

Spatial and Temporal Path Planning

Marc G. Slack
David P. Miller

TR 87-19

Spatial and Temporal Path Planning¹

Marc G. Slack

Committee Chairman: David P. Miller

Abstract

For robots to move out of the lab and into the real-world, they must be able to plan routes not only through space but through time as well. The introduction of a time factor to the planning process implies that robots must reason about other processes and agents that move through space independently of the robot's actions. This thesis presents an integrated route planner and spatial representation system for planning real-time paths through dynamic domains called Robonav. Robonav will find the safest, most efficient route through time and space as described by an evaluation function. Due to the design of the spatial representation and the mechanics of the algorithm, Robonav has an isomorphic mapping onto a machine with a highly parallel SIMD architecture. When Robonav is operated in a predictable domain, paths are found in $O(p)$ time (where p is the length of a path). In unpredictable domains, where Robonav is operated in incremental mode, paths are found and executed in $O(p^2)$ time.

¹This research was supported in part by a grant from the Naval Surface Weapons Center, under contract number N60921-83-G-A165.

Acknowledgements

Thanks to David Miller and Catherine Stein, for their support on both a professional and personal level.

Thanks to my friend Sallie Henry, for her support and my opportunity to do graduate work.

Thanks to Erann Gat, for conversations about this research that produced some good ideas.

Thanks to Steve and Deb Wake, good friends pass through but twelve times in a life.

Thanks to Connie and Gary, for their kind grace.

Special thanks to my wife Pam, for her time and effort helping me to live my life and write this thesis.

Contents

Acknowledgements	iii
Contents	iv
List of Figures	vi
Chapter 1	1
1.1 The Route Planning Problem	1
1.2 Review of Related Works	3
1.2.1 Voronoi Diagrams	3
1.2.2 Free Space Representations	3
1.2.3 Knowledge Bases	5
1.2.4 Potential Fields	7
1.2.5 Vertex Graphs	8
1.2.6 Configuration Space	8
1.2.6 Grid Representations of Space	8
1.3 Planning Routes Through Interesting Spaces	9
1.4 An Outline of the Thesis	10
Chapter 2	11
2.1 Spatial Representation	11
2.2 Finding Paths Through Space	14
2.2.1 Phase 1	
"Throwing the Stone"	17
2.2.2 Phase 2	
"The propagating wave front"	17
2.2.3 Termination	
"A Wave reaches destination"	19
2.3 Finding Paths Through Space and Time	21
2.4 Modeling A Robots Continuous Time Actions	24
2.5 Concluding Remarks	
Algorithms and Models	25
Chapter 3	27

3.1 Predictable Objects in n-Space and Time.....	27
3.2 Refining Object Representation.....	31
3.3 Trade-offs Between Resolution and Efficiency.....	34
Chapter 4.....	36
4.1 Spatial and Temporal Qualities.....	36
4.2 Defining Evaluation Functions.....	37
4.2.1 Using a Single Measure of Quality	
"Openness".....	37
4.2.2 Defining Multi-Variable Evaluation Functions.....	40
4.3 Finding Paths Through Dynamic Spaces.....	42
4.4 Robot Specific Abilities and Limitations.....	45
Chapter 5.....	49
5.1 Incremental Operation.....	49
5.2 Example in an Unpredictable Domain.....	50
Chapter 6.....	53
6.1 Conclusions.....	53
6.2 Further Research.....	53
6.3 Implementation.....	54
Bibliography.....	56
Vita.....	58

List of Figures

Figure 1	The path planning problem.....	2
Figure 2	Voronoi regions of a given 2-space.....	4
Figure 3	Some generalized cones.....	6
Figure 4	Nodes in 1, 2 and 3-space.....	12
Figure 5	Arbitrary examples of 2 and 3-space.....	13
Figure 6	Links of a simple 2-space.....	15
Figure 7	Searching for the destination in 2-space.....	16
Figure 8	Phase one.....	18
Figure 9	Phase 2.....	20
Figure 10	Reflexive links of a simple 2-space.....	22
Figure 11	Shortest path through time.....	26
Figure 12	Representing a revolving door.....	28
Figure 13	Path finding through a simple 1-space.....	30
Figure 14	Overlapping nodes.....	32
Figure 15	Resolution increasing to match the problem.....	33
Figure 16	Refining the revolving door.....	35
Figure 17	Openness in a simple 2-space.....	39
Figure 18	Evaluation function using openness.....	41
Figure 19	Evaluation functions using multiple qualities.....	43
Figure 20	Path finding through a rich space.....	44
Figure 21	Evaluation functions in dynamic spaces.....	46
Figure 22	F Regions in 2-space.....	47
Figure 23	Evaluation function using F regions.....	48
Figure 24	Operation in an unpredictable domain.....	51

Chapter 1

Introduction: Dynamic Domain Path Planning

The research presented in this thesis is directed towards the creation of a path planning system capable of reasoning not only about the spatial aspects of path planning but also the temporal aspects. Several route planning systems have been proposed, each having some degree of success in the domain for which they were targeted. This research represents the first system that begins to adequately address the temporal aspect of the route planning problem.

1.1 The Route Planning Problem

The problem of planning paths for use in moving robots through a given domain is an old one. However, none of the classical systems has an adequate definition of path planning. In this thesis the path planning task will be defined as finding the optimal path between two locations in space. A path is considered optimal if it best satisfies the constraints imposed by the domain in which the planner is operating. Therefore, a route planning system should be capable of planning paths in domains with constraints involving: predictable and unpredictable objects moving through space; the quality of space through which the robot is to move; the abilities and limitations of the robot executing the plans; and the goals and schedules that the robot is to meet. It should also be noted that the definition of optimal path is constrained by the particular spatial representation used by the path planner.

Figure 1 shows an abstract example of a path planning problem set in a two-dimensional space (or 2-space hereafter). Contained in the 2-space are three static objects and one dynamic object. The planning objective is to move the robot along a path that maintains a safe distance from the static objects while avoiding conflict with the dynamic object. The figure indicates the timings of a path that satisfies the given constraints. Notice that the robot pauses during times $t=2$ to $t=4$, allowing the dynamic object to move out of its way before continuing on to the destination. This example is included simply to give an indication of the types of domains in which a robust route planning system should be able to operate.

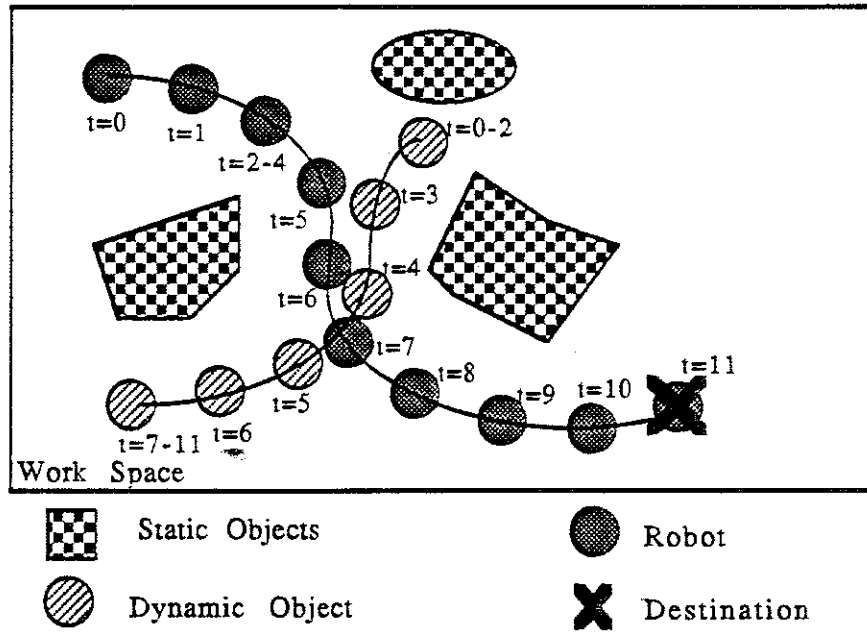


Figure 1 The path planning problem

The problem of path planning is not constrained to the lab. It has many practical real-world applications. Consider the automated space stations of the future, where a number of autonomous robots will need to plan paths to dock with the orbiting station. On a more tangible level, robots working in a job shop require dynamic path planning to maximize resource utilization. These represent only two examples but should illustrate the need for dynamic path planning.

1.2 Review of Related Works

A number of spatial representations and path planning algorithms have been proposed to address the robot navigation problem. This section will describe some of this work and relate advantages and disadvantages of the different approaches taken.

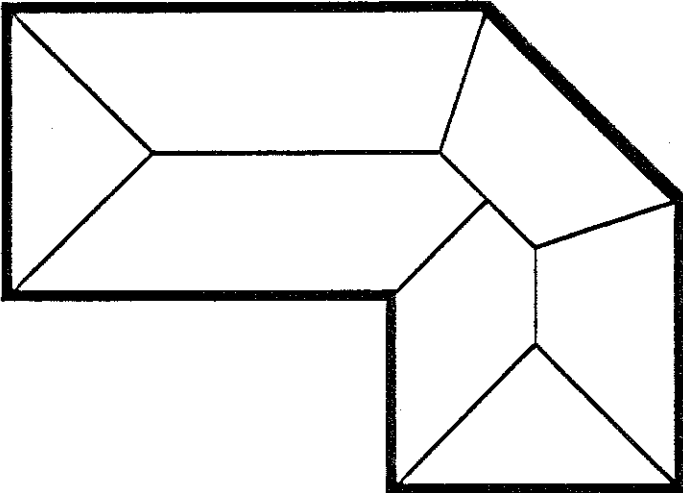
1.2.1 Voronoi Diagrams

Voronoi diagrams have been used to find paths through a finite space [O'Rourke84]. Given a set of points s_1, s_2, \dots, s_n in some space, a Voronoi diagram is defined as a partitioning of space into regions; one region for each of the n points. Each region represents those points in the space that are closer to the point associated with a given region than any other point in the set of points s_1, s_2, \dots, s_n . If the points defining the various regions are generalized to lines, the resulting partitioning can be used by a path planning algorithm. Figure 2 gives an example of how a Voronoi diagram generalized to lines would be defined for a simple 2-space. Path planning algorithms using this paradigm search over the intersections of the regions for a path to the desired location.

The use of Voronoi diagrams allows paths to be generated that maintain a maximal distance from all objects occupying the space. One problem with this technique for the representation of space is that it provides no information on the quality of the surface that the robot is moving across. Another problem is that Voronoi diagrams are not well suited for the representation of dynamic objects and therefore will not operate in dynamic domains.

1.2.2 Free Space Representations

There are a number of systems that approach the path planning problem using generalized cones (similar to Voronoi diagrams) to represent free space. These systems use traditional heuristic search techniques to find collision free paths through the free space representation. Among the researchers taking this approach to robot navigation are Brooks [Brooks82] and Nguyen [Nguyen84].





 Line between two Voronoi regions
 Wall of room to be navigated

Figure 2 Voronoi regions of a given 2-space

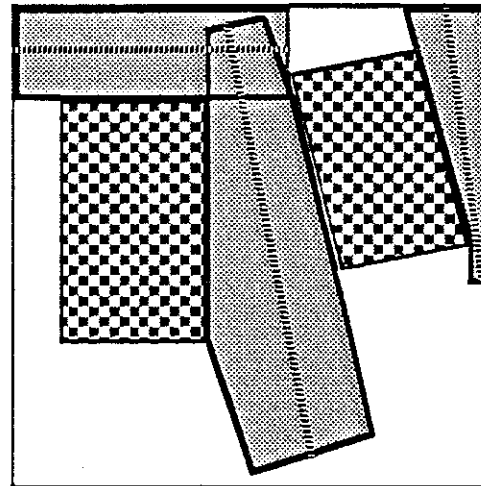
Brooks finds paths through 2-space by using a spatial representation that views free space as the union of, possibly overlapping, generalized cones. This representation is used because the cones provide a description of areas swept out by a moving 2-space object. Figure 3 illustrates how a few of the cones in the given space would appear. The lines bisecting the cones indicate how a robot will move through a given cone. Path finding using this spatial representation then reduces to comparing the area swept out by an object with cones representing free space. Brooks accomplishes the path search using a traditional search technique called the A* algorithm [Nilsson71]. The upper bound on the time complexity of his system is $O(n^4)$ (where n is related to the size of the space).

Among the advantages of Brooks' system is that it deals with the rotation and translation of robot movements. The system also finds paths that move the robot through the space while maintaining the maximal distance from objects occupying the space. The disadvantages of this approach are that the shortest path is the only measure of path quality, and there is no mechanism for dealing with the temporal aspects of movement through space.

Nguyen's work builds on the work of Brooks and others, providing a expedient heuristic to search through the generalized cones representing free space. The approach involves first finding local paths using experts that compute paths along the links between the cones, then applying the A* algorithm. Because the A* algorithm is searching over a collection of local paths for the best global path, the complexity of the algorithm is reduced as compared to Brooks' system. Despite this improvement in the search heuristic, this technique provides no new paradigm for performing route planning.

1.2.3 Knowledge Bases

The SPAM (SPAtial Module) program developed by McDermott is a knowledge base that handles spatial information about a given environment [McDermott84]. The system uses two kinds of knowledge; positional and relational. Positional knowledge, maintained as an assertional data base, represents topological facts such as (in stairway23 building7). The positional knowledge can be seen as a semantic network consisting of a predicate calculus encoding of facts. Relational knowledge, on the other hand, capturing information such as (between Illinois NY California), is represented as a "fuzzy map". The reason that the representation of relational information is referred to as "fuzzy maps" is that it represents facts as a range of values each of which is embedded in a frame of reference. Each frame of reference has its own scale, origin and orientation, all stored as ranges of possible values. This is done to allow inferences about an object to be made simply by looking in the proper map of a given frame of reference.





-  Generalized Cones
-  Objects

Figure 3 Some generalized cones

Path planning using such a spatial representation is done by first finding a fuzzy vector indicating the path from the current location to the destination. The vector is then translated into a list of barriers that must be entered or exited in order to reach the destination. Each of these barriers in turn is searched for a traversable path. As the route is traversed, the original fuzzy vector is refined to provide a better path plan. This process is continued until the destination is reached.

The advantages of this approach include its ability to describe objects in a "fuzzy" manner, thus allowing uncertainty in the position and orientations of objects to be accommodated. The system also provides a framework for the acquisition and assimilation of data when planning in uncertain domains. Another advantage lies in the systems ability to answer queries such as "What Chinese restaurants are within a mile of the Eiffel tower?". The disadvantage of the system is its inability to modify a plan if it finds that the original path has become unexpectedly blocked. The system also fails to address any of the temporal aspects of route planning.

Davis implemented a system he called Mercator, which builds on the work he did with McDermott on the SPAM system [Davis84]. Mercator overcame shape representation and generality problems of the SPAM system by representing the exteriors of obstacles as the edges of a highly connected graph. This allowed detailed knowledge of the environment and its accompanying uncertainty to be captured, and from this information, route plans could be generated. Mercator also handles problems involving spatial containment. For example, consider a house located on the edge of a large field and the following query about the location of the house: "How far is the house from the field?". The SPAM system would respond with the distance between the center of the house and the center of the field, while Mercator would say that the house is in the field. While Mercator is a powerful system for answering queries about the spatial relations between objects, its problem as a route planner are two-fold. First, it is not setup to deal with dynamic situations in a manner that will allow real-time operation, and second, it cannot operate on temporally related problems.

1.2.4 Potential Fields

This approach to path planning has been taken by Andrews [Andrews83] and others. The spatial model used in this approach can be seen as placing the destination at the bottom of a hill (the point of minimum potential energy), and all of the space radially around the destination at increasingly higher locations up the hill. Thus, the space that is furthest from the destination is highest up the hill. This gives rise to the relationship that the further away from the destination an object is, the greater its potential energy. Path planning in this spatial model is then viewed as releasing a ball from the robot's current position and seeing how the ball reaches the bottom of the destination. There are some obvious drawbacks to such an approach that make it impractical for doing general path planning. Besides ignoring all temporal aspects of

path planning, it will not consider any path that must first move the ball uphill away from the destination before proceeding down to the destination. Another disadvantage is the need for special mechanisms that backtrack when the ball enters a local minimum.

1.2.5 Vertex Graphs

This class of algorithms is based on a graph that connects pairs of vertices. The vertices usually represent corners of convex hulls that represent objects that must be navigated. The actual graph is most commonly constructed by associating an edge between those vertices pairs that can be connected by a line that does not intersect an object. A number of local optimizations are available which provide a reasonable method for pruning the search space of the graph. Among the researchers that have successfully used this approach are Laumond [Laumond83], Iyengar [Iyengar85], and Chatila [Chatila85].

Among the problems associated with this approach to path planning is that routes will be calculated that move "too close" to an object. These systems solve this too close problem by enlarging the size of the objects occupying the space. This method also has a very limited definition of what constitutes the best path; shortest is best. These systems also fail to provide a framework for dealing with temporal constraints.

1.2.6 Configuration Space

Configuration space has been used as a spatial representation by a few path planning systems. Lozano-Perez has given this approach a thorough mathematical treatment [Lozano-Perez83]. The principle to this approach is to define each object (rigid solid) as a vector, referred to as its configuration. The vector contains the pertinent information on the objects position and orientation in space. In a similar manner, there is a configuration for the robot. The ways in which the configuration space of the robot interact with the configurations of the objects yields information that is used by a path planner to find the shortest path between two points in the space.

This approach is best suited for calculations involving the manipulation of a mechanical arm through a cluttered work space. It can, however, be used for path planning and will successfully deal with plans involving asymmetric robot dimensions. The approach does not provide a mechanism for dealing with unpredictable dynamic objects nor does it provide or use any information about the quality of the space in which plans are executed.

1.2.6 Grid Representations of Space

Systems that approach spatial representation by using a grid to model space have spatial representations that most closely resemble the model of space used by this research. In 2-space, grid representations are the partitioning of space into equal sized squares upon which path planning schemes are applied. Among the researchers that have used this approach productively are Thorpe and Moravec.

Thorpe's system accomplishes path planning in two steps [Thorpe84]. The system first does a global search over an eight-connected 2-space grid of points, to find a rough path to the destination. The second step then "relaxes" the points on the rough path in order to improve the path quality. Using this technique, Thorpe's system is able to find safe paths through a given 2-space containing static objects. A similar path relaxation technique was also used by Moravec as part of Fido, the navigational system in the CMU Rover [Moravec82].

The major advantage of this approach to path planning is that it provides a framework for dealing with time related aspects of spatial movement. Although not directly addressed in the research, modeling time in these systems corresponds roughly to the addition of an extra dimension to the spatial representation. The added dimension is used to represent the temporal aspect of planning. There is a drawback to systems that use relaxation techniques for finding paths: If the rough path is too rough the system will settle to a local minimum, missing the true global minimum.

1.3 Planning Routes Through Interesting Spaces

Many of the systems covered in the last section suffer from the shortest path obsession. The systems are solely concerned with finding the shortest path while neglecting other qualities that make up the optimal path. For example, none of the systems accounts for the quality of the surface upon which the robot travels, relying on the surface being either traversable or not. This naive treatment of the surface over which the the robot will travel is simply inadequate in real-world planning systems. The most severe limitation of the aforementioned systems is that they assume that the only dynamic object is the robot. This is totally unrealistic and imposes an unacceptable, lab operation only, limitation upon the domains in which the system can operate.

In order to function in a dynamic world, a robot needs to consider the operation of other dynamic processes sharing the space and how they may affect its operation during path planning. For example, Clyde the elephant, knows the local train schedule and needs to get to the other side of the train tracks. He should use that information when planning to get across the tracks. If Clyde has information predicting that a long freight train will be coming just before he can reach the tracks, then given the choice between a short path that involves crossing the train tracks, and a slightly longer plan to go under the tracks, Clyde should choose the latter plan. Similarly, if Clyde's task were to jump on a moving trolley car, then the ability to plot a path that will allow him to jump onto a moving object is necessary.

Unpredictable dynamic processes must also be accounted for by a robust route planning system. Clyde the elephant who is a member of the cavalry and is returning to the safety of the fort. Above all Clyde "fears" attack by a brutal tribe of Native American mice. Clyde would be better off planning to get to the fort across the open plain, rather than traveling through the narrow passageway of *Ambush Canyon*. The

primary reason for Clyde's choice is that an attack in the canyon would effectively block him from his destination, thereby causing the him to backtrack wasting valuable time.

This thesis describes a route planning system called Robonav, that can handle the problems described above. The single property that most distinguishes this work from previous systems is that it deals with both the spatial and temporal constraints imposed by the domain in which the system is operating. That is, Robonav considers the time at which a particular area of space is traversed when determining the optimal path. Robonav allows the designer to model the quality of space that the robot is moving through and use this information in determining the best path. The designer can also model the effects that dynamic objects have on the definition of the optimal path. Constraints involving the abilities and limitations of the robot executing generated plans can also be incorporated into the model. Finally, Robonav has a direct mapping into a parallel architecture, allowing it to operate in unpredictable domains in real-time. The ability to operate in real-time in an unpredictable domain is important for systems operating in a threatening environment.

1.4 An Outline of the Thesis

Chapter 2 covers the spatial representation and path planning algorithms. The spatial representation is used to capture the relevant aspects of space. The spatial representation provides the framework for algorithms that perform the path planning task.

Chapter 3 deals with object representation using the spatial model presented in Chapter 2 and how objects affect the operation of the path planning algorithms. An object is a function of both space and time that maps into the spatial representation.

Chapter 4 is concerned with finding paths that measure qualities, other than length, in determining the best path. In this chapter a number of qualities will be defined along with evaluation functions that operate on these qualities to find the best path.

Chapter 5 gives a brief explanation of how this model can be used in domains where there are unpredictable processes to be accommodated for during actual plan execution. The solution presented alternates between planning the next best step and updating of the state of the world.

Chapter 6 covers some of the conclusions reached as a result of this research. Also in this final chapter are implementational details of the system as well as details on further research needed to extend the system's abilities.

Chapter 2

Path Planning in Dynamic Domains

This section will discuss three similar, yet computationally distinct, message passing algorithms. The algorithms, when combined with the proper spatial-temporal representation, are able to find paths through static and dynamic domains. The basis of the message passing techniques lies in the choice of the proper spatial representation. The spatial representation views space in a discrete manner, as if it were composed of equally sized particles of space. Once the spatial representation has been presented, the three message passing techniques will be described. The first simply finds the shortest path through n -space between two points (source and destination). The second makes modifications to this approach, allowing the destination to be described as a function of time. The final algorithm makes a further modification, allowing the real-time movement capabilities of the robot to be considered in path generation.

2.1 Spatial Representation

For a route planning technique to be effective, it must use a spatial model capable of capturing the essential aspects of the space through which paths are planned. To represent space, this research uses uniformly shaped n -dimensional hyper-cubes called "nodes". Each of the nodes represents a small element of space in a given dimension. For example, in one dimension the nodes are line segments; in two dimensions the nodes are squares; and in three dimensions the nodes are cubes (see Figure 4). This approach to spatial representation has been taken in many previous works. In particular, Thorpe used it as a basis for path planning in two-dimensional static domains [Thorpe84]. Thorpe used the representation in combination with a relaxation algorithm to find locally optimal paths.

Arbitrarily shaped n -dimensional spaces are defined by spatially concatenating nodes along common ($n-1$ dimensional) surfaces. The collective area occupied by the nodes is called "space", while the remaining area is referred to as "void". For example, Figure 5 shows examples of arbitrarily shaped two and three-dimensional spaces constructed from squares and cubes respectively. In general, the size of the nodes will be of at least sufficient size to subsume the size of the robot. This size restriction is based on how objects in n -space are represented.

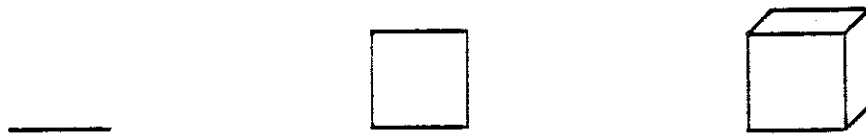


Figure 4 Nodes in 1, 2 and 3-space

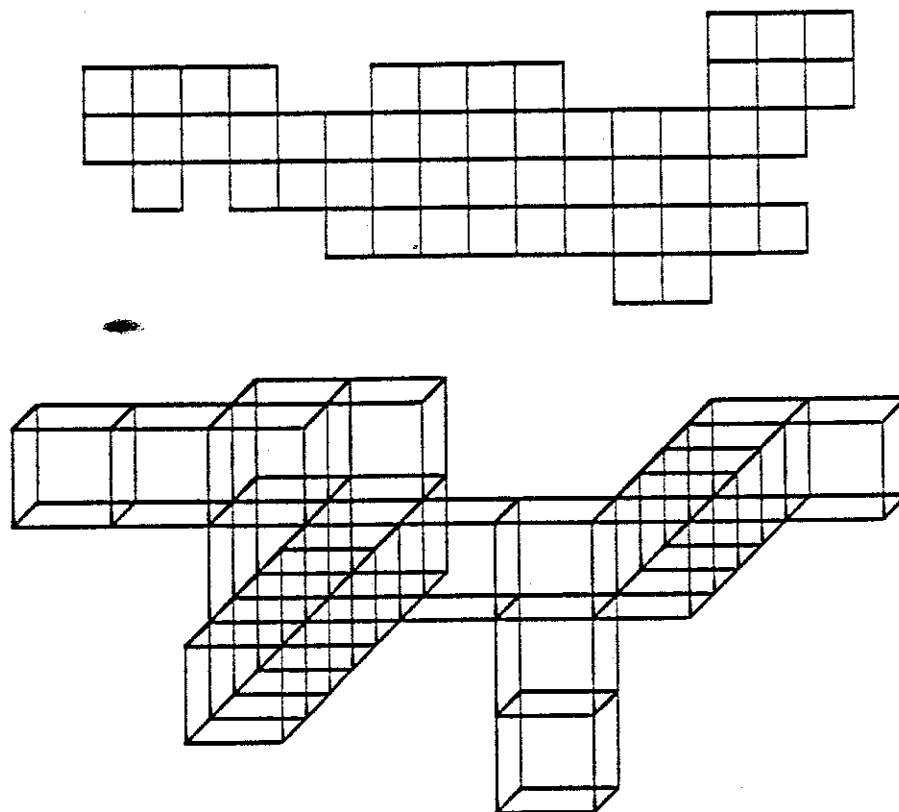


Figure 5 Arbitrary examples of 2 and 3-space

To provide a mechanism for message passing, each of the nodes making up space has a unidirectional link connecting it with each of its $2n$ possible neighboring nodes (i.e. diagonals not included). For example, consider the links that are associated with the two-dimensional space shown below in Figure 6. Notice in the figure that the nodes at the edge of space have fewer connecting links than nodes that are oriented towards the middle of the space. To capture spatial and temporal "quality", each of the links has a cost associated with it. The cost is a function of time and n-space, used to determine the quality of paths through space and time. For example, consider Clyde the commuting elephant: Clyde may plan different paths to the office depending on the time of day (e.g., the freeways are always backed up from 7am to 9am and 4pm to 6pm Monday through Friday, so he takes the back streets if going to or from the office at these times). In general, the links will be used to represent the relative cost of making a transition from one node to another at a particular time.

2.2 Finding Paths Through Space

Using nodes, spatial concatenation, and links, a message passing algorithm will be defined that finds the shortest path through n-space between two given points (ignoring the temporal aspects for the moment). The algorithm is quite similar in nature to the technique used in [Hillis85] for finding the shortest path through a graph. The basic principal used in finding paths by message passing can be visualized as waves moving across a body of water (see Figure 7).

Picture a stone thrown into a pond. The location on the surface of the pond where the stone impacts with the water, represents the current location of the robot or the "source" point. The impacting stone starts a disturbance on the surface of the pond, sending waves out from the source point in all directions. The propagating waves represent a search over the surface of the pond for the desired "destination" point, located somewhere on the surface of the pond. Eventually one of the waves passes over the destination point indicating the existence of a path from the source to the destination.

This symbolizes a message passing algorithm for finding paths over the surface of the pond. The algorithm described in this section operates in a similar fashion, exploiting a message passing scheme to model the movement of a wave through a given n-space. The objective of the algorithm is simply to find the shortest path from the source to the destination. For the model to operate properly, the representation must capture aspects of the path's length and feasibility. To capture the path length, each link between nodes of the space, representing a transition that the robot is capable of making, is assigned a cost of 1.0. The remaining links reflect transitions between nodes that cannot be made by the robot are assigned an infinite cost (e.g., there may be a ledge between the nodes). Using this link cost assignment the necessary spatial aspects are captured. When combined with the path planning algorithm the shortest path through space can be found.

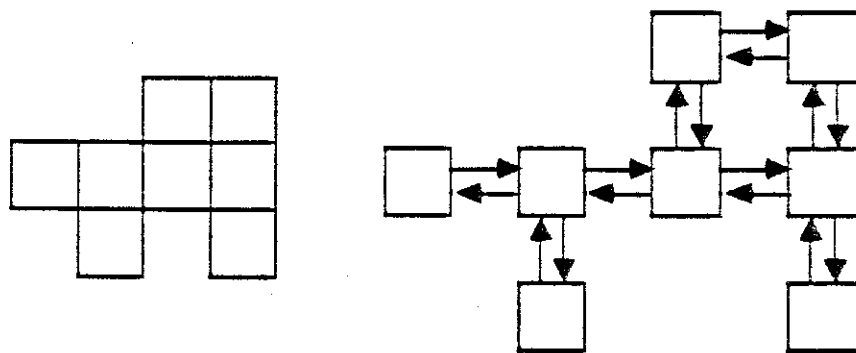


Figure 6 Links of a simple 2-space

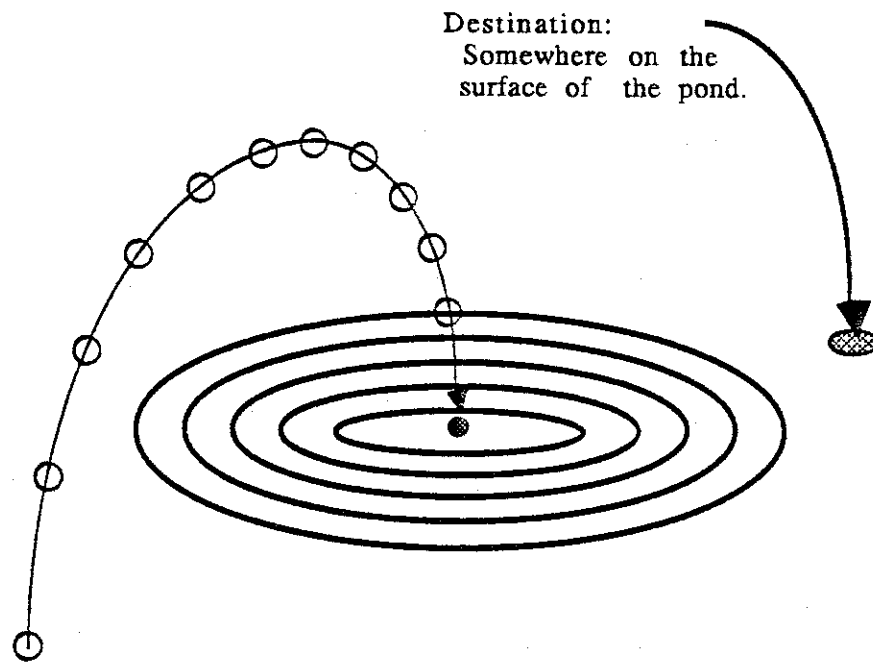


Figure 7 Searching for the destination in 2-space

Each message is a bundle of information that is passed from the node that creates it to one of its connected neighboring nodes. Messages have associated with them an energy value and a "genesis-pointer". The energy value is used to determine the relative value of a particular path through the space. The genesis-pointer points to the node where the message originated and is used to reclaim the path from the representation. Starting at the source node (i.e. the robot's current location), the algorithm uses a synchronous step-wise process of passing messages from node to node as it searches the n-space for a path to the destination. At each step in the process, nodes of the space receive a set of messages from their neighbors and create a new set of messages based on the received messages. The messages of this new message set are sent to the nodes neighboring nodes during the next time step. Using such a process, all possible paths that the robot could take through n-space in attaining the desired destination are considered. To reclaim the path generated by the propagation of the messages through n-space, each node in the given n-space has a "best-message" slot associated with it. This slot contains the message received by the node with the lowest energy value of all the messages it has received during the operation of the algorithm. The actual path planning algorithm is described by its two phases and bounding condition.

2.2.1 Phase 1: "Throwing the Stone"

The first phase sets up the initial message set. At the source node (associated with the robot's current location), a set of messages are created. The set contains one message for each link the source node has with its neighboring nodes (i.e. up to $2n$ messages). Figure 8 shows a source node in 3-space and its connected neighbors; in this case six messages are created for the initial message set. The energy value associated with each message takes on the value of the cost of the link that the message is to travel over. The genesis-pointer of each of the nodes is set to point to the source node. To initialize the model, the best-message slot of the source node gets a nil message set with an energy level of 0.0. Finally, each of the messages in the created set is sent over a link to its intended neighboring node.

2.2.2 Phase 2: "The propagating wave front"

This is the operational phase of the algorithm and consists of having each node in the n-space repetitively perform the following set of tasks in a synchronous step-wise manner:

First, gather together all of the incoming messages sent to it during the last step.

Second, identify the base-message as the incoming message with the minimum energy. All other messages are discarded out because they represent equivalently long or longer paths that attain the same location in space. Note, if there is no incoming message, the node does nothing during the current synchronous step.

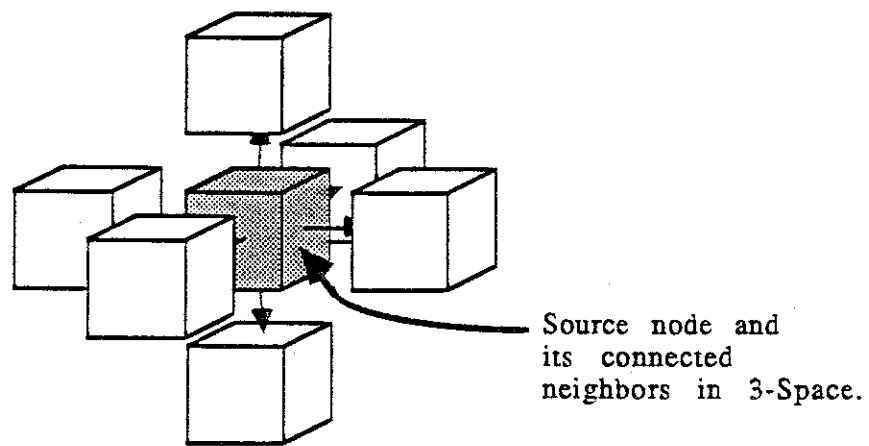


Figure 8 Phase one

Third, do one of the following two actions depending on whether the energy in the base-message greater than or equal to the energy of the message in the best message slot of the node. If the condition is fulfilled, then nothing is done and the node waits for the next step. If the condition fails, or the best message slot is empty, then a new set of messages is created in a manner similar to that described in the first phase. Each message in the new set is assigned an energy that is the sum of the energy in the base-message plus the cost of the link that the message is to be sent over. After the creation of the new message list, the base-message is placed into the best-message slot of the node, thereby replacing any message that might have been there. Finally, the node sends the newly created messages out along their respective links.

The process described in this phase is repeated until the bounding condition is met. Figure 9 shows the spreading node activation in some given 3-space. The figure shows the shape of the expanding wave of messages as they move out into 3-space.

2.2.3 Termination: "A Wave reaches destination"

If the algorithm operates in phase two long enough, it will find a minimum energy path from the source node to every node in the n-space. In this case, the bounding condition is defined as the state of the system when there is no remaining message activity. It should be noted that, as defined, for a finite space inactivity of messages is guaranteed to occur in a finite number of steps. While this will provide a path to the destination, it may operate for many more steps than is necessary for finding such a path. Therefore, to bound the running time of the algorithm, a more restrictive bounding condition must be employed. The bounding condition is defined as the state of the system when the energy associated with each of the messages currently being processed in the system is greater than or equal to the "global bound". The global bound is equal to the energy of the message in the best-message slot of the destination node (similar to zorch decay [Charniak86]). Note, if such a message does not exist then the current bound is considered to be infinite.

After the bounding condition is met, the path through n-space that has the lowest energy associated with it can be retrieved from the space of nodes. This is done by starting at the message in the best-message slot of the destination node and recursively following the genesis-pointer of the current node's best-message until the nil message at the source location is encountered.

Some observations should be made at this point that will clarify the functioning of the algorithm. The algorithm has an obvious isomorphic mapping onto an appropriately connected SIMD (Single Instruction Multiple Data) machine architecture. This gives rise to pragmatic concerns such as the extensive waste of computing power. The waste is easier to visualize when realizing that the propagating messages only

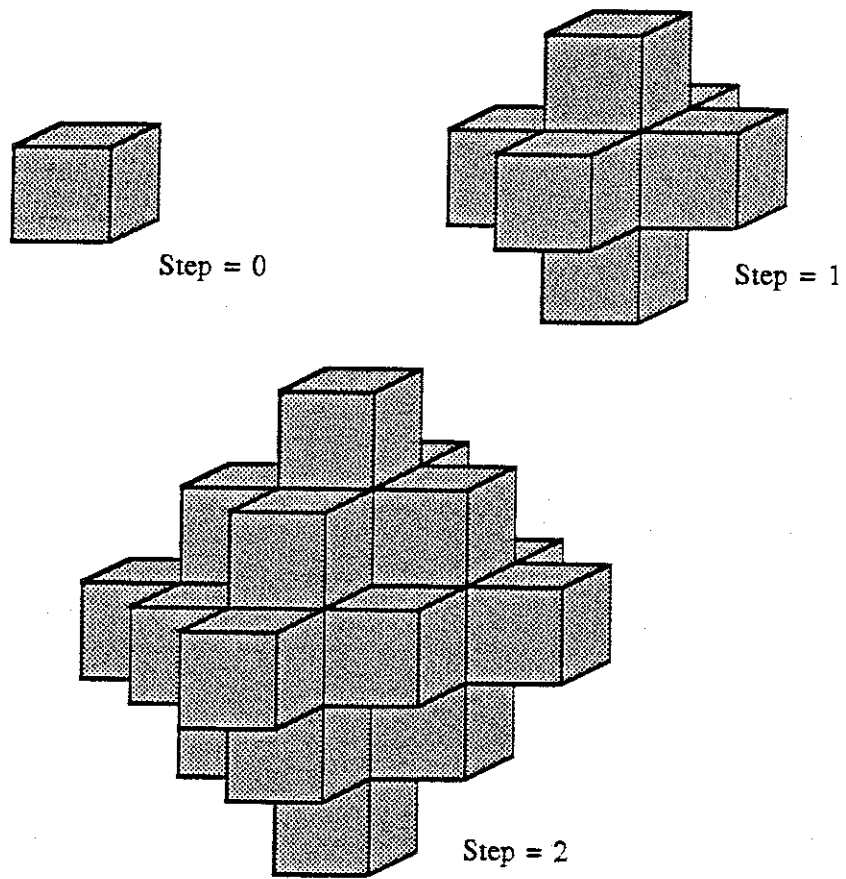


Figure 9 Phase 2

move away from the source and never back up over previously visited nodes. Thus, nodes sit idle for all but 2 of the p steps in the path length. The reason for this is, any message created will have an energy value that is one greater or infinitely greater (all link costs are 1.0 or infinite) than the energy of the message that generated it. Therefore, the message passing activity of the messages sent to any previously active nodes will die out. A further concern is how the path length is acquired. Intuitively, it can be seen that any message reaching the destination will have an energy value that is simply 1.0 times the length of the path that the message represents. Thus, by observing how the bounding condition is defined, one can see that when the first message reaches the destination all messages in the system will have an energy value equal to 1.0 times the minimum path length (messages that go over links with an infinite cost die out immediately). This, however, is exactly the definition of the bounding condition and the system will halt.

2.3 Finding Paths Through Space and Time

The previous section discussed an algorithm that finds the shortest path between two spatial locations. While a useful algorithm, there have been many systems designed that are capable of finding paths between two static points in space. In this section, the current algorithm will be modified in such a way as to enable it to find paths through time as well as n -space. This will enable the destination to be described as a function of both time and n -space. For example, consider Clyde Cassidy robot. For Clyde to be successful he needs to have the ability to plan routes through n -space allowing him to seek a moving destination (e.g. a moving train). Immediately it becomes clear that the paths must consider the value of not moving; it may be advantageous for the robot to wait in one location for the destination to move closer before moving toward the destination.

To provide these new capabilities, the following modifications will be made to both the current spatial representation and algorithm. First, the concept of time must be defined; a unit of time is represented by each synchronous step in the second phase of the algorithm. This assumes that the robot can make a transition from any node to any one of its connected neighbors in one unit of time regardless of the robot's current state. Note, this simplifying assumption will be removed in the next section where modeling continuous time is considered. Second, in order to consider paths that involve non-moves as a step in a route plan, each node must have a reflexive relationship with itself. To accomplish this, a link pointing back to the node is added to each node in n -space (see Figure 10). The reflexive link has all of the same characteristics as the links previously described. Third, the best-message slot of each node is changed to a stack of best-messages. The stack exists so paths involving steps that remain in one location can be reclaimed from the nodes after the bounding condition has been met. The stack also minimizes the amount of inter-node (inter-processor) communication, for without the stack, a list representing the ancestry of each message would have to be passed along with each message in the system.

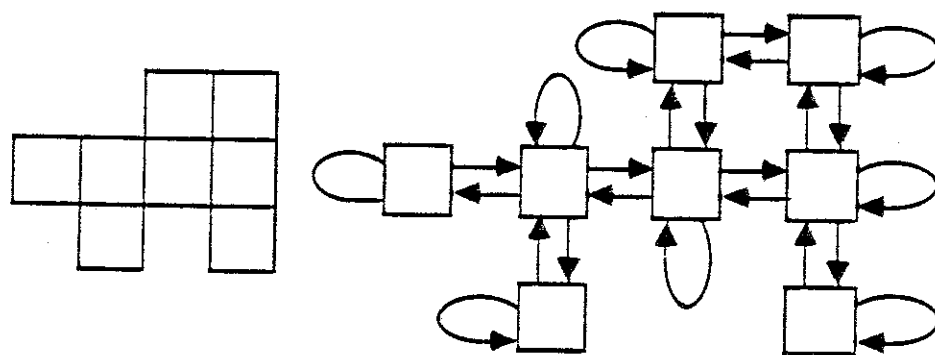


Figure 10 Reflexive links of a simple 2-space

Now that these modifications have been made to the spatial representation, an algorithm that finds paths through this new domain will be defined. As in the previous section, this algorithm is defined by its two phases and bounding condition. The definitions of the phases and the bounding condition are taken from the previously described algorithm and modified to extend the algorithm's power in this new domain.

Phase one is defined almost exactly as in the previous algorithm. There is one new aspect worth noting, however, namely that in addition to the messages previously prepared, there is also a message created and sent out over the reflexive link. This is done to check whether not moving during the first step is more profitable than making a move into one of the source nodes connected neighbors.

Phase two is also a modification of the previous algorithm. In the previous algorithm, if the base-message was not of a lower energy value than the message in the node's best message slot, it was thrown out and the node did not perform any further action during that time step. In the new definition, if the base-message energy is less than the current global bound, the node will always create a new set of messages from the base-message (including one for the reflexive link). The messages in the new set are then sent out over their respective links, initiating the subsequent time step. The reason for a node taking no action when the base-message energy is greater than or equal to the current global bound is because message energies are strictly increasing. In other words, the cost on each of the links is greater than zero. Therefore, any path generated from the node of concern to any of the destination points will provide a path that is of greater energy than the best path known at the current time. Finally, if the base-message has an energy less than that of the global bound, then after the new message set is created the base-message is pushed on the top of the best-message stack. However, if the base-message has an energy value greater than or equal to the global bound or if there were no messages processed at all during the current time step, a place-holding nil message is pushed on the top of the stack.

The bounding condition is changed from that previously defined, enabling the system to consider the destination location as a function of both time and n-space. This is accomplished by simply redefining the global bound to be the minimum energy of all messages that have reached any of the destination nodes. The destination node is no longer simply a location in n-space; it is a location in n-space during a given time period. It should be noted that there can easily be more than one destination location defined during the same time period. For example, consider Clyde who must get to the post office to mail a package to his mother. Clyde's problem is that he lives in a city with two post offices that are each open from 8am to 5pm Monday through Friday. Therefore, Clyde plans a path considering the trade-offs between the two destination locations.

In order to retrieve the path from the stacks of the nodes, the node containing the globally bounding message is identified. Then, until the message associated with the global bound appears at the top of the identified node's stack, all of the nodes in n-space pop their best-message stacks and discard the popped message. At this point, the best path is encoded in the ancestry of this message. So, in an iterative manner, the message's ancestry is found by following genesis-pointers back node by node. At each step in the process, all nodes in the space pop their stacks revealing the next message on the stack. The process of following pointers and popping stacks is continued until all of the stacks are empty, indicating that the source location has been encountered. Note that due to the use of nil messages as place holders, all of the stacks will always have the same depth.

Using this modified algorithm, the system can solve the following problem: Find the optimal time to leave home in order to catch the 4:30 bus at the corner of 1st and Elm streets? This is accomplished by assigning a link cost of 0.0 between all of the links inside the house and a cost of 1.0 to all of the links outside of the house. Thus, the shortest path from the house to the bus stop will result, with the further restriction that the path will not leave the house until the last possible moment. Note that when a node has incoming messages of equivalent values, the selection of the base-message will affect the robot's movements within the house.

2.4 Modeling A Robots Continuous Time Actions

This section will show how the robot's locomotion capabilities can be considered when finding paths through n-space and time. In the last section the fundamentals of time modeling were described. However, an absolute and restrictive assumption was made about the representation of time. Namely, all node to node transitions made by the robot take single uniform units of time. By making a modification to the messages being passed from node to node, the assumption can be removed. The modification involves placing a real-time field in each of the messages. The time field allows messages to contain local times, thus removing the temporal restriction about the robot's abilities to move through space.

The ability to effectively represent the time required by a robot to navigate through space allows plans to be generated that take advantage of the robot's abilities. For example, the transition from one node to another when starting from a resting state should take longer than the same transition when movement has already been initiated in that direction. This is significantly different from the scheme used up to this point, where all moves were considered to take one unit of time. The ability to consider the capabilities of the robot in generated paths has been incorporated into the Robonav system by operating it in a more asynchronous manner. Asynchrony is accomplished by associating a real-time with each message in the system. The time value of created messages is set by summing together the time in the base-message and the time required for the robot to make the move represented by the new message. This process must account for the current state of the robot in determination of the time required to make the next transition. The

ability to effectively predict the performance of the robot is bounded by the precision with which the real-time actions of the robot moving through space-time can be modeled. To model the robot's actions with any precision, each message must keep some representation of its history (e.g., the current state of the robot as reflected by the given messages ancestry). For example, a message must "know" if the robot is currently turning, moving or stationary in determining the time to associate with it. There are many ways to represent this information, all of which are dependent on the robot being modeled. The history can also be represented by a list of attributes tagged onto each message indicating the current state of the robot as defined by a particular message. The stacks must also be modified to push and pop sets of best messages, this is done because the message times are no longer synchronized.

Although this algorithm has an increased message passing complexity, it also has an increased computational power over the previous algorithms. For example, consider the 2-space shown in Figure 11. The figure shows a situation where there are two paths that lead from the robot's location to the desired destination. Using the previous schemes, both of the paths would have equivalent value (assuming that the link weights are all 1.0). With the new scheme, however, each message has a real-time associated with it. Consider the following time complexity assignments: Messages representing the start of motion in some direction from a rest position or a change in the direction of motion have a time complexity of 2.0. Messages representing continuous movement in one direction are given a time complexity of 1.0. Using these assignments, messages from path one will reach the destination at time $t=7$ and messages from path two will reach the destination at time $t=9$. The information provided by such a system is necessary for any system that must meet scheduling constraints (e.g., factory automation).

2.5 Concluding Remarks: Algorithms and Models

This chapter presented the concept of message passing and introduced three different path planning algorithms operating on three different spatial models. The first algorithm, operating on the simplest spatial model, is easy to visualize but fairly trivial in its expressive power. This limitation is caused by the algorithm's inability to effectively operate in dynamic domains. The second algorithm, operating on a modified spatial model has considerably more power, able to utilize both spatial and temporal constraints in planning paths. While this is a vast improvement to the first algorithm, it has discretely uniform robot movements. This simplifying assumption was made at the loss of considerable power. The third algorithm is able to consider the real-time movement capabilities of the robot. However, this expressiveness sacrifices descriptive clarity.

Henceforth, all examples and discussions presented will be set in the frame of the second algorithm and associated spatial model. It should be noted that all of the modeling techniques that follow will operate equally well on the third algorithm and spatial model, with minor modifications made to account for the asynchronous operation of the third algorithm.

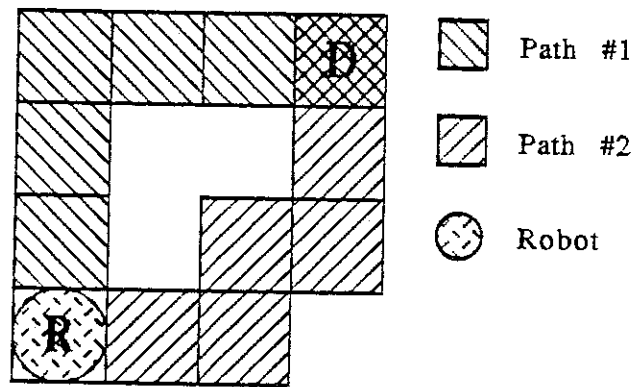


Figure 11 Shortest path through time

Chapter 3

Representing Objects

Object representation is a critical aspect of an effective route planning algorithm expected to operate in real-world domains. This chapter will describe how objects within the defined space are modeled. It will show how the planning algorithm uses the object representation to plan paths that account for the temporal aspects of object interactions. The objects presented in this chapter are predictable; objects are both static and dynamic with fully defined behavior in both time and space. The spatial model and path-finding algorithm developed in the last chapter will provide the basis for the spatial and temporal aspects of object representation.

3.1 Predictable Objects in n-Space and Time

Objects are represented as functions having a given n-space and time in their domain, and are mapped into some subset of the nodes making up n-space during a given period of time. The set of nodes generated by an object's function consists of those nodes in the given space that are occupied (fully or partially) by the object during the given time period. For example, Figure 12 shows how a model of a simple two-dimensional revolving door can be generated. The revolving door is defined by a function that has four nodes forming a square in its range. The function maps onto two of the diagonally adjacent nodes during odd time units and onto the other two diagonally adjacent nodes during even time units.

To find paths through spaces that have predictable objects moving through them, the operation of the message passing algorithm must be modified in its second phase. The modification simply makes any node occupied by an object become inactive during the time period that the object occupies the node. This keeps paths that pass through an object from being generated by the Robonav system.

Consider the example 1-space shown in Figure 13 consisting of three nodes (referred to as left, middle and right nodes). The fact that this example comes from 1-space makes moving right or moving left the only allowable moves, with the appropriate restrictions made at the ends of the 1-space. The objective is to find a path through 1-space that allows

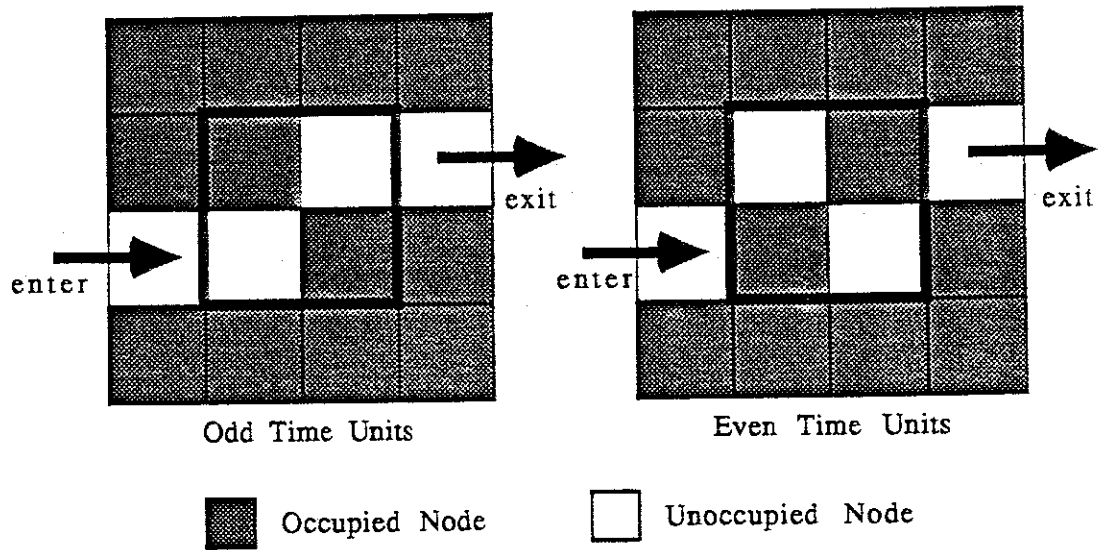


Figure 12 Representing a revolving door

the robot to move from one end of the space to the other in the shortest possible amount of time. Consider the cost on all seven of the links (four neighbor and three reflexive links) in the space to be assigned a value of 1.0. Also shown in the figure are the best-message stacks of each of the nodes and the state of these stacks during the different time periods. Observe that there are two predictable objects whose behavior must be planned for if a safe path is to be found. The first enters the 1-space occupying the left node at the start of time $t=1$. This causes all messages received by the left node at time $t=1$ to be ignored. This is indicated in the stack of the left node at time $t=1$ by the box around the message on the top of the stack. The object leaves the left node before the start of time $t=2$. The other object starts out occupying the right node at the start of time $t=2$, thus preventing the robot from attaining the destination. Now the object moves to the middle node, forcing the robot back to the left node at time $t=3$. At time $t=4$ the object moves back to the right node leaving the middle node before the onset of time $t=5$. The figure shows the actual robot movements instantiated. Actually the robot would not move until the entire path was planned; after time $t=6$ was planned.

First observe that nodes occupied during some time period send no messages into the next time period. This is the mechanism behind finding paths through space and time that avoid objects. Second, observe that figure 13 provides another example of why the stack is needed to store the path. If there was no stack, then paths that need to move back upon themselves could not be generated. Clearly, as the example illustrates, some path planning problems must use this backtracking tactic to find paths. Lastly, the example shows how the generated path can be retrieved from the stacks of the nodes, as indicated by the stacks associated with the state of the space at time $t=5$.

The following question now logically arises; where does object representation actually lie? One answer places the functions for the objects outside of the model. This approach introduces an external communication requirement to get the information about object location to the proper nodes in n -space. This has the disadvantage of introducing a sequential aspect to the otherwise parallel planner. Another approach is to have each node in the n -space maintain a map of times during which the node is occupied. This allows for all required external communication to be accomplished during setup time, thus maintaining a closed system during the algorithm's operation.

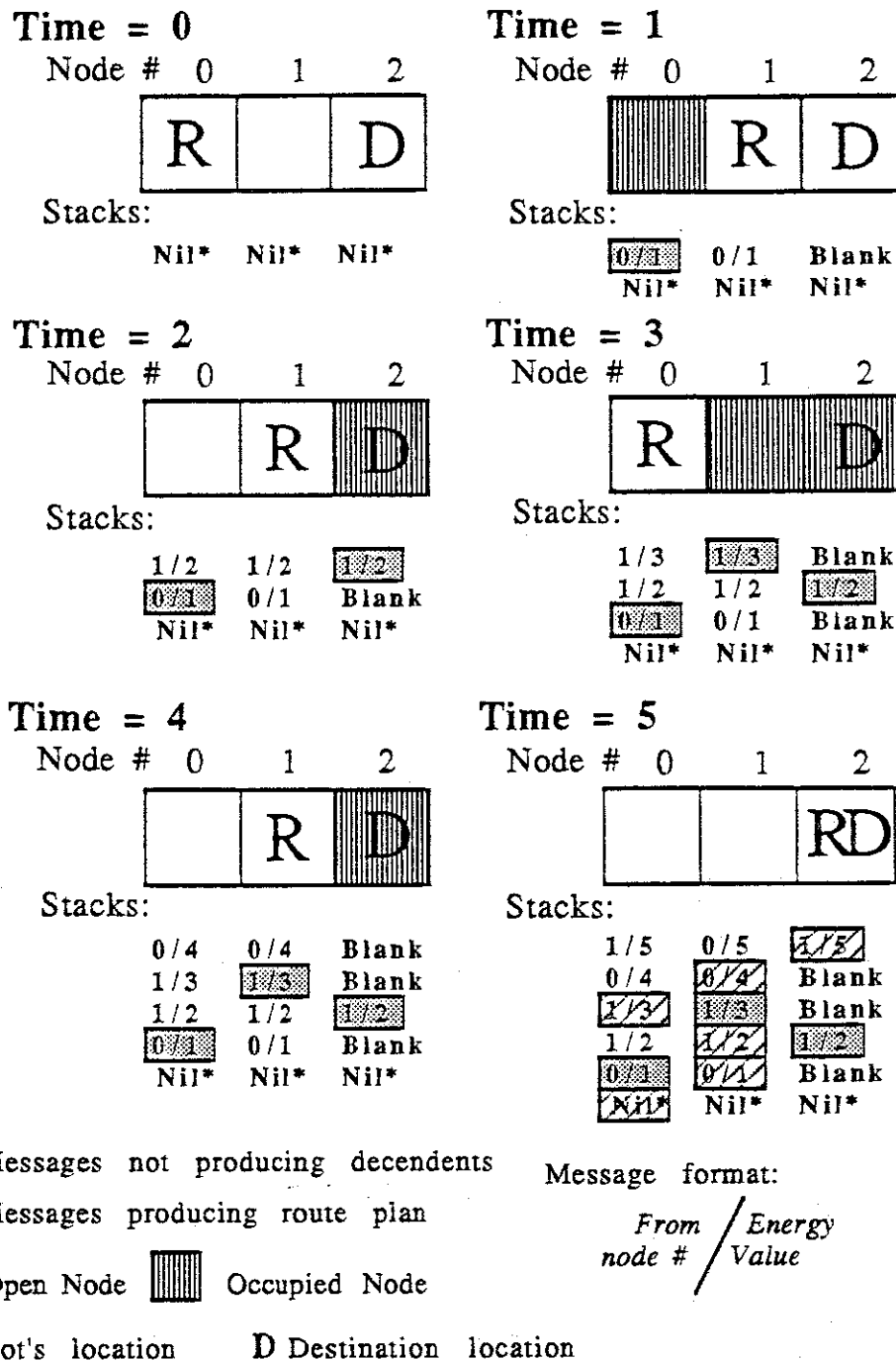


Figure 13 Path finding through a simple 1-space

3.2 Refining Object Representation

Consider the problem depicted in 2-space of Figure 14. The top portion of the figure shows a situation that cannot be solved with the system as it is currently defined. The objective in this situation is to move the robot from the top left corner to the bottom right corner of the space. But, the objects in the top right and bottom left corners will not allow any paths to be generated that reach the destination. This is due to the way in which objects are represented. Namely, any object that partially occupies a node during a given time is considered to entirely fill the node. However, in this example there should be a path to allow the robot to move to the destination. To overcome this problem, a technique that increases the resolution of the space by varying powers (2, 3, 4, . . .) is used. The principle of the technique lies in the addition of overlapping nodes. By using such a resolution technique, continuous n-space can be represented to varying degrees of precision. This resolution technique is similar to the course coding of state spaces used by Hinton [Hinton86]. For the given example, five new nodes, each the same size as the original nodes, are added to the spatial definition as shown in the bottom portion of the figure (note the inter-connection graph). As a result, there are now four possible paths that move the robot from the top left corner to the bottom right corner of the space. All of the resulting paths have a length of two standard node widths but require four half node width moves.

As a further example of the overlapping node solution to the object representation problem, consider the situation depicted in the top portion of Figure 15. Again the objective is to move the robot from the top left to the bottom right corner of the space. Once again there are objects in the top right and bottom left hand corners of the space (notice the different size of the objects). The objects prevent the algorithm from finding any path for the robot to take. As depicted in the middle portion of the figure, if the space is only resolved by a factor of two, there still is no object-free path available. Therefore, the space must be resolved by a factor of three, as shown in the bottom portion of the figure. When this is done, the system introduces twelve new nodes into the spatial representation (as compared with the original situation, which uses four nodes). By doing this, the system will find one of the many possible paths to the destination. The paths found will move through six nodes, each of one third width, resulting in a path that still only covers the distance of two standard width nodes. It should be noted that, while the resolution of the space remains finite, the total length of a path that moves between two diagonally adjacent nodes is two standard node widths (width of nodes at lowest resolution). However, as the resolution of the space approaches infinity, the length of a path between two diagonally adjacent nodes approaches $\sqrt{2}/2$ times the length of a standard node width. This, however, is only of theoretical interest as there is no direct way to provide the infinite resolution required to realize this savings in path length.

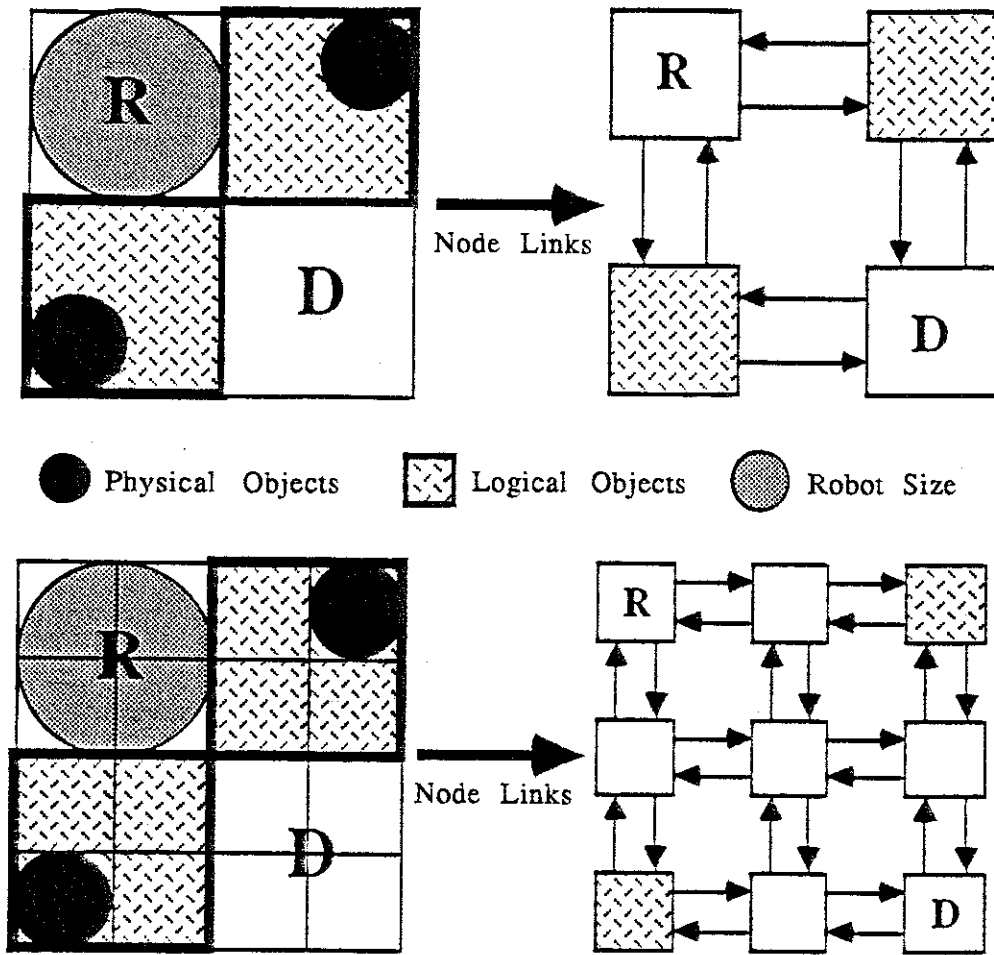


Figure 14 Overlapping nodes

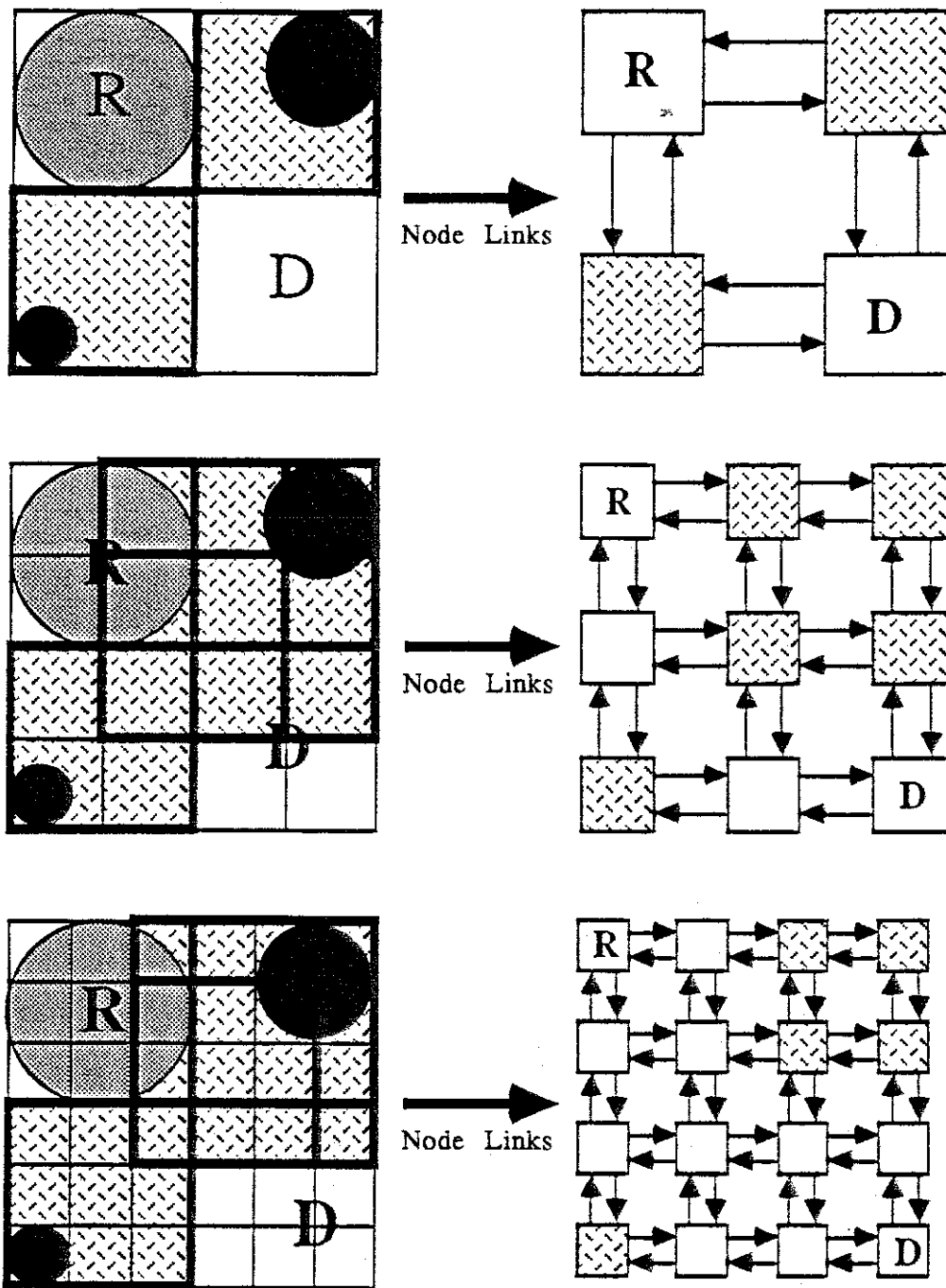


Figure 15 Resolution increasing to match the problem

The revolving door introduced in the previous section has an anomaly, there is no way to represent the direction in which the door is turning (see Figure 13). This is critical, however, because it is "bad" to plan paths that go the wrong way through a revolving door. To resolve this problem, consider the solution represented in Figure 16 below. The solution simply increases the resolution of the space by two in the area around the revolving door. Then, as is shown in the figure, there exists a path through space-time that will adequately consider the direction in which the door is turning. This figure also indicates how the node overlapping technique can be used in a local manner. This allows features that do not fall precisely onto the grid of the spatial map to be accurately represented.

3.3 Trade-offs Between Resolution and Efficiency

With increasing resolution there is a degradation in the performance of the algorithm's efficiency. For example, if the resolution of n -space is increased to $2x$, then the paths that are generated will, on the average, visit twice as many nodes as paths generated in the original scheme. To minimize this cost varying resolutions can be used. In other words higher resolutions can be used on a local level. This will more effectively utilize the computation power of the computer on which the algorithm is running. The use of overlapping nodes also causes problems with the timing of message passing in a second level model (section 2.3) due to the discreteness of time steps. Synchronization is, however, not a problem for the model of section 2.4, where there is a real-time field associated with the messages in the system.

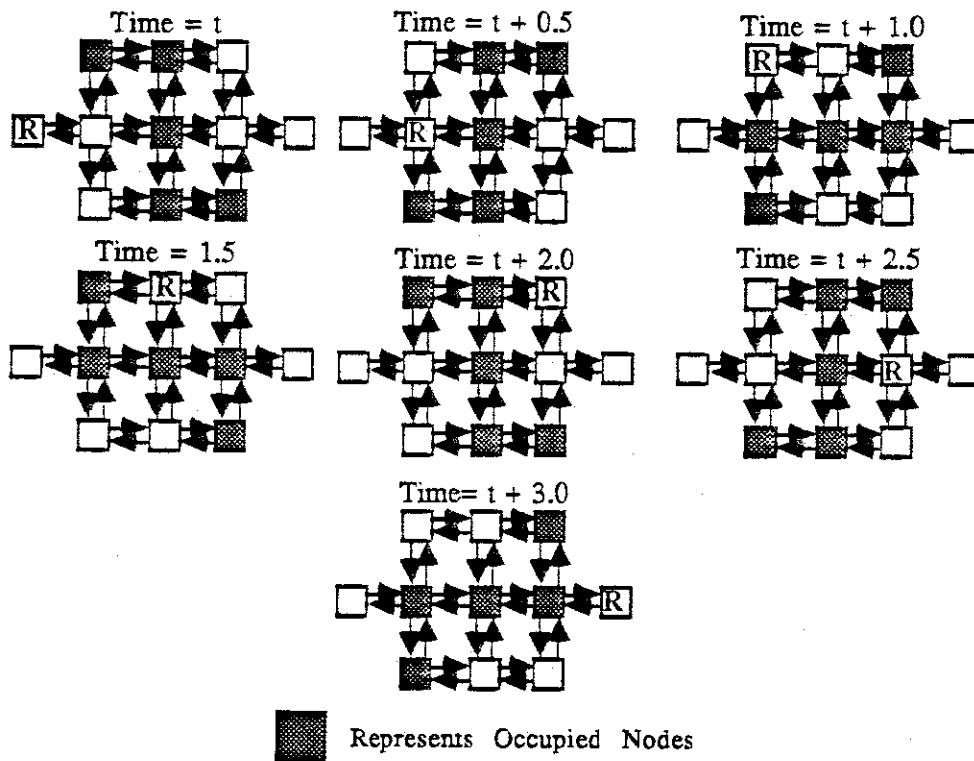
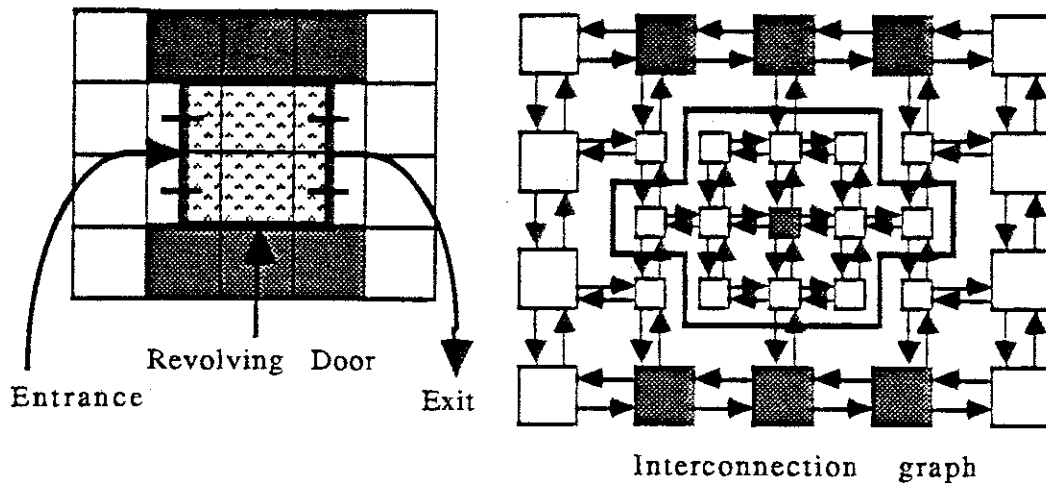


Figure 16 Refining the revolving door

Chapter 4

Evaluation Functions: Planning in Rich Spaces

The key feature to planning the best path through time and space with this model is the creation of properly defined evaluation functions. Evaluation functions provide a means of determining the quality of space and the abilities and limitations of a given robot. Evaluation functions manifest themselves by determining the cost that should be placed on the links in the space. Up until this point, the only evaluation function used to define the best path simply determined the shortest spatial or temporal path available. This section will explore more complicated evaluation functions. Before the evaluation functions can be discussed, the parameters on which they operate will be described. Presented below are three parameters used by evaluation functions to reflect the qualities of space: traversability, openness and topology. This is not to say that they are the only parameters available, but they do comprise three of the more significant ones with respect to robot navigation. Each of the parameters assigns a value in the range of 0.0 to 1.0 to each of the nodes comprising the given n-space. Note that these values are not the same as the costs assigned to the links. The values are determined by functions of both n-space and time and are used by the evaluation functions which determine the cost assignment made on the links between nodes.

4.1 Spatial and Temporal Qualities

There are many qualities that can be used to describe a given space: e.g., climate, political boundaries and terrain. This research focuses on traversability, openness and topology as three qualities of space which allow rich spatial descriptions to be generated. These qualities will in turn be used by evaluation functions to make cost assignments to the links connecting the nodes of a given n-space allowing paths through complicated spaces to be planned.

Traversability is used as a means of measuring the quality of the space through which the robot is moving. Consider, for example, a robot that has the choice of traveling over a paved road or a gravel road in attaining a desired destination. Clearly, if all other factors are equal (e.g. path length), a robust route planning system should choose the paved path over the gravel path. As a further example, a dirt road might have a traversability of 0.5 when it is dry and a traversability of 0.1 when wet

and muddy.

The openness of n-space is used to find paths that move the robot at safe distances from objects as well as finding paths that are least likely to become blocked. For example, if a robot is navigating the streets of New York, it would be advantageous to plan paths that avoid narrow alley ways where its path might become blocked by a garbage truck or other sufficiently impassable obstacle. In the case where routes are being planned through a space occupied by moving obstacles, it is the openness value that allows paths to be planned that maximize the clearance between the moving obstacles.

Finally, topological features are also of concern when routes are being planned. Topology refers to how the spatial relationship between nodes is represented. In 3-space, topology can refer to the modeling of hills. To see how this parameter can be used, consider the following route planning objective. Plan a route to move the robot from one side of mount Everest to the other. In this case it is most certainly best to plan a much longer path that goes around the mountain than the shorter path that goes over the top of the mountain.

4.2 Defining Evaluation Functions

Now that some of the parameters of evaluation functions have been defined, specific examples of how evaluation functions translate the qualitative parameters into a quantitative measure of path quality are in order. First, evaluation functions operating on the value of a single quality parameter will be defined. The openness will be used as the measure of quality to present an evaluation function that operates on the values produced by a function that describes the openness of the space. The evaluation function finds paths through 2-space that move the robot through the most open areas to attain the desired destination. After evaluation functions that operate on a single measure of quality have been presented, an evaluation function that operates on multiple qualities will be discussed. Evaluation functions that operate on more than one parameter have special problems that involve interactions between the different measures of quality. Note, unless otherwise specified, all discussion of this section will assume that the space is static; only static objects in the spatial definition.

4.2.1 Using a Single Measure of Quality: "Openness"

Openness is the quality that describes the relative distance that a node is from other nodes representing objects. As defined here, it is the assigning to each node in the space a value in the range of 0.0 to 1.0 (this is different from link costs). This assignment is done in a manner so that nodes furthest from any object (static or dynamic) are assigned values close to 1.0 and nodes close to objects are assigned values closer to 0.0. Consider the following algorithm for assigning openness values to the nodes in a given space.

- Step 1: for each node in the space that is currently occupied by an object (static or dynamic) assign it a value of 0.0 and assign a value of 1.0 to the remaining nodes.
- Step 2: each node in the space sends its current openness value out over each of its links (including itself) to its connected neighboring nodes.
- Step 3: each node having an openness value greater than 0.0 (nodes not occupied) takes the sum of the openness values being passed into the node ($2n+1$ possible). This value is then divided by $2n+1$ and becomes the new openness value.
- Step 4: the node in the space with the greatest openness value is found. The difference between this node's openness value and 1.0 is added to the openness values of all the nodes in the space.
- Step 5: if the old openness value differs from the new openness value by more than some percentage in any node in the space (e.g., 0.1%) go to step 2, else halt.

Using a scheme such as this, the openness value associated with the nodes will quickly converge to a stable node value pattern. Figure 17 shows the results of this algorithm when it is applied to a 2-space with static objects in it. The left portion of the figure gives the definition of the 2-space showing the location of the static objects. The right portion of the figure shows the node openness value pattern that results if the above openness definition is used. Openness values are indicated through the use of varying levels of gray; lighter shades indicate values close to 1.0 while the darker ones indicate values closer to 0.0. The algorithm in this case converged in only ten iterations.

The ability to consider the openness of a node as a spatial relation between it and the surrounding nodes allows more complex evaluation functions to be written. In particular, evaluation functions that account for trade-offs between path length and spatial path quality can be written. That is, a longer path that avoids moving the robot through tight spaces may be better than a shorter path moving the robot through the tight space. This is true because, in moving through the tight space, the robot may become blocked by an unpredictable object and have to backtrack out and plan again.

Defining an evaluation function that uses this information to find the most open paths through a space can be accomplished as indicated in Figure 18. The first part of the figure gives the evaluation function's link cost definition, which is a recurrence relation based on the openness values of the nodes. For the recurrence relation to be defined, the openness values of the nodes have been broken into 20 equal units in steps of 0.05. In particular, the recurrence relation assigns a cost of 1.0 to the links leaving the particular node if the openness value in that node is greater than 0.95 and less than or equal to 1.0. Links of nodes not in this range are assigned a cost that is determined by the following recurrence relation based on that node's openness value $c(\text{openness value}) = c(\text{openness value} + 0.5) * 2$. The recurrence relation makes paths of length x , moving through nodes with an openness value

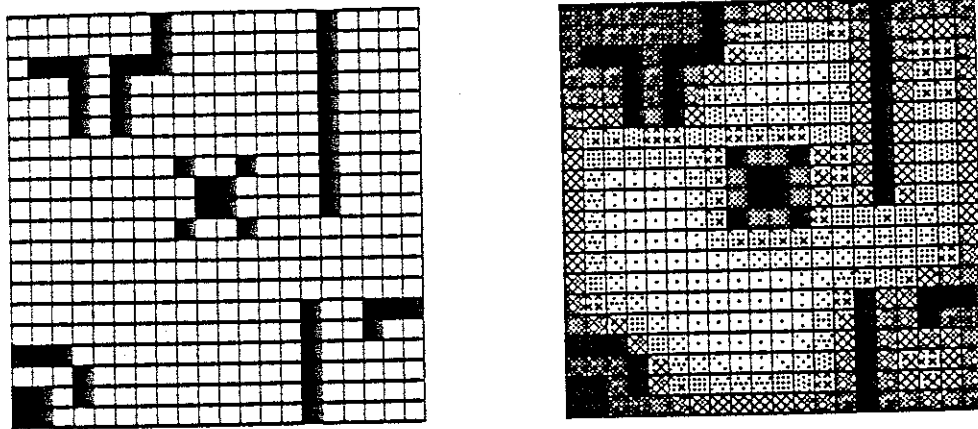


Figure 17 Openness in a simple 2-space

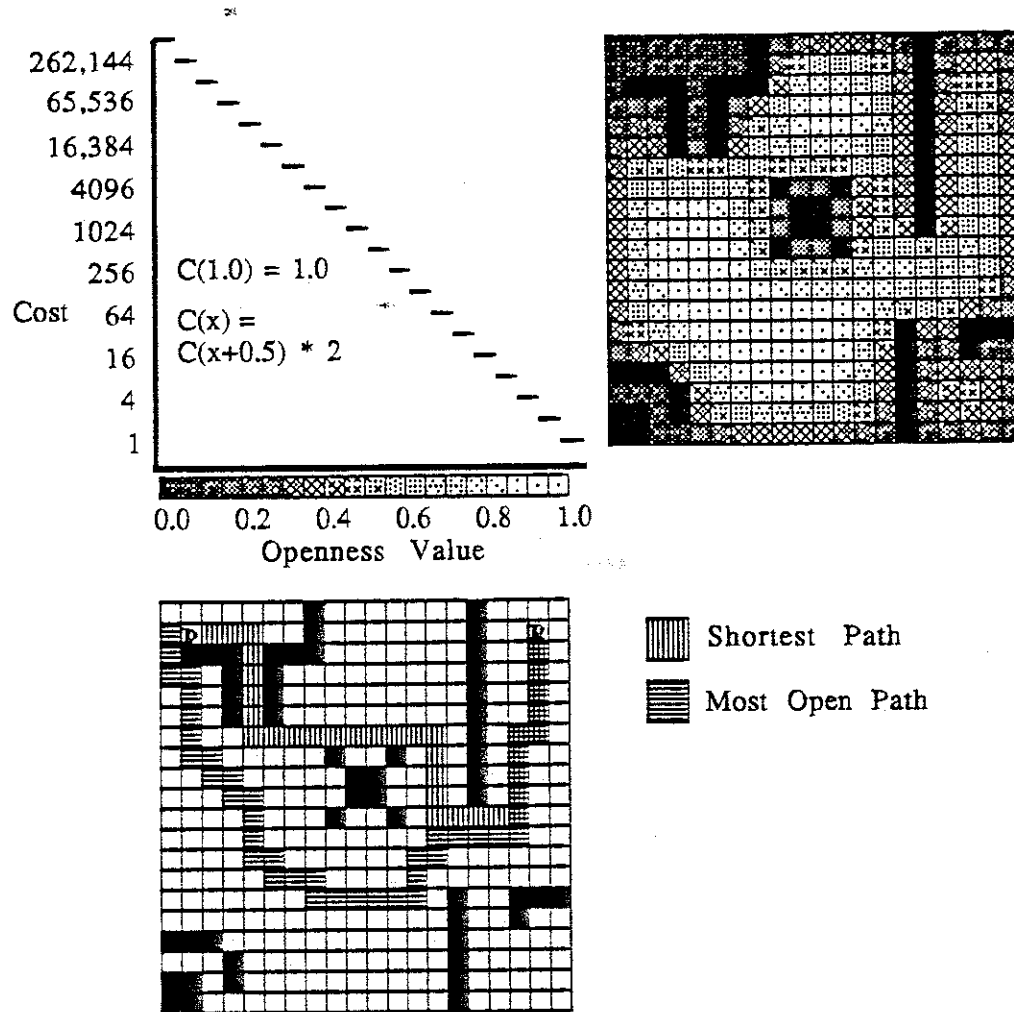


Figure 18 Evaluation function using openness

of v , equivalent to paths of length $2x$ moving through nodes with an openness value of $v - 0.5$. In general, the number of parts into which the step function is broken and recurrence relation should be chosen so that the proper trade-off between path length and path quality is established. The bottom portion of the figure shows two different paths: One results from the evaluation function that finds the most open path, as defined above. The other is chosen when the shortest path evaluation function of the previous chapter is used.

4.2.2 Defining Multi-Variable Evaluation Functions

The previous subsection showed how evaluation functions involving one measure of quality are defined. In particular an evaluation function that finds a path to the destination using the nodes with a high degree of openness was presented. To a certain extent the previous subsection already addressed this, in that it dealt with the trade-offs between a path's length and the openness of that path. This section will discuss how the simultaneous use of multiple measures of quality is incorporated into the model.

To define evaluation functions involving more than one measure of quality, the relative significance of each of the quality measures must first be established. That is, at what point does the cost associated with one quality dominate the others. If an evaluation function is written for each of the qualities with this in mind, then an overall evaluation function that is simply the sum of the single evaluation functions can be defined. Figure 19 shows a graph of the relative costs of two evaluation functions, f and g ; based on two measures of quality, q_1 and q_2 respectively. The overall evaluation function is then defined as $c(q_1, q_2) = f(q_1) + g(q_2)$. In this way, paths will be generated that utilize the relative significance of the two measures of quality in determining the best path. For example, if the value of both q_1 and q_2 is about 0.4 at a given node, then each of the quality measures will equally affect paths moving through that node. If, on the other hand, quality q_1 at a particular node is in the range 0.8 to 1.0 while at the same time quality q_2 is in the range 0.5 to 1.0, then quality q_1 will dominate the cost of paths moving through the given node. This approach can be extended for evaluation functions involving any number of qualities.

Figure 20 gives an example of how the use of an evaluation function defined over two qualities, traversability and topology, can be used to affect plan generation. The figure shows a simple 2-space (or if you like 3-space) in which there are two paths that can be planned to attain the destination. One of the paths moves the robot over the hill, while the other avoids the hill but must traverse nodes that have a lower traversability than the others in the space. Moving through a node up hill adds a cost of 1.5 to the cost of the path. Moving down hill through a node, on the other hand adds a cost of only 1.0 to the path's cost. Moving through a level node adds a cost of either 1.0 or 2.0, depending on the traversability of the node. Using this description of the space, it turns out that moving the robot over the hill yields a path that costs 9.0, while

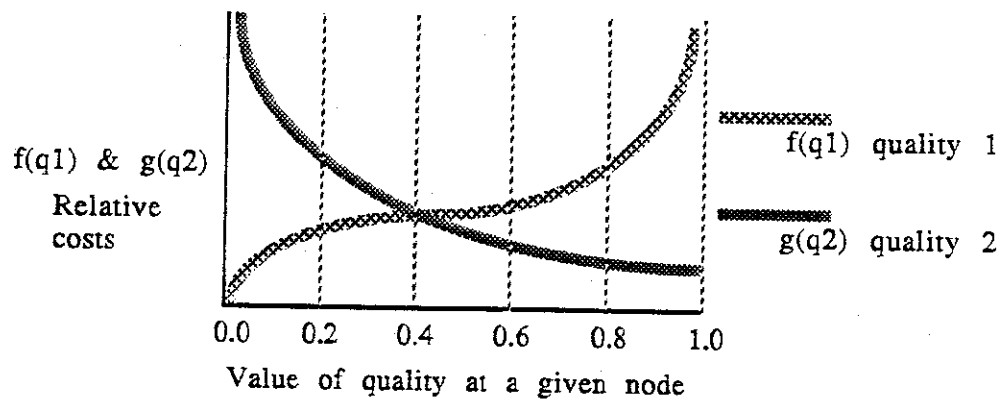


Figure 19 Evaluation functions using multiple qualities

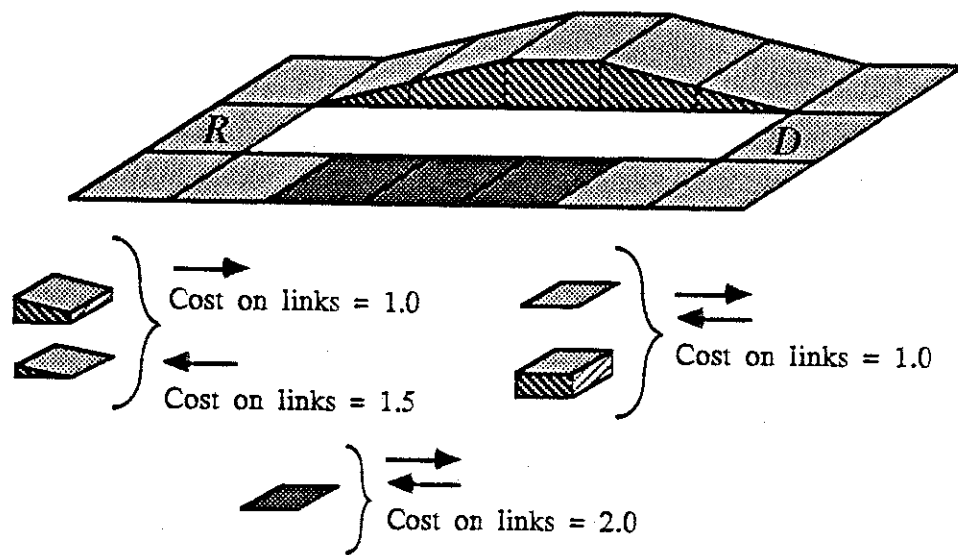


Figure 20 Path finding through a rich space

avoiding the hill yields a path of cost 11.0. So clearly, the preferred path moves the robot over the hill. The various costs used on the links of the example were arrived at in the following way: Nodes with a light shading have a traversability cost of 1.0, while the links of the darker nodes have traversability cost of 2.0. Node links representing movement in the plane or along a down grade have a topological cost of 0.0. Links representing an upgrade, on the other hand, have a topological cost of 0.5.

The particular evaluation functions presented in this section are defined more or less independently of each other. In other words, the evaluation function of one measure of quality does not depend on the value of the other measures of quality being used. If in the above example the traversability of the node was a parameter used by the evaluation function to determine how the topology of the node affects the cost of paths then the qualities of traversability and topology would not be independent. In which case, more complex evaluation functions would need to be written to account for the qualitative interactions between the various measures of quality being used by the evaluation function. This is a topic for further research.

4.3 Finding Paths Through Dynamic Spaces

This chapter has discussed how measures of quality are incorporated into the system providing a path planning algorithm that deals with static spaces. What has not yet been discussed is how dynamic objects moving in the space affect evaluation functions. How the system deals with evaluation functions in dynamic domains is the subject of this section.

The key to evaluation functions defined in dynamic domains is that any quality that is affected by the movement of objects must be updated after each time step of the algorithm. Consider the openness function; it represents a measure of quality that is affected by the movement of dynamic objects. Objects affect a node's openness value because the relative openness of a node changes as an object moves close to it. For example, Figure 21 shows a simple 2-space in which the planning objective is to move the robot from the top center node through the space to the bottom center node. The figure depicts a dynamic object that moves through the space during times $t=1$ through $t=5$. The evaluation function used by this example is the one described above for dealing with openness. The figure gives a lighter shade to the nodes with the highest openness values and a darker shade to the nodes with the lower openness values. Interestingly enough, the introduction of dynamic objects does not necessarily affect the evaluation function or the algorithm at all. The only new requirement of the system is that the openness values of the nodes must be recalculated after each time step of the simulation in which there is a change in the position of an object. Note evaluation functions that consider time and distance constraints of high priority will need to be modified to handle dynamic objects.

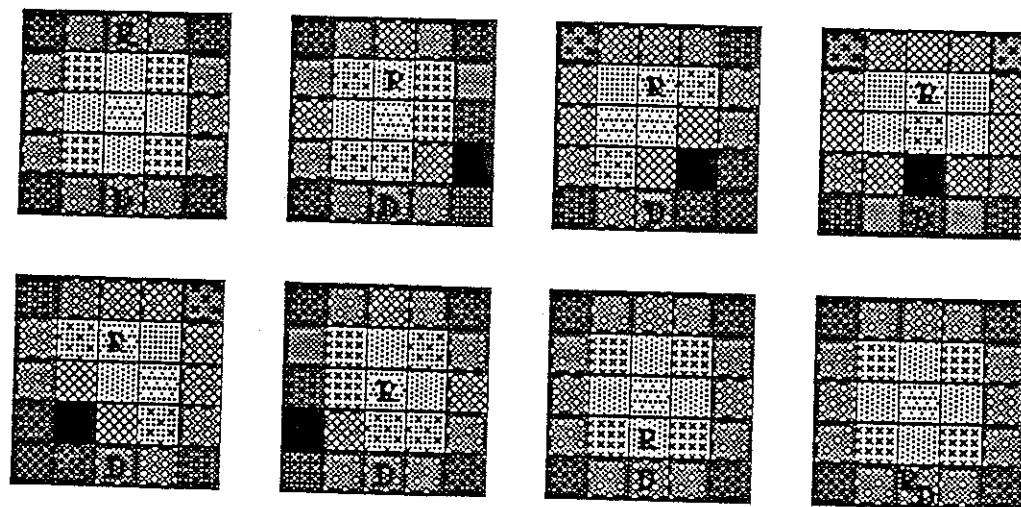


Figure 21 Evaluation functions in dynamic spaces

Thus, the introduction of more complex evaluation functions dealing with various measures of quality has no detrimental effects on the overall system except for performance. The only modification to the system occurs when, in a dynamic environment, one or more of the measures of quality being used is affected by the movements of dynamic objects. The modification to the system is relatively minor, requiring the measures of qualities affected by the movement of objects to recalculate their node values after each time step of the algorithm.

4.4 Robot Specific Abilities and Limitations

There are aspects of the robot's performance that must be accounted for during the planning of paths. Different robots will have different abilities and limitations that must be incorporated in the evaluation functions. For example, consider a robot that has a very small battery and should plan paths that have the robot charging as long as possible. To a first approximation, this can be accomplished by placing zero cost on the reflexive links at the nodes representing charging ports and a high cost on the links of the nodes that are farthest from any charging port.

Consider the more complicated concept of F regions introduced by Miller [Miller85]. F regions are used to define areas in a space that represent the number of positional degrees of freedom that a mobile robot outfitted with sonar can eliminate. This information can be used to find paths that stay within the limits of the robots abilities to handle uncertainty. In this scheme, paths that attain the destination while moving the robot through the space with as high an average F region as possible are considered the best or safest paths. For example, Figure 22 shows an example 2-space indicating the various F regions in which the highest possible F region for a robot is three (two planar and one rotational). The D_{max} of the figure refers to the maximum distance over which the sonar can reliably be used.

To use this concept of F regions in finding "safe" paths, a relatively simple evaluation function can be defined. First, determine how many three-F region nodes should be traversed in order to avoid moving through one two-F node, call this number k_1 . Then, in a similar manner, determine the relationship between two-F nodes and one-F nodes (k_2) as well as between one-F nodes and zero-F nodes (k_3). Each node in a three-F region is assigned a cost of 1.0 to the links leaving the node. A cost of k_1 is assigned to the links of nodes in two-F regions. The nodes of the one and zero-F regions are assigned costs of k_1*k_2 and $k_1*k_2*k_3$ to their links respectively. In this manner, the safest path will be found. If, for example, the values of $k_1=3$, $k_2=6$ and $k_3=4$ are chosen for the situation depicted in figure 22, the resulting path shown in Figure 23 will be generated. Using schemes such as the F region concept, evaluation functions can easily be defined to find paths that remain within the abilities of the robot to navigate through a given space.

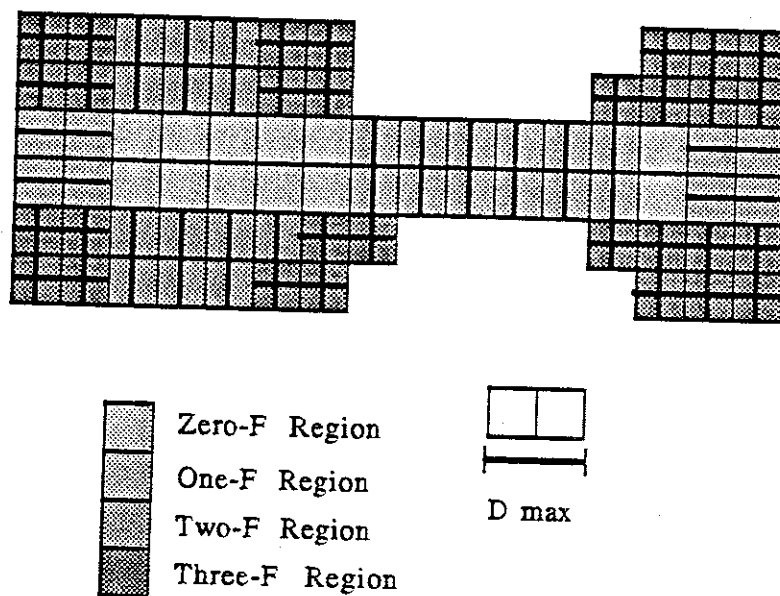


Figure 22 F Regions in 2-space

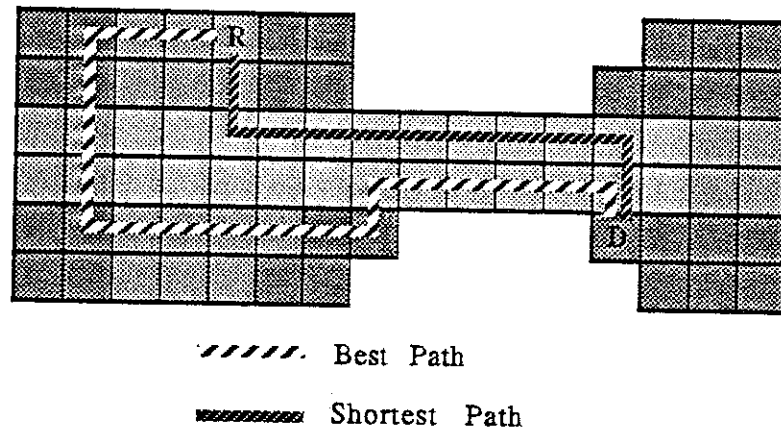


Figure 23 Evaluation function using F regions

Chapter 5

Incremental Route Planning

Thus far, only the generation of plans that involve predictable objects have been considered. To move autonomous robots in the real-world, a route planning system must be able to handle the unpredictability that the real-world has to offer, as in the case of a robot that must walk across a busy street. The process of incremental route planning has been identified to handle this problem.

5.1 Incremental Operation

An incremental route planner can be viewed as the repeated use of a route planner that executes in a predictable dynamic environment. After each step, the state of the world is tested and updated with any new information. In this way any unpredictable objects that have changed their position since the last plan step was generated are identified and can be accommodated for by the route planner. Because Robonav is structured to operate on a parallel architecture it can be easily modified to function as an incremental route planning system. This is because incremental route planners must operate in real-time.

To have Robonav operate in an incremental fashion a simplifying modification must be made to the algorithm. By making an addition to the messages being passed around the system, the stack is eliminated from the processors. The reason that the stack can be eliminated is, the planner is no longer interested in finding an entire path from beginning to end it simply needs to find the next step in the plan. Thus, there is no reason to keep the entire ancestry of the messages active in the system. The modification made to the messages involves the addition of a field indicating the link over which the messages oldest ancestor traveled. More precisely, the headers of the messages created in phase one of the algorithm, are set to a value reflecting the link along which that particular message is to be sent. The header, of the messages created during phase two, is copied from the header of base-message. Thus, the messages active in the system represents both the viability of a path (the messages energy) and the direction of the first step along that path. There is one more modification that must be made in order for Robonav to operate in an incremental mode. The bounding condition is modified to keep track of the message representing the current global message

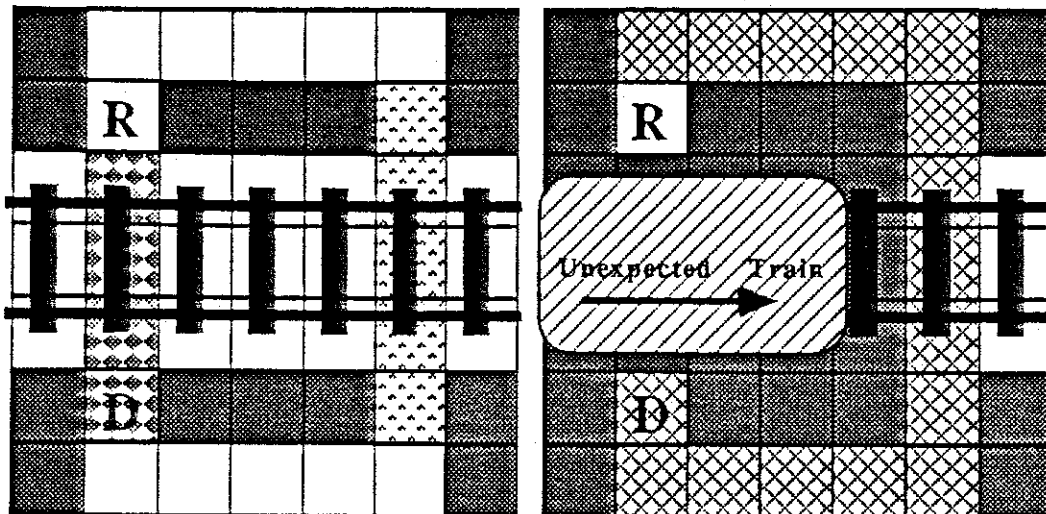
energy bound. Thus, when the system halts, the header of the global message energy bound will indicate the direction of the next best move.

Robonav was initially intended to be operated in an incremental fashion. The reason is that if the system is not operated in this manner, the growth of the stacks containing the ancestries² of the messages is potentially unbounded. For example, consider a robot that is locked inside a vault. Its task is to plan a route that gets it outside of the vault. If there is no scheduled opening of the door the system will continue searching for a plan to move the robot out of the vault until the heat death of the universe. This situation would in effect cause the stacks to grow in an unbounded manner. This situation could be eliminated by introducing a supervisory system that maintained a queue of scheduled events and could alert the planner if there is no chance of a new state of the world occurring. Such a system would be complex and need to detect cyclic events like that of a revolving door and predict how events interact with each other. This is a very hard problem and probably semi-solvable at best. So even if such a system did exist at best it would only help some of the time.

5.2 Example in an Unpredictable Domain

Consider the problem for an incremental planner depicted in Figure 24. The figure shows that there are two paths that will lead the robot to the desired destination location. One takes the robot over the railroad tracks. The other, being substantially longer, uses the tunnel going under the tracks to get to the destination. The robot's goal is to reach the destination in the shortest amount of time. Assume that Robonav is operating in the incremental mode as described above. Given the situation depicted in the left portion of the figure and no oncoming train, Robonav would plan a path that takes the robot over the tracks. But, if a train unexpectedly arrived and blocked the short path over the tracks, Robonav would replan the next best step and start down the path to the tunnel. If the situation depicted in the left portion of the figure were to persist the planner would eventually reach the destination via the tunnel. However, if before the robot started through the tunnel the train had moved and unblocked the path over the tracks, Robonav would once again take advantage of the situation and move to go over the tracks instead of finishing its path through the tunnel.

There is an anomaly that is produced by operating Robonav in an incremental fashion. It represents Robonav's need for a supervisory system to monitor the operation of the path planner. Consider the example presented above. What if before the robot started under the tunnel the train that was blocking the optimal path left clearing the original path? In this case, the robot would change direction and move toward the original path over the tracks. Now consider the situation where just before the robot started over the tracks, another train arrived. The robot would once again start to move along the path that takes it under the tracks. If it turns out that this train length and train frequency is a constant then Robonav will get stuck, never reaching the destination when there is clearly a path available.








-  Railroad Tracks
-  Path Going Under the Tracks (Tunnel)
-  Original Route Going On the Tracks
-  Alternative Route Going Under the Tracks
-  Occupied Node

Figure 24 Operation in an unpredictable domain

The control anomaly is a situation that is common in execution systems. For the Robonav system to operate in these domains, the system must incorporate a form of supervisory system to provide execution monitoring. As this problem is not solvable in general it must be done in a very domain specific manner in order to provide reliable guidance. It should be noted that this is a very hard problem and one that is not addressed in this thesis.

Chapter 6

Conclusions, Further Research and Implementation

6.1 Conclusions

Moving robots through dynamic environments mandates that the other dynamic processes of the environment be modeled to a reasonable level of detail. This thesis has presented Robonav, a spatial representation and temporal reasoning system that plans paths through space and time. Robonav models the dynamic aspects of the environment as well as creating path plans through the environment that consider the performance characteristics of the robot executing the plans. The particular algorithm that operates on the representation is optimal when functioning in an incremental mode. This mode of operation allows Robonav to perform tactical planning in real-time for operation in the real-world.

It is becoming apparent that spatial representation and path planning is an inherently computationally expensive task. To this end, parallel processing becomes one solution to the computational needs. Researchers like Rao, are beginning to address path planning from parallel processing point of view [Rao86]. The Robonav system approaches parallel computation by providing a spatial representation and path planning algorithms with an isomorphic mapping onto a parallel architecture. Thus, by exploiting massive parallelism, Robonav is able to provide real-time operation in unpredictable dynamic domains. This ability is one that has not been approached by any other path planning system.

6.2 Further Research

There are several possible extensions to this model that would increase its representational power. Among the most useful and interesting are:

-Integrating the system with sensors. Ultimately, any system that does robot navigation must be integrated with the sensors that provide the robot with information about the environmental situation. This problem has been addressed by Moravec [Moravec85]. His approach to sensor integration

is compatible with the spatial and temporal model presented here.

- Refining generated paths. The paths generated by the system tend to have a stair-stepping effect that results from the system only allowing movement in four directions.
- Planning for robots with asymmetric dimensions. The problem of asymmetric robot dimensions can be addressed by the use of an extra dimension to deal with the rotational aspects that enter into planning under this condition.
- Introducing an object typing system. The use of object types would allow queries such as "What Moose clubs are within a mile of your house?" to be answered by the system.
- Modeling of unpredictable processes. The power of an incremental route planner can be increased for a particular domain with some model of the typical behavior of the unpredictable objects in that domain. For example, the route planner could provide more useful plans for a robot crossing busy street if the system had a model of the speed, maneuverability, and direction of travel for the autos traveling the road [Sanborn87]. This might be approached from a neural net modeling point of view [Poggio87].
- Representing and coordinating multiple robots. The message passing algorithms could be changed to handle the coordination of multiple robots seeking competing or cooperating goals.
- Implementing the algorithm to operate in a hierarchical fashion. This is similar to that found in [Moravec87] and would allow time efficient heuristics to be defined that would drastically prune the size of the search space.

6.3 Implementation

The algorithm, when fully implemented on an SIMD machine, operates in predictable domains in $O(p)$ time, where p is the length of the longest path through space-time that is bounded by the global message energy bound. In unpredictable domains, the time complexity analysis is less clear. If, however, the incremental approach is taken in a predictable domain, the algorithm will operate in $O(p^2)$ time. This is derived from the following: If the first step of the incremental algorithm takes $O(p)$ time then the second step will take $O(p-1)$. This is continued, with each step reducing the complexity by one. Solving this recurrence relation yields an $O(p^2)$ time complexity. In a truly unpredictable domain the time complexity of operation will vary with the behavior of the objects moving through the space.

A version of the algorithm, written in NISP [McDermott83], is currently up and running on a VAX 11-785. It has been transported onto an *Explorer* and modified to include a graphics interface. It functions

on the examples given herein, as well as others involving more complex spatial definitions and unpredictable dynamic environments. The implementation includes software for simulating the SIMD architecture needed to operate the system in real-time.

Bibliography

- [Andrews83] Andrews, R. J., *Impedance control and a framework for implementing obstacle avoidance in a manipulator*, Masters thesis, Massachusetts Institute of Technology, 1983.
- [Brooks82] Brooks, R. A., Solving the find path problem by a good representation of free space, in *Proceedings of AAAI 82*, AAAI, pp. 381-386, 1982.
- [Charniak86] Charniak, E., A neat theory of marker passing, in *Proceedings of AAAI 86*, AAAI, pp. 584-588, 1986.
- [Chatila85] Chatila, R., Position referencing and consistent world modeling for mobile robots, in *Proceedings of the International Conference on Robotics and Automation*, IEEE, pp. 138-145, 1985.
- [Davis84] Davis, E., *Representing and acquiring geographic knowledge*, Pitman Publishing Limited, 1986.
- [Hillis85] Hillis, W. D., *The connection machine*, MIT press, 1985.
- [Hinton86] Hinton, G.E., Distributed Representations, in *Parallel Distributed Processing*, MIT press, pp. 91-94, 1986.
- [Laumond83] Laumond, J. P., Model structuring and concept recognition: Two aspects of learning for a mobile robot, in *Proceedings of the 8th IJCAI*, IJCAI, pp. 839-841, 1983.
- [Lozano-Perez83] Lozano-Perez, T., Spatial planning: a configuration space approach, *IEEE transactions on computing*, c'32, pp. 681-698, 1983.
- [McDermott83] McDermott, D. V., *The nisp manual*, technical report 274, Yale University Computer Science Department, 1983.
- [McDermott84] McDermott, D. V., Davis, E., Planning routes through uncertain territory, *Artificial intelligence*, v22, pp. 107-156, 1984.
- [Miller85] Miller, D. P., A spatial representation system for mobile robots, in *Proceedings of the International Conference on Robotics and Automation*, IEEE, pp. 122-127, 1985.
- [Moravec82] Moravec, H. P., The CMU rover, in *Proceedings of AAAI 82*, AAAI, pp. 377-380, 1982.
- [Moravec85] Moravec, H. P., Elfes, A. E., High resolution maps from wide angle sonar, in *Proceedings of the International Conference on Robotics and Automation*, IEEE, pp. 116-121, 1985.

- [Moravec87] Moravec, H. P., Certainty grids for mobile robots, in *Proceedings of the Workshop on Space Tele-Robotics*, JPL Pasadena California 1987.
- [Nguyen84] Nguyen, V., *The find-path problem in the plane*, A.I. technical report 760, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1984.
- [Nilsson71]. Nilsson, N. J., *Problem-solving methods in artificial intelligence*, New York: McGraw-Hill, 1971.
- [Iyengar85] Iyengar, S. S., Jorgensen, C. C., Rao, S. V. N., Weisbin, C. R., Robot navigation algorithms using learned spatial graphs, in *Robotica*, 1985.
- [O'Rourke] O'Rourke, J., Convex hulls, Voronoi diagrams, and terrain navigation, in *Proceedings of the ninth William T. Pecora Memorial Remote Sensing Symposium*, IEEE, USGS, NASA, ASP, Sioux Falls, South Dakota, pp 358-360, 1984
- [Poggio87] Poggio, T., Koch, C., Synapses that compute motion, in *Scientific American*, v256 # 5, pp 46-52, 1987.
- [Rao86] Rao S. V. N., Iyengar, S. S. , Jorgensen, C. C., Weisbin, C. R., *Concurrent algorithms for autonomous robot navigation in an unexplored terrain*, technical report 85-048, Louisiana State University Computer Science Department, 1986.
- [Sanborn87] Sanborn, J C., Hendler J. A., *Towards dynamic planning*, technical report 1785, Maryland University Computer Science Department, 1987.
- [Thorpe84] Thorpe, C. E., Path relaxation: Path planning for a mobile robot, in *Proceedings of AAAI 84*, AAAI, pp. 318-321, 1984.

Vita

Education

- M.S.--Computer Science, VPI Blacksburg, VA. June 1987
Thesis title; *Spatial and Temporal Path Planning.*
- B.S --Computer Science, VPI Blacksburg, VA. June 1985
Mathematics minor.

Professional Experience

- Computer Science Dept., Va. Tech 6/86-present
Employed as a GRA (Graduate Research Assistant) to carry out research funded by a grant from the Naval Surface Weapons Center investigating the uses of AI (Artificial Intelligence) in their defense systems. This project culminated in a report detailing the recommendations made.
- Computer Science Dept., Va. Tech 8/85-present
Employed by the department as a graduate teaching assistant. Responsibilities included acting as undergraduate course advisor, teaching assistant to a number of different undergraduate courses and instructor to an introductory level course.
- Agronomy Dept., Va. Tech 12/84-8/85
Employed by the Research Laboratory of the University Agronomy Dept. as an Applications Programmer. The programs developed are currently supplementing classroom education and are in practical use in other laboratories around the country.

Professional Affiliations

- American Association for Artificial Intelligence
- Artificial Intelligence Society fo the Mid-Atlantic States
- Association for Computing Machinery-SIGART

Publications

- [Slack87] Slack, M. G., Miller D. P., Path Planning Through Time and Space in Dynamic Domains, in the forthcoming *Proceedings of the 10th IJCAI, IJCAI*, printing pending, 1987.
- [Slack87] Slack, M. G., Miller D. P., Route Planning in a Four Dimensional Environment, in *Proceedings of the Workshop on Space Tele-Robotics*, JPL Pasadena California 1987.