closed, then the region between the blocking boundary endpoints is taken to lie amid a sensing path, and on either side of this region there is the union boundary from which the path starts, and the intersection boundary at which the path ends.

By discretizing the plane, and filling the partition elements with the derived information, Schneiter's system can find optimal paths without the excessive computational cost entailed by a more analytical implementation. Management of error, in two dimensions, is conducted by determining those spatial regions in which a given face must lie, and forming blocking *regions* for these extended faces; the exterior of these blocking regions can then be used to guide the search for a sensing path.

Extension of this latter method to three dimensions can be done, but rapidly becomes very expensive. The blocking regions of an ambiguous set of interpretations of non-convex polyhedra, from uncertain sense data, are costly to calculate. Furthermore, the search exhibits combinatorial explosion as, at the minimum, a path is defined as having endpoints in the given 3-D volumes and must pass through one of several given 3-D volumes. Although well defined, it is not clear that this approach is computationally feasible.

In summary, Schneiter has provided several insights into the problem of path formation, and has presented a system capable of solving the problem on the plane. Extension to three dimensions is possible, but expensive.

5.4.4 Decoupling

The final approach in our taxonomy attempts heuristically to decouple the four–dimensional search into a pair of two–dimensional searches. This decoupling is an attempt to sacrifice guaranteed optimality for rapid computation of acceptable paths.

In decoupling, the path parameters are broken into orthogonal pairs. Without loss of generality, let us suppose that we attempt to find first the $\{X, \alpha\}$ pair and then the $\{Y, \beta\}$ pair. The first effect of this procedure is that we must approximate each face in a candidate set by some line segment on the **X-Z** plane, and use this linear approximation as an edge in our 2–D path planner.

Two alternatives are to be conservative and plan only guaranteed paths and potentially miss many good paths, or to be liberal and potentially plan poor paths. The advantage of the latter is, extrapolating from our experience with planar objects, a sub-optimal path often provides a good deal of information. Pursuing this route, we would approximate a polygon that is orthographically projected onto the **X-Z** plane as a line whose endpoints are chosen appropriately from the isothetic[2] bounding rectangle of the projected polygon, e.g., a line whose endpoints are polygonal vertices which have the minimum and maximum **X** values. Figure 5.5 shows a projected face, and a liberal approximation of it. When all faces of the candidate set have been approximated as edges, our 2–D algorithm is capable of piercing the edges so as to maximize the information gained.

We have a few observations of this approach. First, the decoupling *is* a heuristic, and can be expected to fail sometimes. Roughly speaking, we can estimate the probability of a path, derived by decoupling, hitting a given face as the integrated difference of areas, as was derived in Equation 5.4. This effect can be mollified somewhat by requiring that all faces be convex, and by selecting paths that contact the central regions of faces. The result of such corrective factors will

[2]Aligned with the coordinate axes.

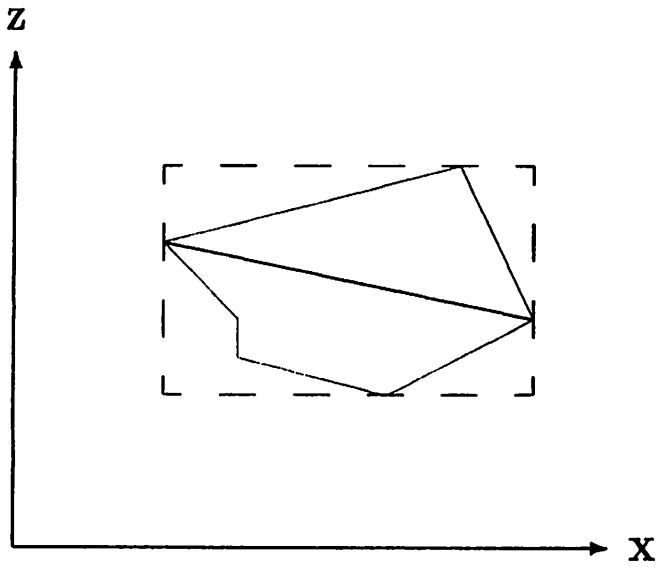

Figure 5.5: Approximation of a projected polygon by a line.

be an improved hit ratio.

The second observation is that the quality of approximation varies as the candidate set is rotated about the generator normal, i.e., as the local coordinate system rotates about the **Z** axis. Clearly, an arbitrary polygon will be better approximated by an isothetic rectangle when the polygon is in certain orientations; a good approach might thus be to rotate the coordinate system until the total error area of all approximated faces is minimized.

Finally, although this approach pretends that the path parameters can be decoupled, they are in fact interdependent. An accomodation might be to find the first pair of path parameters, "slice" the candidate set parallel to the **Y-Z** plane through the **X** parameter, project the 3–D edges onto the **X-Y** plane, and solve for the remaining path parameters. This accomodation accounts somewhat for the varied geometry of the candidate set, but is not commutative, i.e., the path found is dependent on the order in which the decomposition takes place. This contrasts with simple decoupling, where the path parameters are independent of

order.

Two advantages of decoupling are: (a) it is relatively quick, and the cost will be slightly more than twice the cost of solving the 2–D problem; and (b) uncertainty can be taken into account systematically, by finding the rotational uncertainty of each face with respect to the **Y** axis when finding the **X** parameters and *vice versa*. Decoupling is, none the less, a heuristic, and promising though it is, proper evaluation must wait until implementation.

5.5 Summary

We have presented an analysis of the problem of finding sensing paths in three dimensions, and five approaches to its solution. The most rigourous approach is a direct implementation of the analytical description by optimization of the parameters; this is likely far too expensive to be worth implementing in a real-time system, although it might provide some interesting insights into the frequency of easy *vs.* hard cases.

The second approach is a linearization of the projection equations, which can lead to the formulation of the problem of finding a sensing path as a problem in linear programming. This is a promising heuristic, especially as we can derive probabilistic estimates of the path quality as a function of the geometry of the polygonal faces of the polyhedral objects.

Two other approaches fall under our taxonomy of decomposition, the approaches being due to Grimson and Schneiter. The first does not attempt to optimize all of the parameters, but excellently incorporates uncertainty into the analysis; the second attempts a direct attack on the problem of optimizing the parameters, but does so at considerable computational expense.

The final approach, not yet implemented, attempts to decompose the problem into two simpler ones and then combine the results into a path that distinguishes the ambiguous interpretations. This decomposition should work in a large vari-

ety of circumstances, and a derived error bound could be used to predict those cases in which it is likely to fail. Implementation and comparison with the other approaches appears to be a worthwhile enterprise.

We should note that the approaches described, particularly our decomposition taxonomy, are not the only ones possible. These are the approaches that result from fairly direct analysis of the recognition paradigm in which we operate, but there are many other ways of recognizing and modelling objects. Other paradigms, no doubt, have approaches that could efficaciously solve the general problem we pose; they might also suggest methods that could be used within our paradigm. Finding ways to efficiently and effectively gather information to aid the recognition of three-dimensional objects is a hard problem, but one whose solutions would have considerable applicability in robotics.

Chapter 6

Conclusions

"Let us not be dainty of leave-taking"

This thesis has attempted to address the problem of how intelligently and efficiently to acquire sparse tactile data, in order to accomplish model-based object recognition. We used computational geometry, rather than feature-based or statistical methods, in order to provide a general and very effective method. The core of our approach is a representation of paths passing through a section of an object in a parameter space, allowing us to search this parameter space for a path that satisfies the design criteria.

6.1 Contributions

At the time of inception of this thesis, the state of the art was not very advanced. There were several published methods for recognition of objects from sparse tactile data, but only one, [Luo *et al.*,1984a], had been implemented; and at that time, there were no published methods for systematic acquisition of new data. Since then, independent of our efforts, Grimson has developed a geometric method and tested it on laser range data; Schneiter has developed a distinct geometric method and tested it on 2–D tactile data; and Allen has developed and implemented a contour-following approach that uses vision and models to guide tactile data acquisition.

This dissertation has contributed to the field in several ways, and in varying degrees of importance, in that we have:

1. Provided a framework for intelligent acquisition of new data.

2. Found tight bounds on the geometric uncertainty of separated-component model matching.

3. Produced a novel representation for sensing paths.

4. Implemented an efficient algorithm for finding these paths in two dimensions.

5. Developed a method for evaluating the path-finding algorithm.

6. Provided an analysis of the three-dimensional problem.

7. Produced a taxonomy of problem decompositions.

To elaborate on each of these points:

1. Our general framework is similar to Grimson's and Schneiter's, but differs in details. Like them, we observe that the geometry of the situation constrains sensing paths and that intelligent acquisition can be viewed as a search for these paths. We, however, have tended to emphasise the integration of the reasoning system with the hardware, and have tried to incorporate these various needs in a coherent manner.

2. Our formulation of the geometric uncertainty of separated-component model matching has produced very tight bounds on the pose of an interpretation. Because pose uncertainty is one of the most significant factors in path rejection, the tight bounds permit more rapid search and yet maintain the guarantee of distinguishability of contacts that we desire.

3. Our path representation is a novel, concise way of encoding the parameters of all of the linear paths that pass through a polygon in 3–space. This representation facilitates the search for a good path that distinguishes features from different interpretations of the given data.

4. Our implementation in two dimensions has shown that the method works on real data, and indicates that coherent integration of high-level software, low-level software, and hardware considerations can be done, and that the result is a robust and maintainable system.

5. Our representation of distinguishing power, the ambiguity tree, permits explicit evaluation of a set of paths and of the algorithm that produced this set. The tree also facilitates reasoning about probable efficiency and effectiveness of the algorithm when it is presented with a particular mix of models, potentially leading to automated determination of the design parameters to optimize the system's overall efficiency for that model set.

6. Our analysis of the three-dimensional problem indicates, explicitly, how hard the problem is and why it is so hard. Our examination of the problem structure suggests several solutions, the probable effectiveness and utility of which can be abstracted.

7. Finally, our taxonomy of decompositions shows how our path representation permits our approach to be integrated with Grimson's approach and Schneiter's approach into a hierarchy of simplifications of the general problem.

6.2 Critical Assessment

Although we have contributed to the theoretical and practical understanding of how to intelligently acquire sparse tactile data for model-based object recognition, our work is far from perfect. The imperfections range from nagging details to restrictions inherent in our approach.

Of the nagging details, perhaps the most worrisome is that we have been unsuccessful at producing a formal proof that if a path *can* be found, our method *will* find it. So far, the evidence is empirical: whenever it fails to find a path, the

interpretations are either almost indistinguishable (e.g., the rotational parameters are just a few degrees different), or in the presence of uncertainty humans also cannot produce a path. The principal reason for not being able to guarantee that paths will be found is the subtle interaction of path selection with contact positions and angles. A full analysis will probably reveal this to be a higher-dimensional non-linear problem, but we have not yet characterized it fully enough to warrant any firm conclusions on the matter.

Another nagging detail is that the system is often inefficient because of the primitive way in which candidate sets are formed and ordered. Path planning would be far more efficient if the sets were smaller, if distracting faces could be removed, and if a more intelligent method of sorting the candidate sets were employed. We have not had time to explore this matter, but it is the most obvious way in which our algorithm could be improved.

All systems break, and ours is no exception. Notably, it degrades gracefully as the number of interpretations increases, but because of the $O(N^3)$ nature of the search it eventually takes an intolerable amount of time to find paths. What is more insidious is the large difference between the mean planning time and the worst-case planning time; in some circumstances, the system will exhaustively search for that path that is "just a bit better" than one it has already found. Placing a limit on the computation time only partly addresses this problem, which is really one of high-level control of search-based systems.

Our paradigmatic assumptions have tended to restrict the purview of the thesis, and loosening them somewhat would result in a more general system. One such assumption is our model of sensor error as being absolutely bounded; this allows simple, consistent management of uncertainty, but there is a hidden cost. In order to succeed with real data, the error bounds must be very large to accomodate deviations that have low likelihood yet are still possible; but, large error bounds lead to large uncertainty, which leads to inaccuracy, inefficiency, and possibly to failure. A different model of error could be used, but the entire

approach would have to be modified.

Another subtle assumption is that the tactile sensor is much smaller than the object features. This implies that object faces, which are large and easy to hit, become the preferred targets for the sensor. If, however, the sensor is large – for instance, if we assumed that we had available a tactile array sensor the size of a typical robot gripper – then not only can vertices (which exhibit considerable excursion in the presence of uncertainty) become targets, but one can even add such different tactile features as local texture or deformability. This change, obviously, affects large parts of the system structure.

A more serious paradigmatic restriction is that we have assumed that a single known object is fixed in the workspace. Grimson and Lozano-Pérez have extended their recognition strategy to manage multiple objects; similarly, we could extend both the recognition phase and the path-planning phase to include more than one object. In the planning phase, though, considerable care must be given to ensure that a path intended to identify one object is not obstructed by another; this is difficult but not impossible, and the growing literature on path planning for mobile robots indicates that hybrid methods are both potentially very useful.

A more philosophical problem is what to do about unknown objects. When no interpretation of the data is consistent, should one doubt the data and continue to acquire randomly? When there is only one interpretation, should one methodically verify the position and orientation of each object feature to be *sure* that it is the right one? For the domain of automated assembly, the model-based assumption is legitimate; in a domain of naturally occurring objects, it is not. This assumption underlies much of modern robotics, and when we lose it we will be taking a large step towards truly intelligent behaviour.

The last part of the assumption, that the object is fixed, is also quite difficult to relax. Throughout, we have assumed that the world is static, but the real world is dynamic. Some generality can be gained by trying continuously to update the suspected pose of the object being manipulated, but the easiest short-term

solution is to maneuver the object into a stable grasp and hold it fixed. Trying to chase sliding objects around on a table, simultaneously batting them and trying to identify their type and location, is probably a no-win situation (amusing though it would be to watch).

Finally, we have paradigmatically assumed a certain representation of objects, namely as polyhedra. These have permitted us to take many computational shortcuts and derive useful bounding conditions in the search for path parameters, but at the expense of generality. In the large, our problem is to acquire data intelligently and coherently for object recognition; many of our principles can be simply modified to include extended, nonlinear surface patches, which greatly expands the applicability of the method.

6.3 Implications

The principal implications of our work for other areas of research are how our approach might be used with other sensors, and as a general comment on systems design.

6.3.1 Other Sensors

Although our system was produced for use with tactile sensors, it is applicable to other sensors, e.g., laser rangers. The required data are position and local surface normal; these can be extracted from a range image by approximating a patch by a plane. From such data, the recognition phase would produce interpretations.

To plan paths, the system requires that the sensor be mobile and that it require a [relatively narrow] cylindrical channel. Time-of-flight sensors satisfy the last requirement, and accurate positioning is a matter of careful engineering in the specific application. It is thus possible to use our approach with non-contact sensors, as well as with tactile sensors.

This is a straightforward application of our work. Another application might be to try to use it as a focus-of-attention strategy, to select potentially useful regions of a dense range image rather than try to process an enormous amount of raw data. By sampling the image, interpretations result; a method such as Grimson's or ours might then spot those image regions that contain the most crucial information.

We also note that the problem of a mobile robot trying to determine which room it is in, and where in the room it is, is the geometric dual of the recognition problem we address. If a mobile robot equipped with a range sensor (here, sonar might suffice) could extract the range and orientation of walls in the room, our method could tell it what the possibilities were and where next to move to gain the best view. This is a tantalizing prospect, but one which we have not yet had time to explore.

6.3.2 Design of Robotic Systems

Finally, in producing this system, we have had the opportunity to learn several valuable lessons in the design of systems for robotics. First and foremost is the necessity of integrating, as coherently as possible, the various constraints imposed by the nature of the domain.

In order to produce a coherent, integrated system that was efficient and effective, several design iterations were necessary. A good knowledge of tactile sensors allowed us to determine the salient characteristics of the domain, and thus to select a recognition strategy that would probably work. From the object representation and the way that contemporary robots can move contemporary sensors, we selected linear freeways as the path representation. This representation, and the kind of information we needed for recognition, guided the sensor design; and last, the higher-level goals and path types, along with the available hardware, dictated the control design. Working back, we could theoretically estimate the sensor errors, experimentally verify them, and propagate these error estimates

to the recognition system. The error estimates cycled through the uncertainty analysis and path planning so that the robot was instructed to move in a manner that ensured that only reasonable data were acquired.

In sensor-based robotics, one must know about the sensors, how data is extracted from them, how robot motion can be performed, how uncertainty and numerical stability affect calculations, and what the high-level goals are, in order to conduct the necessary trade-offs. Careful analysis, extensive simulation, and real-world experimentation seem to be the keys to development of robotic systems, given contemporary hardware and software tools.

Appendix

Experimental Hardware

"Bloody instructions, which being taught, return to plague the inventor"

The experiments we conducted on tactile recognition were performed in the Laboratory for Perceptual Robotics on a prototype Cartesian manipulator, using a force sensor specifically designed for our research.

The CART manipulator has four degrees of freedom, three being mutually perpendicular translational axes and one being a rotation about the vertical axis. Figure A.1 shows a picture of the manipulator and the gripper. For this research, however, we only moved the two axes that produce motion in the X-Y plane, the other two being left as stationary servos.

A.1 The Force Sensor

The force sensor we used is essentially an instrumented beam of square cross-section, with an ordinary radial ball bearing attached to the tip; see Figure A.2 for a photograph of the sensor. As the sensor beam deflects, strain gauges measure the force in two perpendicular directions. Assuming that there is a single point of contact, the direction of the force points from the beam centre to the contact point; and because the contact occurs on the outer race of a bearing of circular cross section, the local surface can be described as a planar segment tangent to a circle. Thus, we can estimate both the position of contact and the local surface normal from the two strain gauge readings. Figure A.3 shows the geometry of the contact at a slice through the bearing. The bearing is used to ensure that there are only radial forces at the point of contact, and that there are no tangential or shearing forces to corrupt the estimate of the net force direction.

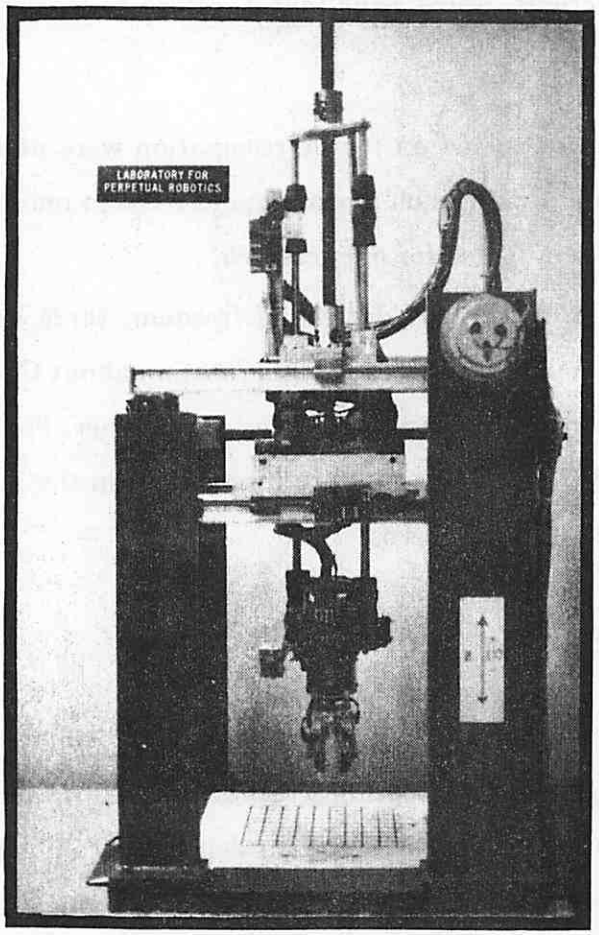

Figure A.1: The CART manipulator.

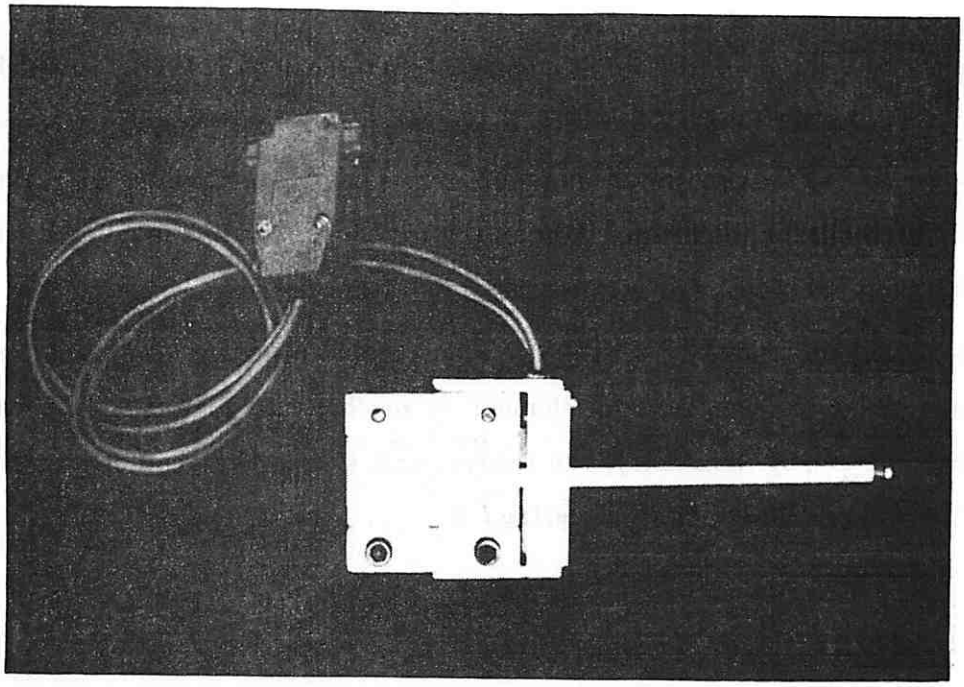

Figure A.2: The force sensor.

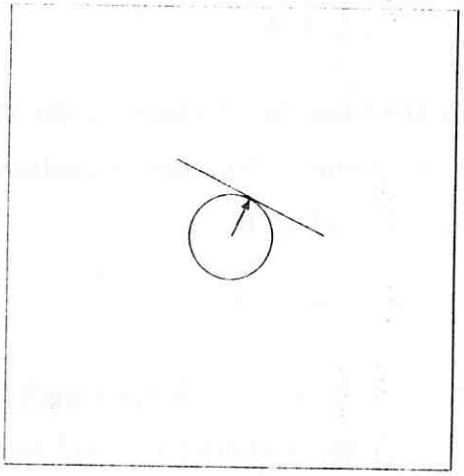

Figure A.3: Geometry of the method used for sensing tactile data.

The sensor was designed to yield at a nominal tip force of 22 N (5 lb), at which the gauge strain would be 0.15%, which is typical of foil-type gauges. Because typical gauges are approximately 4 mm wide, we chose a beam width of 5 mm to allow ample room for mounting. Accounting for the length of the gauges, we chose our sensing point to be approximately 6 mm from the beam base.

We can model the sensor as a cantilever constructed of extruded aluminum, and compute the beam length from elementary mechanics. Figure A.4 shows the cantilever beam; by symmetry, the neutral axis is also the geometric axis. Because it is square, the moment of inertia I is

$$I = \frac{h^4}{12}$$

and the bending moment at the estimated sensing point is

$$M = F \cdot (L - X)$$

The strain ρ is linearly related to the stress, by Hooke's law, because the beam is undergoing non-plastic deformation; it is just

$$\rho = S/E$$

where S is the stress and E is the modulus of elasticity for the beam medium (71,000MPa, or 10.3MPsi, for aluminum). The only remaining quantity to be specified is the stress, the formula for which is

$$S = M \cdot C/I$$

From these equations and the stated values, the beam length can be computed as 110 mm. Finally, the deflection of the tip under the peak load is computed as

$$D = \frac{F \cdot L^3}{3 \cdot E \cdot I}$$

which gives us an estimated tip deflection of about 2.5 mm.

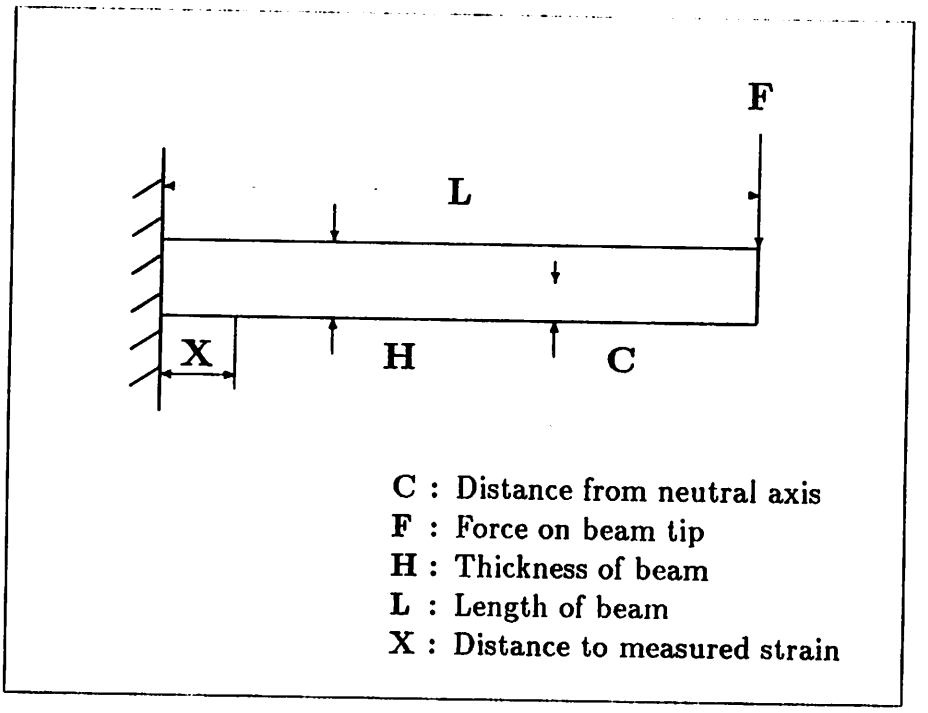

Figure A.4: Cantilever-beam model of the sensor.

In order to deduce the net force applied to the beam, it was necessary to amplify the very small resistance changes by using a Wheatstone bridge and an instrumentation amplifier. Because we were using a sensing bridge, we applied a gauge to each side of the beam and wired the opposite faces as adjacent resistors in the bridge, thus both doubling the sensitivity and providing passive temperature compensation. A schematic of the sensing circuit is shown in Figure A.5.

When the amplified signals were digitized, they were available for use. Because of small differences in actual gauge resistance, in machining of the beam, in placement and efficiency of the glue layer for each sensor, and in the gain provided by the amplifiers, the resulting signals differed in both their zero-force response and in their response to identical forces.

The zero-offset could be compensated by averaging a number of values at startup, and taking this value as the zero-response point. Because all of the previously mentioned inaccuracies are multiplicative, they form a net effective

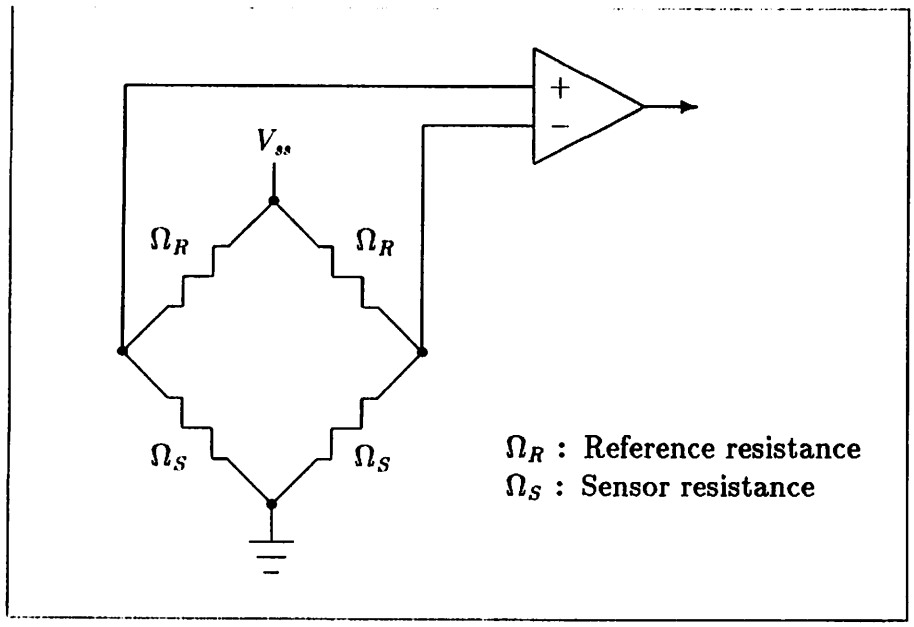

Figure A.5: Sensor circuit diagram.

gauge factor that can be measured for each gauge pair. We simply applied several identical forces to the gauge pairs by arranging the beam parallel to the ground plane and suspending weights from the contact bearing; the gauge responses could then be calibrated.

Two additional problems we faced were gauge/beam creep and shot noise. The former is a slow but noticeable decrease in a gauge's response to a constant applied force; it was very difficult to model, but if the sampling window was kept to less than a second or two it became very minor. However, creep and warming of the amplifiers and gauges resulted in a slow drift, which we compensated by applying a moving window average to near-zero response values and regularly updating the zero response. The latter problem, shot noise, was managed by averaging a few values. The net effect of all of these measures was a sensing system that could reliably measure the local surface normal of a truly planar contact within $\pm 0.5°$ of its actual value, at an applied force of about 9 N (2 lb). Allowing for passive compliant deflection of the sensor, the gripper, and the robot arm itself, the position of a truly planar contact could be deduced within ± 0.5mm

of true, which is principally a reflection of the error present in the deduction of the surface normal.

A.2 Control of the System

In use, the force sensor is secured in the gripper and attached to nearby instrumentation amplifiers. The amplified signals are then fed into a pair of analog-to-digital converters on the PDP 11-23 that controls the CART. In order to accomplish the high-level task of reliably acquiring tactile data from an object in the workspace, we developed a special-purpose control architecture for the robot system. There are four components to this architecture: the outer control loop, which monitors positions and forces; a position servo loop; a force/velocity loop; and at the lowest level, hard-wired velocity-servo control of the motors by means of pulse-width-modulation power amplifiers.

Because the lowest-level control loop determined in part the design of the others, we will describe it first. Briefly, it is composed of a power amplifier for each motor which accepts a reference velocity from the computer, compares it with the velocity derived from tachometer feedback, and varies the width of a constant-voltage, constant-frequency train of pulses. This architecture, shown in Figure A.6, provides a very stiff but highly reactive actuation system. Gains for the amplifiers were determined empirically to closely approximate critical damping.

The object-recognition/path-planning system places interesting demands on the control system. It is necessary to be able to move the sensor freely in the workspace, at high speeds, when the path is clear; to move the sensor guardedly in anticipation of contact, usually at low speeds, to improve control response; and finally to be able to move it snugly against a contacted surface in order to reliably acquire the position and local surface normal of the contact.

Our solution to this problem was a multi-level scheduling architecture. At the

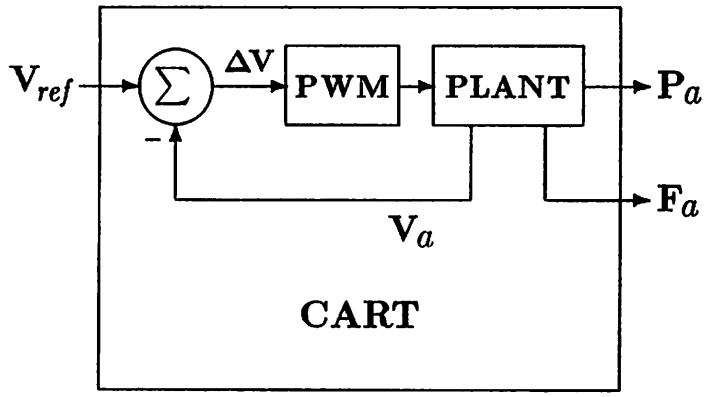

Figure A.6: Hard-wired control structure of the CART.

highest level, compensator scheduling is performed, as shown in Figure A.7. A guarding force is supplied to the loop, along with a desired position. If no contact occurs, the robot is position servoed. But if contact occurs, the control system switches to admittant control, in which a nominal velocity and a desired net force reading are combined to ensure firm contact.

In position control, a desired position is compared to the nominal position. As shown in Figure A.8, the magnitude of the difference determines which gain is selected. This permits issuance of velocity commands which ramp a joint up to a maximum speed, continue until the actual position is near the desired position, and then ramp the speed down so that the joint smoothly achieves its goal. In practice, the system calculates the gains from a maximum desired velocity and maximum acceleration.

When contact occurs and is to be controlled, the system switches compensators to perform *admittant* control, as shown in Figure A.9, in which velocity and force information combine to produce velocity commands.[1] A nominal velocity, usually just the path direction whose magnitude is adjusted by a gain, is combined with the amplified force feedback from the sensor. The force feedback

[1] We are indebted to Judy Franklin for suggesting this control structure.

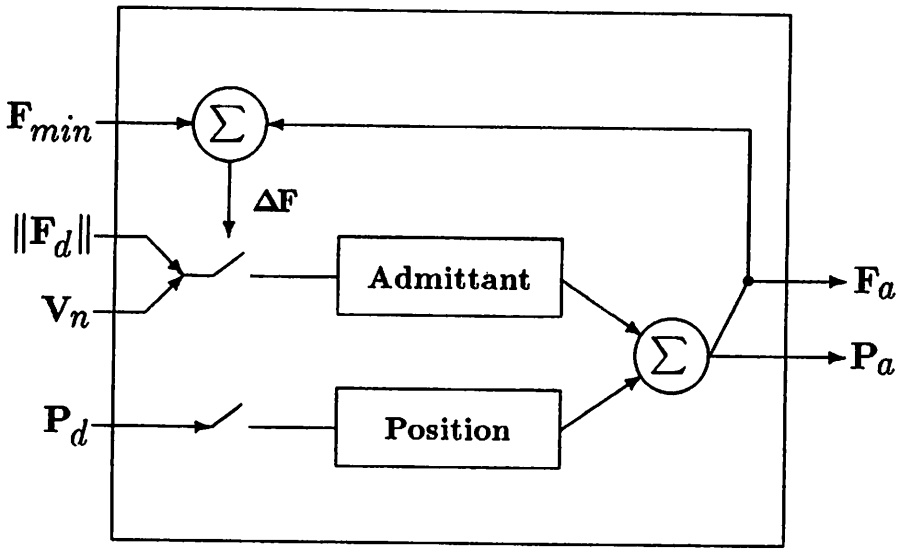

Figure A.7: System control architecture.

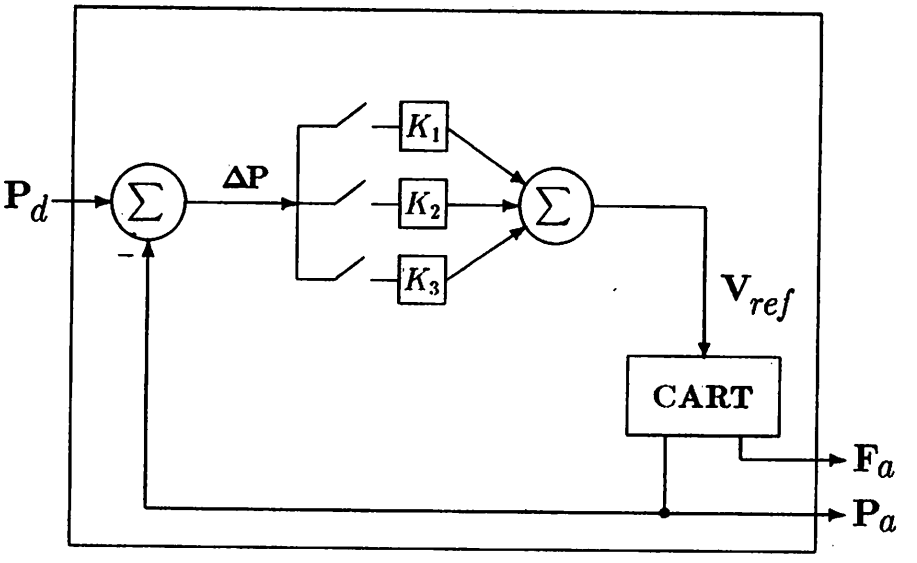

Figure A.8: Position control of the CART.

gain is scheduled by the difference between the actual net force and the desired net force. If the net force falls below the given value, the system selects a low gain that offsets the nominal velocity in the direction of the force; if the force is too high, a negative gain causes the sensor to back off slightly from the contact. When the net force is within an acceptable range, the force feedback dominates the computation and the sensor is pushed firmly agianst the surface. All of the gains were determined empirically, with the preference being for slight under-damping. This preference was dictated by the high stiffness of the system, which could easily result in damaging oscillations. However, the gains chosen proved to be quite robust; it is not only difficult to induce even small oscillations, but the resultant system is reasonably reactive to applied forces, especially considering the low servo sampling rate of the PDP 11/23 (60 Hz).

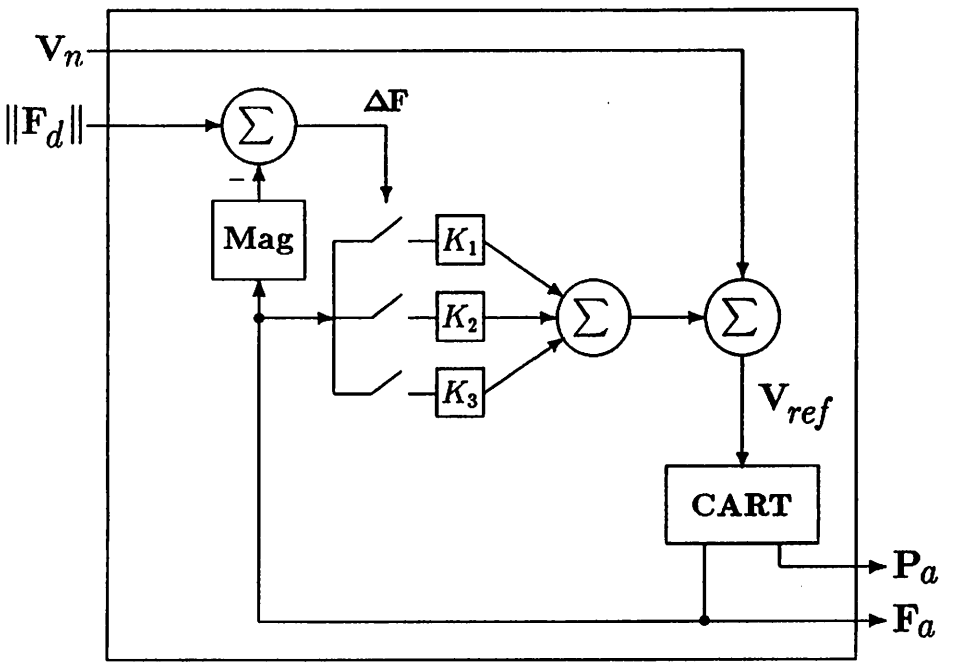

Figure A.9: Admittant control of the CART.

Our control architecture thus permits rapid and reliable acquisition of tactile data. We also learned that it is important to understand both the high-level

problem to be solved and the low-level control; it is by designing them in tandem, balancing the strengths and weaknesses of each, that a cleanly specified and effective system results. There is a wealth of literature in the field of control theory on optimality and performance guarantees, which we were regretfully unable to employ because of limited computational resources; but we hope that in the future robotic systems will be better integrated with better parts, thus improving performance of the whole.

BIBLIOGRAPHY

Allen, P., and Bajcsy, R.: 1985. Object recognition using vision and touch. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 1131–1137.

Bajcsy, R.:1984. What can we learn from one finger experiments? *First International Conference on Robotics and Automation Research*, M. Brady and R. Paul (eds), Cambridge: MIT Press, pp. 509–527.

Begej, S.: 1985. A tactile sensing system and optical tactile-sensor array for robotic applications. *Technical Report 85-06*, Department of Computer and Information Science, University of Massachusetts.

Bhanu, B.: 1984. Representation and shape matching of 3–D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3)340–351.

Boie, R.A.: 1984. Capacitive impedance readout tactile image sensor. *Proceedings of the IEEE Conference on Robotics (1984)*, pp. 370–378.

Bolles, R.C., and Cain, R.A.: 1982. Recognizing and locating partially visible objects: The Local-Feature-Focus method. *International Journal of Robotics Research*,1(3):57–82.

Bolles, R.C., Horaud, P., and Hannah, M.J.: 1983. 3DPO: A three-dimensional parts orientation system. *First International Conference on Robotics and Automation Research*, M. Brady and R. Paul (eds), Cambridge: MIT Press, pp. 413–424.

Boissonnat, J-D.: 1984. Representing 2D and 3D shapes with the Delaunay triangulation. *Proceedings of the Seventh IEEE Conference on Pattern Recognition*, pp. 745–748.

Brooks, R.: 1981. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17:285–349.

Cole, R. and Yap, C.K.: 1983. Shape from probing. *Technical Report No. 104*, Courant Institute of Mathematical Sciences, New York University.

Dario, P., De Rossi, D., Domenici, C., and Francesconi, R.: 1984. Ferroelectric polymer tactile sensors with anthropomorphic features. *Proceedings of the IEEE Conference on Robotics (1984)*, pp. 332–340.

Dario, P., Bergamasco, M., Femi, D., Fiorillo, A., and Vaccarelli, A.: 1987. Tactile perception in unstructured environments: a case study for rehabilitative robotics application. *Proceedings of the IEEE Conference on Robotics and Automation (1987)*, pp. 2047–2054.

Duda, R.O., and Hart, P.E.:1973. *Pattern Classification and Scene Analysis*, John Wiley & Sons: New York.

Ellis, R.E.: 1984. Extraction of tactile features by passive and active sensing. *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision (1984)*, pp. 289–295.

Ellis, R.E.: 1986. A multiple-scale measure of static tactile texture. *Proceedings of the IEEE Conference on Robotics and Automation (1986)*, pp. 1280–1285.

Ellis, R.E., Riseman, E.M., & Hanson, A.R.: 1986. Tactile recognition by probing: Identifying a polygon on a plane, *Proceedings of the AAAI Conference (1986)*, pp. 632–637.

Ellis, R.E.: 1987. Acquiring tactile data for the recognition of planar objects, *Proceedings of the IEEE Conference on Robotics and Automation (1987)*, pp. 1799–1805.

Faugeras, O.D., and Hebert, M.: 1983. A 3–D recognition and positioning algorithm using geometrical matching between primitive surfaces. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 996–1002.

Faugeras, O.D., Hebert, M., Pauchon. E., and Ponce, J.: 1984. Object representation, indentification and positioning from range data. *First International Conference on Robotics and Automation Research*, M. Brady and R. Paul (eds), Cambridge: MIT Press, pp. 425–446.

Fearing, R.S., and Hollerbach, J.M.: 1984 Basic solid mechanics for tactile sensing. *Proceedings of the IEEE Conference on Robotics (1984)*, pp. 266–275.

Fearing, R.S.: 1987 Some experiments with tactile sensing during grasping. *Proceedings of the IEEE Conference on Robotics and Automation (1987)*, pp. 1637–1643.

Gaston, P.C., and Lozano-Pérez, T.: 1984. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):257–265.

Grimson, W.E.L.: 1984. The combinatorics of local constraints in model-based recognition and localization from sparse data. *AI Memo 763*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Grimson, W.E.L.: 1986. Sensing strategies for disambiguating among multiple objects in known poses. *IEEE Journal of Robotics and Automation*, RA-2(4)196–213.

Grimson, W.E.L., and Lozano-Pérez, T.: 1984. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35.

Grimson, W.E.L., and Lozano-Pérez, T.: 1985. Recognition and localization of overlapping parts from sparse data in two and three dimensions. *Proceedings of the IEEE Conference on Robotics and Automation (1985)*, pp. 61–66.

Hillis, D.W.: 1984. A high-resolution imaging touch sensor. *International Journal of Robotics Research*, 1(2)33–44.

Horaud, P., and Bolles, R.C.: 1984. 3DPO's Strategy for matching three-dimensional objects in range data. *Proceedings of the IEEE Conference on Robotics (1984)*, pp. 78–85.

Horn, B.K.P.: 1983. Extended Gaussian Images. *A.I. Memo 740*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Horn, B.K.P., and Ikeuchi, K.: 1983. Picking parts out of a bin. *A.I. Memo 746*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Ikeuchi, K.: 1981. Recognition of 3-D objects using the Extended Gaussian Image. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 595–600.

Jacobsen, S.C., Wood, J.E., Knutti, D.F., and Biggers, K.B.: 1984. The Utah/MIT dextrous hand: Work in progress. *First International Conference on Robotics and Automation Research*, M. Brady and R. Paul (eds), Cambridge: MIT Press, pp. 601–653.

Kuan, D.T., and Drazovitch, R.J.: 1983. Model-based interpretation on range imagery. *AAAI-83 Proceedings*, pp. 210 215.

Lozano-Pérez, T.: 1983. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C32(2)108–120.

Luo, R.-C., Tsai, W.-H., and Lin, J.C.: 1984. Object recognition with combined tactile and visual information. *Proceedings of the Fourth International Conference on Robot Vision and Sensory Controls*, pp. 183–196.

Luo, R.-C., Wang, F., and Liu, Y.-X.: 1984. An imaging tactile sensor with magnetostrictive transduction. *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision (1984)*, pp. 264–270.

Nevatia, R., and Binford, T.O.: 1977. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98.

Overton, K.J.: 1984. The acquisition, processing, and use of tactile sensor data for robot control. *Technical Report 84-08* (Ph.D. thesis), Department of Computer and Information Science, University of Massachusetts.

Oshima, M., and Shirai, Y.: 1983. Object recognition using three-dimensional information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(4)353–361.

Perkins, W.A.: 1978. A model-based vision system for industrial parts. *IEEE Transactions on Computers*, C-27(2)126–143.

Raibert, M.H.: 1984. An all digital VLSI tactile array sensor. *Proceedings of the IEEE Conference on Robotics and Automation 1984*, pp. 314–319.

Rebman, J., and Morris, K.A.: 1983. A tactile sensor with electro-optical transduction. *Proceedings of the SPIE Conference on Robot Vision and Sensory Controls (1983)*, pp. 210–216.

Requicha, A.A.G.: 1980. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4)437–464.

Schneiter, J.L.: 1986. An objective tactile sensing strategy for object recognition and localization. *Proceedings of the IEEE Conference on Robotics and Automation (1986)*, pp. (2)1262–1267.

Seigel, D.M., Drucker, S.M., and Garabieta, I.: 1987. Performance analysis of a tactile sensor. *Proceedings of the IEEE Conference on Robotics and Automation (1987)*, pp. 1493–1499.

Stansfield, S.A.: 1987. Visually-aided tactile exploration. *Proceedings of the IEEE Conference on Robotics and Automation (1987)*, pp. 1487–1492.

Togai, M., Wang, P.P., and Rebman, J.: 1984. Design criteria and recognition schemes for an arrayed touch-sensor. *Proceedings of the IEEE Conference on Robotics (1984)*, pp. 385–393.

Turney, J.L., Mudge, T.N., and Volz, R.A.: 1985. Recognizing partially hidden objects. *Proceedings of the IEEE Conference on Robotics and Automation (1985)*, pp. 48–54.