A Tactile Recognition Strategy for Planar Objects

R. E. Ellis

COINS Technical Report 86-32

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 uniquely 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 for a 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 identify the object. Using assumptions that satisfy real-world sensing constraints, it is shown that polygons on a plane can be identified by finding a linear path that passes through an unsensed face of each interpretation. A computationally feasible method is presented for the two-dimensional case, based on a conversion of the problem to that of finding the intersection of all paths passing through each unsensed face. Extensions of the method, including the important case of recognizing polyhedral objects, are briefly examined.

This research was supported in part by the Office of Naval Research under Contract N00014-84-K-0564, and by the General Dynamics Corporation under Grant DEY-601550. Publication as a COINS Technical Report was recommended by Prof. Allen Hanson.

1. Introduction

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 identify 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 determined. An object model, in this initial analysis of the problem, is a polygon located on a plane.

The problem we address – 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 uniquely identifies the object, or it reduces the set of interpretations to a new, smaller set.

This work concentrates on the problem of intelligently acquiring new data (the third principal item). The questions of how to acquire the initial data and how to interpret them, while important, are peripheral to the present discussion.

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 models to be superposed, then some of the unsensed faces "line up" – if a sensor placed on the tip of a long rod were moved along a special line, 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 little circles indicate where an automatically planned linear path would contact each interpretation. If executed, this path would uniquely identify 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.

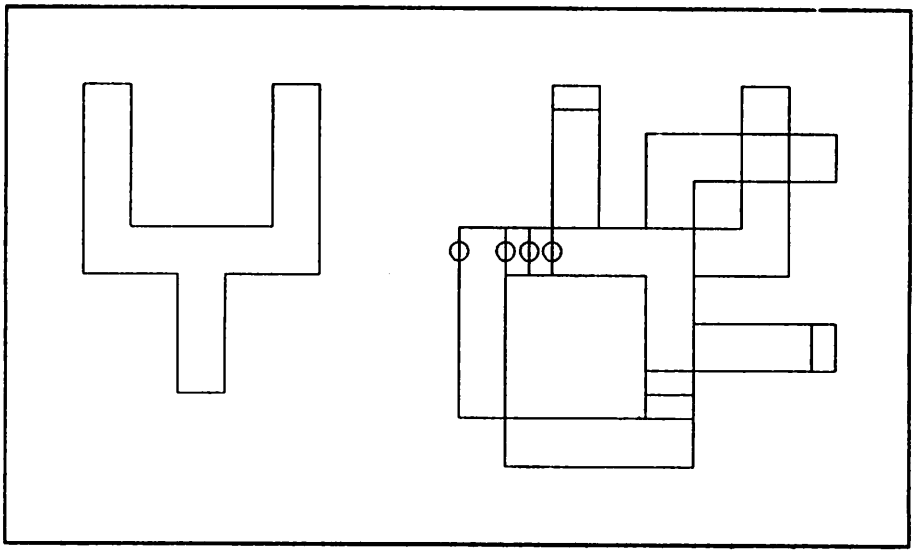

Figure 1.1: An object model, four interpretations of data, and where a path might contact each interpretation.

1.1 Object Models and Tactile Data

The recognition methodology we follow here is that of Grimson and Lozano-Pérez [2], [4], [5]. In this methodology, an object is represented as a polygon, i.e., as a set of line segments. The tactile data consist of a position, a local surface normal, and the maximum error value of each of these quantities (an error circle and error cone, respectively). **Interpretation** of tactile data consists in finding an assignment of each datum to a model face.

The interpretation occurs in two phases: the first is finding a set of feasible interpretations, which is done by applying local geometric constraints to the known relationships of the data; and the second is verification of these interpretations, which ensures that the assignments of data to faces are globally consistent. The output of this procedure is a set of valid rotations and orientations – the *poses* – of the object models such that the data lie on or near their assigned faces, and the sensed local normals are close to the normals of their assigned faces.

Our approach extends this methodology by showing how to acquire new data when multiple valid interpretations exist. Interpretation of these new data is rapid, because the path along which the sensor is moved has been calculated from the known valid interpretations; there are very few assignments of a newly acquired datum, and most of the possible assignments can be rapidly predicted from the geometric relationships between the path and the models.

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 the efforts of Allen and Bajcsy, [1], and Luo *et al.* [6]. 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 very 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 efforts of Grimson [3] and Schneiter [7]. 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).

1.3 Nomenclature

Table 1.1 summarizes the nomenclature used in this paper, and gives the standard usage of the symbols which will be described below.

Table 1.1: Notation

P_i	:	Position vector; point
$\hat{Z}_i$	:	A unit normal vector, in model coordinates
F_i	:	A model face
D_i	:	Distance, difference, or direction vector
$L_i(\lambda)$	:	Parametric form of a line in space
Q_P, Q_L	:	Projection of a point or line into a Euclidean space
θ	:	An angle as measured from the Y axis
α	:	Tangent, or slope; $\alpha = \tan\theta$
ϵ_{sensor}	:	The minimum spatial distance resolvable by a sensor
δ_{sensor}	:	The minimum angle resolvable by a sensor
Ω	:	The maximum permissible angle between a sensor and a surface

2. Piercing a Set of Line Segments

The principal conceptual problem in finding a path along which to move a tactile sensor is that of finding a line that pierces a number of line segments (or faces – the terms will be used interchangeably below). The line is the sensing path, and the segments are unassigned faces 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.

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

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 faces is powerful, and greatly facilitates the development of algorithms for finding sensing paths to distinguish planar objects.

2.1 Finding the Path Parameters

In the plane, a line has two degrees of freedom, one of position and one of direction. 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 face can be expressed as a pair of points, and the face 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.

The path parameters can 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 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

$$\mathbf{Q}(\alpha) = P_X - \alpha \cdot P_Y$$

This function yields the point on the **X** axis that pierces the point **P** with a path whose slope is α. Figure 2.1 shows the geometry of the projection of a point **P** onto the **X** axis for a given value of α.

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

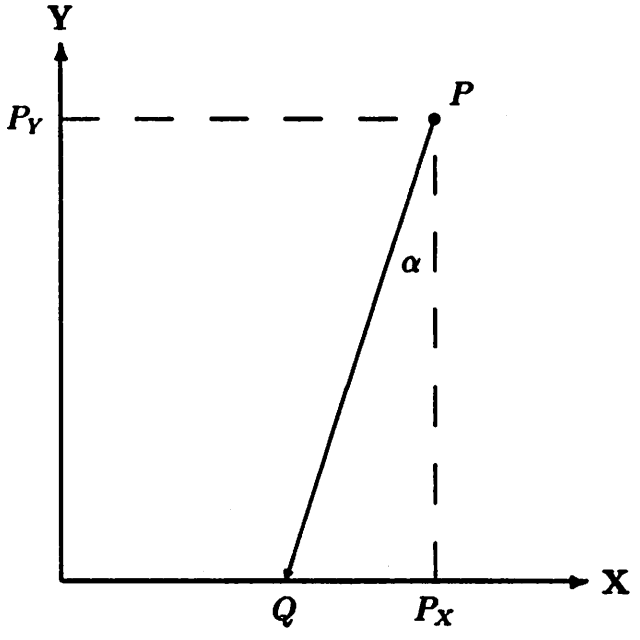

Figure 2.1: Projection of a point onto the **X** axis.

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

$$\mathbf{Q}_P(\alpha) = (P_X - \alpha \cdot P_Y\ ,\ \alpha) \tag{2.1}$$

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 2.2 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 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

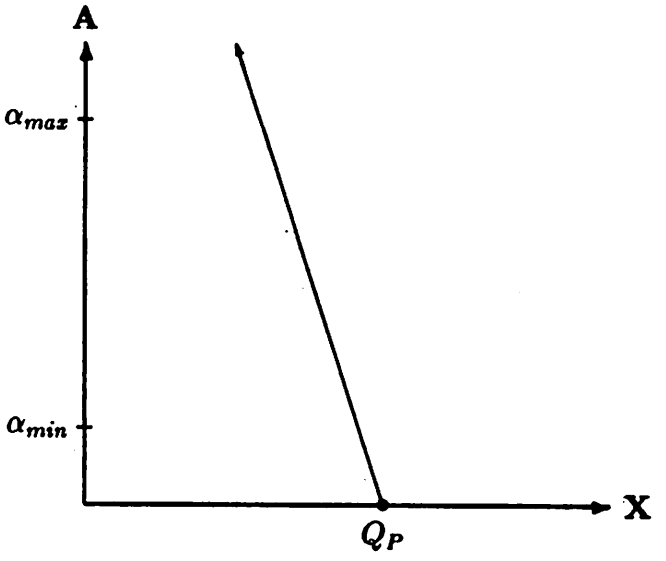

Figure 2.2: Representation of a point in projection space.

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 2.1 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 2.1. The new formula, which is a function of the path slope α and the line parameter λ, is

$$\begin{aligned} \mathbf{Q_L}(\lambda, \alpha) &= (L_X(\lambda) - \alpha \cdot L_Y(\lambda)\,,\ \alpha) \\ &= (P_X + \lambda \cdot D_X - \alpha \cdot P_Y - \lambda \cdot \alpha \cdot D_Y\,,\ \alpha) \end{aligned} \tag{2.2}$$

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 2.2 the projected lines of the endpoints will have different slopes. Figure 2.3 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.

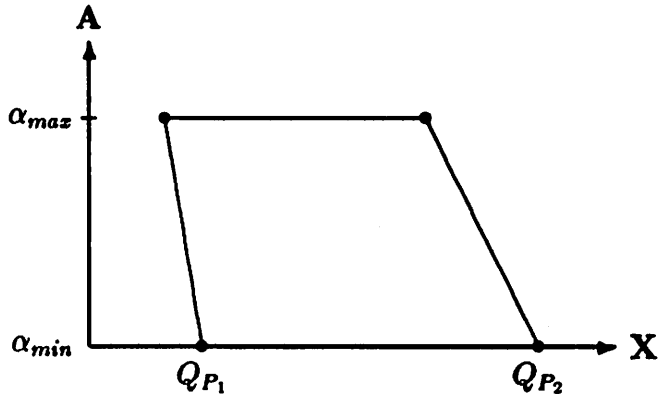

Figure 2.3: Representation of an edge in projection space.

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 2.4 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.

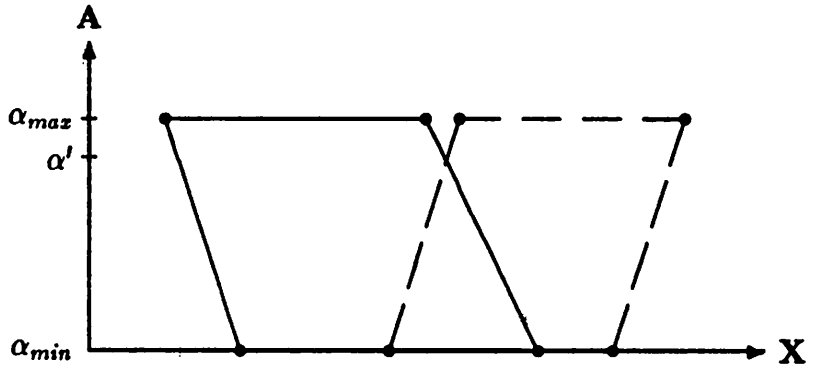

Figure 2.4: Two edges in projection space, and their intersection.

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 faces 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.

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

3. Computing a Sensing Path

We have described the theory of finding a sensing path in two dimensions. Implementation of this theory requires attending to the two difficulties of efficient computation of a path, and of ensuring that the path 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 unassigned face F_i in turn, and trying to find a path through it and as many other faces as possible. Assuming that the sensor can contact a face at very oblique angles, we can form the setq of faces $\{F_j \cdots F_k\}$ such that the angle between the normal $\hat{Z}_i$ of F_i and the normal $\hat{Z}_j$ of any face 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 face** (or generator, for short). Note that the candidates in this set must include the faces already sensed; it is possible that a path through the generator also contacts an assigned face, 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 unassigned face.
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 faces with the generating face.

(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.

3.1 Computing the Path Parameters

In projection space, the representation of a given face is a convex polygon, and the coordinates of points in its interior and boundary represent parameters of the sheaf of lines passing through the face. Thus, the parameters of the sheaf passing through a set of faces is represented by the intersection of the sheaves of each face, 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 as many faces, from different interpretations, as is possible.

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 faces $\{F_i \cdots F_j\}$ 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 face 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. Fig-

ure 3.1 shows the projections of three edges and the critical points that lie within the $[\alpha_{min}, \alpha_{max}]$ bounds; some of the critical points in this example have the same α value. The region labelled **S** (which is bounded above by the line $\alpha = \alpha_{max}$) is the sheaf polygon for all three edges.

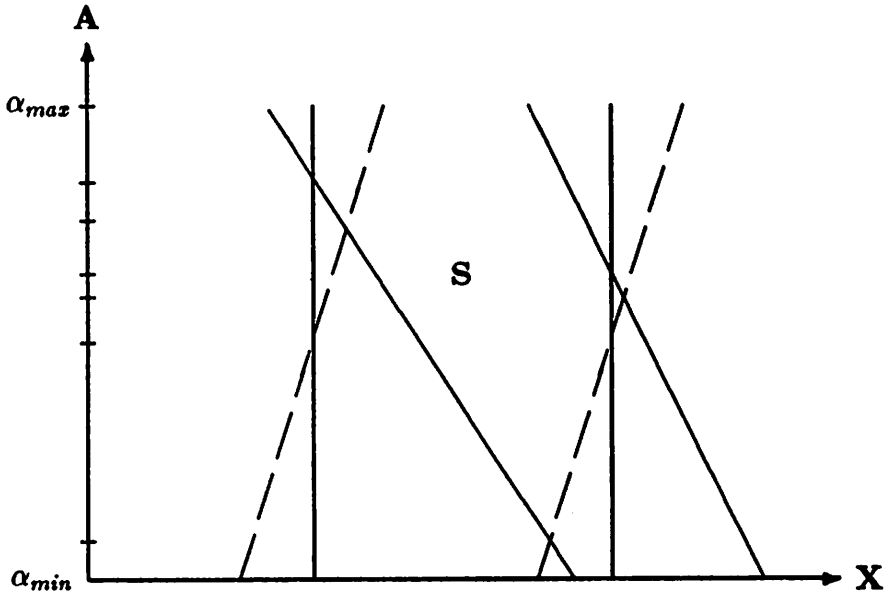

Figure 3.1: Three edges in projection space. The α value of their critical points are marked on the **A** axis.

Observe that 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 faces overlap on the **X** axis when projected by the value α'. Equivalently, the face representations in projection space must overlap at the value $\alpha = \alpha'$ if **c** is a critical point; in particular, all of the face projections must overlap $\mathbf{x}'$, which is a very simple and efficient test.

This means that in order to find a point in the intersection region of the largest number of faces, we need only project the faces at the values given by the critical points and examine the 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 face boundaries encode exactly those regions of projection space where qualitative changes of face overlaps occur.

Checking for intersection at a critical point is quite straightforward. For a pair of face endpoints P and R, the critical point occurs when the projections are equal,which is when $\mathbf{c} = Q_P = Q_R$. 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 face in turn; the critical point bounds a sheaf polygon including the face if and only if the **X** value of the critical point lies between the **X** value of the face endpoint projections at α'. Thus, we need only count the number of faces whose projections bracket the critical point to determine how many faces 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 faces. This can be remedied, if necessary, by re-examining the faces under the projection at α', finding the extent of the overlap of the face projections, and taking some other **X** value, e.g., the middle of the overlap. Because the maximum width of a convex polygon (in any given direction) occurs at a vertex, the critical points can be searched to find not only the largest *number* of overlapping faces 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 be available.

The present system sorts the critical values from the middle of $[\alpha_{min}, \alpha_{max}]$ out-

wards, and determines how many distinct interpretations have a face 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.

3.2 Evaluating a Sensing Path

That a path intersects several unassigned faces does not imply that a tactile sensor can determine which face has been contacted. There are limits to the ability of sensors to discriminate depth and orientation, and regardless, it is possible for several unassigned faces 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. These practical constraints can be summarized as:

- The face 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 faces must be distinguishable by the tactile sensor, i.e., for any pair of faces 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 face F_i must not exceed the maximum permissible contact angle Ω, i.e., $\angle(D\hat{Z}_i) < \Omega$.

Once a path has been found, all pierced faces must be examined to determine how these constraints apply. It often happens that unassigned faces appear in similar

configurations, and so they could not be distinguished by a tactile sensor.

In order to actually identify objects with sensor-based manipulators, then, we must address the issue of how to proceed when either there is no single path that can provide sufficient information, or when such a path can be expected (on the basis of the above combinatorics) to take an unreasonably long time to compute. For a given set of initial tactile data, a set of known object model models, and a particular path-finding algorithm, it is possible to derive a structure which completely describes the possible performance of our tactile recognition strategy.

3.3 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., does not invariably reduce the number of possible interpretations to a unique one. 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 face 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 faces 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 a face is struck at too oblique an angle for the sensor to detect reliably the local surface normal or position. (The path-finding process can be prematurely stopped according to various criteria, e.g., by it reaching a limit on computation time, or by it 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 3.2. 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 faces from distinct interpretations that were indistinguishable; hence, if the sensor contacted that face 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. A more complex tree, derived from a simulation experiment, is given in Figure 4.4 and described more fully below.

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.

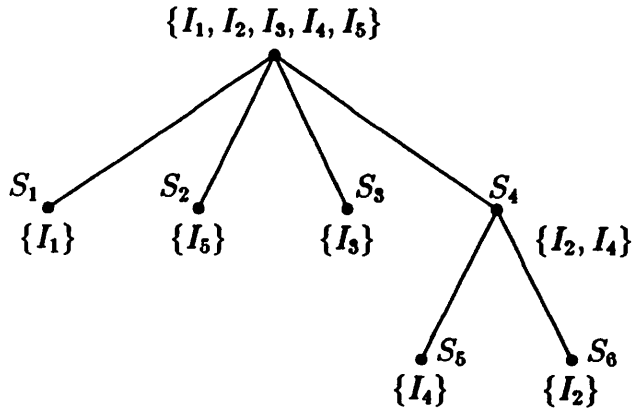

Figure 3.2: Ambiguity tree for a set of interpretations.

3.4 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: it entails determining which faces could conceivably have a path pass through both the generating face F_i and the candidate face 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 faces, so if there are M unsensed faces from all possible interpretations, and K sensed faces, 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 face 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 α_{min}, α_{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 face 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 faces, in case there are faces 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 faces pierced by the path has been found, we determine the first contact on each interpretation by linearly searching; 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 $O(I^2)$.

In summary, 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 be examination of just the most populous candidate sets, so finding a sensing path is a quadratic function of the number of unsensed faces. The dominant part of the calculation is determining and testing overlaps of candidates with the generator.

4. Experimental Results

An extensive series of simulations have been performed using this algorithm and the six polygonal test objects shown in Figure 4.1. 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, and a normal direction error of about 4 degrees. Table 4.1 gives the number of faces of each object, and the number of valid interpretations of the initial data for each object.

Table 4.1: Interpretations of 2 points.

Object Name	Number of Faces	Number of Interpretations
robot-hand	12	4
human-hand	18	3
telephone	12	2
boot	13	3
camera	12	6
beer-bottle	8	3

The simplest experiments attempted to distinguish among the four poses of the **robot-hand** object. A number of paths will uniquely identify these four interpretations. Figure 4.2 shows the four interpretations; circles indicate where the identifying path contacts the object, and the dots and spikes indicate the position and sensed normal of the given tactile data. (The four interpretations of these data, superposed, are presented in Figure 1.1.) 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 degrees, i.e., any slight touch of the 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 the result.

This simple experiment showed that even where there are a number of coincident faces, the algorithm can select paths that distinguish all of the interpretations. A more extreme test of the system's capability is when there are a large number of interpretations, with many faces either overlapping or very close together. When all of the models are included as possible objects, there are a total of 21 distinct interpretations of the data within the error bounds given above.

Using the large skid angle value, and supposing that the sensor has the stated error in determining position and local surface normal values, it turns out that there exists a *single* path that contacts 20 of the 21 interpretations. Of these 20 contacted, 17 are terminal nodes of the ambiguity tree; the remaining non-terminal nodes can be distinguished with a second path. Figure 4.3 shows the objects, with the faces that would be pierced by a path marked with a circle.

A large number of other runs were made. Table 4.2 summarizes the results of some of these runs, indicating the models used, the number of possible interpretations, and the number of interpretations that had at least one face pierced. Because of the design parameters of these tests, they approximate the performance of the best tactile sensors currently available.

The final series of experiments were designed to explore some of the computational issues of the approach, especially in multi-path instances. For these tests, we reduced the skid angle to 45 degrees (which is available with cheap but accurate sensors), and stopped the path-finding when 7 distinct interpretations had been contacted. This arrest occurred before the path assessment stage, so it was not always the case that a path could uniquely distinguish amongst the 7 contacted by the path.

Table 4.3 summarizes the results of the most interesting run, in which all 21

Table 4.2: Distinguishing Multiple Objects.

Objects	Interpretations Found	Interpretations Distinguished
robot-hand human-hand	4 3	7
robot-hand human-hand telephone boot	4 3 2 3	12
camera beer-bottle	6 3	9

interpretations of the 6 models were included; Figure 4.4 gives the ambiguity tree for the results, with {} representing the class of unexpected data or no contact with any surface. The first path found can identify 6 classes of interpretations, 5 of these being terminal; if none of these are correct, the next path can identify 7 interpretations.

The third path, which is the optimal path, has some curious characteristics that are made evident by the ambiguity tree. There are 6 equivalence classes, 4 of which are terminal and one of which contains a pair of interpretations; the {} node occurs because Interpretation 8 was not contacted at all by the path. If the object was guaranteed to be one of those modelled, no further paths would be necessary if contact didn't occur when the sensor was moved along the third path.

This run shows that when we limit the amount of computation permitted in the planning phase, no more than 4 probes are needed to uniquely determine which of the 21 original interpretations of the data is correct. Table 4.3 also includes relative computation times for the interpretation phase, each path-finding phase, and the time needed to find the best single path indicate the efficacy of a multi-path approach to object recognition. In these units, a manipulator could be expected to take about 60

timesteps to execute a path, so after the first one is found the time to compute the next path is comparable to physical transit time.

Table 4.3: Multiple-Path Identification.

Pass	Interpretations			Cost
	before	after	pierced	
Verification	–	21	–	42
Candidate Formation	21	21	–	151
Path 1 (limited search)	21	14	7	137
Path 2 (limited search)	14	7	7	54
Path 3 (exhaustive search)	7	1	6	87
Path 4 (trivial search)	1	0	1	1
TOTAL TIME				**472**
Optimal Path	21	2	19	**1389**

The ambiguity tree for this large run is shown in Figure 4.4. Each level shows the interpretations identified by some distinct datum along the sensing path. The entry {} indicates that the next level should be explored if the datum found is not one of those expected.

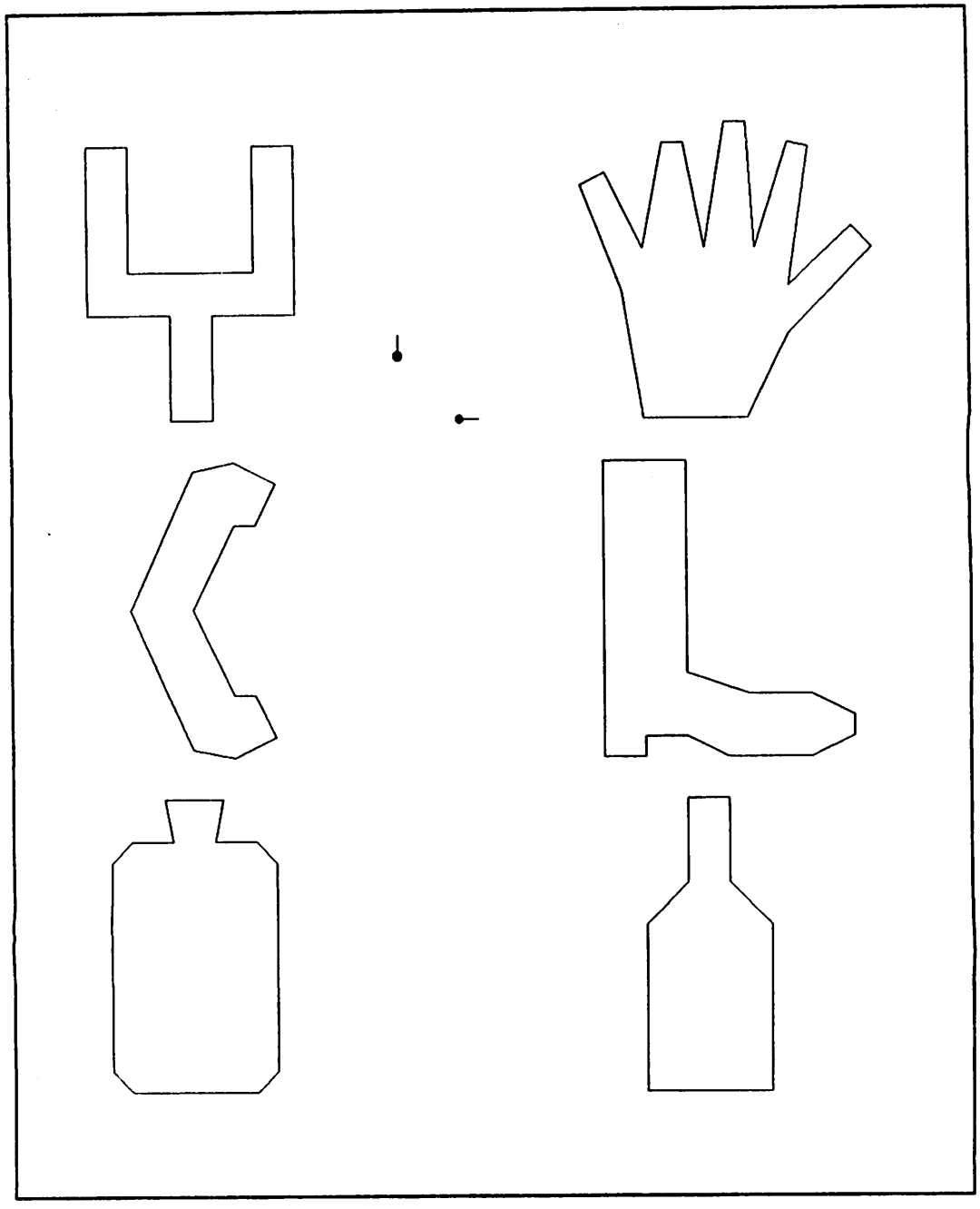

Figure 4.1: Object models and initial tactile data used in the experiments. (The reader can find interpretations of these objects by copying the tactile data onto a transparent sheet, and moving the sheet about to find places on the models where the position and local-surface-normal constraints given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

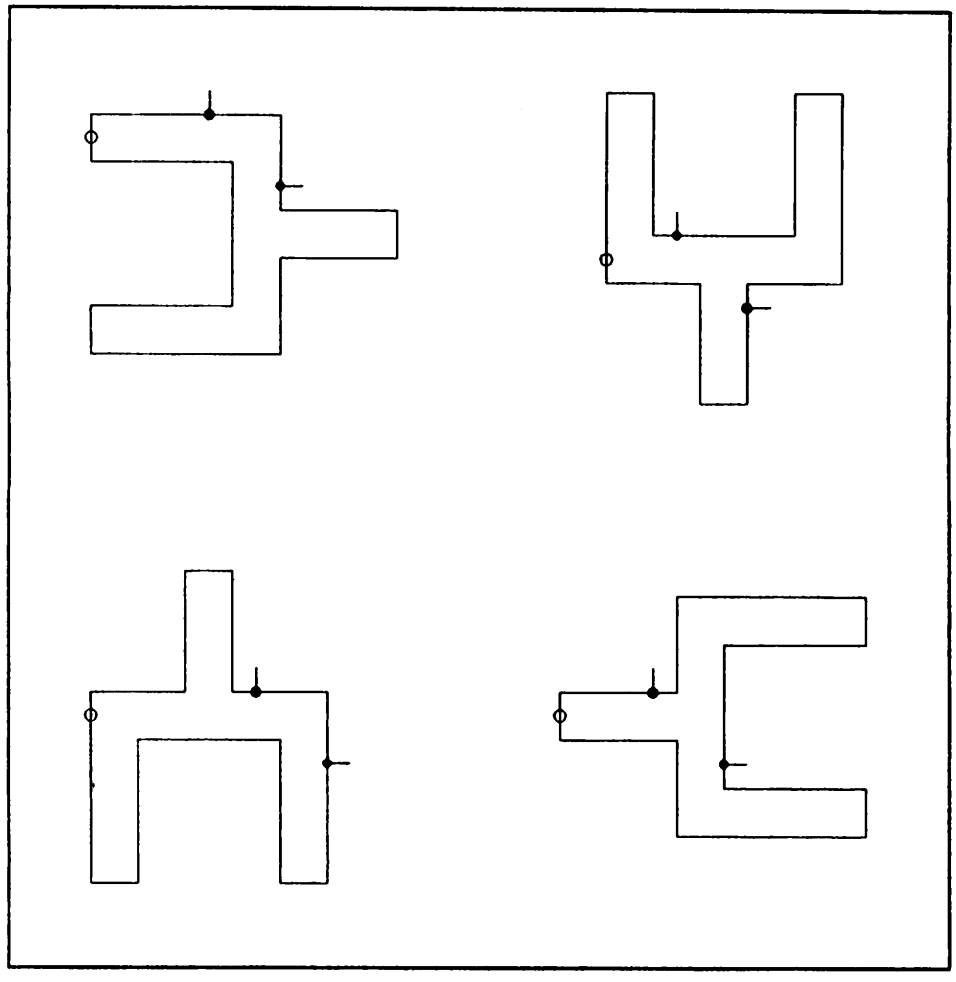

Figure 4.2: Four interpretations, and where the path contacts each.

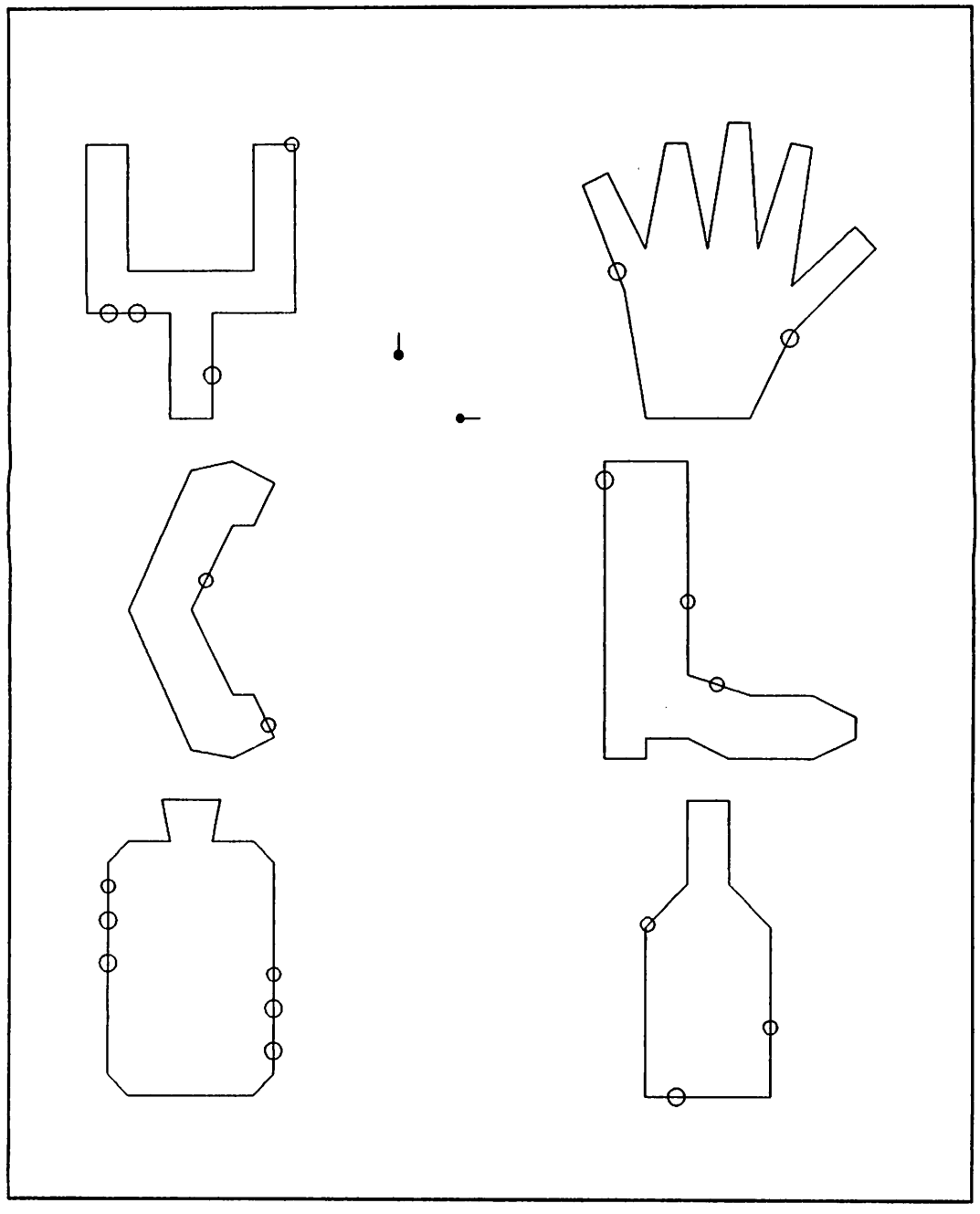

Figure 4.3: Where a path contacts 20 distinct interpretations of the objects.

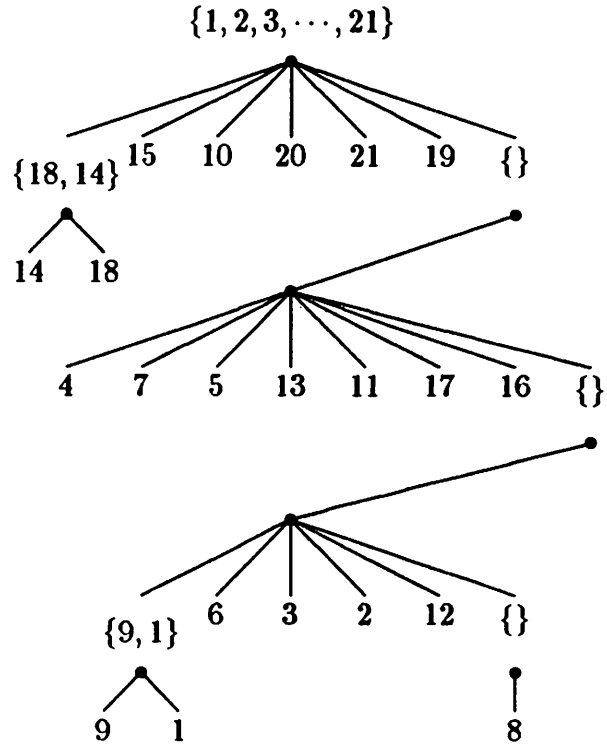

Figure 4.4: Ambiguity tree for 21 interpretations, limited path search.

5. Extensions

There are three obvious extensions to the current system. One is to include error in the location and orientation of the interpretations; another is to reduce the restrictiveness of some of the underlying assumptions; and the third extension is to attempt to identify three-dimensional polyhedral objects.

5.1 Error in the Interpretations

Because there is error in the initial tactile data, there is error in the estimated pose of the model that interprets the data. The present system, for simplicity, used the given position and local surface normal values; since the true values could deviate from these sensed values, a given model instantiation could actually have a range of translations and rotations that would fit the data.

There are two ways to extend the present system to handle this error. One is to find the range of poses for an interpretation, find the face locations at these ranges, and form candidate sets only of faces that do not overlap. The other way is to find a worst-case estimate of the positional and orientational deviation of each face, and just add these in to the sensor error in the path-evaluation stage of the algorithm.

Which extension is preferred depends upon how the system is likely to be used. The former method is costly at candidate-formation time, because sorting through candidates for the best combination of non-overlapping faces is not trivial. The second method is costly because it can result in the rejection of paths which met the optimality criteria of the previous stage; what is optimal in one stage can be sub-optimal in another, resulting in excessive computation.

The second method, however, would be preferable when optimal paths are not of concern, e.g., where computational resources are limited or multiple paths are so

likely that several paths will have to be found anyway. In such cases, the number of interpretations that are lost to error in the initial data can be expected to be small compared to the number that just can't be distinguished by the path found; when such a situation obtains, inserting interpretation-pose error into the path evaluation stage can be justified. When optimal path performance is critical the first, systematic method is preferable.

5.2 Changing the Assumptions

A number of restrictive assumptions are made in the present work. Two that are especially open to challenge are that there is a single object in the workspace, and that the sensor is of effectively infinite length (the present methodology has placed no *a priori* restriction on the length of a path, and thus no restriction on the length of the sensor).

Path planning in a crowded workspace is non-trivial, but possible. The present methodology would require first that a recognition engine work on data from a crowded workspace, and then that a method be developed for piercing only those faces that are of interest, avoiding the extraneous objects when possible.

Regarding the first issue, we note that the recognition scheme used can be extended to handle the case of multiple objects in two and three dimensions [5]. In such cases, the recognition engine would return all possible interpretations of the data. Secondly, the present method could be modified to attempt to identify objects when there are restrictions on such parameters as starting position and location; such a modification would permit, if not identification, at least a considerable reduction in the ambiguity. Clearly, this is an issue that requires much further work.

Changing the second assumption does not require such significant modifications to the present system. If the sensor configuration is changed to suppose that it is a projection of finite length from some body – mimicking a finger sticking out from a hand – we can observe that for a given path we can find the position along this

path where the hand might contact one of the interpretations. This would represent the point on the path where the manipulator would *have* to stop if no sensors were on the hand, or where it *might* stop if the hand was equipped with simple bump-contact sensors. Either way, the maximum path length can be incorporated into either the path-planning stage or the path-evaluation stage, with the trade-offs being similar to those examined when we considered incorporating interpretation error into the system. Changing this assumption would also introduce the possibility that the object could not be uniquely identified with the sensor at hand, a possibility present with errorful sensors anyway but which is increased when obstructions are added.

5.3 Three-Dimensional Objects

The most intriguing extension of the present method is to attempt to use the same framework to identify three-dimensional objects. Conceptually, the extension is possible; computationally, there are significant difficulties to be overcome when attempting to find the optimal path.

In three dimensions, a face of a polyhedral object is a polygon, which in turn has line segments as boundaries. In the projection formula for lines, even in two dimensions, a non-linearity is present; but in two dimensions, because the boundaries are points, we were able to sidestep the issue. In the polyhedral case, the projection of an arbitrary line segment in even one angular degree of freedom becomes a ruled hyperbolic paraboloid; a polygon (even a convex polygon) will in general project to a non-convex volume bounded by a set of these curved surfaces. In finding the optimal path, we seek a point in the interior of the intersection of a number of these structures. (Since there are actually two angular degrees of freedom, the problem is four-dimensional, and consequently much worse.)

This extension is currently under active investigation. Because of the considerable cost of analytically finding the optimal path, we prefer to linearize the projection equations and reduce it to a simpler problem. Linearization entails missing some

paths, finding some that are unacceptable, or both; we are currently examining the issues in approximation, linearization, and computation time, and believe that reasonable results can be achieved even when attempting to optimize over all four of the path parameters simultaneously.

6. Conclusions

We have defined a methodology for acquiring new tactile data in a model-based recognition scheme when the available data are not sufficient to uniquely identify the object in question. A method was proposed for finding a path along which to move a tactile sensor so that the maximum amount of information can be gained from the sensor motion. Simulations show that this method is practical and effective in gathering tactile data to recognize polygonal objects which lie on a planar surface.

A number of extensions of this method can be performed, including elaboration of the way in which sensor error is managed, loosening of some of the restrictive assumptions, and extension of the method to finding paths for three-dimensional objects. These extensions vary in conceptual and practical difficulty, but are all possible with some effort.

A possible future direction arising from this work is that it may permit us to address the question of when to stop performing recognition, which is essentially data-driven, and begin performing verification, which is essentially model-driven. It is possible to estimate the complexity of computation of recognition, and how many interpretations we can expect from an initial set of tactile data; we can also place bounds on the amount of computation needed to plan verifying paths and execute them with a sensor. Combining these parts into a computational framework may permit us to develop a hybrid control strategy for identifying objects in minimum time. These, and other directions, are currently under investigation in our Laboratory.

Acknowledgements

Many of the ideas explored in the paper were developed in conversations with Ed Riseman and Al Hanson, both of whom also commented extensively on the manuscript. Thanks also to Les Kitchen and Damian Lyons, for their comments on the preliminary draft of the paper.

This research was supported in part by the Office of Naval Research under Contract N00014-84-K-0564, and by the General Dynamics Corporation under Grant DEY-601550.

References

[1] **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.

[2] **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.

[3] **Grimson, W.E.L.: 1986.** Disambiguating sensory interpretations using minimal sets of sensory data. *Proceedings of the IEEE Conference on Robotics (1986)*, pp. (1)286–292.

[4] **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.

[5] **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 (1985)*, pp. 61–66.

[6] **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.

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