Path Planning Through Time and Space in Dynamic Domains

Marc G. Slack
David P. Miller

TR 87-5

Path Planning Through Time and Space in Dynamic Domains

Marc G. Slack and **David P. Miller**

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
(703) 961-5605
miller%vtcs1@bitnet-relay.arpa

ABSTRACT

Realistic robot problems involve navigating the robot through time as well as space. The obstacles that a robot must avoid and the pathways on which it travels are subject to changes throughout time. These changes can occur in a predictable or unpredictable fashion. This paper presents an integrated route planning and spatial representation system that allows paths to be calculated in dynamic domains. The path planner finds the "best route" through a given n-dimensional space. The "best route" is defined as the path through space-time with the best score as determined by a set of user-defined evaluation functions. The algorithm takes into account the following: Capabilities of the robot executing generated plans, traversability of space, and interactions with both predictable and unpredictable dynamic objects. The route planning algorithm is highly parallel and can be run on an SIMD (Single Instruction Multiple Data) machine in O(p) time, where p is the length of a path. In domains with unpredictable processes, this system may be run in an incremental fashion, allowing the robot to create real-time routes along the optimal path as described by its current information. This paper will discuss: Spatial representation, an SIMD algorithm for route planning in a dynamic domain, and results from an implementation on a traditional computer architecture.

1 - Introduction: Route Planning in Dynamic Domains

The ability to represent and plan movements through space is necessary for any autonomous mobile robot. Mechanical error and uncertainty make it impractical to maneuver a robot through a series of complex tasks strictly by dead-reckoning. If dead-reckoning is of limited use, then some navigation capabilities must be brought into play; navigation depends on having some knowledge of the world outside of the robot. Towards this end, a wide variety of spatial representation systems have been developed in recent years.

A variety of techniques have been used to attack different aspects of the spatial representation problem. Topological graphs [Laumond83], [Chatila85] have been used for guiding route planning through a loosely connected set of convex polygons representing free-space areas in an indoor environment. Regions mapped with traversable conduits [McDermott84] have been used successfully for large scale navigation in uncertain environments. Representation of the exteriors of obstacles as the edges of a highly connected graph was used by Davis, allowing detailed knowledge of the environment and its accompanying uncertainty to be captured [Davis84].

Other representations have been used for capturing movement or navigational details necessary for a robot to plan its activites. Configuration space [Lozano-Perez83] provides a computationally tractable approach to calculating the practical steps for moving a robot from one position to another. Using Voronoi diagrams and representations of free space, movements in three-dimensions have been calculated to maintain a robot the maximal possible distance from any obstacle [Brooks82]. Similar methods, when combined with an analysis of the robot's sensors, can calculate a path that is both relatively safe and easy to navigate [Miller85].

Despite the variety of the techniques mentioned above, all of the systems discussed share some basic limitations. None of the systems takes into account the quality of the surface upon which the robot travels, relying on the surface being either traversable or not. Such a restricted view is continually contradicted by the way people move about. People stray off the sidewalk or jay-walk across a street whenever it is convenient and safe, hence a realistic robot should be able to behave similarly. A further limitation is that all the aforementioned systems are designed to operate under static conditions, where the only aspect of the world that changes is the position of the robot. This is an unrealistic and unacceptable limitation for almost all applications.

In addition to being able to function in a dynamic world, a robot should be able to reason about dynamic processes and how they may affect it. For example, if a robot knows the local train schedule and needs to get to the other side of the train tracks, then it should use that information when planning to get to the other side. If the robot has information predicting that a long freight train will be coming just before it 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, the robot should choose the latter plan. Similarly, if the robot's task is to rob a train, then the ability to plot a path that will allow the robot to jump onto the moving train is necessary.

Unpredictable dynamic processes must also be taken into account during route planning. A cavalry robot that "fears" an attack by a tribe of Indian robots would be better off planning to get to the fort across the open plain, rather than passing through the narrow passageway of *Ambush Canyon*. The primary reason for this is an attack in the canyon would more effectively block the robot from its destination, thereby mandating backtracking.

The single property that most distinguishes this work from previous systems is that it models not only space, but time as well. Rather than making a calculation about whether the robot can traverse a particular area independent of time, this system models the ability of the robot to traverse that area at different times. We have accounted for temporal as well as spatial changes in the environment. The message passing technique used allows time to be considered while allowing the system designer to model qualities of the domain, such as the cost of moving from one position to another and the ease of traversing a particular area of space. The remainder of this paper describes a representation and route planning system for use in unpredictable dynamic domains.

2 - The Algorithm

This section will describe an algorithm that finds the best path through a predictable n-dimensional space using user-defined evaluation functions. The algorithm provides for an effective representation of space-time and the ability to functionally define predictable static and dynamic objects that map into a subset of a given space-time.

2.1 - Spatial Representation

A path-finding algorithm that is of any use must provide for: 1) an effective representation of space, 2) the relationship between the elements making up the space, and 3) the ability to define the quality of that space with respect to a robot using the generated plans.

To effectivly represent space, this model uses uniformly shaped, n-dimensional hyper-cubes called "nodes". Each node represents a small chunk of space. Arbitrarily shaped n-dimensional areas can be defined through the spatial concatenation of nodes along common (n-1 dimensional) surfaces. The collective area occupied by the nodes is called "space", while the remainder of existence is referred to as "void". For example, Figure 1 shows an arbitrary two-dimensional space constructed from the spatial concatenation of square shaped nodes. In general, the size of a node will be of at least sufficient size to subsume the size of the robot.

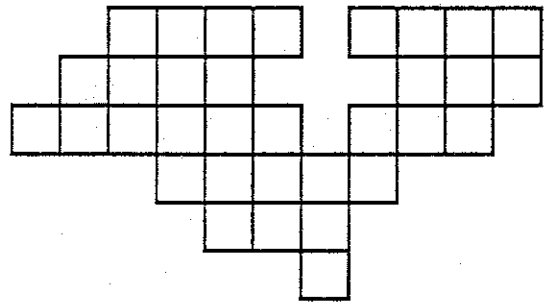

Figure 1

The relationship between adjoining nodes (such as the ability to move from one node to another) is represented by unidirectional links between each of the nodes (see Figure 2b). The reflexive relationship a node has with itself is represented as a link that points back to the node and represents the ability to remain at that node. Associated with each of the links is a cost. The cost represents such things as: whether the surface between the two nodes is continuous, a downgrade, in three-dimensions directly below, or if a node represents a safe place to stop. The function of the links is to provide a communication path over which messages can be sent. A node can have up to $2n$ communication links with its neighboring nodes and one reflexive communication link. Nodes that lie on the edge between space and the void will have fewer communication connections.

The topological features of the space are represented by the cost on links between nodes, whereas the actual traversability of a particular region of space is specified by the node's traversability constant. The traversability constant represents the relative ability of the robot to move over a given node. For example, for a robot with wheels, a concrete floor would have a traversability constant near 1.0 while that of quicksand might be on the order of 0.001.

Consider the spatial representation of the two-dimensional space shown in Figure 2. Part **a** of the Figure shows the physical layout of the example space. Part **b** shows how the nodes that define the space are interconnected with one another. The reason for the missing links between some of the nodes is that the cost associated with the link is infinite (the cost on other links is not indicated). As a result, no paths are generated that make those transitions. While the surface is three-dimensional, only a two-dimensional representation is necessary (for this particular example) because the costs on the links between nodes allows for the representation of hills, absorbing the three-dimensional aspect of the space. For example, a link going down hill could have a lower cost than one going up hill. If the features described above are taken together, a robust representation of the salient features of space can be created.

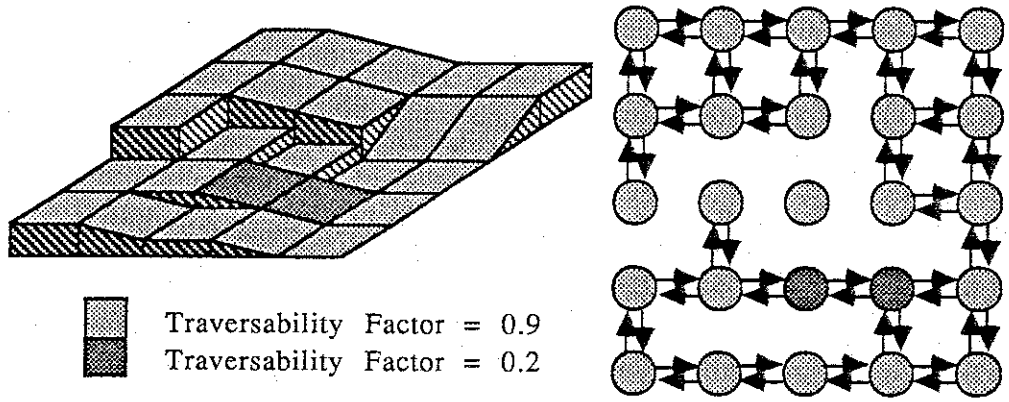

Figure 2

2.2 - Representing predictable objects

For a route planning system to be of value, not only does it have to represent the salient features of space, but it must also provide an effective means for representing predictable objects. Predictable objects are those objects, both static and dynamic, that have fully predictable behavior in both time and space. An object is represented as a function having the defined space and time in its domain, and some subset of the nodes that make up space in its range. The set of nodes generated on any application of the function consists of those nodes in the given space that are occupied (fully or partially) by the object during the given time. For example, consider a model of a simple two-dimensional revolving door. This can be represented by defining a function that has four nodes forming a square in its range. Then, by setting the function to map onto two of the nodes that are diagonal during odd time units and onto the other two nodes during even times, a simple, predictable, revolving door can be created. Such functions can be encoded into each of the nodes at setup time, thus reducing the amount of outside communication during operation.

2.3 - An SIMD Algorithm for Route Planning

The representation of the spatial features and predictable objects thus far described provides a basis for an algorithm that can be directly implemented on an SIMD machine, such as that in [Hillis85]. This is accomplished by assigning each node to a processor. Each processor has message communication links to other processors that transcribe directly from the node links. A message represents the value of a possible transition from one node to another and the quality of the entire path leading up to that transition. To perform the task of reclaiming the generated paths from the processors, each processor must maintain a stack. The stack represents a storage place for logging the history of the activity at the processor. For now, the simplifying assumption will be made that each of the moves made by the robot being simulated will take one unit of time. For example, the time required for the robot to move from one node to an adjacent node, takes one unit of time regardless of the robots previous state. The removal of this

assumption will be discussed later (see additional features section 2.5).

Using a synchronous, step-wise process of passing messages from processor to processor, all possible paths that the robot could take through time and space in attaining the desired, static destination location can be considered. The process has two phases and a terminating condition.

The first phase sets up the initial message set. Using the node in space that represents the current location of the robot, a set of messages is created, one message for each communication link associated with the node. Each message has associated with it an energy value that reflects the cost of moving the robot to the space represented by the adjoining node. The particular value of a message's energy is determined by a user-defined evaluation function. The evaluation function considers such things as: the current state of the robot, the cost on the link that the message is to be sent over, the traversability of the node currently being occupied by the robot, etc. The set of messages is then sent to its respective destinations along the communications links of the node.

The second phase is the operational phase. It is defined by having each node in space that is not occupied by an object during the current simulation time perform the following process in a synchronous manner: form the base message by setting it to the incoming message with the minimum energy. All other messages can be thrown out because they represent more costly paths that attain the same location in space-time. In a manner similar to that described in the first phase, a new set of messages is created. Each message in the new set is assigned an energy that is a function of the base message and the link over which the message is intended to travel. The base message is then tagged with the time and a pointer indicating the node that created it. The base message is then added to the node's stack. Finally, the node sends the newly created list of messages out along its respective communications links. This process is repeated, until the termination condition is met, each repetition representing a subsequent time unit.

The terminating 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 the global bound. The global bound is the minimum energy for all the messages that have reached the destination node (similar to zorch decay in [Charniak86]).

After the ending condition is met, the path through space-time that has the lowest energy associated with it can be retrieved from the destination node. This is done by locating the message on the destination node's stack with the lowest energy value. Once this is done, the path can be obtained by following the pointers back through space (other processors) and time (the stacks associated with the processors) until the robot's original location in space-time is encountered. The stack allows interprocessor communications to be kept to a minimum.

2.4 - A Predictable Example

This section shows how predictable dynamic objects are represented and anticipated. Shown below in Figure 3 is a two-dimensional space made of ramps that are to be navigated by the robot. A dynamic object enters the world at time t = 8 and leaves the world after t=15, indicated by the blackened square in the second of the

two Figures. While the object is present, the node it occupies cannot be traversed by the robot. The nodes over which the robot must travel are, for the most part, easily traversable and thus have a high traversability constant of 0.9. Two of the nodes, however, are made of loose sand and have the low traversability constant of 0.2.

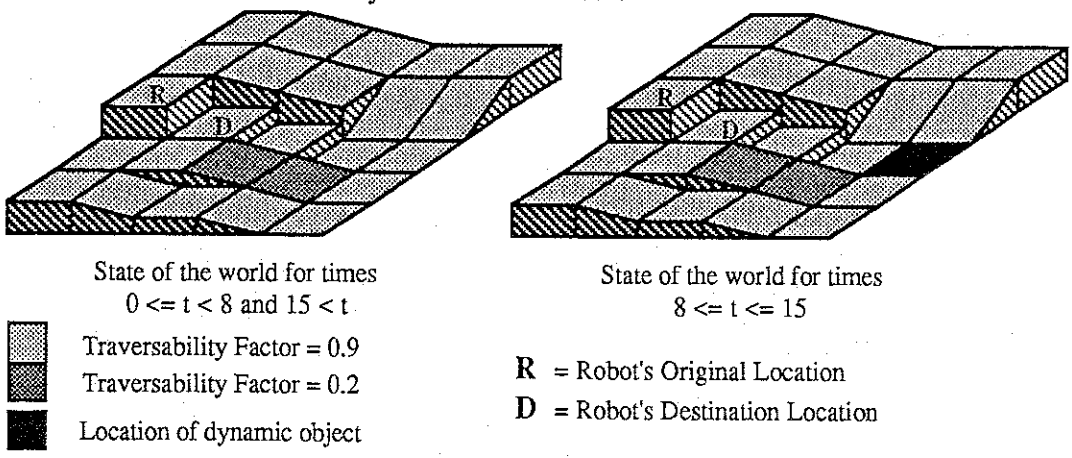

Figure 3

The route planning objective is, starting at t=0, to move the robot from its current location (indicated by the **R**) to the static destination location (indicated by the **D**) along the space-time path of minimum energy without going over a cliff. By counting the number of node transitions that must be made in traveling from **R** to **D**, one can determine that the shortest path the robot could take would be 13 moves long and would use 13 time units. But, when the 8th move is being made, the dynamic object enters the space and disables the indicated node, keeping it from receiving or sending messages. The key is that after seven time units, the message passing process has not propagated to the node with the dynamic object in it. This causes all message activity to be confined to the back portion of the defined space until after t = 15. The blockage is due to the fact that all paths from **R** to **D** must pass through the node that is occupied by the object. After t=15 units, the occupied node is freed and resumes processing and passing messages, eventually allowing the system to meet its terminating condition.

A number of points can be made here that are based on the choice of the evaluation function used to determine the message energies. One is that a proper evaluation function will allow the best path to avoid the nodes that are made of sand by giving a high energy value to any message that represents a transition into such a node. A more important point is that the evaluation function determines how and where the robot spends its time while it waits for the object to leave. For example, the robot could remain at **R**, charging its batteries, wander along the path, or hurry to the object and wait there until the dynamic object goes away. So, depending on the requirements and the situation (e.g., maximize charge time), an evaluation function can be written to determine how and where the robot waits (e.g., add a cost for stopping and starting, or time spent not charging).

2.5 - Additional Features

Some of the system's most powerful features have been omitted thus far for clarity. Among these features are: The ability to describe the destination as a function of both space and time, the ability to consider the openness of a node with respect to its spatial location, and the ability to accurately consider the movement capabilities of the robot using the generated plans.

The ability to describe the destination as a function of both time and space allows the system to solve problems involving alternative planning (e.g. if you can't get to the post office by five, go to the drug store for the stamps) and problems involving coordinating actions with dynamic objects (e.g. jumping on a moving train). This ability is incorporated into the system, by modifying the termination condition of the algorithm to consider a time ordered set of possible destination nodes.

The ability to consider the openness of a node as a spatial relation between it and the nodes surrounding it can be used to generate paths that avoid narrow passages, if possible. Generated plans avoid moving the robot through paths that would become blocked if an unpredictable object were to be encountered during the execution of the plan.

The following gives an example of how an openness function might be defined iteratively for a two-dimensional space. First, assign each node in space a value of one if it is not occupied and a value of zero if it is occupied. Second, have each node send its value along all of its communication links. Third, each node creates a temporary value by summing the values of all incoming information and then integer-dividing them by 1 the first iteration and 2,4,8,16, ... in each subsequent iteration. Lastly, if the temporary value is greater than zero, a new node value is set by multiplying the original node value by the temporary value. Using this scheme, the openness value associated with the nodes will eventually converge to a stable node value pattern. The pattern will be such that the nodes that are in the biggest, most spacious areas will have the highest values, and the nodes in corners or alcoves will have the lowest values. The addition of openness to nodes allows evaluation functions to be written that considers the trade-offs between short path length and increased chance of backtracking due to the chosen route becoming blocked by an unpredictable object.

The ability to effectively represent the time required by a robot to make simulated moves allows plans to be generated that take full advantage of the robot's abilities. For example, moving from rest to another node should take longer than moving from one node to another when the robot is already moving in the desired 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 the generated paths has been incorporated into the model by setting the model to operate in a more asynchronous manner. This asynchrony is accomplished by associating a real-time with each message. The time value of created messages is set by adding to the time in the incoming message the amount of time that is required for the robot to make the move represented by each of the new messages. 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.

3 - Dynamic vs. Unpredictable

Thus far, only the generation of plans that involve predictable objects has 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.

Incremental route planning 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, for identification of any unpredictable objects. Incremental route planning is effectively handled by this system because it is structured to operate most efficiently in the incremental form. Unpredictability is handled by the system's ability to rapidly calculate the next best step after every primitive move the robot makes.

By making a simple modification, an incremental version of the algorithm has been constructed from the framework of the previously defined algorithm for predictable domains. The stack is eliminated from each of the processors by making an addition to the messages being passed around the system. The modification involves the addition of an initial direction header. This change is made because all that is needed is the next best move and not the entire path. The headers, of the messages created in phase one of the algorithm, are set to a value representing 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 the incoming message. To identify the next best move, the terminating condition is modified to keep track of the message representing the current global message energy bound. When the system halts, the header of the global message energy bound indicates the direction of the next best move. This represents a significant simplification of the system, as it eliminates concerns involving the potentially unbounded growth of the node stacks.

4 - Implementation

The algorithm, when fully implemented on an SIMD machine, operates 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.

A simulation version of the algorithm, written in NISP [McDermott83], is currently up and running on a VAX 11-785. It functions on the examples given plus others involving unpredictable dynamic environments. The implementation includes software for simulating the SIMD architecture.

5 - Further Research

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

- The granularity of the nodes: It becomes difficult to ensure that there is an adequate path for the robot to traverse, when the granularity of the nodes used to represent space is smaller than the

size of the robot or the exact size of the nodes is undetermined.

- Uncertainty in dynamic times: Special concern should be given to route planning where objects entering the space are predictable in their behavior but have uncertain arrival times. For example, the subway train that is running a few minutes behind schedule.
- Achieving maximal efficiency over a set of destinations: This is similar to the traveling salesman problem.
- Modeling 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 give more useful advice to a robot crossing a street if the system had a model of the speed, maneuverability, and direction of travel for the autos traveling the road.
- Representation and coordination of multiple robots moving through space-time: For example, getting Huey, Duey and Luey to meet in the garden on the east end of the space ship at 3pm.

This list represents some of the extensions to our model that are currently under investigation. Extensions into more abstract domains, such as general problem solving using state transition graphs, are also under consideration.

6 - Summary and Conclusions

Planning robot movement in dynamic environments demands that the dynamic aspects of the environment be modeled in at least as much detail as the movements of the robot. We have created a representation system that allows dynamic aspects of the environment and performance aspects of the robot to be easily modeled. It also integrates this model with a high-performance route-planning algorithm. This system has been extended into an incremental route planner which can be used for real-time tactical planning in unpredictable domains. This system has been implemented in an SIMD simulator running on a VAX.

7 - Bibliography

[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*, Technical report 292, Yale University Computer Science Department, 1984.

[Hillis85] Hillis, W. D., *The connection machine*, MIT press, 1985.

[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.

[McDermott84] McDermott, D. V., Davis, E., Planning routes through uncertain territory, *Artificial intelligence*, v22, pp. 107-156, 1984.

[McDermott83] McDermott, D. V., *The nisp manual*, technical report 274, Yale University Computer Science Department, 1983.

[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.