Algorithmica (1993) 10: 41–63

Algorithmica
© 1993 Springer-Verlag New York Inc.

Planning a Time-Minimal Motion Among Moving Obstacles[1]

Kikuo Fujimura[2] and Hanan Samet[3]

Abstract. Motion planning for a point robot is studied in a time-varying environment. Each obstacle is a convex polygon that moves in a fixed direction at a constant speed. The point to be reached (referred to as the destination point) also moves along a known trajectory. The concept of "accessibility" from a point to a moving object is introduced, and is used to define a graph on a set of moving obstacles. If the point robot is able to move faster than any of the obstacles, then the graph exhibits an important property: a time-minimal motion is given as a sequence of edges in the graph. An algorithm is described for generating a time-minimal motion and its execution time is analyzed.

Key Words. Time-varying environments, Visibility graphs, Accessibility graphs, Motion planning.

1. Introduction. The motion-planning problem is one of finding a path that connects given start and destination points in an environment that contains a predefined set of obstacles so that the path is collision-free. The visibility graph [As], [Gh] has been an important combinatorial structure in planning paths among stationary polygonal obstacles [Lo], as well as in acquiring information about the environment while exploring it in two dimensions [Oo]. An important property of the visibility graph is that a shortest path is a finite sequence of edges of the graph. For surveys of research on motion planning, see [Wh] and [Ya].

In this paper we study the concept of *accessibility*, which is a generalization of the concept of visibility [Fu]. We make use of accessibility to represent moving objects for the purpose of planning the motion of a robot. The robot is assumed to move in a two-dimensional world in which polygonal obstacles, as well as the destination point, are in motion. The motions of the polygonal obstacles as well as the destination point are assumed to be known. The *accessibility graph* is a generalization of the visibility graph. Paths to the destination point are found as sequences of edges of the graph. In fact, when all the obstacles have zero velocity,

[1] The support of the National Science Foundation under Grants IRI-88-02457 and IRI-90-17393 is gratefully acknowledged. This work is supported in part by the Office of Engineering Research Program, Basic Energy Sciences, of the U.S. Department of Energy, under Contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. and in part by an appointment to the U.S. Department of Energy Postgraduate Research Program administered by Oak Ridge Associated Universities.
[2] Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210, USA.
[3] Computer Science Department and Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742-3411, USA.

Received October 11, 1989; revised November 17, 1990, and October 5, 1991. Communicated by David P. Dobkin.

the accessibility graph becomes the visibility graph of these polygonal obstacles. Moreover, a time-minimal motion is represented as a sequence of edges of the accessibility graph.

Previous work dealing with moving obstacles includes the following: Reif and Sharir [Re] show that motion planning in a three-dimensional environment containing moving obstacles is PSPACE-hard given bounds on the robot's velocity, and NP-hard without such bounds. Canny and Reif [Ca] show that motion planning for a point in the plane with bounded velocity is NP-hard, even when the moving obstacles are convex polygons moving at constant linear velocity without rotation. These results indicate that motion planning with moving obstacles is computationally harder than motion planning with stationary obstacles. Nevertheless, there are some approaches to solving the problem. These approaches are successful in limited domains.

Kant and Zucker [Kan] decompose the problem of motion planning with moving obstacles into two parts. In the first part they plan a path among the stationary obstacles while ignoring the moving obstacles. In the second part a graph is used to define regions through which the robot may not pass when following the path computed in the first part. The positions of these regions influence the choice of the speed. Erdmann and Lozano-Perez [Er] make use of stacks (i.e., piles) of two-dimensional configuration spaces. These spaces are created each time some object changes its velocity. A path consists of a sequence of vertex-to-vertex transitions between two adjacent elements of the piles. None of these approaches deals with the case where the destination point can also be in motion.

In this paper we study the case in which the robot can move faster than any of the obstacles. This condition is the key that makes our problem tractable. Given a velocity (i.e., a speed and a direction in which to proceed), a robot can travel a certain distance before it meets with some obstacles (or it may never meet any obstacles). Let us call such a meeting point, if any, the collision point with respect to the velocity. The collision points around the start point, with respect to a certain speed, can be decomposed into subsets that we call *collision fronts*. They consist of a number of curves or line segments. Having computed collision fronts at the start point, the robot moves to an endpoint of one of the collision fronts, say E. At E the robot is coincident with one of the vertices of the obstacles. Next we compute another collision front with E as a start point. Now the robot moves from E to one of the endpoints of the collision fronts that are generated about E. We repeat this process until we reach the destination point.

Dealing with moving obstacles and a moving destination point is of current interest in robotics. The ability to avoid moving obstacles leads to an increase in the mobility of a robot for navigation. It also results in higher productivity for factory manipulators. By allowing the destination point to move, we are able to consider a larger class of applications. For example, suppose that a robot arm must pick up an object that lies on a conveyor belt. As another example, consider an autonomous vehicle whose goal is to catch up with another vehicle that is in motion.

In a stationary environment a path is specified as a subset in the plane and the term "trajectory" is usually used to refer to velocity and acceleration information along a path. In a dynamic environment a collision-free path for a particular time period may not be collision-free for another time period. Therefore, a path must also be specified as a function of time. In this paper we use the term "motion" to mean a path with timing information, while the term "path" is used to specify a curve in space without timing information. Throughout our paper a goal point in motion is called a *destination point*, and the term *point robot* is used to refer to a point to be moved from the start point to the destination point. Note that we use the term "velocity" to include both the speed and direction of a motion, while we use the term "speed" to mean the magnitude of a motion.

The rest of this paper is organized as follows. In Section 2 we define the motions that obstacles are permitted to have, and introduce the concept of accessibility. Section 3 describes the procedure for finding a path and its time minimality. Section 4 contains an analysis of the execution time of the algorithm. Section 5 contains a few concluding remarks.

2. The Accessibility Graph.

Throughout this paper O, G, and R are used to denote the start point, the destination point, and the point robot, respectively. First we define the motions of the obstacles, the point robot, and the destination point.

MOTION OF AN OBSTACLE. An obstacle is a convex polygon that moves in a fixed direction at a constant speed. We call such a straight motion a *movement*. To simplify the explanation we treat the line segments (edges) that constitute polygons as the basic units of our discussion. A *movement* is defined by a tuple (L, d_L, v_L) that represents the motion of line segment L in direction d_L at speed v_L. We consider an environment that contains a finite set of movements,

$$M = \{M_1, M_2, \ldots, M_n\},$$

where each M_i represents a movement as defined above. Note that M corresponds to the motions of all the edges in the environment, not just those in one polygon. Hence, if a polygon P_i consists of l_i edges (thus vertices), we have $n = l_1 + l_2 + \cdots + l_k$, where k is the number of polygons in the environment.

MOTION OF A POINT ROBOT. After leaving the start point at the start time, R can have any motion as long as its speed does not exceed a given maximum speed and does not pass through the interiors of obstacles. We assume that the maximum speed of R is greater than that of any of the obstacles, and is also greater than that of the destination point.

Motion of a Destination Point. The motion of the destination point consists of a finite series of steps. Within a step, the destination point moves in a constant direction at a constant speed. We assume that the number of steps is bounded.[4]

Now let us define the concepts of accessibility, collision front, accessible vertex set, and accessibility graph.

Accessibility. Consider a set of movements $M = \{M_1, M_2, \ldots, M_n\}$ and G, the destination point. Let R be a point robot located initially at O at time t_0. Suppose that R starts moving at time t_0 at a speed v. After R starts moving, it moves in a fixed direction. A point V(V is either the destination point or a point on a polygonal obstacle) is said to be *accessible* from O if there exists a direction of the motion of R such that R meets V without prior interception by any other movement. We say that V and R *meet* if there exists a location X through which both V and R pass at the same time t, where $t_0 < t$. The location X is called an *accessible point* of V. The time t is called the *accessible time* of X with respect to V and is denoted by $t(X)$. Observe that the accessible point of a particular point V varies for different values of the speed and the initial location of R. Also note that if a stationary point V (V is either a point on a stationary obstacle or a stationary destination point) is accessible, then its accessible point is V itself.

Collision Front. Consider an environment that contains one movement, say (L, d_L, v_L). Let V_a and V_b be the two endpoints of L, and let P_a and P_b be accessible points corresponding to V_a and V_b, with respect to R's initial location O, start time t_0, and speed v. The set of accessible points corresponding to all points in L forms a segment. As is shown later in Section 4, this segment takes the form of either a straight line or a quadratic curve. We call this segment a *collision front* of L (with respect to O, t_0, and v). Figure 1 contains an example of the collision front. It can be shown that P_a and P_b are the two endpoints of the collision front. For an environment that contains more than one movement, there can be more than one collision front, each of which corresponds to some movement. In this case,

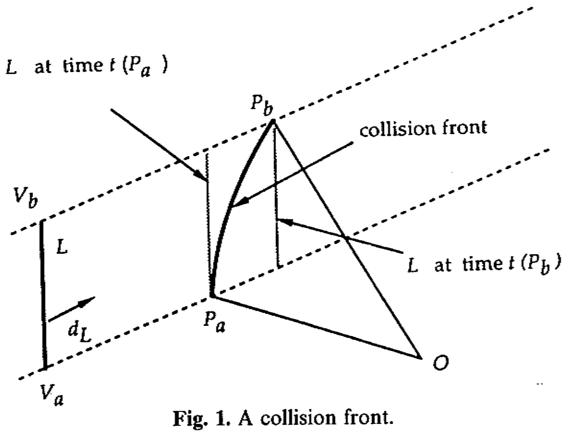


Fig. 1. A collision front.

[4] This condition can be relaxed. See Section 5 for more details.

however, it is possible that only a part of an obstacle is accessible. Notice that if two moving obstacles (e.g., edges) do not collide, then the corresponding collision fronts do not intersect. The characterization of the collision front is as follows. Suppose that R departs O at time t_0 and keeps moving at a constant speed v along a ray, say l, emanating from O. The point robot R does not meet any of the obstacles in motion if and only if l does not intersect any of the collision fronts. The collision front of a stationary edge L is L itself or consists of subsets of L.

Accessible Vertex Set. Let VS be the set consisting of the destination point and the vertices of the polygonal obstacles in motion M, and let O, t_0, v be as in the definition of accessibility. The set of accessible points corresponding to elements of VS with respect to O, t_0, v is called the *accessible vertex set* and denoted as $AVS(M, O, t_0, v)$. Since some vertices in VS may not be accessible from O, the size of AVS is at most $|VS|$.

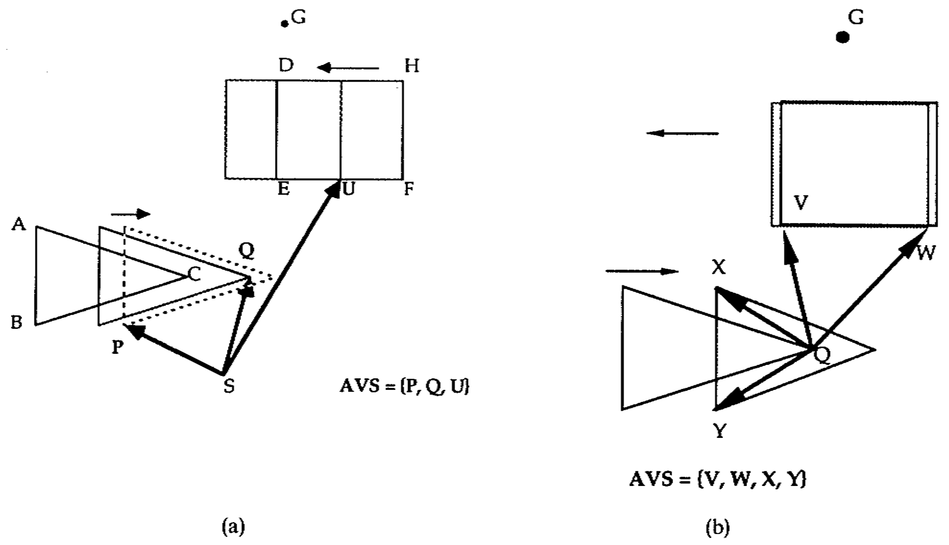
Accessibility Graph. Let O be the start point (i.e., the point at which point robot R is initially found), let t_0 be the start time, let v be the start speed, and let G be the destination point. With each accessible point X corresponding to a point V, we also associate a time value (i.e., timestamp), say $t(X)$, to denote X's accessible time with respect to V. We define a directed graph called the *accessibility graph*, or $AG(M, O, G, t_0, v)$, by the following construction:

1. Insert O in the vertex set of AG, and set its default accessible time to t_0.
2. For every newly added vertex U in the set of vertices in AG, consider the accessible vertex set with U as the initial point, i.e., $AVS(M, U, t(U), v)$. Insert the elements of this AVS in the vertex set of AG, and the edges from U to these points in the edge set of AG.

The graph is a (directed) tree[5] rooted at O. Note that for a given set of M, O, and G, AG varies for different values of t_0 and v. This graph is infinite according to our definition. However, we can make it finite by adopting the rule that when a vertex of an input polygon is used more than once to define vertices of AG, then the instance with the youngest (i.e., earliest) accessible time is the one that is retained. Also note that the vertices of AG are the accessible points and not the original points on the obstacles.

Figures 2(a) and (b) demonstrate how to construct an accessibility graph. Points S and G are the starting and destination points, respectively. Triangle ABC and rectangle $DEFH$ are obstacles moving in the indicated directions. First, S is inserted in the vertex set of the accessibility graph (step 1). Vertices B, C, and F are accessible from S at P, Q, and U, respectively, but E is not accessible because obstacle ABC is in the way. Therefore, $AVS(M, S, t(S), v) = \{P, Q, U\}$. Points P, Q, and U are inserted in the vertex set of the accessibility graph. Accordingly, SP, SQ, and SU are inserted in the edge set of the accessibility graph (step 2). Next,

[5] In degenerate cases the graph might not be a tree but only a DAG (e.g., when there are two shortest paths to a vertex.)



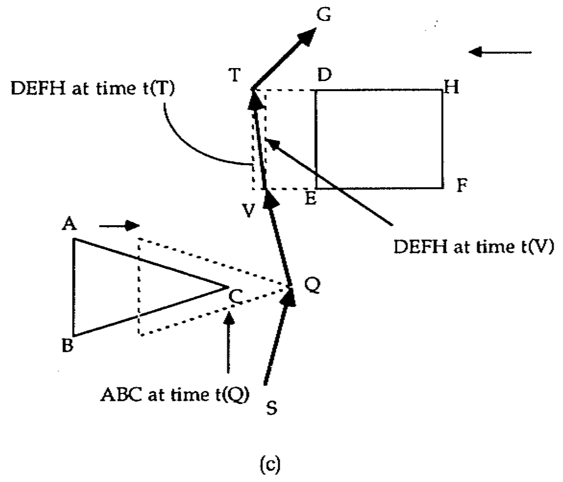


Fig. 2. Construction of an accessibility graph. (a) Insert the start point in the vertex set of *AG*. (b) The *AVS* with *Q* as the start point. Points *Y* and *W* will not be included in the accessibility graph when the finite version of the definition of the accessibility graph is used. (c) The result of applying procedure *PlanMotion* to (a).

let us take Q as a newly added vertex in step 2 of the construction (Figure 2(b)). Vertices A, B, E, and F of the obstacles are accessible at locations X, Y, V, and W, respectively, i.e., $AVS(M, Q, t(Q), v) = \{X, Y, V, W\}$. Thus, X, Y, V, and W are inserted in the vertex set of the graph, and QX, QY, QV, and QW are inserted in the edge set of the graph. This process is applied to other vertices such as P, U, etc.

Note that according to the finite definition of the accessibility graph, points Y and W (Figure 2(b)) will not appear in the graph since P and U (the accessible points of B and F from S, respectively, as can be seen in Figure 2(a)) have younger accessible times when they are reached from S.

The accessibility graph is related to the visibility graph in the following manner. Consider a stationary set of movements M whose elements do not overlap. Construct the (infinite) accessibility graph for M, and make all directed edges in AG undirected and remove the timestamps at all the vertices. The resulting graph is the visibility graph for M. Since we will need only the finite part of the accessibility graph in the subsequent sections, we retain its finite formulation.

3. Time-Minimal Motion.

In this section we first describe a procedure that finds a motion to the destination point using AG as defined in the previous section. Next we prove that the motion found by the procedure is time-minimal. In this process we use a priority queue of vertices where the accessible time of each vertex serves as the vertex's priority.

Procedure *PlanMotion*:

1. Insert the starting vertex O into the queue.
2. Remove a vertex, say V, from the queue whose associated accessible time is the youngest.
3. If vertex V in step 2 is the destination point G, then report the motion and exit; otherwise, insert all the vertices that are adjacent to V in AG into the queue, and repeat steps 2 and 3.

Step 3 needs some care. An element of AVS is inserted into the queue when the corresponding vertex has not been seen before or it has a younger accessible time than the one (for the same corresponding vertex) that is already in the queue in which case the old element is deleted. When a new element has an older accessible time than the one for the same vertex that is already in the queue, the new element is not added. As a result, the size of the queue never exceeds the sum of the total number of vertices of the input polygons plus the destination point. By constructing the graph in this way, we obtain the finite version of the accessibility graph defined in Section 2.

Figure 2(c) is an example of planning a motion using procedure *PlanMotion*. Points S and G are the start and destination points, respectively. Triangle ABC is an obstacle moving toward the right, and rectangle $DEFH$ is an obstacle moving toward the left. Points Q, V, and T are the accessible points of C from S, of E from Q, and of D from V, respectively. The dashed objects show the locations of the corresponding obstacles at the indicated times. Note that while the robot is moving between V and T, it is coincident with some point on edge DE of obstacle $DEFH$. The sequence of line segments (SQ, QV, VT, TG) constitutes the final motion.

In the rest of this section, let $v_{\max}$ denote R's maximum speed, and let O and G be the start and destination points, respectively. We subdivide the motions from O to G into two groups, i.e.:

Group 1. Motions whose initial segments (i.e., start at O) are traversed at speed $v_{\max}$ in a constant direction through a point in $AVS(M, O, t_0, v_{\max})$, say P. The motion from P is arbitrary but must eventually reach G.

Group 2. The other motions.

It is assumed that speeds along motions in both groups do not exceed $v_{\max}$. In the rest of this paper we assume the following two conditions:

1. The maximum speed of the point robot $v_{\max}$ is greater than the speed of any of the obstacles.
2. The obstacles never collide with each other nor do they collide with the destination point.

PROPOSITION 1. *Given a set of movements M and a destination point G, suppose that point R and vertex V, the common endpoint of edges UV and VW of the same polygonal obstacle, are at location X at time t_1. Let M_i and M_j be the movements of edges UV and VW, respectively, and let M_k be the movement corresponding to the motion of the new edge UW. If there exists a motion starting from X at time t_1 and terminating at G at time t_2 $(>t_1)$ in $M - \{M_i, M_j\} + \{M_k\}$ (i.e., an environment formed by removing the triangle UVW from the polygonal obstacle, but leaving edge UW), then there exists a motion in M, starting from X at time t_1 and terminating at G at time t_2.*

PROOF. Let α be a motion from X to G in $M - \{M_i, M_j\} + \{M_k\}$. There are two cases depending on whether or not α collides with the interior of the moving triangle UVW.

Case 1: Motion α Does Not Collide with the Triangle Interior. By construction, α is collision-free in M.

Case 2: Motion α Collides with the Triangle Interior. We construct an alternative motion β such that β is collision-free in M in the following manner. Let Y be the location of the occurrence of the last collision of α with the moving triangle UVW (see Figure 3). Without loss of generality, let L denote the movement (i.e., M_i or M_j) with which α collides last. (Point Y can coincide with an endpoint of L.) Starting at time t_1, define β as the straight-line motion followed by R from X to Y with a constant speed such that β reaches Y exactly at the same time as α reaches Y for the last time. This means that until reaching Y, R is always coincident with some point on L. The length of this motion is shorter than the corresponding segment of α. To see this, note that the shortest distance between two points is a straight line. After Y, R follows the remaining part of motion α. The point R reaches G at the same time regardless of whether motion α or β is used. Along β, R moves at a speed slower than $v_{\max}$ from X to Y. To see this, consider two motions from X to Y, namely, α and β, both starting and arriving at the same time. Since β is shorter than α, the speed of R between X and Y must be less than $v_{\max}$. □

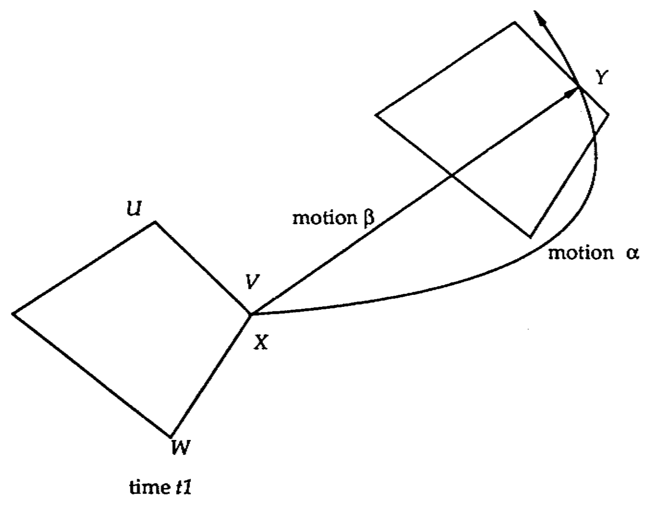


Fig. 3. Illustration of Proposition 1.

As a special case of this proposition, we consider a case in which an obstacle is just a single edge. We have the following proposition; its proof is similar to the proof of Proposition 1.

PROPOSITION 1′. *Assume the same environment as in Proposition* 1 *except that* UV*, with movement* M_i*, is a single edge obstacle. Suppose that at time* t_1*, point* R *and vertex* V *are at location* X*. If there exists a motion starting from* X *at time* t_1 *and terminating at* G *at time* t_2 *in* $M - \{M_i\}$*, then there exists a motion in* M *starting from* X *at time* t_1 *and terminating at* G *at time* t_2.

PROPOSITION 2. *Let* σ_1 *be a motion of a point robot, say* R_1*, consisting of two linear movements. Let the first movement be from* O *to* A *at a constant speed* v_1 *along a line segment* OA *and let the second movement be from* A *to infinity at a constant speed* v_2*. Let* $v_1 < v_{\max}$ *and* $v_2 < v_{\max}$*. For any point* P *on* σ_1*, there exists a motion of a point robot, say* R_2*, from* O *to* P *such that* R_2 *leaves* O *at the same time as* R_1*,* R_2 *moves along a straight line* OP *at a speed* v' ($v' < v_{\max}$)*, and* R_2 *arrives at* P *either before* R_1 *or at the same time.*

PROOF. Choose $v' = \max\{v_1, v_2\}$. This proposition holds by applying the triangle inequality to OAP (Figure 4). □

PROPOSITION 3. *Suppose that* G *is not accessible from* O*. Let* π_2 *be a collision-free motion from* O *to* G*. Let* π_2 *be in Group* 2*. There exists a collision-free motion from* O *to* G *in Group* 1*, say* π_1*, that is identical to* π_2 *starting at some point* J*. The speed along some portion of* π_1 *before* J *is less than* $v_{\max}$.

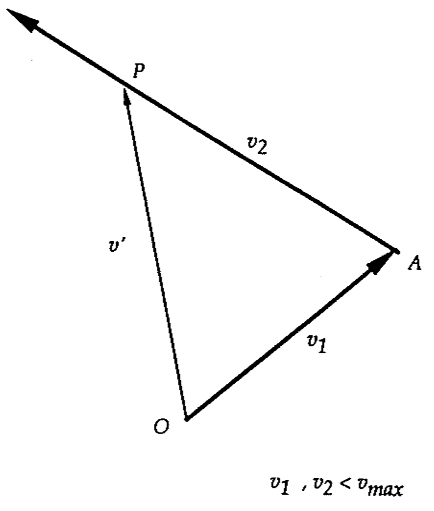


Fig. 4. Illustration of Proposition 2.

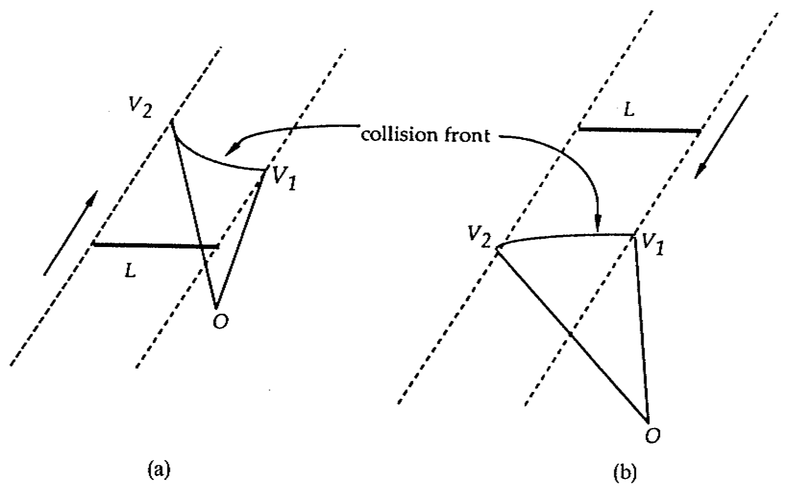
Proof. In the first part of the proof let us assume that there is only one collision front about O for $v = v_{\max}$. Let V_1 and V_2 denote the two endpoints of the collision front. Let v_o denote the maximum speed of the obstacles in the scene.

The outline of the proof is as follows: as candidates for π_1 in Proposition 3, we construct two motions, τ_1 and τ_2, that start with segments OV_1 and OV_2, respectively, each moving at speed $v_{\max}$. Next, each of them moves at a speed infinitesimally less than $v_{\max}$ (i.e., $v_{\max} - \varepsilon$ for a small ε; the value of ε remains to be determined) and heads for the earliest point J on π_2 that they can reach.

In the following we show that it is possible to choose one of τ_1 or τ_2 as π_1. To carry out the proof, we classify the collision front into three types: *receding obstacle*, *proceeding obstacle*, and *degenerate obstacle*, as defined below. To see this classification, consider a point robot that moves straight from O to a point in the collision front V_1V_2 at speed $v_{\max}$, and let the point robot stand still at that spot for a brief moment, while the obstacle continues moving.

Type 1 (receding obstacle): The point robot will not collide with the obstacle.
Type 2 (proceeding obstacle): The point robot will collide with the obstacle.
Type 3 (degenerate obstacle): Until the point robot stops moving, it is always collinear with the obstacle.

Figures 5(a)–(c) contain examples of a receding obstacle, a proceeding obstacle, and a degenerate obstacle, respectively. The type can be determined by the relation between the initial location of the obstacle (line segment) and O. Let l be the line in motion that is the extension of the obstacle. When l intersects O after it starts moving, l is a proceeding obstacle (type 2). When it does not, it is a receding obstacle (type 1). When O is on l, it is of type 3.



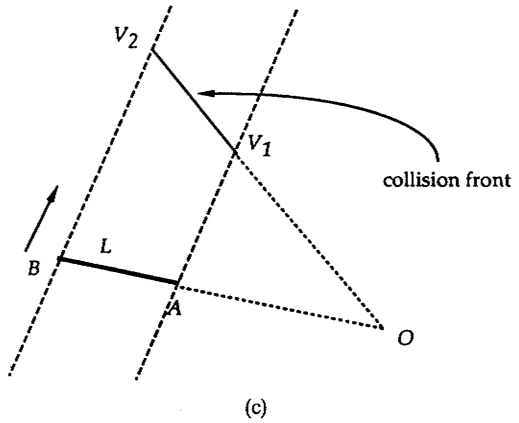


Fig. 5. Three types of obstacles: (a) receding, (b) proceeding, and (c) degenerate.

Case 1: Receding Obstacle.

Proof. We consider two auxiliary motions, σ_1 and σ_2, starting from V_1 at time $t(V_1)$ and from V_2 at time $t(V_2)$, respectively (see Figure 6(a)). Motions σ_1 and σ_2 move at constant speeds v_{c1} and v_{c2}, respectively, where $v_{ci} < v_{\max}$ ($i = 1, 2$), and they are always coincident with some point in L. Let W be the point at which σ_1 and σ_2 meet (Figure 6(a)). Choose v_{ci} so that the ray OW crosses the collision front. Let l_1 and l_2 be the two rays emanating from V_1 and V_2 that are extensions of rays OV_1 and OV_2, respectively. An open-ended serial chain of line segments l_1, V_1W, WV_2, and l_2 partitions the plane into two regions. Let A be the region containing O and let B be its complement.

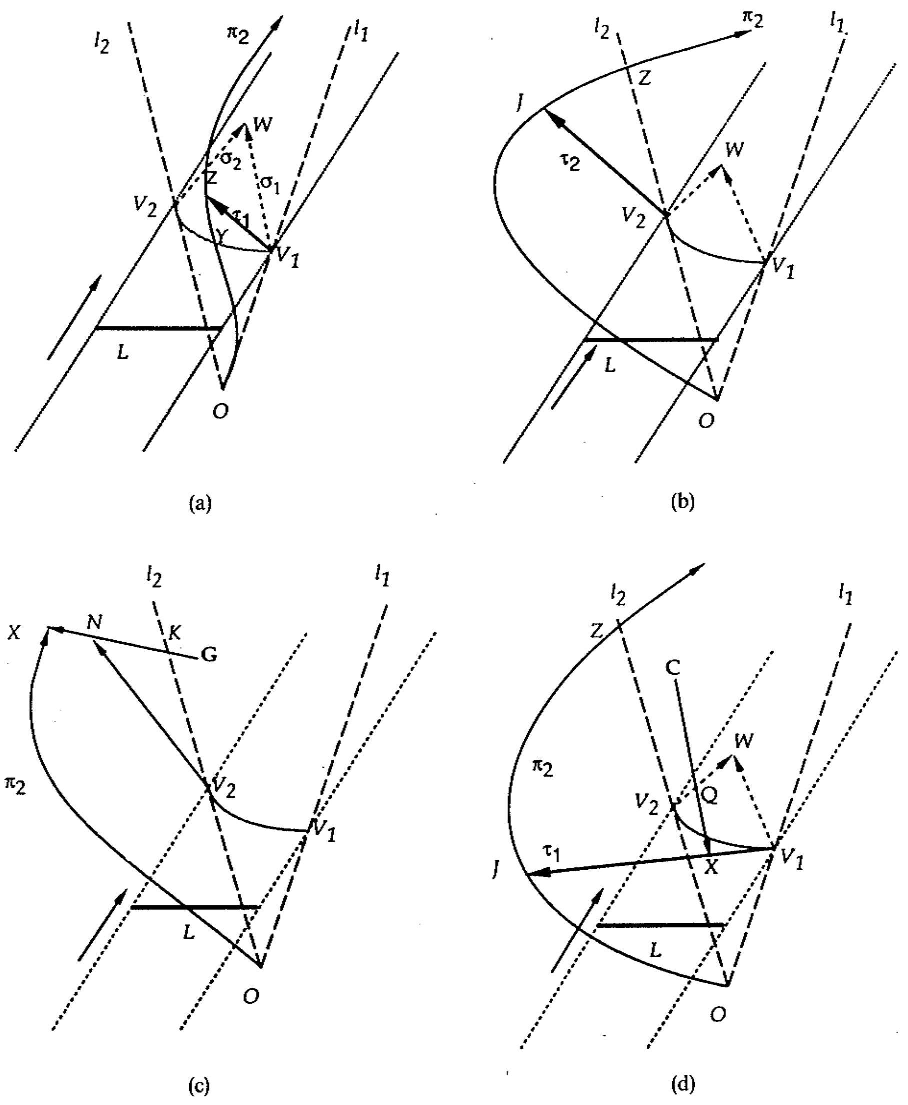


Fig. 6. A receding obstacle for the proof of Proposition 3.

First we show that if there are no moving obstacles besides L, then one of τ_1 or τ_2 will reach π_2 in region A. There are two cases depending on whether or not π_2 crosses the border of the two regions. In the first case we assume that π_2 crosses the border of the two regions and let Z be the first point at which such a crossing occurs. We show that either τ_1 or τ_2 can reach π_2 before π_2 enters region B (i.e., J occurs at or before Z along π_2).

Suppose that Z lies on WV_1 or WV_2 (see Figure 6(a)). Let Y be the point at which π_2 crosses the collision front V_1V_2. Motion π_2 goes through Y at the same time as or after L has gone through Y, because Y is an accessible point and π_2 is a Group 2 motion. Since it is impossible for π_2 to penetrate L between Y and Z, π_2 also goes through Z at the same time as or after L has gone through Z. Now, recall that segment V_1W (or V_2W) is a trajectory for a point robot whose motion is always coincident with some point in L. Therefore, the point robot goes through Z at the same time as or earlier than π_2 goes through Z. This implies that ε can be chosen so that one of τ_1 or τ_2 will reach π_2 in region A. For the case that Z lies on l_1 (or l_2 as in Figure 6(b)), it should be clear that, by choosing ε to be sufficiently small, τ_1 (or τ_2) with a speed $v_{\max} - \varepsilon$ will reach π_2 in region A.

In the second case we assume that π_2 does not cross the border of the two regions. If π_2 reaches the destination point without crossing the border, then either π_1 (one of τ_1 or τ_2 depending on whether we are departing from V_1 or V_2) can reach the destination point before π_2 does, or π_1 can reach π_2 before π_2 reaches the destination point. This can be shown as follows.[6] Let K be the point at which G crosses the border (Figure 6(c)). Let K lie on l_2 as in Figure 6(c) (proofs for the other cases are analogous). Let X be the point at which π_2 meets G. Consider a point robot that departs V_2 at time $t(V_2)$ and moves straight to X at speed $v_{\max} - \varepsilon$. If the point robot arrives at X before π_2, then we are done since it implies that there is a motion from V_2 at $v_{\max} - \varepsilon$ that reaches π_2 before π_2 reaches X. Otherwise, we can show that there exists a location (say N) between K and X at which π_1 can meet G as follows. Consider a motion that departs V_2 at time $t(V_2)$ with a speed $v_{\max} - \varepsilon$. This motion is directed toward K or X. When this motion is aimed toward K, it can reach K before G reaches K. When this motion is directed toward X, it reaches X only after G has reached X. This means that there exists a point, say N, between K and X having the property that if the motion is directed toward N, then it will reach N just as G passes N. We choose this motion as τ_2 (a candidate for π_1).

Now we must show that one of τ_1 or τ_2 is indeed collision-free, even when there are other obstacles present besides L.

At this point let us describe how we choose a speed along τ_1 or τ_2 (i.e., the choice of $v_{\max} - \varepsilon$). Consider all obstacles that exit from region B into region A and that possibly intercept τ_1 or τ_2. Since V_1V_2 is the only collision front in the scene, the motions of the obstacles must cross l_1, l_2, V_1W, or V_2W at some point. Moreover, the crossing times have the following characteristic. For any obstacle that possibly intercepts τ_1 or τ_2, let Q be the last crossing point between regions A and B, before it intercepts τ_1 or τ_2. When Q lies on l_1 (or l_2), the crossing time must be strictly later than the time at which a point that moves along l_1 (or l_2) at $v_{\max}$ departing O at time t_0 would reach Q. Otherwise, it contradicts our assumption that there is only one collision front. When Q lies on V_1W (or V_2W), the crossing time must be strictly later than the time at which motion σ_1 (or σ_2) passes Q, since the obstacles cannot collide (in this case this would have required the obstacle to collide with L).

This implies that we can always construct a motion[7] from V_1 (or V_2) at a speed

[6] The following proof does not require that the robot move faster than the destination point.

[7] This motion itself may or may not be collision-free. However, it does not affect our proof.

less than $v_{\max}$ that reaches any of those obstacles when it crosses l_1 or $V_1W(l_2$ or $V_2W)$. Let v_d be the speed of the fastest of all such motions. Let $v_{\max} - \varepsilon = \max\{v_o, v_{c1}, v_{c2}, v_d\}$. Recall that v_o is the maximum speed of the obstacles in the scene. Then applying Proposition 2 (letting the appropriate one of V_1 and V_2 correspond to O, and letting Q correspond to A), τ_1 will not collide with any obstacle crossing l_1 or V_1W, and τ_2 will not collide with any obstacle crossing l_2 or V_2W. For π_1 of Proposition 3, choose the one of τ_1 or τ_2 that reaches π_2 earlier. We prove that motion π_1, chosen as such, is collision-free. Without loss of generality, let τ_1 be chosen as π_1. By construction, τ_1 does not collide with any obstacle that crosses l_1 or V_1W. Now, suppose that τ_1 collides with an obstacle crossing l_2 or V_2W. Figure 6(d) shows such a scene, where V_1 and V_2 are the accessible points of P_1 and P_2, respectively. Let C be an obstacle that intercepts motion τ_1 at X. Let Q be a point at which C crosses V_2W. Note that the time at which C passes Q must be strictly later than the time at which motion σ_2 passes Q. Applying Proposition 2 to V_2QX (letting V_2 and Q correspond to O and A, respectively) as well as to V_2XJ (letting V_2 and X correspond to O and A, respectively), it can be shown that there is a motion that will reach J before τ_1. This implies that τ_2 can reach π_2 earlier than τ_1. This contradicts the choice of π_1. (The same argument works if Q is on l_2).

Case 2: Proceeding Obstacle.

Proof. The proof for this case is analogous to the proof for receding obstacles.

This time the plane is divided into two regions by l_1, l_2, and the collision front. Let A be the region containing O and let B be its complement. Let Y be the first point at which π_2 crosses the border. We consider only the case in which Y lies on the collision front. The other cases (i.e., the case in which Y lies on either l_1 or l_2 and the case in which π_2 reaches the destination point without crossing the border) are the same as the corresponding parts of the proof for receding obstacles.

We consider motions σ_1 and σ_2 at a speed v_c, where $v_o < v_c < v_{\max}$. Let W be the point at which σ_1 and σ_2 meet. It should be clear that π_2 must cross either V_1W or V_2W (as in Figure 7(a)) before π_2 reaches Y. Note that when the trajectory of L contains O (Figure 7(b)), it is impossible for π_2 to reach the destination point without placing itself outside L's trajectory (even just momentarily), since π_2 cannot penetrate L.

Let Z be the point at which π_2 crosses V_1W or V_2W for the last time before π_2 passes Y. Note that portion ZY of π_2 is entirely inside the trajectory swept by L, and that π_2 passes Y after L has passed it. Motion π_2 must pass Z only after L has passed Z, since π_2 cannot penetrate L. This means that π_2 passes Z later than the time at which σ_1 or σ_2 passes Z. For the same reason as in the case of receding obstacles, τ_1 departing from V_1 or τ_2 departing from V_2 can reach π_2 before π_2 reaches Z. As π_1, pick one of τ_1 or τ_2 that reaches π_2 earlier. Motion π_1 is shown to be collision-free as in the case of a receding obstacle.

Case 3: Degenerate Obstacle. In this case the collision front is collinear with O. The destination point G must be on the extension of the collision front (otherwise,

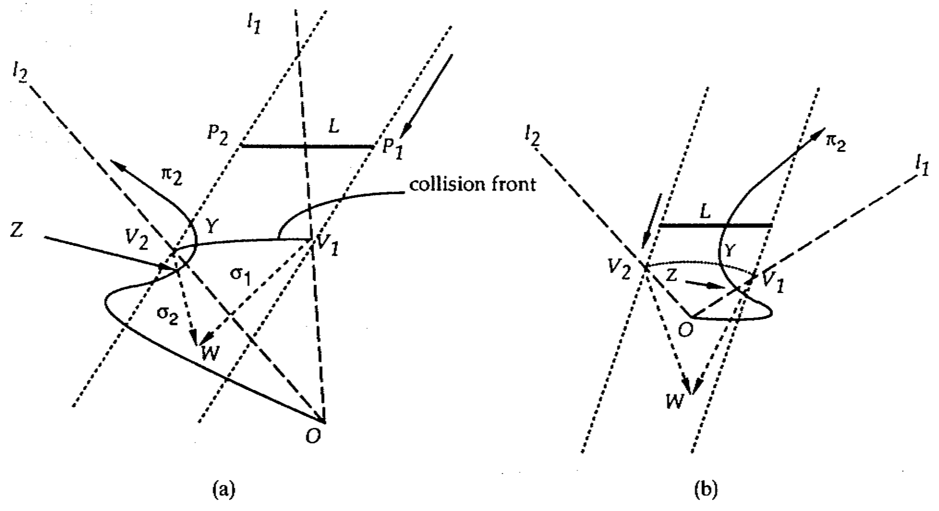


Fig. 7. A proceeding obstacle for the proof of Proposition 3.

G is accessible from O). For a sufficiently small ε, it is clear that τ_1 or τ_2 will reach π_2 before π_2 reaches G.

Next we consider the case in which there is more than one collision front generated at O. If there are n collision fronts, then there are at most n regions designated as B. Let us use B_i to denote such a region behind collision front CF_i (Figure 8). There are at most $2n$ choices for motion π_1—two for each collision front. For each of the collision fronts, CF_i, there is a range of speed values $[v_{\max} - \varepsilon_i, v_{\max}]$ such that the corresponding motion π_{1i} joins motion π_2 at point J_i. Pick a value of the speed v_ε such that v_ε is in each of $[v_{\max} - \varepsilon_i, v_{\max})$. Choose a motion π_{1i} such that J_i is reached at the earliest time of all J's. Now

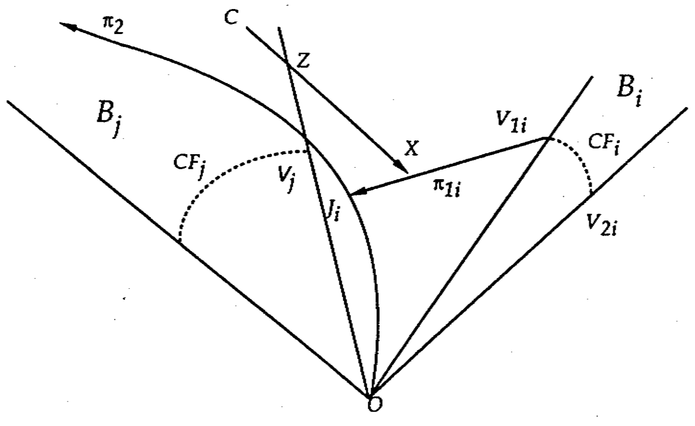


Fig. 8. Multiple collision fronts.

we must show that this motion is collision-free. Suppose that it is not. In this case the obstacle intercepting π_{1i} must come from some B_j region ($j \neq i$). Then it can be shown by applying Proposition 2 twice that a motion π_{1j} (a candidate for π_1 departing from an endpoint of CF_j) will join motion π_2 at an earlier time. In Figure 8 let Z be a point at which C crosses the boundary between A_j and B_j, let V_j be an endpoint of collision front CF_j. Applying Proposition 2 on V_jZX and V_jXJ_i, we can construct a motion that reaches J_i before π_{1i}. This means that π_{1j} can reach π_2 earlier than π_{1i} (i.e., J_j occurs before J_i along π_2), which contradicts the choice of π_1. □

THEOREM. *A time-minimal motion is given as a sequence of edges of*

$$AG(M, O, G, t_0, v_{\max}).$$

As a result of this theorem, point robot R moves at the maximum speed along a time-minimal motion. In other words, if a motion contains some portion during which R moves at a slower speed, then that motion is not time-minimal. We prove the theorem by induction on n, the total number of movements in the given environment.

PROOF (base step). In the base case (i.e., $n = 0$) G is accessible from O, the start point, since there are no obstacles (stationary or moving) in the scene. Recall that we treat stationary obstacles as obstacles with zero velocity. Thus, G's accessible point, X, is in $AVS(M, O, t_0, v_{\max})$, and the procedure terminates with motion OX along which R moves at $v_{\max}$. Now we show that this motion OX is time-minimal. Suppose that it is not. Let γ be a time-minimal motion and let Y be the point at which R reaches G. Motion γ can be extended to point X such that R is always coincident with G from Y to X. Now there are two motions from O to X, i.e., OX and γ plus its extension, both of which reach X at the same time. This is a contradiction since R moves at $v_{\max}$ along straight line OX.

(Inductive step) Next we consider an environment that contains n movements. We assume that the theorem holds for an environment with $n - 1$ movements. If G is accessible from O, then the theorem obviously holds as in the case of $n = 0$. Therefore, assume that G is not accessible from O. The theorem follows from the following two properties proved below. Let V be a point in $AVS(M, O, t_0, v_{\max})$. First, we prove that a time-minimal motion from V to G in M consists of a sequence of edges in $AG(M, V, G, t(V), v_{\max})$. Second, we prove that for any motion in Group 2, a better motion in Group 1 exists. Thus a motion in Group 2 is never time-minimal.

Proof of the First Property. Let V be an accessible point (from O) of a vertex in movement M_i. Note that at time $t(V)$, R is on a vertex of some polygon. Suppose that L_i and L_j are the two edges that are incident on the vertex. We remove L_i and L_j from the polygon and introduce a new edge L_k that connects the two endpoints of L_i and L_j that are not incident at the vertex. This results in a polygon that has one less edge than before. By the induction hypothesis, a time-minimal motion γ exists from V to G in $M - \{M_i, M_j\} + \{M_k\}$, consisting of the edges of

$AG(M - \{M_i, M_j\} + \{M_k\}, V, G, t(V), v_{\max})$, where M_i, M_j, and M_k are the movements of L_i, L_j and L_k, respectively. We show that this motion γ is also collision-free in M. In other words, γ does not collide with M_i or M_j.

Assume that γ collides with M_i. Since R is located at V, we can use Proposition 1 to construct a motion γ' in M that terminates at G at the same time as γ in $M - \{M_i, M_j\} + \{M_k\}$. In case M_i is a single edge, then Proposition 1′ is applied and $M - \{M_i\}$ is used in place of $M - \{M_i, M_j\} + \{M_k\}$ in the following argument. Motion γ' must be one of the time-minimal motions from V to G in $M - \{M_i, M_j\} + \{M_k\}$, as it terminates at G at the same time as γ. However, γ' contains a path segment along which R moves at a slower speed than $v_{\max}$ (as noted in the proof of Proposition 1). Therefore, by the induction hypothesis, γ' cannot be a time-minimal motion from V to G in $M - \{M_i, M_j\} + \{M_k\}$. Since γ and γ' terminate at G at the same time, γ is also not a time-minimal motion, which is a contradiction (recall our initial assumption was that γ is time-minimal). Thus, motion γ does not collide with M_i and is a collision-free time-minimal motion from V to G in M as well. This can be seen by noting that if a collision-free motion is time-minimal in $M - \{M_i, M_j\} + \{M_k\}$, then it is also time-minimal in M.

Let π be a motion from V to G in M that contains a segment during which R moves at a speed less than $v_{\max}$. We now show that π will terminate at G at a time later than γ. By the induction hypothesis, in $M - \{M_i, M_j\} + \{M_k\}$, π terminates at G later than γ. Since γ is also a motion in M, as we showed above, π terminates at G later than γ in M, and thus π is not time-minimal in M. Thus our first property has been proved.

Proof of the Second Property. Assume that there exists a motion α in Group 2. We show that there exists a motion in Group 1 that reaches G earlier than α. Let β be a motion from O to G constructed in the following manner. After leaving O at time t_0, β moves to one of the vertices in $AVS(M, O, t_0, v_{\max})$ (let us call this vertex V) at speed $v_{\max}$, and then turns to rejoin motion α at a speed v less than $v_{\max}$. Proposition 3 assures that this rendezvous is possible. After rejoining α, the remaining parts of motion β are identical to motion α, and β terminates at G at the same time as α. On the other hand, in every time-minimal motion γ from V to G in M, every motion segment has speed $v_{\max}$, as described in the first property. Since β has a motion segment along which R moves at a slower speed than $v_{\max}$, β is not time-minimal from V to G and β terminates at G later than γ. This implies that we can construct a motion in M that terminates at G earlier than α by concatenating motions β and γ at V. Thus our second property has been proved.

From the first property, the time-minimal motion from V to G consists of edges in $AG(M, V, G, t(V), v_{\max})$, which is a subgraph of $AG(M, O, G, t_0, v_{\max})$. From the second property, a time-minimal motion from O to G must contain edge OV (for some V); this is an edge of $AG(M, O, G, t_0, v_{\max})$. Thus the time-minimal motion is a sequence of edges of $AG(M, O, G, t_0, v_{\max})$. □

4. Execution Time. We now analyze the time required to generate a collision-free motion using the algorithm described in Section 3. First we show that the shape of the collision front for a single moving segment is either a parabola, hyperbola,

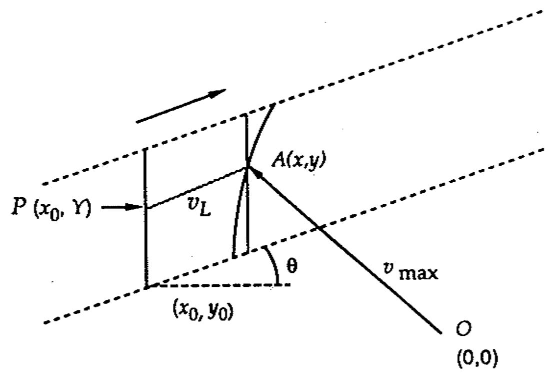


Fig. 9. A movement.

or ellipse. Without loss of generality, we consider a line segment L lying parallel to the y-axis. Let (x_0, y_0) be the location of the lower one of L's endpoints at the start time, let l be the length of L, let v_L be the speed of L, let $v_{\max}$ be the speed of the robot, and let θ be the angle formed by the direction of the movement and the x-axis (Figure 9). Suppose that a point P at (x_0, Y) in L is accessible from the origin $(0, 0)$ at point A at (x, y). Coordinates x and y must satisfy

(1) $$t(A) = \frac{\sqrt{x^2 + y^2}}{v_{\max}} = \frac{\sqrt{(x - x_0)^2 + (y + Y)^2}}{v_L},$$

(2) $$y = (x - x_0) \tan \theta + Y,$$

(3) $$y_0 \le Y \le y_0 + l.$$

The points (x, y) that satisfy (1) and (2) form the collision front. These equations define a quadratic relationship, i.e., the collision front lies either on a parabola, hyperbola, or ellipse.[8] The curve degenerates to a straight line when the direction of motion is parallel to L and when the initial position of L is collinear with the origin. Also, we can show that the origin is on the same side of the collision front as one of the foci of the curve. Generally, the two endpoints of a collision front (accessible points) can be computed by setting Y to y_0 and $y_0 + l$ in (1) and (2).

Having computed two endpoints of the collision front of a single movement, we must now compute all the endpoints of the collision fronts (AVS) when the robot is at a given start point with a given speed v. For a set of polygonal obstacles with n vertices, there are n candidate points. However, some points may not be accessible from the start point since some other movement intercepts the accessibility of that point. We now show that it takes $O(n \log n)$ time to compute AVS.

[8] In general, the collision front is a portion of (i) a hyperbola, if $v_{\max} > v_L \cos \theta$, (ii) a parabola, if $v_{\max} = v_L \cos \theta$, (iii) an ellipse, if $v_{\max} < v_L \cos \theta$. In our problem where $v_{\max} > v_L > v_L \cos \theta$, the collision front is always a portion of a hyperbola except for degenerate cases.

The set of accessible points is determined by using a similar technique to that used to compute the visibility of a given set of line segments. In the following we show two methods for finding elements of AVS.

Method 1 (*Plane-Sweep*). This is based on a plane-sweep algorithm [Pr]. This is a two-step process. The first pass sorts all the vertices, say in a clockwise direction with respect to O. This sorting process takes $O(n \log n)$ time. The second pass rotates a line about O. It halts each time the line intersects a vertex, and checks whether or not the collision front associated with the vertex is accessible from O. This process can be achieved in $O(\log n)$ time by using a 2–3 tree [Ah] to maintain the active collision fronts based on their distance from O. In this way the closest collision front is marked as accessible from O. Since it takes at most $O(n \log n)$ time to build the initial 2–3 tree and $O(\log n)$ time for each update at a vertex, the determination of accessibility of n candidate points takes $O(n \log n)$ time. Therefore, given a start point and a set of candidate points, it takes $O(n \log n)$ time to compute accessible vertices from the start point. Alternately, since no two collision fronts intersect, we can use the approach of Asano *et al.* [As] to find the accessible points in $O(n)$ time after the sorting pass. This gives no asymptotic speedup, but it does avoid the use of 2–3 trees.

Method 2 (*Merge Sort*).[9] In this method the sorting and closest-front finding passes are combined into a single process. The set of collision fronts is divided into two equal-sized groups. For each group we find recursively a polar-order list that describes the collision fronts visible from O. Each list element corresponds to an angular interval in which a particular collision front is closest to O. The list for the whole set of collision fronts is found by merging the lists for the two half-sets. The endpoints of the angular intervals in the two lists determine a finer set of intervals. These finer intervals are computed during a walk through the two lists in tandem; in each of the new intervals we determine which collision front is closer to O. Finally, we merge adjacent intervals in which the closest collision front is the same to produce the list for the whole set. Since it takes $O(n)$ time to combine the two lists, the entire algorithm for finding AVS takes $O(n \log n)$ time.

At this point note that, as a result of our theorem, a motion that meets with the same vertex of an obstacle more than once is not time-minimal. As there are n vertices in the environment, an AVS needs to be generated at most n times before a time-minimal motion reaches the destination point. This observation leads to an asymptotic computation time of $O(n^2 \log n)$ to compute a motion using procedure *PlanMotion*.

Figure 10 shows a scene with 16 moving obstacles. Points O and G are the start and destination points, respectively. Each polygonal obstacle moves in the direction indicated by the arrow near the obstacle at a velocity given in parentheses. In Figures 11(a)–(d), the thick arrows show the motion generated by our algorithm.

[9] This alternative algorithm for finding AVS is suggested by a referee.

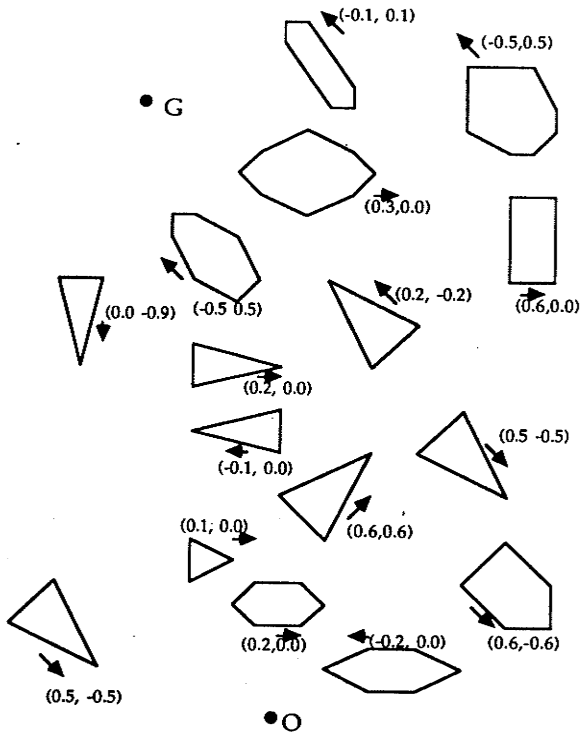


Fig. 10. A scene containing 16 moving obstacles ($V_{\max} = 1.0$).

Hatched obstacles indicate the locations of the obstacles when the point robot is at the end of the thick arrow, and dotted obstacles show the locations of the obstacles from the previous figure. For an environment containing this number of obstacles, it takes about 0.5 seconds to compute a motion using a VAX 11/785.

Note that for m stationary convex polygonal obstacles with a total of n vertices, a shortest length path can be computed using the visibility graph in $O(n^2)$ time [As], [We], $O(E + n \log n)$ time [Gh], or $O(E_S + n \log n)$ time [Kap], where E is the number of edges in the visibility graph and E_S is the number of edges in the visibility graph that are locally tangent at both endpoints. The quantities E and E_S are bounded by n^2 and m^2, respectively.

5. Concluding Remarks.

We have studied the problem of moving a point robot among a set of obstacles moving at constant velocities in the plane. We have discussed the case where the robot can move faster than the obstacles and the destination point. Making use of the concept of accessibility, we have demonstrated an $O(n^2 \log n)$ algorithm to find a motion, and proved that the motion found is time-minimal.

Our main result required that the speed of the destination point be less than

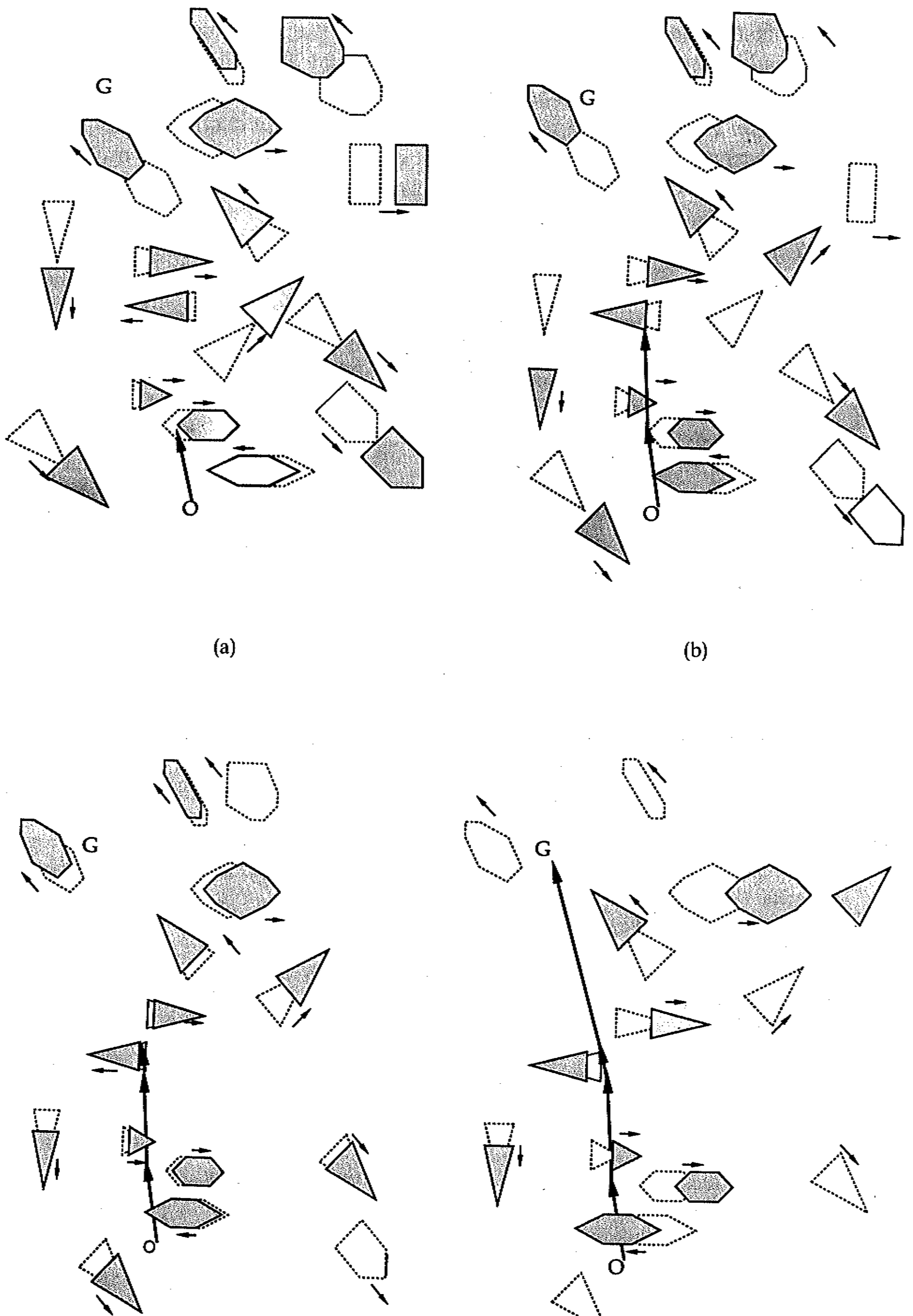


Fig. 11. A time-minimal motion: (a) $t = 2.5$, (b) $t = 6.1$, (c) $t = 7.1$, and (d) $t = 14.4$.

that of the robot. This restriction is not necessary. Of course, when the destination point moves faster than the robot, it is possible that the robot will not be able to reach the destination point. This can be resolved by applying procedure *Plan Motion*.[10] If all the vertices have been visited before finding the destination point, then *PlanMotion* reports that it is not possible to reach the destination point. To conclude correctly that the destination point is unreachable, we must modify the proof of the base case of the theorem. The proof of the inductive case needs no modification since we did not have to use this restriction.

We now show how to relax our original assumption that the number of steps in a motion of the destination point is bounded. Let n_{goal} be the number of steps made by the destination point. Earlier we required it to be bounded so that the accessible point of the destination point could be determined in constant time. This can be done in $O(n_{\text{goal}})$ time, so it does not affect the total computation time of $O(n^2 \log n)$ as long as $n_{\text{goal}} = O(n)$.

Acknowledgments. We have benefited greatly from the incisive critical comments of an anonymous referee. We would also like to thank Azriel Rosenfeld for his comments.

References

[Ah] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[As] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, Visibility of Disjoint Polygons, *Algorithmica*, **1**(1) (1986), 49–63.

[Ca] J. Canny and J. Reif, New Lower Bound Techniques for Robot Motion Planning Problems, *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA, October 1987, pp. 49–60.

[Er] M. Erdmann and T. Lozano-Perez, On Multiple Moving Objects, *Algorithmica*, **2**(4) (1987), 477–522.

[Fu] K. Fujimura and H. Samet, Accessibility: A New Approach to Path Planning Among Moving Obstacles, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Ann Arbor, MI, June 1988, pp. 803–807.

[Gh] S. K. Ghosh and D. M. Mount, An Output Sensitive Algorithm for Computing Visibility Graphs, *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA, October 1987, pp. 11–19. (To appear in *SIAM Journal on Computing.*)

[Kan] K. Kant and S. W. Zucker, Toward Efficient Planning: The Path-Velocity Decomposition, *International Journal of Robotics Research*, **5**(3) (Fall 1986), 72–89.

[Kap] S. Kapoor and S. N. Maheshwari, Efficient Algorithms for Euclidean Shortest Path and Visibility Problems with Polygonal Obstacles, *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 1988, pp. 172–182.

[Lo] T. Lozano-Perez and M. A. Wesley, An Algorithm for Planning Collision Free Paths Among Polyhedral Obstacles, *Communications of the ACM*, **22**(10) (October 1979), 560–570.

[10] A point that moves faster than $v_{\max}$ may be accessible either at zero, one, or two locations. If the destination point is accessible at two locations, then the one with a younger accessible time is adopted at its accessible point.

[Oo] B. J. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap, Robot Navigation in Unknown Terrains Using Learned Visibility Graphs. Part I: The Disjoint Convex Obstacle Case, *IEEE Journal of Robotics and Automation*, **3**(6) (December 1987), 672–680.

[Pr] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[Re] J. Reif and M. Sharir, Motion Planning in the Presence of Moving Obstacles, *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, Portland, OR, October 1985, pp. 144–154.

[We] E. Welzl, Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time, *Information Processing Letters*, **20**(4) (May 1985), 167–171.

[Wh] S. H. Whitesides, Computational Geometry and Motion Planning, in *Computational Geometry* (G. T. Toussaint, ed.), North-Holland, Amsterdam, 1985, pp. 377–427.

[Ya] C. K. Yap, Algorithmic Motion Planning, in *Advances in Robotics*, Vol. 1 (J. T. Schwartz and C. K. Yap, ed.), Lawrence Erlbaum, Hillsdale NJ, 1986, Chapter 3.