From: AAAI-86 Proceedings. Copyright ©1986, AAAI (www.aaai.org). All rights reserved.

# Tactile Recognition by Probing: Identifying a Polygon on a Plane

R. E. Ellis
Edward M. Riseman
Allen R. Hanson

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 uniquely the model instance and model pose that best interpret the object. In this paper, we consider the situation in which some tactile data about the object are already available, but can be ambiguously interpreted. The problem is thus to acquire and process new tactile data in a sequential and efficient manner, so that the object can be recognised and its location and orientation determined. An object model, in this initial analysis of the problem, is a polygon located on a plane; the case of planar objects presents some interesting problems, and is also an important prelude to recognition of three-dimensional (polyhedral) objects.

## 1. Introduction

This work addresses the question of how a robot equipped with a tactile sensor can recognise and locate an object in its workspace. 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 research was supported in part by the Office of Naval Research under Contract N00014-84-K-0564, by the General Dynamics Corporation under Grant DEY-601550, and by the National Science Foundation under Grant DCR-8318776.

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.

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 often 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. In some cases there will still be ambiguity, but it will usually have been reduced.

Identification of an object from ambiguous data can thus be accomplished if a line can be found that passes through an unsensed face of each valid model interpretation. Our method for finding these lines involves changing the representation of the problem, and asking what sheaf of lines can possibly pass through each unsensed 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 show 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.

### 1.1 Related Work

There is very little work, past or present, on ways of intelligently and automatically acquiring tactile data for the

purposes of object identification. There are only two reasonably well-known works of significance: that 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 simple tactile features subsequently, to search through a decision tree; the sensing strategy is very simple, consisting of repeated rotation of the sensor about the object. While it is effective in simple cases, the authors point out its shortcomings in dealing with smooth or highly symmetric objects.

The recognition methodology we follow here is that of Grimson and Lozano-Pérez, who in several papers have detailed a model-based tactile recognition approach [3], [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.

Our approach extends this methodology by showing how to acquire new data when multiple valid interpretations exist. Interpretation of these new data is very 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. The approach presented here, and related issues, are discussed more fully in [2].

## 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 unsensed 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.

### 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; 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 an **X** intercept, and a direction. In order to make it clear that some later formulae have important linear forms, the path direction will be expressed as the slope $\alpha$ of the line, taken with respect to the **Y** axis. That is, if the angle between the path and the **Y** axis is $\theta$, we will deal only with the slope of the line which is $\alpha = \tan\theta$.

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 $\alpha_{min}$ and $\alpha_{max}$.

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 $\alpha$, 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 $\alpha$.

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

$$\mathbf{Q}_P(\alpha) = (P_X - \alpha \cdot P_Y\ ,\ \alpha) \qquad (1)$$

This function may be interpreted as giving the position of a point $Q_P$ in **X-A** space, derived by projecting the original point **P** (in **X-Y** space) onto the **X** axis in the direction $\alpha$.

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 projection line $Q_P$ in **X-A** 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 Equation 1 to determine how each point on the line segment would project.

We can represent a line parametrically as a point and

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

a displacement along some direction; letting the point be $P$ and the direction vector $D$, the equation for a line is $L(\lambda) = P + \lambda \cdot D$. Substitution in Equation 1 gives the projection formula of a line, which is a function of the path slope $\alpha$ and the line parameter $\lambda$, as

$$\begin{aligned} \mathbf{Q_L}(\lambda, \alpha) &= (L_X(\lambda) - \alpha \cdot L_Y(\lambda), \alpha) \qquad (2) \\ &= (P_X + \lambda \cdot D_X - \alpha \cdot P_Y - \lambda \cdot \alpha \cdot D_Y, \alpha) \end{aligned}$$

where the subscripts indicate components of the line, point, and direction. This formula is non-linear in $\alpha$ and $\lambda$. However, for fixed $\lambda$, it is linear in $\alpha$; in particular, the endpoints of a segment project into lines in **X-A** space. If $D_Y$ is nonzero, i.e., if the line segment is not parallel to the **X** axis, then by Equation 2 the projected lines of the endpoints will have different slopes. Figure 1 shows a projection of an edge into **X-A** space; the boundary is not a parallelogram because the edge was originally tilted with respect to the **X** axis. The left and right line segments represent the projections of the edge boundary points $P_1$ and $P_2$ into **X-A** space.

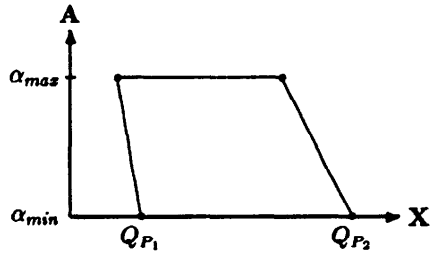


**Figure 1:** Projection of an edge into **X-A** space.

Note that the locus of points in **X-A** that is described by this projection, when $\alpha$ and $\lambda$ 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 project each into **X-A** 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. Figure 2 shows the projection of two distinct edges in **X-A** space; the area of intersection is the set of points that specifies the sheaf of paths that pierce both original edges.

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 **X-A** space. The coordinates of any interior point represent the position on the **X** axis and the direction $\alpha$ of a line

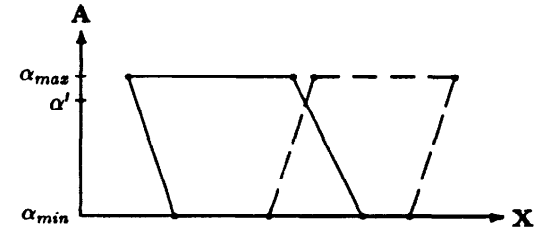


**Figure 2:** Two edges in **X-A** space, and their intersection.

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.

## 3. Computing a Sensing Path

Our approach to calculating a sensing path involves examining each unsensed face $F_i$ in turn, and trying to find a path through it and as many other faces as possible. A real-world constraint is that when a sensor contacts a surface at too oblique an angle, it either skids off or returns unreliable data. We can thus form the set of unsensed faces $\{F_j \cdots F_k\}$ such that the angle between the normal of $F_i$ and the normal of any face in the set is less than the sensor skid angle; let us call this the **candidate set** formed from $F_i$.

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

1. Calculate the candidate set of each unsensed face.
2. Sort the candidate sets according to how many interpretations are present in each.
3. Find and test a feasible path through each candidate set:
   (a) Find the projection parameter $\alpha'$ which creates the maximum overlap of candidate faces with the generating face $F_i$.
   (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.

This algorithm is more fully described in [2]. Here, we will only outline the computational approach, describe the complexity of the algorithm, and discuss some of the path evaluation issues.

### 3.1 Computing the Path Parameters

In **X-A** space, the projection 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.

Our method for finding a point in the sheaf polygon is to examine only the regions near its vertices. Because a vertex is formed from the intersection of the projections of endpoints of different faces – which we call the critical points of the projection – this simple geometric observation reduces the search from a full two-dimensional one to a search over a finite set of points. Combinatorially, there are in general $O(N^2)$ points to examine if there are $N$ faces in a candidate set. Figure 3 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 $\alpha$ value. The region labelled **S** (which is bounded above by the line $\alpha = \alpha_{max}$) is the sheaf polygon for all three edges.

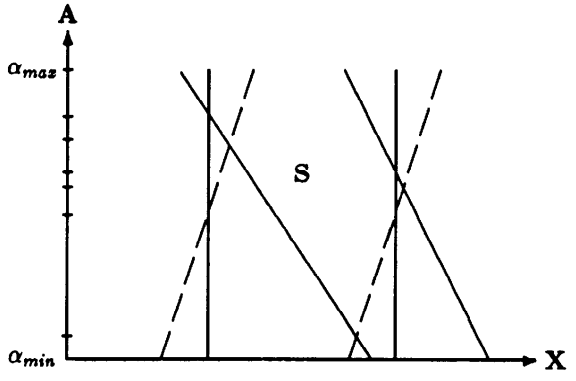


**Figure 3:** Three edges in **X-A** space, and their critical points.

Once these critical points have been found, we must determine how many faces have projections that contain each of these points. This test is simple, but adds a level of complexity to the algorithm, since we must check $N$ faces for each of $O(N^2)$ critical points; the net worst-case complexity is thus $O(N^3)$. In practice, however, very good paths can be found well before this worst-case behaviour becomes significant.

## 3.2 Evaluating a Sensing Path

That a path intersects several unsensed 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 unsensed faces to be exactly coincident. 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 is either unreliable, or unavailable because the sensor skids off the object surface. Once a path has been found, all pierced faces must be examined to determine how these constraints apply. It often happens that unsensed faces appear in similar two-dimensional configurations, and so they could not be distinguished by a tactile sensor.

We address such cases by forming an **ambiguity tree** for the interpretations. At each level of this $n$-ary tree, the nodes represent the set of interpretations that are possible if a particular sense datum (or class of sense data) are found. The width of the tree at any level indicates how many equivalence classes of interpretations there are with respect to the path from which it was formed, and the depth indicates how many paths must be followed by the sensor, in the worst case, if the object in the workspace is to be uniquely identified.

An example ambiguity tree is given in Figure 4. Beginning at the root, there are 21 interpretations; the first path uniquely identifies 5 of these, one datum indicates an equivalence class of 2 interpretations, and if any other datum were detected (or if no contact took place) then more motion would be required do distinguish among the remaining 14 interpretations. For this complex tree, at most 4 paths would have to be traversed to identify the object, but most likely only 2 would be needed.

## 4. Experimental Results

An extensive series of simulations have been performed using this algorithm and the six polygonal test objects shown in Figure 5. 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 1 gives the number of valid interpretations of each object.

**Table 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 the four poses of the **Robot-Hand** object. A number of paths will uniquely identify these four interpretations. Figure 6 shows the four interpretations; the 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. It was 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.

Of interest was how many distinct interpretations of these data could be identified with a *single* path. The answer is, with the above path constraints, that 20 out of the 21 can be contacted, 17 of these being terminal nodes in the ambiguity tree. (A second path is required to distinguish among the remaining interpretations.) Table 2 summarises the results of various runs, indicating the models used, the number of possible interpretations, and the number of interpretations that had at least one path pierced.

**Table 2:** Distinguishing Multiple Objects.

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

It is not always possible to distinguish all of the interpretations with a single path; in such cases, multiple paths must be found. To test the system's capability, we reduced the sensor skid angle to 45 degrees, permitted the path-finding to stop when 7 interpretations could be distinguished, and ran it successively on the full object set. As is summarised in Table 3, the first path would distinguish 7 of the 21 interpretations; if none of these interpretations was the correct one (the worst case), the second path would distinguish 7 of the remaining 14, the third path would distinguish 6 (which is optimal), and the last path is trivial. 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 3:** Multiple-Path Identification.

| Pass | Interp's pierced | Cost |
|---|---|---|
| Verification | – | 42 |
| Candidate Formation | – | 151 |
| Path 1 (limited search) | 7 | 137 |
| Path 2 (limited search) | 7 | 54 |
| Path 3 (exhaustive search) | 6 | 87 |
| Path 4 (trivial search) | 1 | 1 |
| TOTAL TIME | | **472** |
| Optimal Path | 19 | **1389** |

The ambiguity tree for this large run is shown in Figure 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.

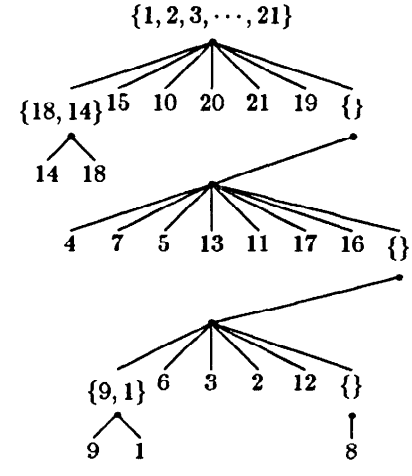


**Figure 4:** Ambiguity tree for 21 interpretations, limited path search.

## 5. 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 recognise simple objects on a planar surface.

This method extends to the three-dimensional case, in which objects are represented as polyhedra, but that problem is significantly harder. The non-linearities of the projection equations are not simplified by the boundary conditions (as was the case here), so the problem becomes one

of finding a point in the intersection of a four-dimensional structure which is bounded by curved hypersurfaces. Linearising the 3–D problem, and producing both analytical characterisations and search heuristics, is a topic of ongoing research.

## 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] **Ellis, R.E., Hanson, A.R., and Riseman, E.M.: 1986.** A tactile recognition strategy for planar objects. *COINS Technical Report*, Department of Computer and Information Science, University of Massachusetts.

[3] **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.

[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 Symposium 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.


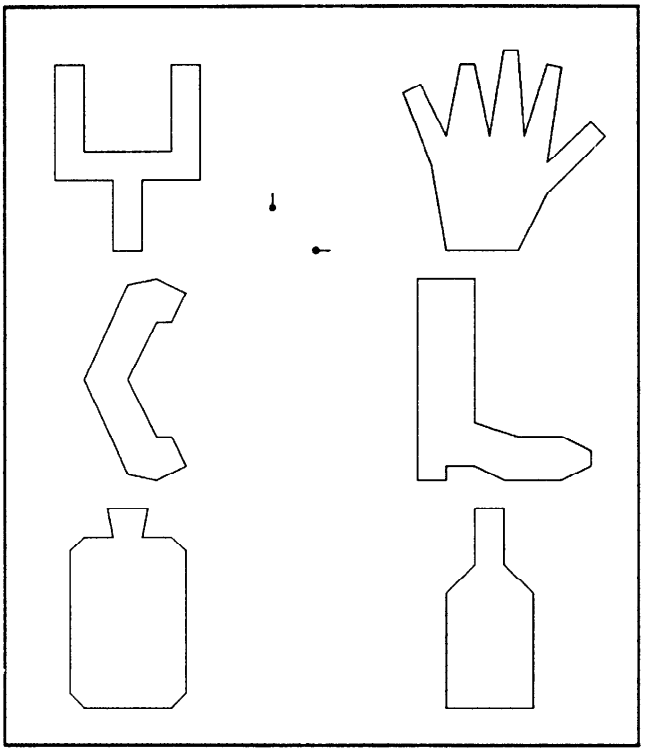

**Figure 5:** Object models and initial tactile data used in the experiments. (The reader can find interpretations of these objects by copying the tactile data onto a transparent sheet, and moving the sheet about to find places on the models where the position and local-surface-normal constraints are simultaneously satisfied.)

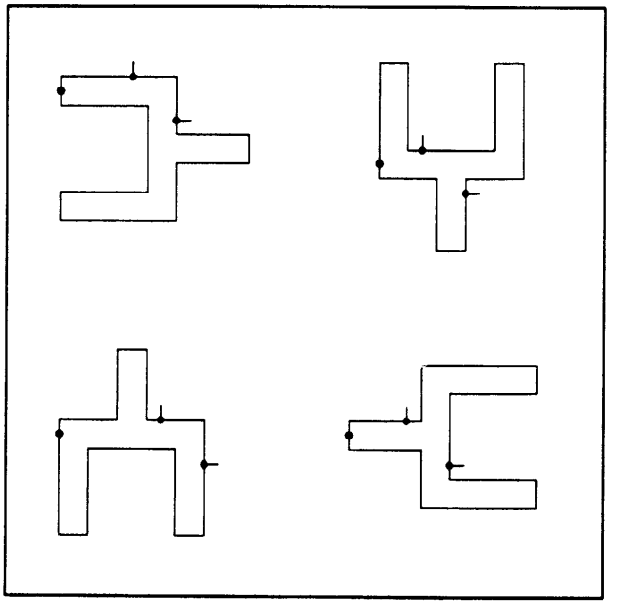

**Figure 6:** Four interpretations, and where the path contacts each.