Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

NAVIGATION IN A DYNAMIC ENVIRONMENT

by

E. Adin and A.M. Bruckstein

CIS report #9101

March 1991

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

Navigation in a Dynamic Environment

Einat Adin * Alfred Bruckstein †

March 3, 1991

*Faculty of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel. Currently with GALAI Labs., Migdal Haemek, Israel.

†Faculty of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

Abstract

To accomplish various tasks, robots need to navigate in environments that contain stationary or moving obstacles. The trajectory design problem requires finding a continuous collision-free path for the robot, from a given origin to a destination. A dynamic environment calls for a trajectory design that determines both the physical path and the time table for carrying out the movement along it.

This work describes the problem of navigating a point robot among obstacles whose movement is known in advance. The solution provides a collision-free path under the given constraints that minimizes the traversal time.

The approach to the solution is based upon adding the time dimension to the spatial dimensions, resulting in a new space in which all the obstacles are represented as stationary bodies.

The complexity of the problem is first reduced by limiting the movement of the robot : a network of allowed tracks is defined in the navigation space, and the movement is confined to these tracks. The navigation problem is then solved in two stages: the design of the movement along each track of the network, and an algorithm for determining the path over the entire net. The analysis of the movement along each track, for all possible times, provides the data for computing the navigation path over the network.

A time-dependent directed graph is constructed which describes the work-space and the moving obstacles. A search for the shortest journey in the graph between the origin and destination is carried out. The solution is equivalent to finding the shortest route in a network with edges crossings described by the calculated time-schedules.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

1 Introduction

The need to navigate in environments that contain stationary or moving obstacles arises in many robotic tasks. The ability to plan and execute collision-free motions is important for the control of manipulator arms and mobile robots. In both cases, the trajectory design problem requires finding continuous collision-free paths for the robot from a given initial position to a final configuration.

Most previous research on trajectory planning in robotic has focused on the problem of finding paths among static objects. The ability to deal with time-varying environments will increase the potential capabilities and the range of applications of robots. For example, the situation where several robots operate together in a common workspace, each having to transfer payloads from one place to another, is commonly encountered. In such a dynamic environment, one needs to find a trajectory design procedure which will determine both the physical path and the timetable for carrying out the movement along it.

Note that we face similar navigation problems daily, when walking in a room or in the street while avoiding other moving people, when planning a safe crossing of a lawn where sprinklers operate, or while crossing a busy road or driving in the city.

Related Work

The classical so-called "piano-movers" problem [1] is to plan the movement of a robot (in two or three dimensions) from an initial to a final position, so as to avoid collision with obstacles that remain fixed throughout the

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

movement.

A variety of motion planning problems in a static environment have been studied [1] [2] [3]; For instance, the robot may be modelled as a disk, a rigid polygon or a set of articulated arms, and its motion may consist of pure translations or may also involve rotations.

One simplification for purely translational robot motion is achieved by reducing robot's geometric dimensions [2] [4]: the robot is 'shrunk' to a point while the obstacles are expanded. The simpler point navigation problem is then equivalent to the original one.

The task of path planning may be viewed geometrically in two complementary ways: planning the path so as to avoid the obstacles or planning a path that lies entirely in the free space between them.

An approach of the first type was used by Kant and Zucker [1]. Based on their MinPath theorem, they construct a visibility graph in which the nodes are the obstacles' vertices. The solution path is then found by a search for the minimum cost route in the graph.

The basic idea of the free-space approaches is to represent explicitly the safe space among the objects. Generally, that space may be divided into smaller cells. Such an approach has been used by Brooks [6], who represented the free space by generalized cones, their spines indicating the suggested path.

The approaches dealing with stationary objects only are not compatible with the problem of moving obstacles. The dynamic problem poses two combined questions: "where" and "when" will the robot pass on his way to the predefined final position ? Indeed, given two different velocity functions on the same path - one could lead to a collision while the other may not. We

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

must also be aware that the physically shortest path is not necessarily the path with the minimal travel-time. The problem of designing the optimal path with respect to both the elapsed time and the distance traveled seems to be a formidably difficult one.

Kant and Zucker [5] presented a heuristic decomposition of the problem into two subproblems: first planning a path to avoid collision with static obstacles, and then planning the velocity along the path to avoid collision with moving objects. This serial procedure clearly leads to solutions which are not optimal.

Samet and Fujimura [7] introduced the concept of "accessibility" from a point to a moving object, and used it to define an infinite graph on a set of moving obstacles. Another of their contributions [8] is a hierarchical strategy for path planning. They include time as one of the dimensions in the model world. Discretization of the search space was done by quadtree representation, and by the requirement that the path will consist of a sequence of specific points in space-time (chosen from a larger finite set of such points).

Sutner and Maass [9] introduced time dependent graphs as a discrete combinatorial model for the continuous geometric motion planning problem. Furthermore, they introduce the notion of the c-hulls of objects for the development of an algorithm for motion planning in one dimension.

Erdmann and Lozano-Perez [10] explored the motion planning problem for multiple moving objects. For each moving object the planner constructs a configuration space each time a change of velocity occurs. Two adjacent configuration planes are inspected to see if a path exists between them.

For most practical applications, all these methods have drawbacks such

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

as high computational complexity, non optimality, lack of safety margin.

In this paper we present an approach to the problem of motion planning among moving obstacles whose motion is completely known in advance.

The approach is based on restricting the motion of the robot to a predetermined network of paths or links connecting fixed points or vertices in the environment and analysing the problem of collision-free movement along each link of the network. For each link, a collision-free motion schedule is determined, for each vertex of the network a parking schedule is obtained and then the global path design involved the application of a dynamic graph search algorithm. This algorithm yields the time-optimal path on the network.

2 Problem Definition and Outline of Solution

The general 2D dynamic motion planning problem is the following: given a robot and a collection of moving objects in two dimensional space, decide whether there is a continuous, collision-free movement of the robot from a given initial position to a final position, subject to the condition that the robot cannot move any faster than some fixed maximal speed. If a collision-free motion exists , determine its path and schedule.

2.1 Work-Space

In this paper, we restrict our attention to a mobile point robot who can move autonomously among the objects, not faster than some fixed speed. Let O denote a set of obstacles. We assume that each obstacle is a rigid body, whose shape and movement (direction and velocity) are completely

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

deterministic and provided in advance to the robot.

Our 'world' is a region of the two-dimensional plane. The time interval is bounded by some 'max-time' setting a limit on the time allocated for executing the planned movement. Fig.1 illustrates such a work-space.

2.2 The Time Axis

Following the usual procedure [5] [8] [9], we include time as a third dimension of a 3D configuration space, usually called the Space-Time (ST) environment.

The addition of the time axis results in an expanded space in which all obstacles are stationary bodies, generated as the volumes swept as the original obstacles move. If a point (x, y, t) is inside some swept volume in ST, this means that the 2D point (x, y) was occupied by an object at time t.

Now our task is to look for a curve in ST space joining the initial and final points, while avoiding the 'new' obstacles. This problem is quite different from the 3D classical piano-movers problem with static obstacles since movement along the additional time axis here is forced and irreversible: any feasible path in ST has to be monotonically increasing in time, and the robot's predefined maximum velocity constrains the slope of the trajectory in ST space $(v(t) = \frac{ds}{dt})$.

2.3 Complexity and Simplification

Planning trajectories even among static obstacles is a computationally difficult problem. The problem of dynamic motion planning is even harder, and its complexity is large from a practical point of view.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

We reduce the very high complexity of the problem [9] [11] by confining the movement of the robot to a predetermined network of allowed tracks linking fixed points in the navigation space (see Fig.2). The so-discretized navigation problem is then solved in two stages: the analysis of collision-free movement on each track of the network, and an algorithm for determining the path over the entire net of links.

The analysis of the movement along each track, for all possible times, will henceforth be referred to as the one-dimensional problem. It must provide the data for computing the navigation path over the entire network (the two-dimensional problem). A complete solution is thus derived using a novel combination of the navigation problem and an algorithm for path searching in a time-varying graph.

3 Motion Planning Along a Given Link

To solve the simplified 2D navigation problem over the predefined network of tracks, one needs to first analyse the robot's motion possibilities along each track. Hence, the solution of the 1D motion problem along a given path, in the presence of moving obstacles, is essential. The question is "when" will the robot be at each point of the predetermined path ?

The basic step here is adding the time dimension to the given 1D track, resulting in a static 2D ST plane. A preprocessing procedure is then run on the set of obstacles, in order to eliminate some redundant regions from the ST free-space. The one dimensional problem is then solved by projecting the ST plane in a particular direction. Moreover, a comlete link-crossing "time-schedule" is computed for all possible departure times.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

The path itself can be represented as $\Psi(s)$, where s is the length parameter. Let L be the path length, $\Psi(s=0)$ is the representation of the initial point, and $\Psi(s=L)$ is the target location. The robot's velocity limitations are V_+ for movement along the track, and V_- for the oposite direction. Now, the problem can be stated as follows: Given a 1D path in a 2D working space containing moving obstacles whose movement is predefined, find a time-optimal feasible collision-free motion of the robot, starting in $\Psi(s=0)$ at time t_o, and reaching $\Psi(s=L)$.

3.1 Transforming the Problem to Space-Time Plane

The intersections of the volume swept by the moving obstacles with the given path, provide time-varying constraints for the robot's position on the path. The descriptions of these intersections in Space-Time generate a set of stationary objects. Their shape is dependent upon shape and movement (direction and velocity) of the obstacles and the given path.

Motion planning is equivalent to the problem of finding a feasible collision-free curve in the ST plane, joining $(s=0)$ with $(s=L)$ in minimum time. In the new plane the mobile robot remains a point-body, and its velocity limitation is transformed to constraints on the motion's permitted slopes. Combining the monotonicity requirement with the slope constraints, once the robot is at location (s,t), it can proceed inside a conus defined by slopes $a_\pm = \frac{1}{V_\pm}$ (see Fig.3).

3.2 Improving the Representation of ST Plane Obstacles

A preprocessing stage is first executed on the ST stationary obstacles, before the analysis stage. The obstacles are expanded by specific sub-areas, where

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

the future trajectory cannot pass. These areas may be divided into two groups:

- Forbidden regions - the collection of all positions in ST at which a future collision with an obstacle cannot be avoided.
- Hidden region - the collection of all positions in ST which cannot be reached without colliding with an obstacle first.

The existence and exact shape of these regions is determined from the geometries of the ST obstacles (see Fig. 4). The future motion will never pass through points belonging to a hidden or forbidden region.

For that reason, analysis of motion through them is redundant, as well as at the obstacles' positions. The union of these areas with the obstacles that caused them, thus leads to a construction of expanded obstacles' set, (exampled at Fig. 5). The obstacles' expansion is implemented using a shading model: assume a point-light source located at $t = \infty$ illuminating with a parallel beam in the θ direction, towards $t = 0$. Assuming each obstacle is opaque. A shadow is created behind each obstacle, as illustrated in Fig. 6. Let $\theta_{\pm}$ be the angles representing motions in velocity $V_{\pm}$. Illuminating the plane both in θ_+ and θ_- directions at the same time, generates a common- shaded area 'under' every obstacle, identical to its forbidden region (see Fig. 7).

Using a similar model in which the light source is at $t = -\infty$, we can mark the hidden regions.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

3.3 Solution of the 1D Problem

It is feasible to plan a time-optimal motion along a given path with respect to the initial departure time t_0. First, special "t-hidden" regions are defined as the collection of all positions in ST, a robot leaving $s = 0$ at time t cannot reach without colliding with an obstacle first. These regions (see Fig. 8) are projections of the obstacles at angle θ_+ towards the line $s = L$.

For constructing the "t-hidden" region we use a back-projection model : assume a mobile point light source, which starts moving along the time axis at time t, illuminating in θ_+ direction. Projected shadows are generated 'behind' the opaque obstacles (see Fig. 9). The union of all overlapping shadows (starting with a shadow created at time t), results in a combined shadow which is the t-hidden region.

A solution to the given-path motion problem is then determined by the following procedure:
Given the initial time t_0 builds its t_0-hidden region.

- if an empty region was generated, then the robot can move forward along the path at its maximum velocity. Arrival at the target occurs at a fixed delay: $\Delta t = L \cdot a_+$ (the case of t_0' at Fig. 10).

- otherwise, a projected region was created. Let t_+ denote the highest time-value of the corresponding hidden region. The robot can reach the target not earlier than at time t_+. Since the robot cannot move forward at its maximum velocity along all the path, it can choose its desired motion (moving slower all the way, move fast and wait or even go backwards). An optional trajectory is illustrated in Fig. 10, the case of t_0''.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

3.4 Time Schedules

There are cases in which the initial time is not known exactly, and a solution trajectory must be determined for a continuous range of possible departure times.

An obvious localization property in time helps to solve that case without necessitating the construction of infinite t-hidden regions or time discretization. The basic idea is that a solution found for a specific time t_0 remains valid for a finite time period afterwards (Fig. 10):

- if the robot can move at its maximum velocity starting at time t_0' - this is the minimum delay for leaving at later times $t \in [t_0', t_2]$.
- if the robot left $s = 0$ at t_0'' and reached $s = L$ at time t_+, then if it leaves at $t \in [t_0'', t_1]$ it can reach at t_+ but no sooner.

Based on the above property, an efficient procedure is implemented to generate a complete time-schedule for motions along the path. A finite number of projected regions are computed, each for the time defined by one of the "extremum" values of an obstacle (extremum in the θ_+ direction).

3.5 The "Parking" Table

The vertices of the net of tracks, i.e. the initial and final location can be occupied by a moving obstacle. For that reason we determine the time intervals when passing and parking are allowed at the source point (Fig. 11). This information can be extracted directly from the ST representation, by scanning the overlaps of obstacles with $s = 0$.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

4 Motion Planning on the Network of Tracks

The main drawback of motion planning in a 2D dynamic environment is its very high complexity. For simplification, we limited the movement of the robot by restricting it to a predefined network of "allowed" tracks. The solution trajectory will determine both the physical path on the networks and the time charts for carrying out the movement.

The first stage of the planning process was the 1D motion analysis along all possible path-segments: each segment defines two different Space -Time planes (one for each direction). A complete time-schedule and a parking table are built for each ST plane, as the result of the one-dimensional process described in Section 3. This stage of collecting the possible motions-data and parking over the net, is essential for solving the problem.

4.1 Transforming the Navigation Problem into a Graph Search Problem

After analysing the possible motion on each link of the network, the navigation problem is transformed into a graph search problem. A time-varying directed graph is built, which describes exactly the given motion planning problem (Fig. 12). The graph nodes are the vertices of the network. Each segment is represented by two directed edges, one for each motion direction. The weights of the edges are functions of time, since the delay of motion over each edge depends on the departure time. These time functions are the time-schedules computed earlier.

Each node also has a parking table, denoting the time periods when parking or passing through the node are permitted. The initial and target

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

positions are represented by nodes of the graph. Solving the dynamic 2D motion planning problem now becomes equivalent to finding the shortest route in the time-varying graph.

4.2 Formulation of the Graph Problem and Notations

Let $G = (N, E)$ be a finite directed graph, where $N = \{1, ..., n\}$ is the set of nodes with $E \subset \{(i, i) | i, j \in N\}$ being the set of directed edges. Assume time is measured from the moment that it is possible to initiate travelling on the graph. Let $d_{ij}(t)$ be the travelling time from node i to node j, leaving i at time t. $d_{ij}(t)$ is a non-negative, finite and picewize continuous function of t.

Once travelling along an edge (i, j) has been started at node i at time t, it continues until node j is reached at time $t + d_{ij}(t)$.

Parking is allowed only along the nodes. The limitations on parking on the nodes takes the form of "no parking" periods of time. Let P_i be the finite set of time intervals when parking is permitted on node i.

Let A_j be a finite set of feasible arrival time intervals to node j, and $D_j = f(P_j, A_j)$ is a finite set of feasible departure time intervals from node j.

Define N_k as a subset of N, including all nodes which are neighbours of node k.

A feasible journey joining two specific nodes in G is identified by an ordered set of nodes $\{(i_1, i_2, ..., i_q\}$ such that $(i_k, i_{k+1}) \in E$ for all $k = 1 \cdots q - 1$, and an ordered set of pairs of points of time $\{(t'_1, t''_1) \cdots (t'_q, t''_q)\}$ where t'_k is the arrival time to node j and t''_k is the departure time from this node. Feasibility requirements imply that $(t'_k, t''_k) \in P_k$ and $d_{k,k+1}(t''_k) < \infty$.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

The problem may be formally defined as: "Find a feasible journey connecting the source node (1) and destination node (n) which minimizes t'_n ".

4.3 Outline of Solution

A variety of algorithms [12] [13] have been proposed for the solution of the shortest route problem with time-dependent length of edges and free parking. These approaches are based on the fact that nothing can be gained by delaying the arrival time to any given node. Such a delay in leaving nodes may sometimes be desirable due to the features of the functions $d_{ij}(t)$, but not always allowed in our case which includes restricted parking. For that reason, an optimal route may include cycling.

Halpern and Priess [14] [15] have suggested an efficient algorithm for the solution of the shortest route problem in networks with time dependent weights of edges and limited parking possibilities in nodes. Their assumptions are identical to our problem's conditions, except for their demand for permanent possibility of passing through a node. Luckily, our weight functions $d_{ij}(t)$ were computed so that they avoid the robot's arrival or wait on a node while it is occupied.

Halpern and Preiss' algorithm defines two subsets of nodes:
- The first group, N^*, includes the nodes to which the length of the shortest route from the source node is already known.
- The second group, N^{**}, is a subset of nodes for each of which there exists a nonempty set of arrival times, A_j, when arrival is via nodes contained in N^*.

At the initiation step, the N^* group includes only the source node. By

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

iterative procedure, the algorithm expands the sets N^* and N^{**}. The possible arrival times from nodes in N^* to their neighbours in N^{**} are updated, and a node contained in N^{**} is selected for transfer to N^*. The algorithm terminates after a finite number of iterations. The corresponding cost when a node is selected for the first time to be transferred from N^{**} to N^*, is the length of the shortest route from the origin to that node. Once the destination node is added to N^*, the length of the shortest route to it is found. If N^{**} becomes empty before the definition node appears on it, then there is no feasible route leading from the source node to the destination node.

4.4 The Graph Search Algorithm

Let 1 be the source node and n is the destination node.

Step 0: Let $N^* = \emptyset$, $N^{**} = \{1\}$, $A_1 = \{0\}$, $D_1 = \emptyset$,
$A_j = D_j = 0$ for all $j \neq 1$. Set $k = 1$.
Go to step 1.

Step 1: Set $N^* = N^* \cup \{k\}$, $N^{**} = N^{**} \backslash \{k\}$, $D = f(A_k, P_k) \backslash D_k$,
$D_k = D_k \cup D$, $A_k = \emptyset$.
Go to Step 2.

Step 2: For all $j \in N_k$ compute $A_j = A_j \cup \{t + d_{kj}(t) | t \in D\} \setminus D_j$.
Let $\bar{N}_k = \{j \in N_k | A_j \neq 0\}$ and $N^{**} = N^{**} \cup \bar{N}_k$.

If $N^{**} = \emptyset$, terminate. There is no feasible route from source to destination. Otherwise go to Step 3.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

Step 3: For all $j \in \bar{N}_k$ compute $T_j = \inf\{t | t \in A_j\}$.
Let $T^* = \min\{T_j | j \in N^{**}\}$ and $k \in N^{**}$ be the node for which $T_k^* = T^*$.
If k is the destination node, terminate. T^* is the length of the shortest route from the origin to the destination. Otherwise return to Step 1.

5 Examples

In this section we present some simple examples for motion planning in the presence of moving obstacles. The simulation results were obtained by using the technique described in the previous sections.

We restrict the simulation to deal with the case in which the obstacles are simple polyhedral objects, not necessarily convex. Each obstacle is given by its shape, location at a specified time, and trajectory (velocity and direction). Note that the time and velocity values are give concerning some fixed time and distance units.

We considered a 2D navigation space, including 16 triangular obstacles. Each obstacle moves with a different velocity in its "arrow-head" direction parallel to the X or Y axis. The robot is a point object whose maximum allowable speed is $V_+ = V_- = 1$ (i.e., same velocity for moving forwards and backwards). The limit on the elapsed time of the planned movement was set to 100.

To create an interesting problem, we restricted the obstacles' velocities so that the robot could move faster then each one of them.

A square network of tracks was defined over the navigation area.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

Fig. 13 shows the work-space and demonstrates the changes in the environment as a result of obstacle movements.

We shall illustrate our motion planning technique for different pairs of source (S) and destination (D) locations. In each case, a legal collision free trajectory is generated.

In the first case, shown in Fig. 14(1), the robot has to proceed from S_1 to D_1. The minimal arrival time to the target is 75.9 time units. The bold lines mark the designed trajectory.

The second example, Fig. 14(2), illustrates a situation in which the robot must move forwards and backwards along a track to avoid collision with one obstacle which moved "up", and with another that was "moving down" towards it. The robot can reach its destination D_2 no sooner than at time 51.4.

The third case (Fig. 14(3)) is a nice example that illustrates the difference between a short path and minimal-time movement. Although there exists a track connecting S_3 and D_3, the robot cannot proceed along it at time 0 due to the combined effect of two obstacles which are close to S_3 at the initial time. The robot had no other choice but to run away along two tracks and then return to its original location before it could reach D_3 at minimal time 51.4.

6 Discussion

An approach has been proposed for designing continuous collision-free trajectories for mobile robots moving in a time-varying environment.

The motion planning problem was first simplified by limiting the robot's

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

movement to a pre-defined network of tracks. This effective "discretization" of the navigation space clearly influences the quality of our solution - the proposed trajectory will not be time-optimal over all possible trajectories in the continuous 2D physical work-space. However, unlike other approaches, our trajectory is continuous in time and therefore it is time-optimal for the given constrained motion.

Since the robot is allowed to proceed along tracks, selecting a good network of tracks for a given motion problem is of great significance, and it is yet an unsolved problem. Given a network of tracks, we suggested an efficient method for optimal motion planning along each track (the 1D problem) that results in a "time schedule" describing all possible departure and corresponding times.

Our geometric method is general and is clearly not confined to polyhedral or convex obstacles being capable of handling any simple object.

The complete solution involves a novel combination of the navigation problem with an algorithm for searching an optimal path in a time-varying graph.

Acknowledgements

We wish to thank Michael Fleisher for his help with this document.

References

[1] J.T. Schwartz and M. Sharir, " On the "Piano Movers' " Problem I. The Case of a Two-Dimensional Rigid Polygonal Body Moving

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

Amidst Polygonal Barriers", *Communications on Pure and Applied Mathematics*, vol. XXXVI, pp. 345-398, 1983.

[2] T. Lozano-Perez and M.A. Wesley, "An Algorithm for Planning Collison-Free Paths Among Polyhedral Obstacles", *Communications of the ACM*, vol. 22, No. 10, pp. 560-570, Oct. 1979.

[3] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces", *Proceedings of the 16th Annual Symposium on Theory of Computing*, pp. 144-153, Washington, 1984.

[4] S.M. Udupa, "Collison Detection and Avoidance in Computer Manipulators", *Proceedings 5th International Joint Conference Artificial Intelligence*, pp. 737-748, 1977.

[5] K. Kant and S.W. Zucker, "Toward Efficient Trajectory Planning: The Path Velocity Decomposition", *The International Journal of Robotics Research*, vol. 5, pp. 72-89, 1986.

[6] B.A. Brooks, "Solving the FindPath Problem by Good Representation of Free Space", *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-13, No. 3, pp. 190-197, Mar. 1983.

[7] K. Fujimura and H. Samet, "Time Minimal Paths Among Moving Obstacles", *Center For Automation Research, University of Maryland*, CAR-TR-398, Oct. 1988.

[8] K. Fujimura and H. Samet, " A Hierarchical Strategy for Path Planning Among Moving Obstacles", *IEEE Transactions on Robotics and Automation*, vol. 5 No. 1, pp. 61-69, Feb.1989.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

[9] K. Sutner and W. Maass, "Motion Planning Among Time Dependent Obstacles", *ACTA Informatica*, vol. 26, pp. 93-122, 1988.

[10] M. Erdmann and T. Lozano-Perez, "On Multiple Moving Objects", *Algorithmica*, vol. 2, No. 4, pp. 477-521, 1987.

[11] J. H. Reif, and M. Sharir, "Motion Planning in the Presence of Moving Obstacles", *Center for Research in Computing Technology, Harvard University*, TR-06-85, May 1985.

[12] S.E. Dreyfus, "An Appraisal of Some Shortest Path Algorithms", *Operations Research*, vol. 17, pp. 395-411, 1969.

[13] K.L. Cooke and E. Halsey, "The Shortest Route Through a Network with Time-Dependent Internodal Transit Times", *Journal of Mathematical Analysis and Applications*, vol. 14, pp. 493-498, 1966.

[14] J. Halpern, "Shortest Route with Time Dependent Length of Edges and Limites Delay Possibilities in Nodes", *Zeitschrift fur Operations Research*, vol. 21, pp. 117-124, 1977.

[15] P. Yigal, "Shortest Path Problem in a Network with Restriction on Movement and Parking", *MSc Dissertation*, Technion-Israel Institute of Technology, Haifa, Israel, Jun. 1973.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

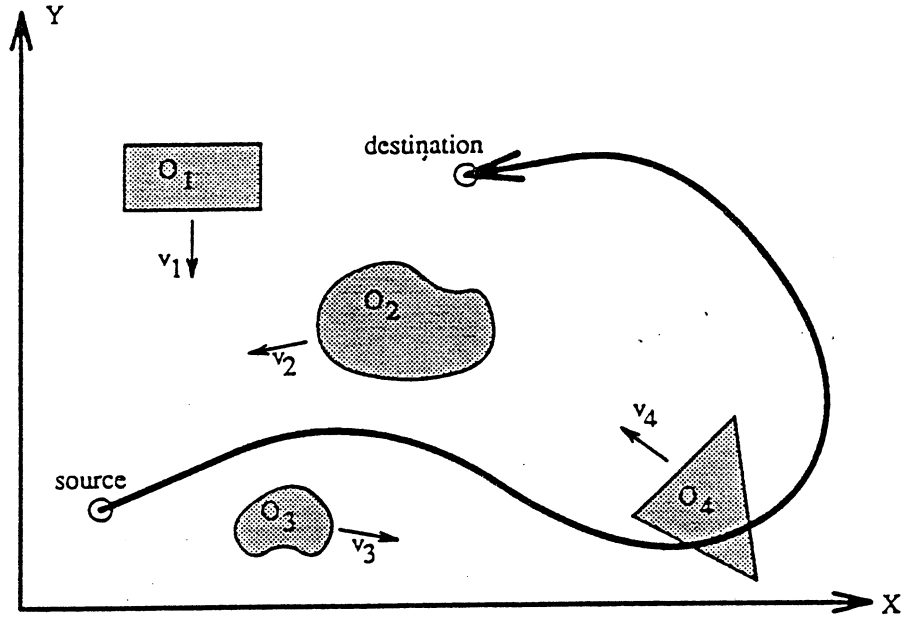


Fig. 1: An example of the work-space.

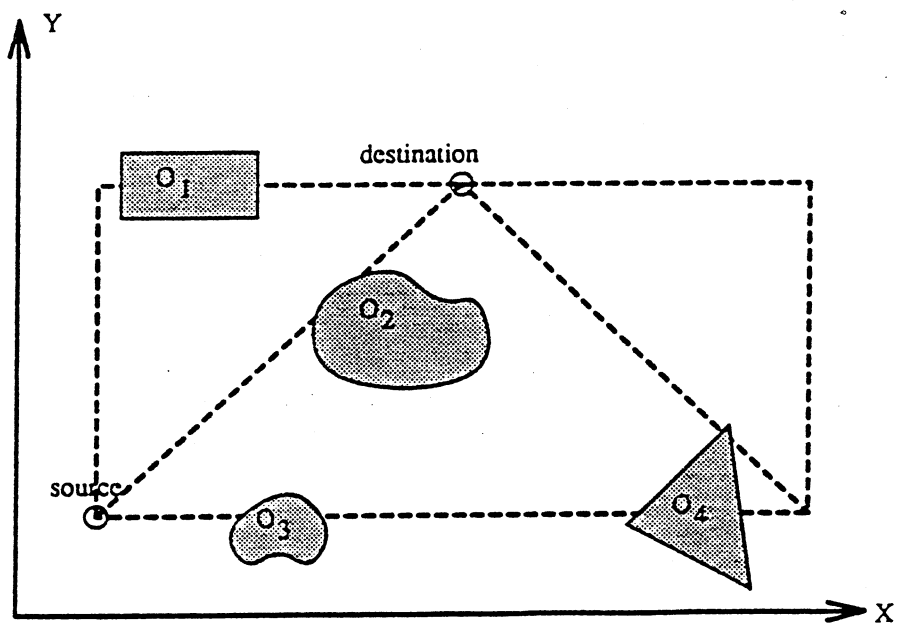


Fig. 2: Restriction of the work-space to a network of tracks.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

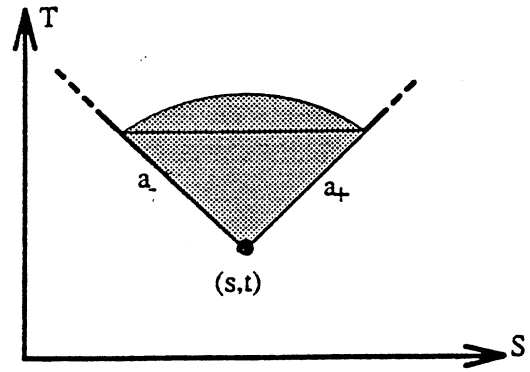


Fig. 3: Possible movements according to maximum velocity limitation.

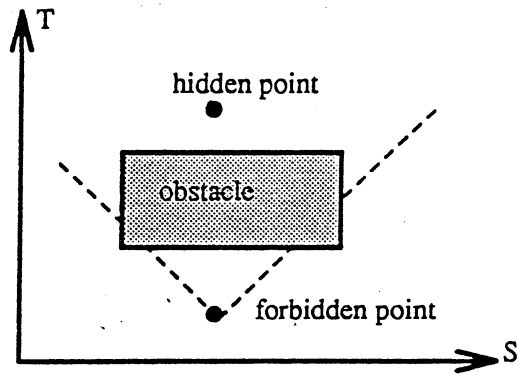


Fig. 4: An example of a hidden point and a forbidden point.

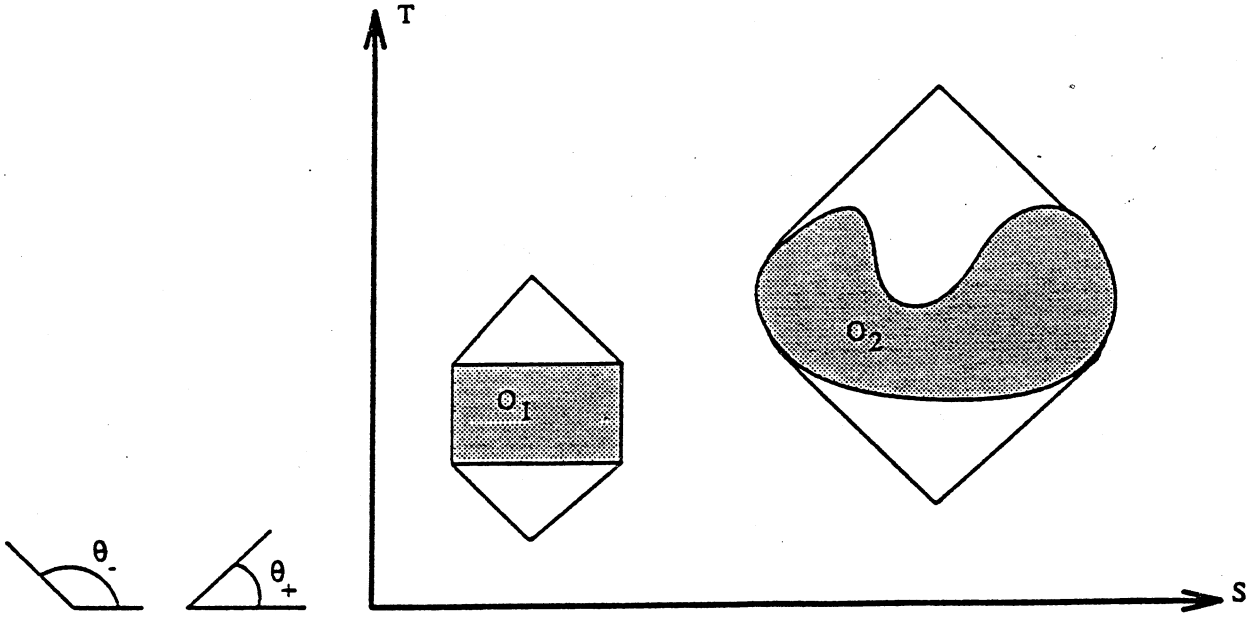


Fig. 5: Expansion of the obstacles with the hidden and forbidden regions.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

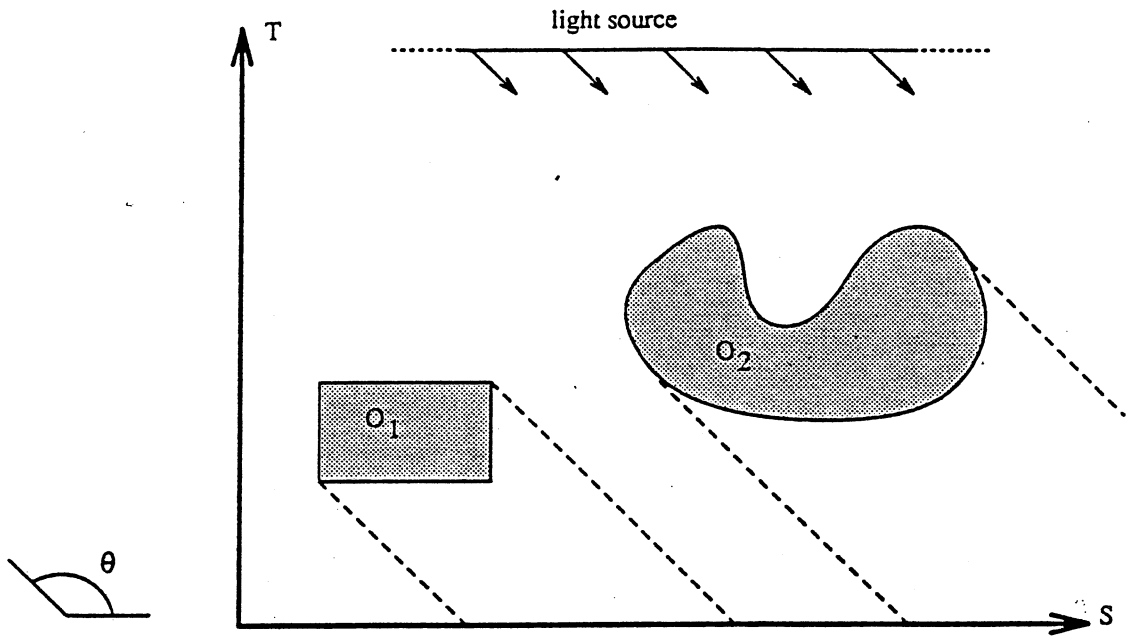


Fig. 6: Shadows generated by illumination in the θ direction.

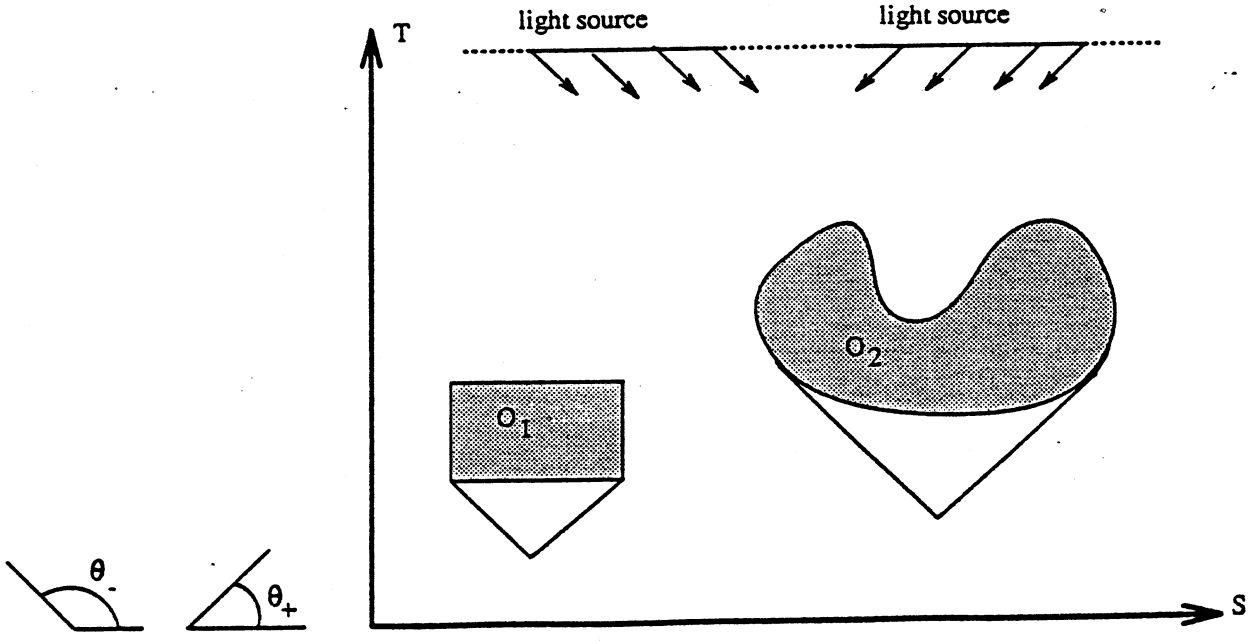


Fig. 7: Shadows generated by illumination in the $\theta_\pm$ directions.

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

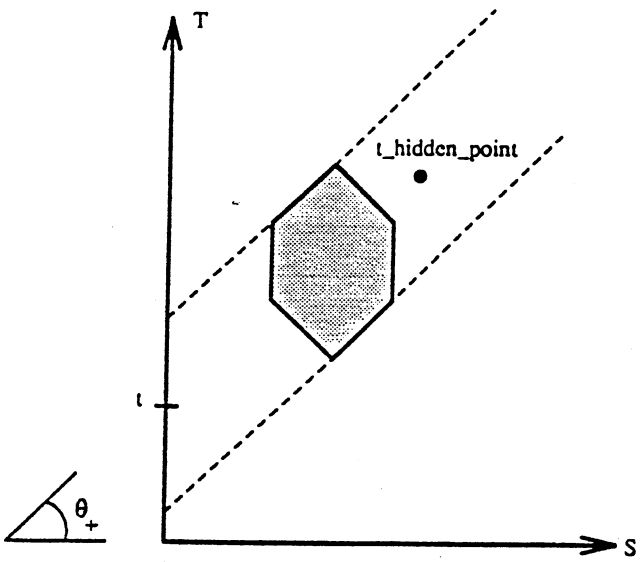


Fig. 8: An example of a point hidden relative to time t.

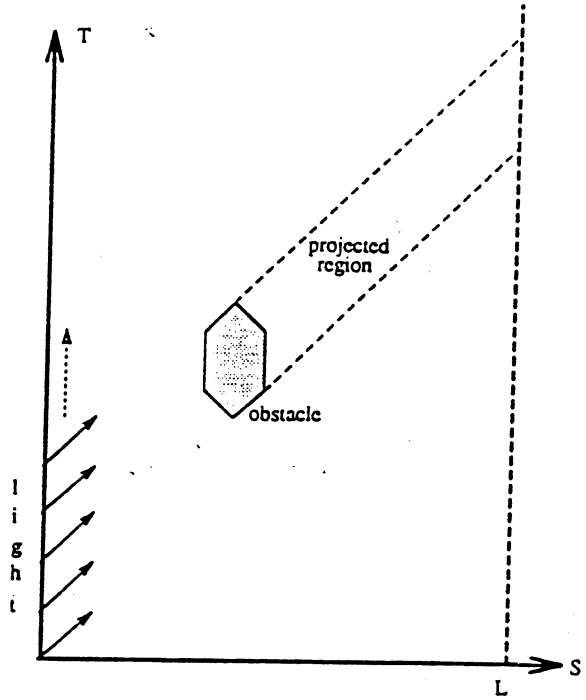


Fig. 9: An example of a projected region.

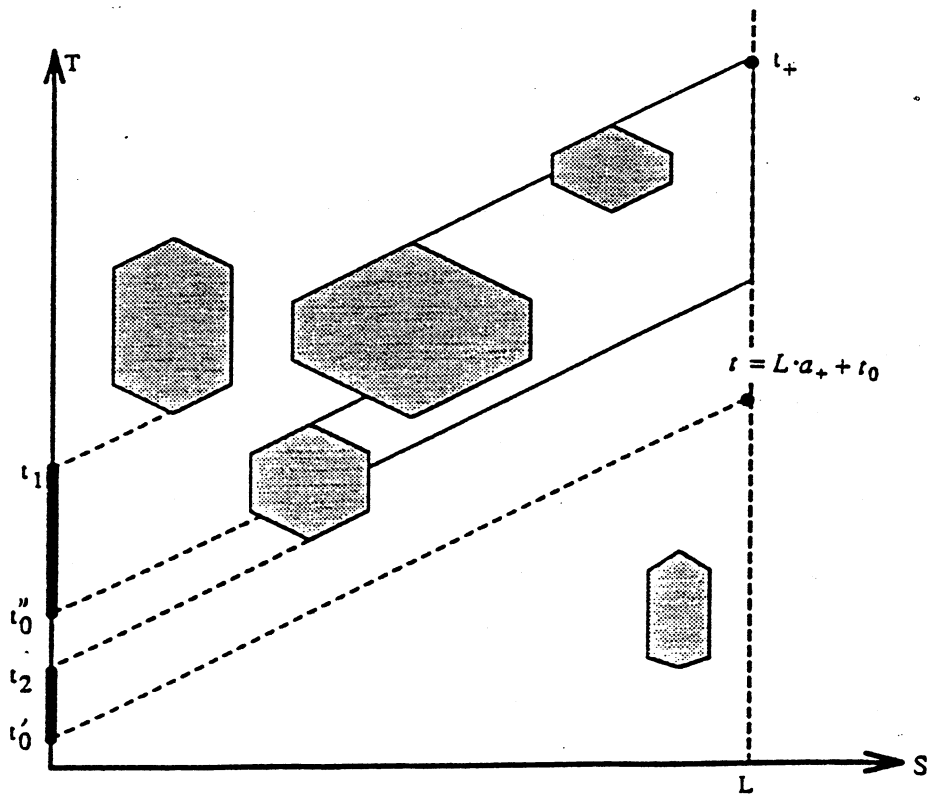


Fig. 10: The solution of the 1D problem - navigation along a given path.

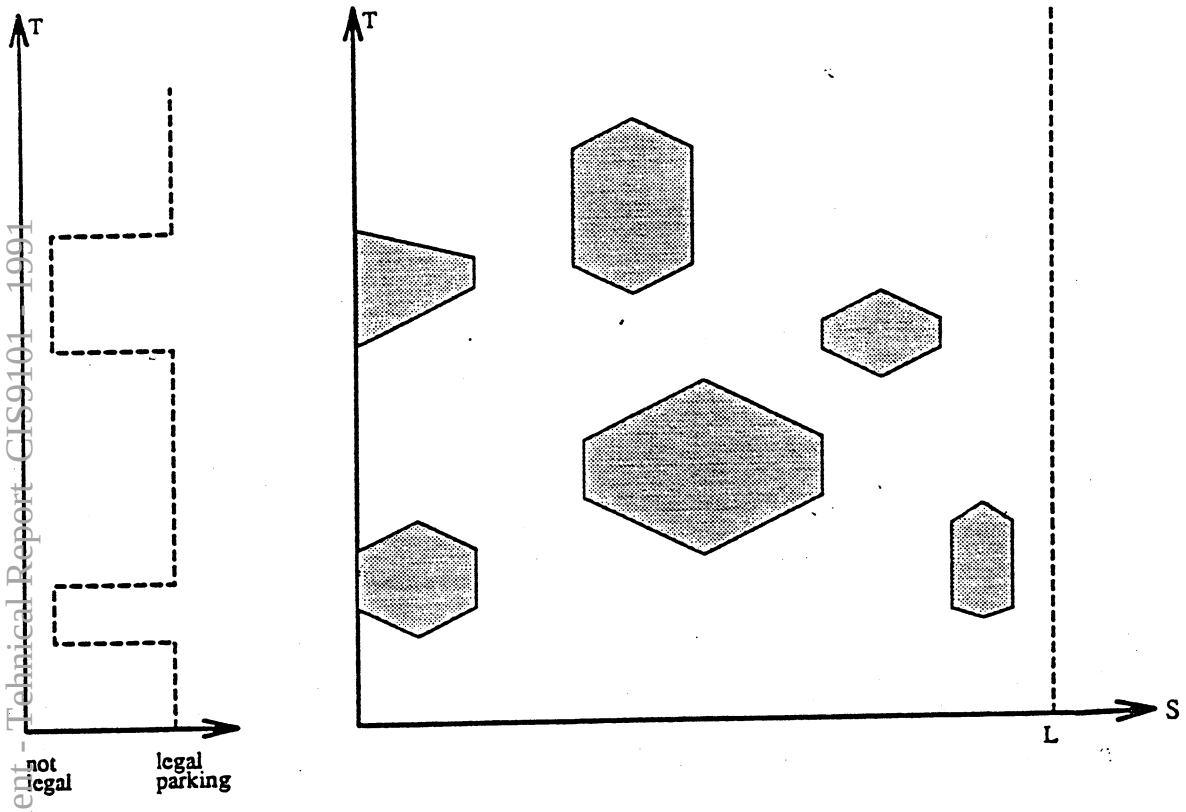
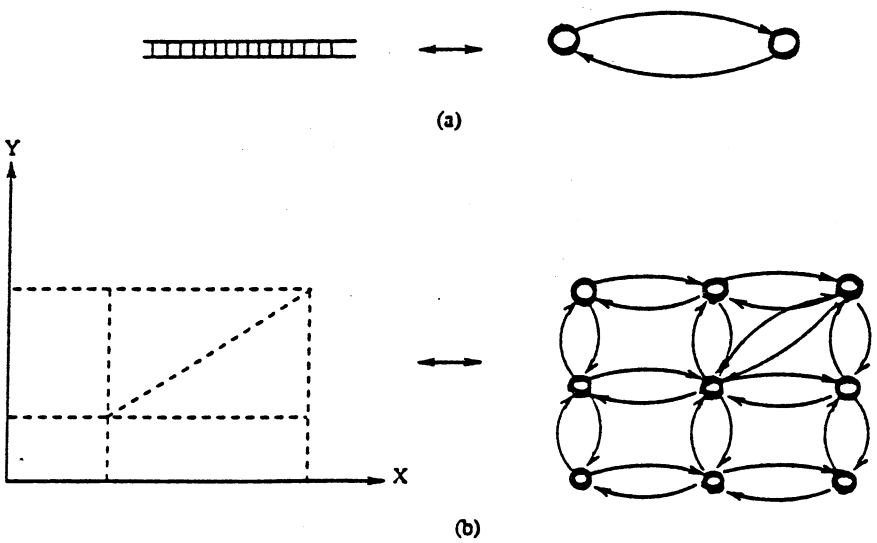


Fig. 11: Extraction of legal parking periods at the source point.



Technion - Computer Science Department - Technical Report - CIS9101 - 1991

Technion - Computer Science Department - Tehnical Report CIS9101 - 1991

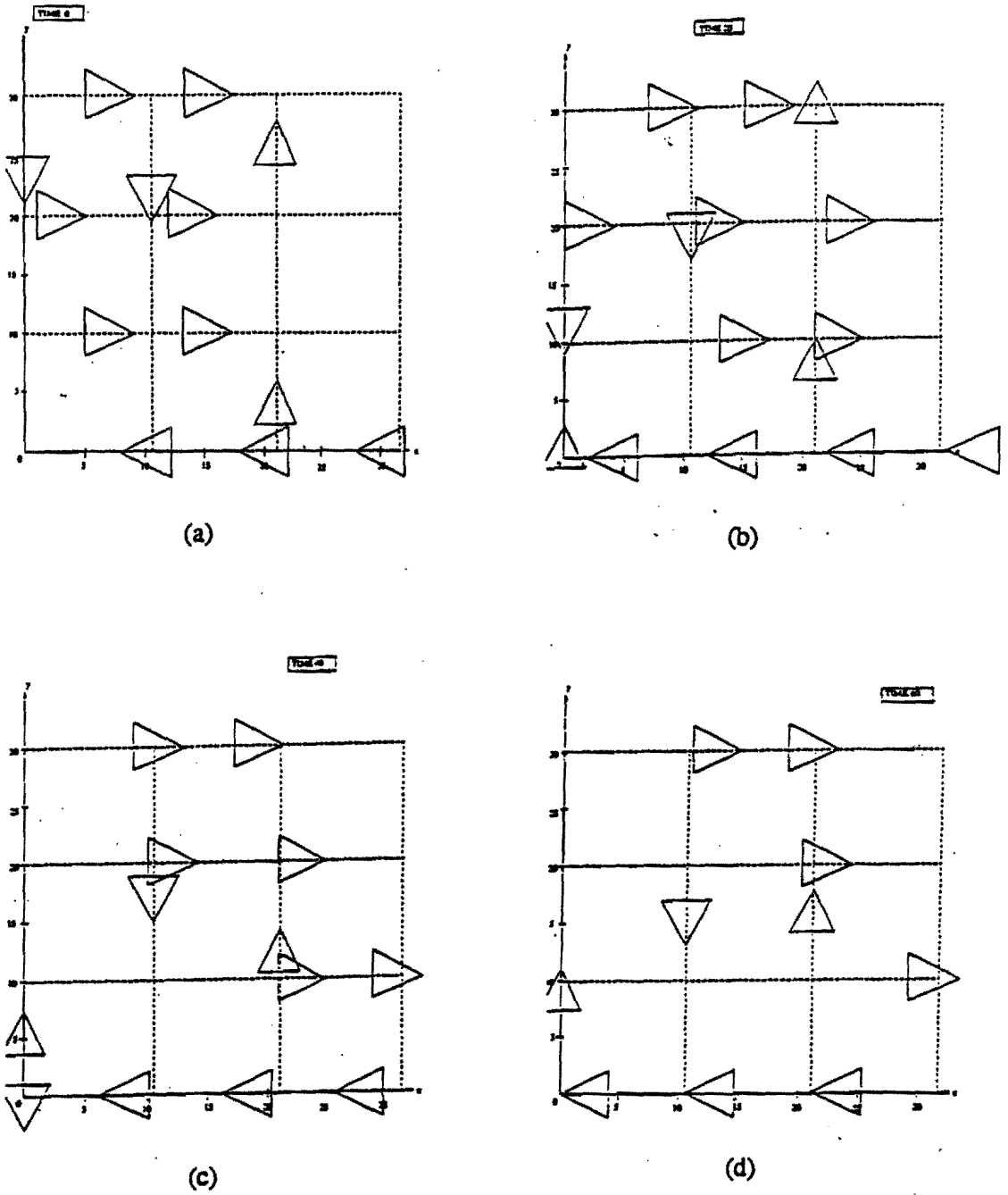


Fig. 13: Work-space map at various times:
(a) $t = 0$ (b) $t = 20$ (c) $t = 40$ (d) $t = 60$.

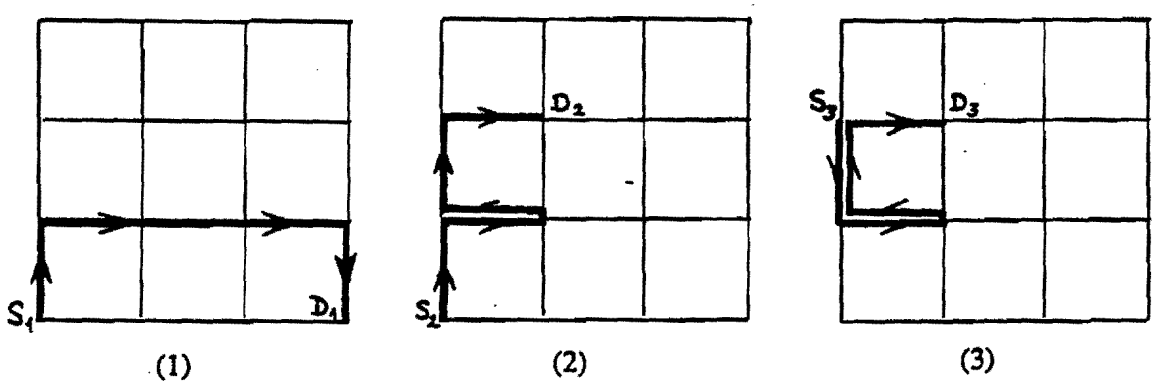


Fig. 14: Examples of different navigation trajectories.