An Algorithm for Planning 'Sensible' Routes*

Ian Pratt
Department of Computer Science
University of Manchester
Manchester, M13 9PL
England
ipratt@uk.ac.man.cs

Abstract

The problem of finding an optimal route between two points in a space strewn with obstacles has been extensively studied in robotics. The standard approach to such problems is first to reduce the original geometrical specification to a finite graph of possible routes, and then to use graph-searching techniques together with the metrical information available in the original geometrical specification to select the best of these routes. The algorithm presented here, by contrast, employs a 'first-past-the-post' approach to route planning. Specifically, we show that the algorithm is guaranteed to produce a route satisfying a certain criterion of 'sensibleness', because routes failing this criterion are always slower to get constructed. The approach taken here differs in three respects from most current work on route planning: (i) it finds sensible, but not necessarily optimal, routes; (ii) it is uninfluenced by perspective (though it is limited by occlusion problems); (iii) it produces its output in a qualitative form, which is nevertheless well-suited to guide the appropriate route-following behaviour.

1 Background

The problem of finding an optimal route between two points in a space strewn with obstacles has been extensively studied in robotics. As the problem is normally posed, the set of obstacles cluttering space, the dimensions and degrees of freedom of the moving robot, and the start and goal points, are all specified in some numerical coordinate system. In the special case of planning routes for a point object moving in a plane with polygonal obstacles, this specification usually amounts to a Cartesian coordinate description of the scene in plan view, with the output being given in terms of the coordinates of line segments defining the optimal route. The standard approach to such problems is first to reduce the original geometrical specification to a finite graph of possible routes, and then to use graph-searching techniques together with the metrical information

*Appeared in *Engineering Applications of Artificial Intelligence*, vol. 4 no. 2, (1991), pp. 97–108.

available in the original geometrical specification to select the best of these routes. The route planning problem in 2-d for a point-mover with polygonal obstacles has been shown to be solvable by means of the well-known *V-graph* in $o(N^2)$ time, where N is the number of vertices in the scene in question.[1]

However, casual reflection on, for example, the way people negotiate their way across rooms cluttered with tables and chairs and briefcases and litter bins suggests a rather different picture. First, though people often find good routes in such situations, it is unlikely that they reliably choose the very best. After all, it is very difficult to compare, by eye, the perimeters of objects, especially when those objects have curved boundaries. Secondly, when they look at the obstacles in front of them, people do not immediately perceive, and very probably never compute, a numerical coordinate description of of the scene in plan view. In particular, the obstacles in question may be partially or totally occluded; and even when occlusion is no problem, the scene is always presented in perspective. Thirdly, bearing in mind that the point of route planning is the eventual motion of the planner, it is unclear that the description of the chosen route—that is to say, the output of the route planning process—need or should be given in terms of the numerical coordinates of the critical line segments. Much more useful, for example, might be a list of directions as to which side of the various obstacles one is to pass (more on this way of specifying routes anon).

This paper presents a route planning algorithm which, though by no means equal to human navigation abilities, takes seriously the three points of the previous paragraph. Specifically, the algorithm (i) finds *sensible* routes rather than optimal ones (*sensible* in a sense to be made precise below), (ii) is uninfluenced by perspective (though it is limited by occlusion problems) and (iii) produces its output in qualitative terms (which are nevertheless well-suited to guide the appropriate route-following behaviour). The more 'qualitative' approach adopted here is close in spirit to some recent robotics research which strives to achieve useful behaviour with the minimum of internal—and especially numerical—representations.[2]

The plan of the paper is as follows. Section 2 sets out a criterion of 'sensibleness' which we will expect a chosen route to satisfy, a criterion which serves to rule out many sub-optimal routes in spaces which are not too cluttered. Section 3 presents the route planning algorithm. Section 4 shows that this algorithm is guaranteed to yield only sensible routes (this section may be omitted). Section 5 goes on to indicate how the algorithm might be modified so as efficiently to rule out additional suboptimal routes not ruled out by the original criterion. Section 6 discusses applications and further work. In this paper, we restrict ourselves to route planning for a point object moving in a plane cluttered with convex (not necessarily polygonal) obstacles.

[1]Weltzl [5]. For an introduction to the V-graph technique, see Lozano-Pérez [3]. For a more general review, see Schwartz and Sharir [4].

[2]See, for example, Chongstitvatana and Conkie [1] and Conkie and Chongstitvatana [2]

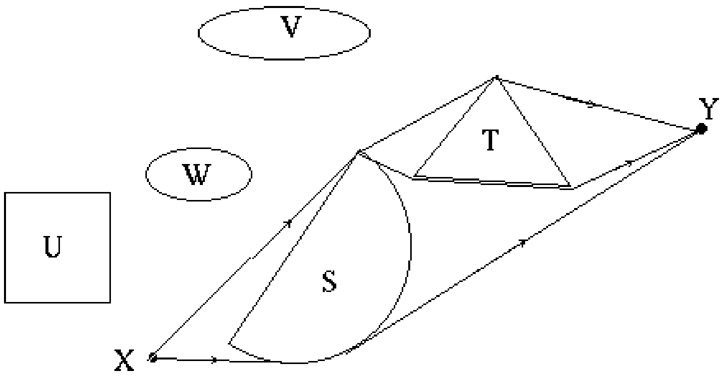

Figure 1: Some routes through a cluttered plane

2 The Criterion

As already mentioned, *people* are frequently unable to select the optimum route through a scene cluttered with obstacles; yet they seldom choose a very bad one. In this section, we present the criterion of 'sensibleness' which we shall expect the route chosen by our algorithm to satisfy. To be sure, the criterion we use deems as sensible many routes which, even to the unaided human eye, are clearly not. Thus, our method performs less well in some respects than people. But that is only to be expected: the results of this paper are not intended as the last word on route planning, but as a concrete proposal for a new way of approaching the problem. The criterion will be presented formally below, but fig. 1, which shows three routes from point X to point Y, helps to give an intuitive idea of its force. All routes from X to Y other than the three shown either cross over themselves (and hence are blatantly suboptimal) or fail to satisfy the criterion.

We begin by introducing the notion of a *directed joint tangent.* Given any two non-intersecting convex regions with nonempty interiors, S and T, there will be exactly four line segments which are tangent at their endpoints to S and T, as shown in fig. 2. We call these line segments the *joint tangents* involving S and T.[3] When one of S or T shrinks to a point, the number of joint tangents reduces to two; and when both S and T are point regions, there is just one joint tangent between them. A *directed joint tangent* (djt) is simply a joint tangent with a direction associated with it. Hence, ignoring regions with empty interiors, there are exactly eight djt's associated with any pair of non-intersecting convex regions. A djt has a well-defined *orientation*—clockwise or anticlockwise—

[3] Some authors use the term *lines of support* to denote the (infinitely extended) lines along

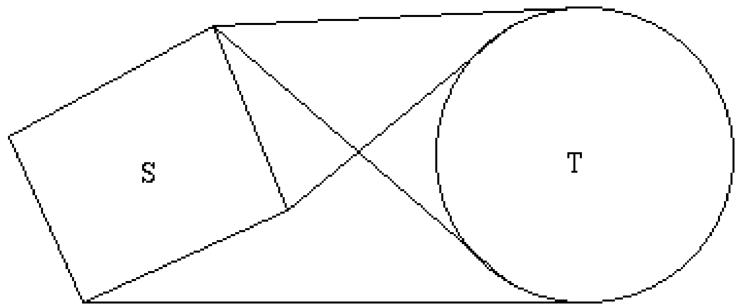

Figure 2: The joint tangents associated with two convex regions

about the regions (with nonempty interiors) it runs between. We say that a djt has *clockwise* orientation about a region if that region lies to the right of the djt, and *anticlockwise* orientation about a region if that region lies to the left of the djt (fig. 3).

Djt's are important in route planning because of the following observations.

Observation 1 *If p is an optimal route between two points in a plane cluttered with convex obstacles, then p consists of a sequence of djt's interspersed with segments of the boundaries of obstacles.*

Referring back to fig. 1, we see that the lowermost route from X to Y consists of a djt from the point-region X to the obstacle S, followed by a segment of S's boundary, followed finally by a djt from S to the point-region Y. Similarly, the middle route consists of a djt from X to S followed, after touching the boundary of S at a single point, by a djt from S to T, followed by a segment of T's boundary, followed finally by a djt from T to Y.

Observation 2 *If p is an optimal route between two points in a plane cluttered with convex obstacles, then any two successive djt's in p have the same orientation about their common obstacle.*

Again, referring back to fig. 1, we see that, in the lowermost route from X to Y, the two djt's both have anticlockwise orientation about S. Similarly, in the

which joint tangents lie.

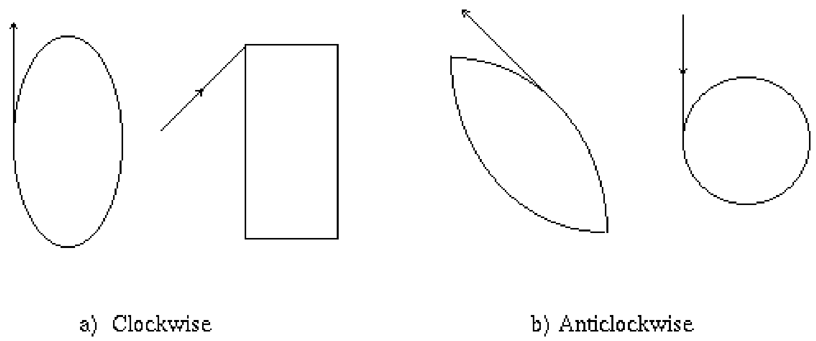

Figure 3: The orientation of djt's

middle route, the first and second djt's (which both involve the region S) both have clockwise orientation about S, and the second and third djt's (which both involve the region T) both have anticlockwise orientation about T. We can see immediately that any path which fails to satisfy the conditions of observations 1 and 2 must be suboptimal, by imagining a rubber band stretched taut along it. Such a rubber band, when released, would obviously snap into some other position around the obstacles in such a way as to reduce its overall length.

It is also clear that a route satisfying these conditions is determined by the sequence of shapes it visits together with its orientation about those shapes. Thus, the lowermost route in fig.1 can be specified as:

$$[X, S^{-}, Y]$$

and the the middle route as:

$$[X, S^{+}, T^{-}, Y]$$

where $+$ denotes clockwise orientation and $-$ anticlockwise orientation. Because of the central importance of shape-orientation pairs in route planning, we will henceforth refer to them as *oshapes*: thus, S^{+}, S^{-}, T^{+}, T^{-} etc. are all oshapes. In the sequel, we use the letters σ, τ etc. to range over oshapes. Notice that djt's do not have determinate orientations about point-regions such as X and Y, so no superscripts are needed.[4] Henceforth, when we speak of a *route*, we shall take for granted that it satisfies the conditions in the above observations and hence can be specified as a sequence of oshapes (together with a start and finish); no other paths need concern us.

These preliminaries having been dealt with, we now introduce two notions which are crucial to our criterion of sensibleness. The first notion is that of a route's being *clockwise definite* or *anticlockwise definite*.

[4]In some cases, djt's involving regions which are themselves line segments can lack determinate orientations. However, we ignore line-shaped obstacles in this discussion.

Definition 1 *Let p be a route from point X to point Y. We say that p is* clockwise definite *if, for any obstacle O encountered by p between X and Y, p passes clockwise around O. We say that p is* anticlockwise definite *if, for any obstacle O encountered by p between X and Y, p passes anticlockwise around O.*

Intuitively, clockwise definite routes always turn clockwise—i.e. always veer to the right; anticlockwise definite routes, by contrast, always turn anticlockwise—i.e. always veer to the left. The route p' schematically represented in fig. 4a) is thus clockwise-definite; the route p' schematically represented in fig. 4b) is anticlockwise-definite. Trivially, all routes from X to Y visiting no intermediate obstacles (i.e. just consisting of one djt) are both clockwise and anticlockwise definite, since they do not encounter any intermediate obstacles.

The second notion is that of one route's being on the *clockwise side* or on the *anticlockwise side* of another.

Definition 2 *Suppose p and p' are two routes from X to Y which cross over neither themselves nor each other, and which have only their endpoints in common. Let p^* be the route formed by first following p, and then following p' in the reverse direction, so that p^* is a simple circuit. If p^* winds clockwise, then we say that p is* on the clockwise side *of p'. Otherwise, if p^* winds anticlockwise, then we say that p is* on the anticlockwise side *of p'.*

Again, the intuitive idea is best conveyed by a diagram such as fig. 4. Fig. 4a) schematically represents a route p lying on the clockwise side of another route p'; following p and then p' in the reverse direction results in a clockwise circuit. Fig. 4b) schematically represents a route p lying on the anticlockwise side of another route p'; following p and then p' in the reverse direction results in an anticlockwise circuit. Notice that, provided p and p' do not cross over themselves or each other, and have only their endpoints in common, p^* is always defined and winds either clockwise or anticlockwise. Thus, either p lies on the clockwise side of p' and p' on the anticlockwise side of p, or vice versa.

The following observation establishes the relevance of these notions to the question of suboptimality.

Observation 3 *Let p and p' be routes from X to Y which cross neither themselves nor each other, and which have only their endpoints in common. If p' is clockwise (anticlockwise) definite and p lies on the clockwise (anticlockwise) side of p', then p is longer than p', and is hence sub-optimal.*

In both the cases shown in fig. 4, we can see immediately that p must be suboptimal by imagining rubber bands stretched taut along the routes p and p': removing the obstacles enclosed by p and p' would cause p to snap back to the line of p', indicating that p is longer than p'.

Thus, the notions of clockwise/anticlockwise-definite and clockwise/anticlockwise-side yield a sufficient criterion for identifying suboptimal routes. Whence we define:

Criterion 1 *A route p between points X and Y in a plane cluttered with a collection Ω of convex obstacles is said to be* sensible *if there do not exist points*

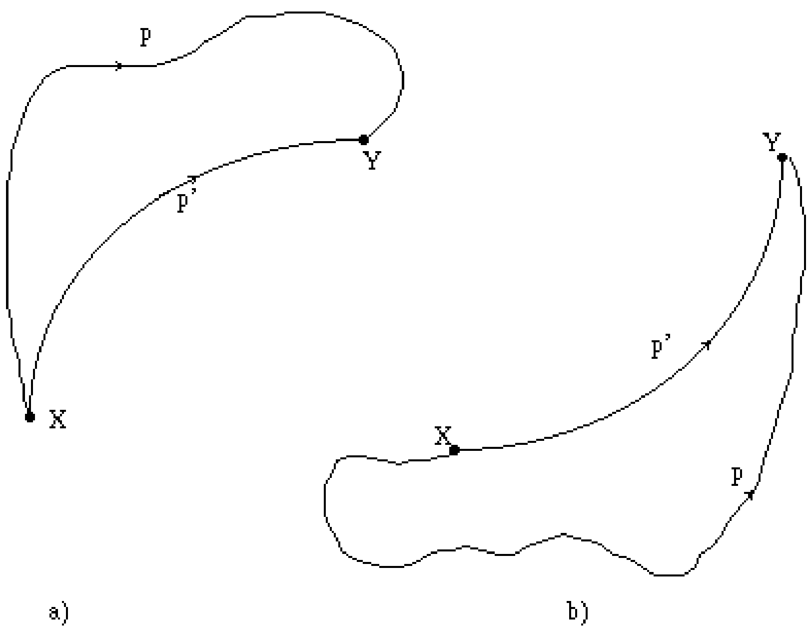

Figure 4: Some non-sensible routes

A and B on p where either:

> *there exists a clockwise definite route p' between A and B around the obstacles in Ω such that p' and the portion of p between A and B cross neither themselves nor each other, and such that and the portion of p between A and B lies on the clockwise side of p',*

or:

> *there exists an anticlockwise definite route p' between A and B around the obstacles in Ω such that p' and the portion of p between A and B cross neither themselves nor each other, and such that the portion of p between A and B lies on the anticlockwise side of p'.*

Basically, a *sensible* route is one which cannot be seen to be suboptimal by applying observation 3. However, it should be stressed at this point that the above criterion is satisfied by many obviously suboptimal routes. In particular, some routes which cross over themselves—and all such routes are blatantly suboptimal—may be sensible according to criterion 1. Section 5 indicates briefly how the algorithm presented here might be modified so as to avoid routes which cross over themselves.

3 The Algorithm

The algorithm presented here will find a route between two points in a plane strewn with convex obstacles which is guaranteed to be *sensible* in the sense made precise above. The route will be expressed as a list of oshapes, again as explained above.

We begin with the observation that unobstructed djt's can be ordered in a clockwise fashion according to their orientation. Imagine the entire plane being rotated anticlockwise about some fixed point through a complete revolution. We can associate with each unobstructed djt the time at which it is horizontal and pointing to the right; the resulting cycle of djt's defines the clockwise order in question. If two or more djt 's are cooriented, then let the tie be broken arbitrarily. We assume that we have some facility for finding the unobstructed djt's in a scene, in clockwise order. In the rest of this section, when we speak of *djt's*, we shall understand *unobstructed* djt's, since these are the only ones of interest for route planning. [5]

The idea behind the algorithm is then as follows. The (unobstructed) djt's in the scene are considered in clockwise order. Each of these djt's is simply specified as an ordered pair $\langle \sigma, \tau \rangle$ of oshapes, and constitutes a route from σ to τ. As more and more djt's are considered, a network of routes between the oshapes is built up. If the start point X and goal point Y are also included among the oshapes (with arbitrarily assigned orientations), the algorithm can terminate when a route is found between X and Y. The way in which the network of routes is built up is such as to ensure that the first route found between X and Y is *sensible* according to criterion 1.

The following example illustrates the procedure for building up routes. Suppose the first djt to be considered runs from S^- to T^+. Then we have the one-step route $S^- \to T^+$. If, subsequently, the djt from R^- to S^- is considered, not only do we have the new one-step route $R^- \to S^-$, but also, by joining this new djt onto the beginning of the existing route $S^- \to T^+$, the two-step route $R^- \to S^- \to T^+$. If, still later, the djt from T^+ to U^+ is considered, we can form the three-step route $R^- \to S^- \to T^+ \to U^+$. Notice that, in this example, the original route $S^- \to T^+$ was extended both *backwards* by linking the djt $R^- \to S^-$ onto the first oshape S^-, and *forwards* by linking the djt $T^+ \to U^+$ onto the last oshape T^+. For reasons which will become clear later, we impose the following rule concerning the extension of existing routes by new djt's: an existing route may only be extended forwards if it ends in a clockwise oshape; and it may only be extended backwards if it starts in an anticlockwise oshape. Or, to put matters another way: only djt's running from oshapes with clockwise orientation will be allowed to link onto the end of existing routes; and only djt's running to oshapes with anticlockwise orientation will be allowed to link onto the front of existing routes.

The above procedure thus allows existing routes to grow by addition of suc-

[5] A minor complication concerning the algorithm presented below arises when three or more shapes all lie along a single common tangent. To simplify matters, whenever shapes R, S, T lie along a common tangent in that order, we say that the relevant djt from R to T is *not* obstructed by S.

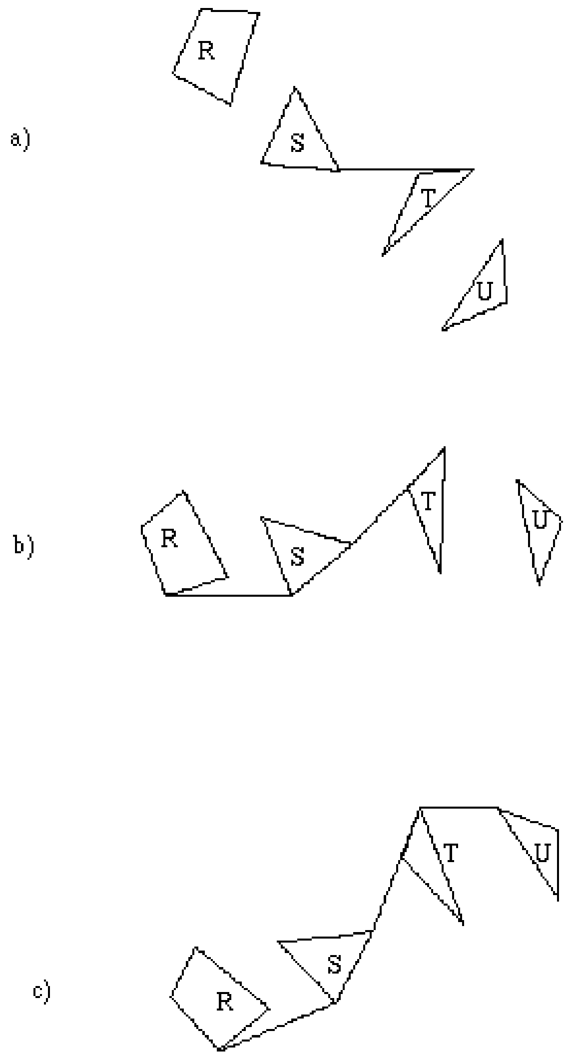

Figure 5: The formation of some routes

cessive djt's. Pictorially, if we imagine the entire scene rotating anticlockwise, we can think of routes which end in clockwise oshapes as growing forwards by winding clockwise about the last obstacle, perhaps linking up to another oshape when the appropriate djt is horizontal and pointing to the right. Similarly, we can think of routes beginning with an anticlockwise oshape as growing backwards by winding anticlockwise around the first obstacle, perhaps linking up to another oshape when the appropriate djt is horizontal and pointing to the right. Fig. 5 illustrates the stages in the formation of the route $R^- \to S^- \to T^+ \to U^+$ as the scene is rotated anticlockwise. Notice that, given the above restrictions, a route ending with an anticlockwise oshape and starting with a clockwise oshape cannot grow any more.

The above procedure also allows two existing routes to be joined together. Suppose, for example, that when the djt from S^+ to T^- is considered, we already have a route from R^- to S^+ and a route from T^- to U^+. Then, the new djt can link onto the clockwise end S^+ of the former route and the anticlockwise beginning T^- of the latter route to from a route all the way from R^- to U^+ via the oshapes S^+ and T^- (fig. 6). By thus growing and joining routes as successive djt's are considered, we will eventually find a route from the start point X to the goal point Y.

There is one final, but crucial element in our algorithm: given any two oshapes σ, τ in the scene, we stipulate that once a route is found from σ to τ, no other route from σ to τ will be recorded. In effect, we are adopting a first-past-the-post approach to route planning: as successive djt's are considered (in clockwise order), more and more routes are found; but once a route between two oshapes is found in this way, we do not change it or find alternatives.

The basic algorithm can be implemented in various ways, and I shall present the most readily comprehensible, if not the most efficient, of these. The implementation proceeds by constructing a table of 'directions' like the one shown in fig. 7. The rows and columns of the table are labelled by the oshapes in the scene in question. The row labelled by a given oshape represents journeys *originating* from that o-shape, while the column labelled by a given oshape represents journeys *terminating* at that oshape. Thus, the cell

```
table(fromoshape, tooshape)
```

will be used to store information about how to get *from* `fromoshape` *to* `tooshape`. Specifically, `table(fromoshape, tooshape)` gives the name of the first oshape after `fromoshape` on a complete route from `fromoshape` to `tooshape`. (Recall that routes are completely specified by the list of oshapes that they visit.) Suppose the entry in `table(fromoshape, tooshape)` is `oshape1`: then to find the next oshape in the route we examine the contents of the cell `table(oshape1, tooshape)`, obtaining, say `oshape2`; to find the next oshape in the route we examine the contents of the cell `table(oshape2, tooshape)`, and so on, until we eventually find a cell containing `tooshape` itself, meaning that we have reached our destination. Of course, if there is a djt leading directly from `fromoshape` to `tooshape`, then the cell `table(fromoshape, tooshape)` can simply contain the entry `tooshape` to represent this fact. Thus, following a route encoded using such a table is a matter of jumping from entry to entry in a fixed column of

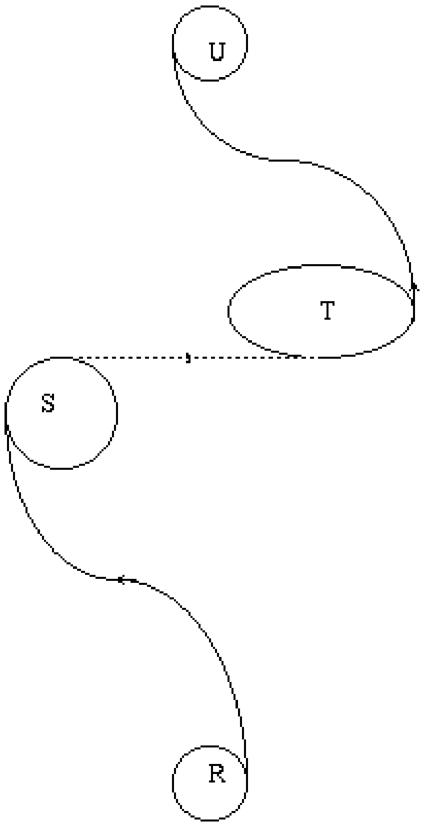

Figure 6: The linking up of existing routes with a new djt

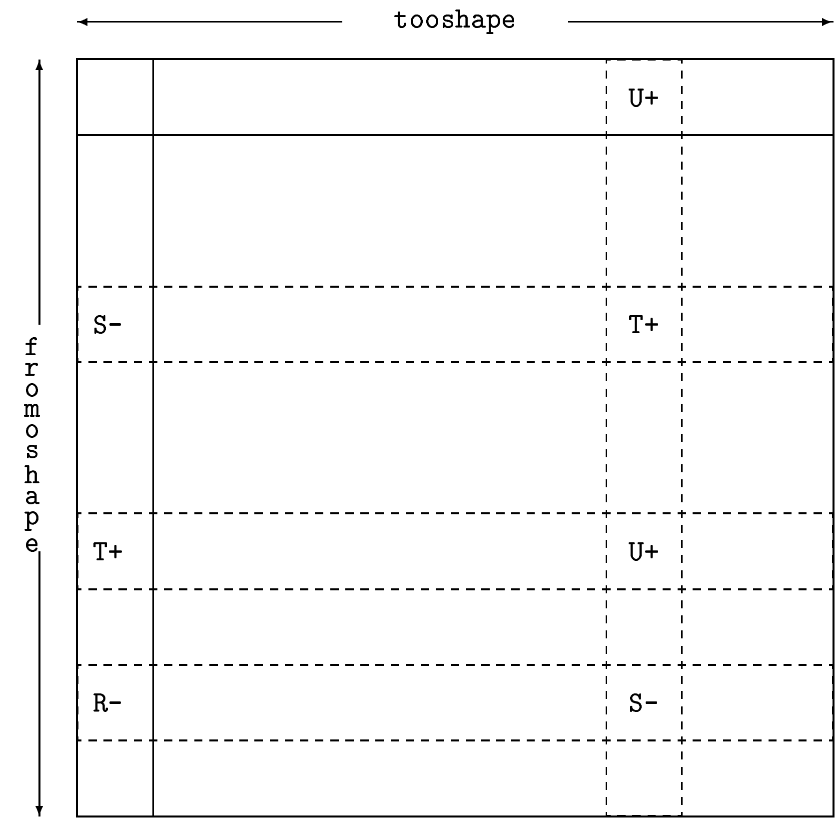

Figure 7: A table of directions

the table (the column corresponding to the destination oshape). Fig. 7 shows the critical entries corresponding to a route

$$[R^-, S^-, T^+, U^+]$$

from R^- to U^+. Obviously, such a table cannot encode all possible routes in the given scene. The idea behind the route planning algorithm presented here is that the table be filled in such a way that only 'sensible' routes are included.

Entries are made in the table of directions as djt's are considered in clockwise order. For example, if a djt from S^- to T^+ is considered, the name T^+ is entered into the cell `table`(S^-, T^+), because a way to get from S^- to T^+ is simply to go directly there. Similarly, if the djt from R^- to S^- is subsequently considered, the name S^- is entered into the cell `table`(R^-, S^-). Notice, however, that after the second of these djt's, we have not only found the routes $S^- \to T^+$ and $R^- \to S^-$, but also the route $R^- \to S^- \to T^+$. We must also therefore enter the name S^- into the cell `table`($R-, T^+$) to indicate this fact: if you are at R^- and you want to go to T^+, proceed first to S^-. More generally, suppose we find a new djt, say from `fromoshape` to `tooshape`. As explained above, if `fromoshape` has clockwise orientation, we should be prepared to link the new djt onto the end of any existing route terminating in `fromoshape`, and if `tooshape` has anticlockwise orientation, we should be prepared to link the new djt onto the start of any existing route beginning with `tooshape`.

Taking `table(oshape1, oshape2)` to be undefined if a route from `oshape1` to `oshape2` has not yet been found, we can set the procedure out formally as follows. First, we define functions `predecessors` and `successors`:

```
function predecessors(oshape)
begin
        if orientation(oshape) = anticlockwise then
            { oshape }
        else
            { oshape1 | table(oshape1, oshape) ≠ undefined }
        endif
end
```

```
function successors(oshape)
begin
        if orientation(oshape) = clockwise then
            { oshape }
        else
            { oshape1 | table(oshape, oshape1) ≠ undefined }
        endif
end
```

Then we simply say that considering the djt from `fromoshape` to `tooshape` should result in new routes joining all members of `predecessors(fromoshape)` to all members of `successors(tooshape)`—excepting, of course, those pairs for which routes have already been found. Entering the new routes in the table is completely straightforward if we remember that routes are always specified in terms of the first oshape encountered on the way. If `oshape1` in `predecessors(fromoshape)` is to be joined to `oshape2` in `successors(tooshape)`, then the first oshape encountered along the new route is simply the first oshape on the existing route from `oshape1` to `fromoshape`. Thus, the procedure for processing a new djt from `fromoshape` to `tooshape` is:

```
procedure merge
begin
        tooshape → table(fromoshape, tooshape)
        for oshape1 in predecessors(fromoshape) do
            for oshape2 in successors(tooshape) do
                if table(oshape1, oshape2) = undefined then
                    table(oshape1, fromoshape) → table(oshape1, oshape2)
                endif
            endfor
        endfor
end
```

It is worth observing (and is a consequence of the theorem 1 below) that, provided djt's are presented in clockwise order, the table entry `table(fromoshape, tooshape)` is undefined at the time when the djt from `fromoshape` to `tooshape` is encountered, so that the first line in the procedure `merge` does not overwrite any existing table entries.

The importance of the procedure `merge` lies in its relationship to criterion 1. We have two results:

Theorem 1 *If the djt's in a given scene are processed in clockwise order through one cycle according to the procedure* `merge`*, then all routes encoded in the table will be sensible.*

Theorem 2 *If the djt's in a given scene containing point regions X and Y are processed in clockwise order through one cycle according to the procedure* `merge`*, starting at the orientation $Y \to X$, then the table entry* $\texttt{table}(X, Y)$ *will be filled.*

Theorem 1 says that only sensible routes are ever found by the algorithm provided we never have to go through more than one cycle. Theorem 2 says that, provided we start the process of scanning for unobstructed djt's at a certain orientation—namely, in the orientation pointing from Y to X—a route, and therefore, by theorem 1, a sensible route, will be found within one cycle. Theorem 1 alternatively states that, provided the djt's are passed in clockwise order, a sensible route from the start to the goal is always completed first: non-sensible routes never get constructed. Our next task is to prove the theorems.

4 Proofs of theorems

Before coming to the proofs proper, we must do a little more work on the relations of clockwise- and anticlockwise-side. Suppose L_1 and L_2 are two successive djt's on a route p involving a common region S. Then the angle between L_1 and L_2 is simply the angle through which the route p turns while it is on the perimeter of S. As observed above, L_1 and L_2 must be either both clockwise or both anticlockwise about S; and this enables us to adopt a clockwise-is-positive convention regarding the angles of turn in a route. If L_1 and L_2 are both clockwise about S, then we can take the angle through which p turns on the perimeter of S to be the angle measured clockwise from L_1 to L_2. If L_1 and L_2 are both anticlockwise about S, then we can take the angle through which p turns on the perimeter of S to be *minus* the angle measured *anti*clockwise from L_1 to L_2. Thus, if p visits S with clockwise orientation, p will have an angle of turn about S lying in the range $(0, 2\pi)$; if p visits S with anticlockwise orientation, p will have an angle of turn about S lying in the range $(-2\pi, 0)$. The total angle through which a route p turns is then simply the algebraic sum of these individual angles of turn. Clearly, this total angle of turn may take any real value.

One last definition involving angles of turn. Suppose a route p visits obstacles S and T in that order. We define the angle of turn of p *between* S and T

to be the algebraic sum of all the individual angles of turn on p *after* p has left S and *before* p arrives at T. That is, we do not count the amount of turn on S and T themselves. If p visits T immediately after visiting S, then the angle of turn of p between S and T is therefore zero.

Now for an important fact.

Lemma 1 *Let p be a route from X to Y which does not cross over itself, and which visits distinct oshapes σ (belonging to obstacle S) and τ (belonging to obstacle T) in that order. (We allow $S = X$, and $T = Y$.) Let p' be a route from σ to τ which does not cross over itself, and whose endpoints A and B lie on the perimeters of S and T, respectively, such that p and p' have exactly the points A and B in common, and such that the section of p between A and B lies on the clockwise side of p'. Fig 8 illustrates the situation in the particular case where σ has anticlockwise orientation and τ clockwise orientation.*
Denote the djt from S in p by L, and the djt from S in p' by L'. Then if θ_0 is the angle measured clockwise from L to L', θ the the angle of turn of p between S and T, and θ' the the angle of turn of p' between S and T, we have:

$$\theta > \theta' + \theta_0.$$

Proof: Denote the last djt in p (i.e. the djt in p running to to T) by L_e, and the last djt in p' (i.e. the djt in p' running to to T) by L'_e. Let θ_e be the angle measured clockwise from L'_e to L_e. Consider the simple circuit p^* formed by p followed by p' in the reverse direction. By hypothesis, the section of p between A and B lies on the clockwise side of p', so that p^* winds clockwise—that is, turns through an angle 2π. By adding up all the angles of turn on this circuit, we obtain the equation:

$$\theta + (\pi - \theta_e) - \theta' + (\pi - \theta_0) = 2\pi.$$

Hence,

$$\theta - \theta_e = \theta' + \theta_0.$$

But if p and p' touch only at A and B, so that L_e and L'_e are not cooriented, $\theta_e > 0$, and the lemma follows. QED.

Recall that, in defining the clockwise order of the djt's in a scene, I suggested that we imagine the scene being rotated anticlockwise, with each djt being associated with the time at which it is horizontal and pointing to the right, and with ties being broken arbitrarily. Of course, the exact times at which djt's are passed to `merge` are inessential: only the order of presentation makes any difference to the outcome of the algorithm. However, for the purposes of proving theorems 1 and 2, it will be useful to suppose that each djt is passed to `merge` precisely when it is horizontal and pointing to the right as the scene is rotated anticlockwise with constant unit angular velocity.

We have the following lemma:

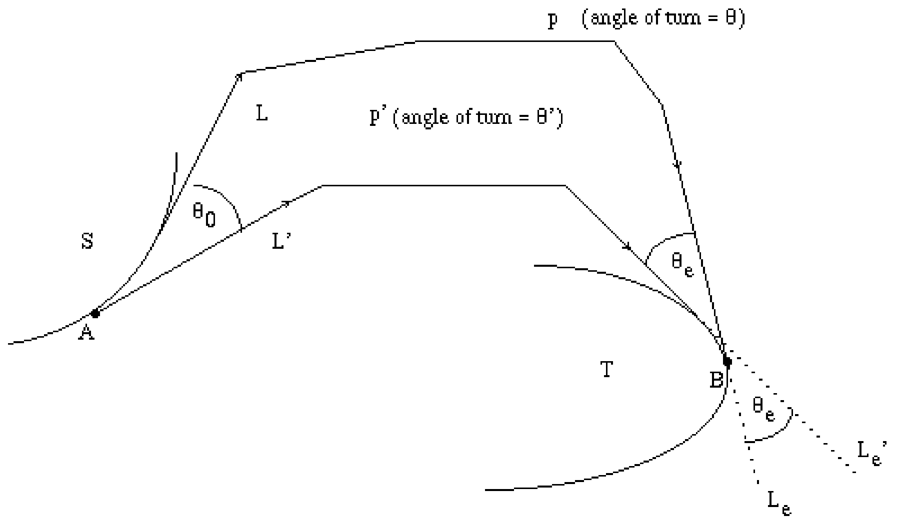

Figure 8: Routes p and p' between A and B

Lemma 2 *Let the unobstructed djt's in a scene be passed to* `merge` *at the times when they are horizontal and pointing to the right, as the scene is rotated anti-clockwise at unit angular velocity. (We assume that the cycle is presented only once.) Let p be a route visiting oshapes σ (belonging to obstacle S) and τ (belonging to obstacle T) in that order, and let L be the first djt after S (i.e. the djt running from S) in p. Suppose that the djt L is passed to* `merge` *at time t_0, and that the angle of turn of p between S and T is θ. Then, if the route p gets entered in* `table` *at all, the entry* `table`(σ, τ) *will get filled no earlier than time $t_0 + \theta$.*

If p is clockwise definite, then if p gets entered into `table`*, the entry* `table`(σ, τ) *will get filled at exactly the time $t_0 + \theta$.*

The above lemma holds because routes are basically constructed by stepping through djt's in clockwise order, and the proof is straightforward. However, two points are worth noting. First, the condition that the cycle of djt's is presented only once is crucial to lemma 2. It is possible, as mentioned above, for routes to wind through an angle $\theta > 2\pi$; and such routes could be entered into the table of directions if the cycle of djt's were repeatedly processed by `merge`. Suppose, then, that the cycle of djt's were passed to `merge` repeatedly, and that p_1 is a route, winding through an angle of more than 2π, which would be entered into the table shortly after the start of the second cycle. Then it might be that, following a djt presented early on in the second cycle, the end of p_1 were joined to the front of a second route, p_2, with a net angle of turn somewhat less than 2π, that had been entered into the table during the first cycle. The route resulting from combining p_1 and p_2, being entered into the

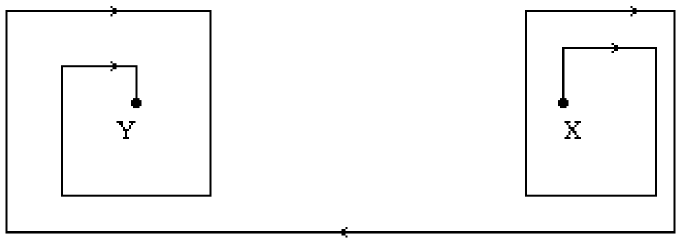

Figure 9: A very curly route (obstacles not shown)

table shortly after time 2π, could have a total angle of turn near to 4π, thus violating the lemma. At the time of writing, the behaviour of the procedure `merge` has not been analysed for multiple presentations of the cycle of djt's. However, theorem 2, which states that we never need present the cycle of djt's more than once, means that this issue may be ignored.

Secondly, we can now see the rationale for the restriction which we built into the procedure `merge` using the functions `predecessors` and `successors`—namely, that new djt's going to clockwise oshapes cannot link onto the front of existing routes and that new djt's going from anticlockwise oshapes cannot link onto the end of existing routes. Absent such a restriction, the very curly route of fig. 9, which has all its djt's either vertical or horizontal, would be entered into the table at time $7\pi/4$, since by then, all the djt's involved would have been passed to `merge`.

Now to theorem 1. To prove that the procedure `merge` can generate only sensible routes, it suffices to show that no route can be generated part of which lies on the clockwise side of a clockwise definite route; the anticlockwise case follows by symmetry.

Suppose, then, that p visits oshapes σ (belonging to obstacle S) and τ (belonging to obstacle T) in that order, and that there exists a clockwise definite route p' from σ to τ with with endpoints A on S and B on T. Suppose in addition that p and p' have only the points A and B in common, that p and p' do not cross over themselves, and that the section of p between A and B lies on the clockwise side of p' (fig. 8). Let L be the djt from S on p, and L' the djt from S on p'. Since p and p' touch only at the points A and B, L and L' must have different orientations. Let the angle measured clockwise from L to L' be θ_0. Then if the djt L is passed to `merge` at t_0, the djt L' can be passed to `merge` no later than $t_0 + \theta_0$. Let the angle of turn of p between S and T be θ, and let the angle of turn of p' between S and T be θ'. By lemma 2, the section of p between S and T could be recorded in the table entry `table`(σ, τ) no earlier than:

$$t_0 + \theta.$$

Also by lemma 2, and making use of the fact that p' is clockwise definite, p' would, unless beaten to it by some other route, cause the table entry `table`(σ, τ) to be filled no later than

$$t_0 + \theta_0 + \theta'.$$

But by lemma 1, since p' lies on the clockwise side of p:

$$\theta > \theta_0 + \theta'$$

In which case, the no-overwriting rule will ensure that p is never entered into the table, because p' would always be found before the portion of p between S and T. QED.

Finally, we come to theorem 2. Consider the straight line segment between start point X and goal point Y in any scene. One way to get from X to Y (it is certainly not the optimal way) is to follow this line, hugging the boundaries of any obstacles that intersect it. For the sake of argument, let us suppose that we pass clockwise around all such obstacles. Following each detour around an obstacle, we return to the line and continue to the next obstacle, or to Y. The resulting journey is shown in fig. 10. Of course, such a journey will not constitute a route as we have been using the term here, for it will not consist of a sequence of djt's interspersed with segments of the boundaries of the obstacles. But we can put this right if we add tiny obstacles at the corners marked C in fig. 10, around which we suppose the route to pass anticlockwise. That way, we can ensure that there is a route (in the proper sense of a sequence of djt's interspersed with segments of the boundaries of the obstacles) arbitrarily close to the route drawn in fig. 10. The djt's in the route in question would have clockwise orientation about all the original obstacles and anticlockwise orientation about all the additional obstacles. If we start scanning for unobstructed djt's at the orientation $Y \to X$, the djt's going *to* the original obstacles will be found (and entered into the table) within time π, the djt's between pairs of successive additional obstacles will then be found at precisely time π, and the djt's *from* the original obstacles to the following additional obstacles will be found at times between π and 2π. The restrictions on linking in new djt's to existing routes embodied in `merge` will allow the resulting route to be constructed in the order just described, unless, of course, another route is found first. Now, it is clear that, assuming the additional obstacles are very tiny, the route just described will not be sensible. By theorem 1, then, it will not be entered into the table. But this can only be because some other sensible route

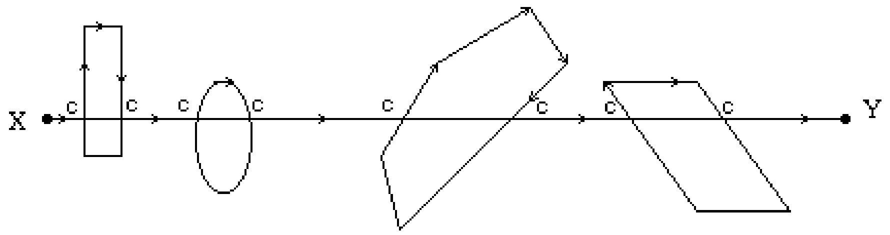

Figure 10: A route involving added obstacles

from X to Y has been found first. Therefore a sensible route p will be found by `merge` within a single cycle, when we process the scene with the additional obstacles. But a sensible route cannot involve any of the additional obstacles. In which case, a sensible route p will be found by `merge` within a single cycle, when we process the original scene. QED.

5 Eliminating loops

As already mentioned, criterion 1 need not rule out as non-sensible a route between X and Y which crosses over itself. An example is shown schematically in fig. 11. Notice that, since p crosses over itself between points A and B, we cannot apply definition 2 and say that that section of p lies on the clockwise side of the clockwise definite route p' from A to B. Indeed, the route p is anticlockwise definite, and so has a negative angle of turn between S and T, and so does not have a greater angle of turn between S and T than does p'. The simplest, but not the most efficient way, to avoid such routes is simply not to enter into the table of routes those routes which cross over themselves (assuming we have is some way of detecting them). Omitting such routes will not compromise theorems 1 and 2. A better way of avoiding looping routes employs the following observation. Consider the route p'^{-1} from B to A defined simply as the reversal of route p' from A to B. It follows from the rules for linking new djt's onto existing routes that p'^{-1} must be found (if it is not

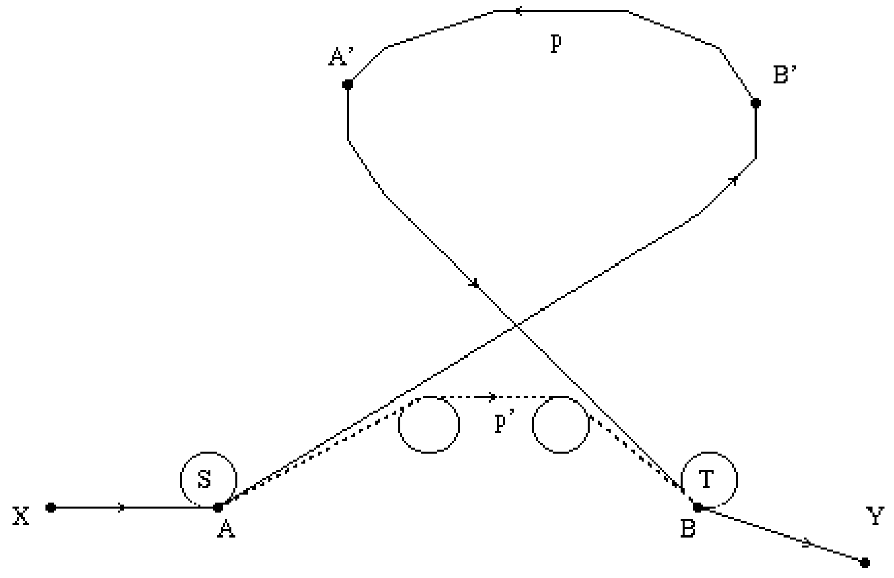

Figure 11: A route which crosses over itself

beaten to it by some other path competing for the same table entry) while the loop on p is being constructed. Specifically, p'^{-1} will be constructed whilst the section of p between A' and B' is being constructed, which will be after the section of p between T and Y has been found. What is then needed to prevent the looping route p from being entered in the table is a modification of the procedure `merge` to permit the route p'^{-1} to be reversed and linked on to the beginning of the existing route from T to Y, so forming a route from S to Y. Then the looping route could not be completed because the table entry $\texttt{table}(T^{-}, Y)$ would already have been filled. The precise details of this modification, however, demand a more thorough analysis than that attempted in this paper.

6 Applications and future work

I mentioned in the introduction three characteristics of the approach to route planning taken here. (i) It finds *sensible* routes rather than optimal ones; (ii) it is uninfluenced by perspective; and (iii) it produces its output in qualitative terms. The first of these characteristics has been explained. By way of conclusion, we now discuss the remaining two, for it is therein that the approach's potential advantages lie.

The criterion of sensibleness on which our approach is based is perspective-invariant. Consider the scene depicted in plan view in fig. 12a). Fig. 12b) shows this scene in perspective as viewed from a point above the plane, and can of course be obtained from fig. 12a) by applying an appropriate projective transformation. Any such perspective projection maps lines to lines and preserves the relation of intersection and the sense of angles. Therefore, the unobstructed djt's in the original scene, fig. 12a), will be mapped to the unobstructed djt's in the transformed scene, fig. 12b), with their orientations about the relevant obstacles preserved. In addition, the properties of clockwise-definiteness and anticlockwise-definiteness are preserved, as are the relations of clockwise-side and anticlockwise-side. Notice that the drawn route, which is not sensible in the original scene, is mapped to a route which is not sensible in the transformed scene either. More generally, it is easy to see that the clockwise order of the unobstructed djt's is preserved by this transformation. The usefulness of this latter fact in certain situations is clear enough: in choosing a sensible route across a floor on which are strewn a collection of relatively flat obstacles—i.e. obstacles whose overall outline is close to their base outlines—there is no point in first computing a plan view of the obstacles we can see: the depth information would only be thrown away again. True, nothing I have said in the above helps deal with the problem of occlusion, which arises when the obstacles in question are not flat. This problem must await separate treatment; but it is a ultimately a problem which any navigating robot must somehow tackle.

Finally, there is the question of the output of the algorithm described here. Once the cell $\texttt{table}(X, Y)$ is filled, the table of directions can be read to obtain a recommended itinerary in terms of a list of oshapes. Such an itinerary might

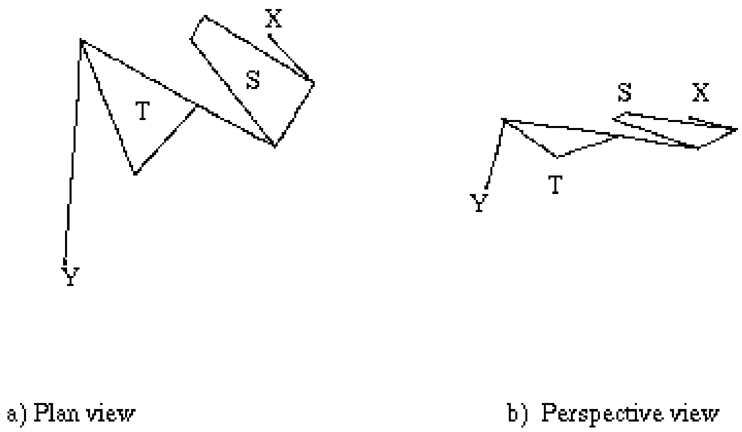

Figure 12: A perspective view of a scene

look like:

$$[X, S^+, T^-, Y],$$

for example. Given only this qualitative information, we cannot draw a plan view of the corresponding route. But, surprisingly, we can follow that route, assuming that we can identify the named objects in our field of view. Imagine yourself initially situated at point X, and given the instruction to follow the route specified in the above itinerary. You turn to face the obstacle S. You must pass clockwise around it. But that just means that you should pass just to the left of it. So if you face directly at the left-most visible point of S, and head off in a straight line, you will be on the right tack. When you come close to S, then you must hug its boundary. (You will find yourself following that boundary in a clockwise manner). Since the next obstacle to visit is T with anticlockwise orientation, you should keep going around S until the right hand limit of T comes into view directly ahead. When it does so, you know you are facing along the djt from S^+ to T^-, so you must head off in a straight line until you close on T. When you do so, you must again hug its boundary (proceeding anticlockwise) until the goal, Y, comes into view, at which point you head straight for Y.

Thus, the above itinerary can be transformed into the program:

`Face` left `limit of` S
`Go straight until rendezvous with` S
`Hug boundary until` right `limit of` T `visible`

```
Go straight until rendezvous with T
Hug boundary until Y visible
Go straight until rendezvous with Y
Stop
```

It is clear that the generation of the program from the itinerary is completely mechanical. Moreover, such a program constitutes a sequence of commands which a robot equipped with a reasonable array of sensor and effector mechanisms could execute. All the commands are of the form `do` x `until` y, where x denotes a relatively straightforward action and y a perceivable condition. True, included in the robot's supposed repertoire of actions are such non-trivial and interactive matters as boundary-hugging and straight-line navigation. True also, the robot is supposed to be able to perceive the margins of objects, and to be able to tell when it is near an object that it is moving towards. But these are things which robots can do to a greater or lesser extent already, and which we would certainly want a robot to be able to do as well as possible. Thus, the specification of a sensible route in terms of the objects it visits and the orientations with which it visits them is well-suited to the generation of commands which can guide a robot along that route. There is no need to obtain any coordinate specifications of the route's critical points.[6]

References

[1] Chongstitvatana, Prabhas and Conkie, Alistair: "Behaviour Based Assembly Experiments", *Technical Report, Dept. of Artificial Intelligence, University of Edinburgh.*

[2] Conkie, Alastair and Chongstitvatana, Prabhas: "An Uncalibrated Stereo Visual Servo System", *Technical Report no. 475, Dept. of Artificial Intelligence, University of Edinburgh*, 1990.

[3] Lozano-Pérez, T. and M.A. Wesley: "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", *Communications of the Association for Computing Machinery*, vol.22, **10** (1979), pp.560–570.

[4] Schwartz, J.T. and Sharir, M: "A survey of motion planning and related geometric algorithms", *Artificial Intelligence*, **37** (1988), pp.157–169.

[5] Weltzl, Emo: "Constructing the visibility graph for n-line segments in $O(n^2)$ time", *Information Processing Letters*, vol. 20, no. 4 (1985), pp.167–171.

[6] I am indebted to my colleague, Dr. Behnam Banieqbal, for helpful criticism of an earlier draft of this paper.