

A Spatial Representation System for Mobile Robots

David Miller

Department of Computer Science
Yale University
New Haven, Ct. 06520

Abstract

A two dimensional spatial representation system is presented which makes efficient use of distance information for accomplishing common mobile robot tasks. Ways to use the representation for route planning, positioning, and execution monitoring are presented. The representation is very space efficient, and all the transformations to be performed on it are computationally tractable.

1 The Function of a Spatial Representation

In the domain of mobile robots space, and how it is represented, plays a pivotal role in any problem solution. The spatial representation system serves three purposes:

1. It provides a framework for incorporating newly discovered information about the robot's environment.
2. It provides the necessary information to do route planning.
3. It gives information for monitoring the position of the robot during task execution.

Over the years several representation systems have been developed to tackle each of the above systems. Most of these systems have been designed to attack only one or two of the above problems. Only a very few of these systems have been designed to work with a real mobile robot.

The representation system to be presented allows easy calculation of the robot's position using only its onboard sensors. Methods for plotting navigable routes and monitoring their traversal are also given. The emphasis of this system is not to find the shortest route, nor to calculate the robot's position to the highest accuracy. Rather, this system is designed to find the *easiest* route for the robot to follow and to provide corrective information should the robot's path deviate from the precalculated one.

Most research on mobile robots has dealt with *idealized robots*. Mobile robots that are currently available differ from their idealized counterparts in the following areas:

- Mobile robots have low resolution, noisy sensors
- Mobile robots have very limited effector accuracy, i.e. limited dead reckoning abilities
- *Clean* surroundings are necessary for most robots to function in a satisfactory manner (e.g. smooth floors, flat walls, static environment)
- Most mobile robots have a projection onto the plane of the floor that is roughly circular

This paper will be used to present a representation system that allows a class of simple, *real* robots to plan and execute maneuvers about their world. The prototype robot at which this representation is aimed is the Heathkit *Hero-1* robot, or any of the other personal robots currently available. All robots in this class have:

- low mechanical accuracy
- some sort of low acuity ranging sensor (usually a sonar)
- roughly circular projections onto the plane of the floor (the *Hero's* is a square)

2 Representing Space

For the purposes of this paper the world consists of a flat open plane on which vertical walls and obstacles are placed. The walls are also flat, but can be joined with other walls to form rooms and obstacles of arbitrary polygonal shape. Since the robot is limited to motion on the plane of the floor, the projection of walls and obstacles onto the plane of the floor capture all the relevant information about the world.

The basic unit of the spatial representation system is the *Map*. A map is made up of linked *regions*. Regions have a local coordinate frame. Walls and obstacles that exist in the area represented by the region are themselves represented by line segments whose endpoints' positions are designated by coordinates in the frame of the region. The borders of the region are marked with labels that specify the adjoining regions.

A distance sensor such as a sonar reports the distance to the nearest object that occurs inside its sensory cone. The sensory cone has a dispersion angle ϕ and a length D_{max} (see figure 1). At a distance of D_{max} the sensor cone has either intercepted the floor and is reflected back, or is no longer strong enough to return a reliable signal. A map is broken into regions by labeling every point within D_{max} of a wall with the name of the wall. All contiguous areas with the same markings are in the same region (see figure 2).

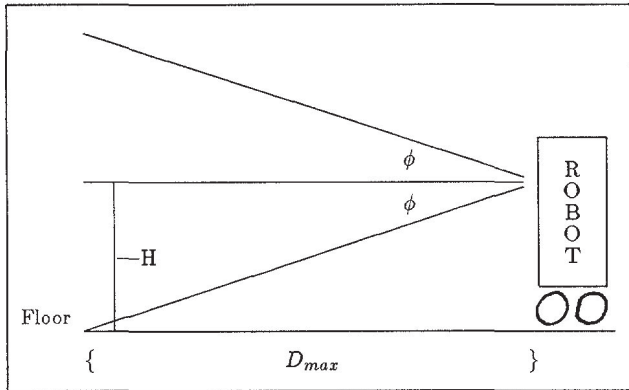


Figure 1: Sonar Dispersion & Definition of D_{max}

Regions can be any of four types: zero-F, one-F, two-F, or three-F. A floor-dwelling mobile robot has three degrees of freedom: the two planar dimensions and the robot's orientation. A type designation of j -F means that the sensor can be used to eliminate j degrees of freedom. So a zero-F region consists of an empty space that is at least D_{max} in any direction from the nearest landmark. Neither the robot's position or orientation can accurately be determined.

In a one-F region the robot's position can only be known for certain if two of its three coordinates (x, y, θ) are known a-priori. A two-F region can be formed by a single-sided large wall. The area within D_{max} of the wall and at least D_{max} away from each corner is a two-F region because the robot would be able to calculate its orientation and distance from the wall, but not its position along the wall. The areas within D_{max} of the corners are three-F regions since the robot's position and orientation can be fixed absolutely within the frame of the region.

A one-F region exists when there are two or more symmetrical positions where the robot might be located. For example, if the robot is in a hallway with two smooth parallel walls of a length greater than $2D_{max}$ its position can only be located relative to the walls, but it cannot tell which wall is which, and therefore which of two possible orientations it has. If a one-F region is entered from a two-F or three-F region then the orientation can be kept track of as long as the robot takes measurements often enough to keep its orientation errors less than one half the smallest change between the region's possible orientations (e.g. in the one-F region in figure 3, the robot must sample often enough to keep its errors below 90° , since there are two possible orientations 180° apart)

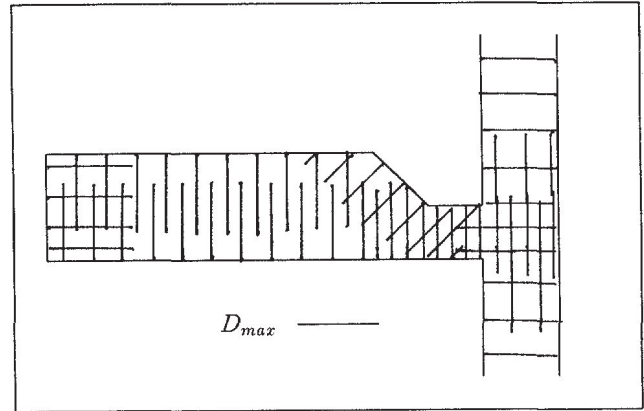


Figure 2: Dividing a Map into Regions

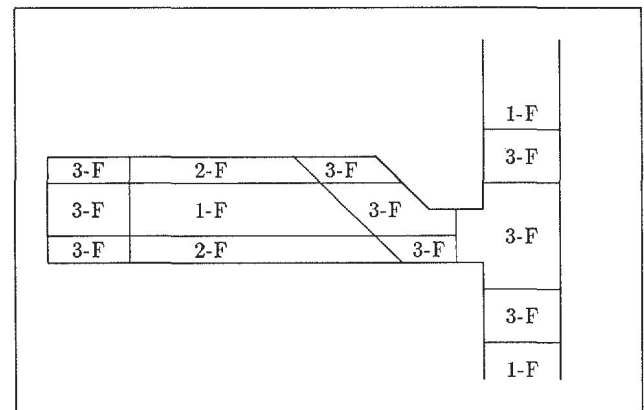


Figure 3: Typing the Regions in a Map

Each region has its own coordinate frame. A three-F region has exact positions for each measurement in it. Two-F and one-F regions use fuzzy numbers in one or more of their dimensions. The range of the fuzziness depends on the sensor used to determine it; in this research it is a combination of sonar and the robot's odometer.

Regions are made up of a set of edges. Each edge is represented by a pair of end-points whose Cartesian coordinates are specified in the frame of reference of the particular region. Zero-F regions have a coordinate frame, but do not have any edges. Each zero-F region contains the approximate coordinates of the nearest edges of each region which borders it.

The relative positions of features in two different regions cannot be known with great precision. The more zero-F, one-F, and two-F regions on the path between the regions in question, the less accuracy with which the two regions may be related. However, it is possible to get an approximate idea of the distance that must be traveled in order to go from one place to another. This is accomplished by adding up the tour distance for each region that must be traversed. A lower bound is found by adding up all the lower bounds for each of the one-F and two-F regions, along with the distances from the three-F regions. A

similar process is done to find the upper bound. But this does not help in relating position and orientation information from one region to another. The propagation of spatial information through the fuzzy regions is non-trivial. Such a computation is beyond the scope of this research, but has been discussed in detail in [1].

3 Robot Positioning

When a robot first starts up it must be able to determine its position given the vaguest information about where it is (e.g. which map covers its area). When the robot is traveling from one place to another it must be able to check and take corrective action to ensure that it stays on its desired path. This section discusses how to use the representation, introduced in the previous section, in combination with a sonar positioning system, in order to accomplish these tasks.

The type of region that the robot is in determines the amount of positional information that can be calculated. If the robot is in a zero-F region then the only positional information available would be extrapolations from the last known position, based on the robot's ability to do dead reckoning.

If the robot is known to be in a region that is one-F or greater, then positional information can be found using the system described in [2]. That system takes several sensor readings and does a heuristic search over the tree of possible matches between the observations and the edges in the map (see figure 4). Once a consistent matching between the observations and map edges has been found, a simple geometric construction is done to pinpoint the robot's location and orientation. A tree of m edges and k observations has m^k possible matchings. The various heuristics used in the search algorithm usually keep the complexity to $O(m^2)$.

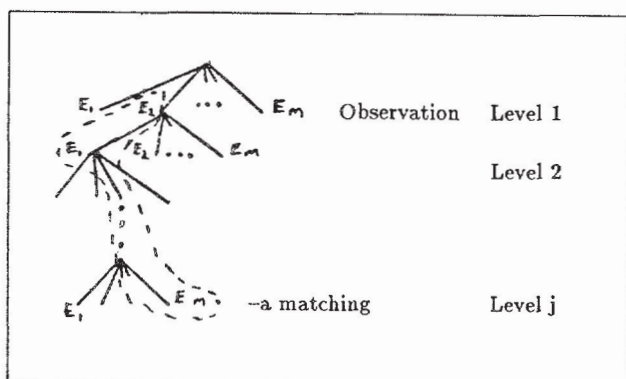


Figure 4: The Tree of Observations and Edge Matches

This performance could further be improved if the system could know which map edge the robot was closest to. With that information the positioning system could rate the partial matchings according to how consistent they were with that information, i.e. a matching that had the nearest observation matched to the wall known to be closest to the robot would be pursued before a matching that had that observation matched to some other wall.

In the positioning system, most of the heuristics are not effective during the first two levels of the matching tree; therefore its m^2 performance. If the first observation can be assigned to a single map edge with a very high probability of being correct, the performance of the system would become approximately linear.

One way to do this would be to divide the regions of the map into areas such that every point in a specific area was closer to one specific edge in the region than any other edge (see figure 5). Such a construction is based on the standard Voronoi diagram [3]. Algorithms that can find the Voronoi diagram of an arbitrary polygon in $O(m \log m)$ time were developed by Kirkpatrick in [4].

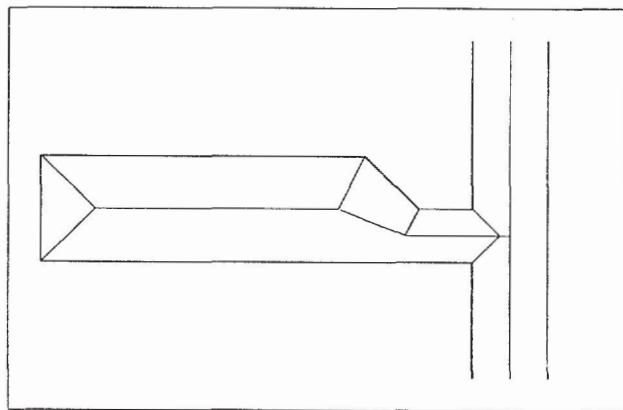


Figure 5: Voronoi Diagram of the Example Map

As the robot moves about the world, it keeps track of which V-area (Voronoi diagram area) it is in. Whenever an exact positioning of the robot is needed, a rough sonar scan¹ is made of the robot's surroundings. The shortest observation is then assigned to the map edge associated with that V-area. This gives the positioning algorithm a rough orientation with which to start pruning the search space. The rest of the positioning process is carried out in the normal manner.

The V-areas also allow the robot's distance from, and orientation to, a specific wall to be trivially calculated in many instances. If the region the robot is in is convex then the shortest observation taken in a fine scan² of the robot's surroundings will be the observation that is orthogonal to the map wall associated with the V-area where the robot is located. (see figure 6). Such a check is very useful for monitoring the robot's progress as it moves about.

¹A rough sonar scan is done by panning the sensor in a full circle, collecting four to eight approximately evenly spaced readings.

²A fine scan is when the sensor is panned in a full circle with observations being taken every few degrees. Since the time consuming part of a scan is the aiming of the sensor, fine and rough scans require approximately the same amount of time to execute.

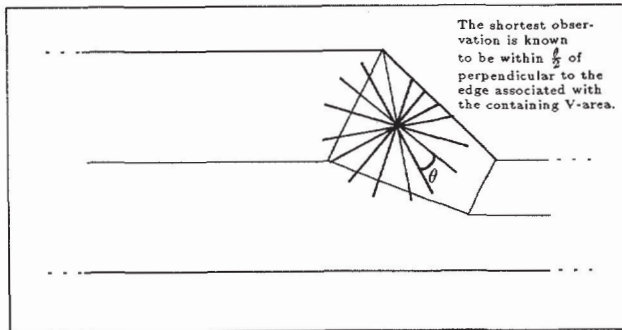


Figure 6: A Quick Check of Orientation and Distance to the Nearest Wall

4 The Route Planner

Route planning is a vital function in accomplishing most mobile robot tasks. It is therefore not surprising that a large amount of work has been done on the subject. The route planning method presented below is based heavily on previous systems created by other researchers. However, the other systems tried to find the shortest or safest paths, although these are often the most difficult for a robot to follow. A strategy employed by many systems is to keep as far away from obstacles as possible. Other systems plot the most direct course possible, barely skimming past obstacles. Unfortunately, both systems often produce routes that require large amounts of dead reckoning (see figure 7). The system presented here finds routes that are navigable by a robot with very limited abilities.

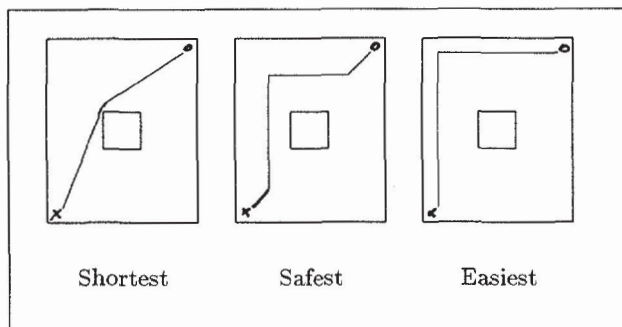


Figure 7: Navigating Across a Gymnasium with a Table in the Center

4.1 Finding the Easiest Route

The Voronoi diagram of a polygon is a made up of line segments that are equidistant between their associated edges of the polygon (see figure 5). Therefore, a path made up of the segments of the Voronoi diagram will stay as far away from the walls of a polygon as is possible. Because of this property, Voronoi diagrams have been used as the basis for several route planning systems: [5], [6], [7], [8].

Unfortunately, routes from such a system would eliminate any advantage received from the modifications to the positioning algorithm described in the previous section. The advantage that the Voronoi diagram brought to the positioning algorithm was based on the system knowing what V-area the robot was in before the positioning process was started. A route made up of the Voronoi diagram segments would keep the robot on the borders between the V-areas.

What is desired is a route that travels down the centers of the appropriate V-areas, but which areas are these? The ease of navigation for a specific V-area can be determined by the type of region that contains it. Zero-F regions are very difficult to travel through because they rely on dead reckoning. Three-F regions contain ample information for execution monitoring, but tend to be more crowded than other regions. They therefore require more care than other regions. One-F and Two-F regions can cover large distances with continuous low level monitoring. If the destination lies along the path of the one-F or two-F region then large distances can be easily and reliably traversed.

The general rules for finding an easy to navigate route are:

1. Traverse a two-F or three-F region as early as possible in the route. This allows the robot to totally orient itself early on, making it possible for the robot to traverse a one-F region without ambiguity.
2. Minimize the total number of regions to be traversed
3. Travel along one-F and two-F regions when possible to get the most distance for the fewest regions.
4. Traverse zero-F regions only under extreme circumstances; plot a course that is safe from dead reckoning errors (see figure 8).
5. Traverse a three-F region as close to the destination as possible in order to allow a full positioning of the robot.

If the projection of the robot into the plane of the map is circular, then the easiest to follow path can be found by the following algorithm:

1. Grow the walls and obstacles in the map by an amount R which is equal to the radius of the robot (see figure 9). This allows the robot to be considered as a point for the remainder of the route planning process.
2. Calculate a conduit, [9], made up of connecting V-areas that contain the starting and destination points. This can be found by doing a heuristic search with a rating function based on the rules in the paragraph above (see figure 10)
3. Compute the Voronoi diagram of the polygon formed by the areas in the conduit.
4. Plot a direct course from the starting point of the robot to the nearest vertex of the new Voronoi diagram. Do the same for the destination point.
5. Do a heuristic search of the graph formed by the segments of the newly calculated Voronoi diagram, where the cost function is the length of each proposed route. The result

of the search is the shortest path through the conduit.
(see figure 11)

6. Annotate each part of the route with the V-area it traverses.

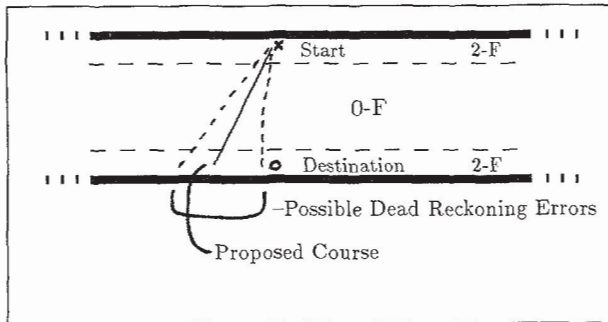


Figure 8: Plotting a Course Across a Zero-F Region

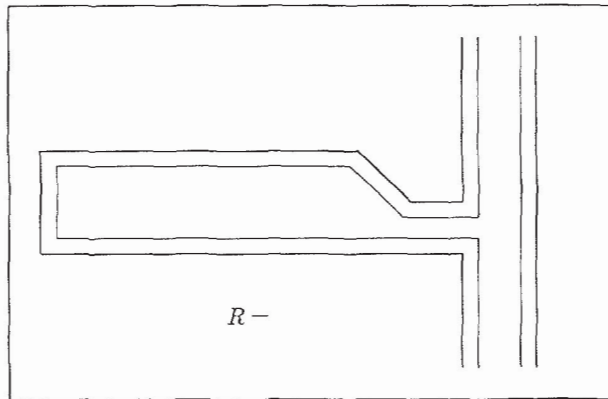


Figure 9: Growing the Walls by R

5 Applicability and Conclusions

Since this representation system is limited to man made environments (e.g. homes and warehouses) it may not be immediately apparent why dead reckoning should be avoided. However, as soon as one sees a robot perform in less than ideal conditions the answer becomes obvious. A robot's speed, turning rate, and reliability can differ greatly from surface to surface. A program that makes a robot roll in a perfect circle on smooth tile may cause the robot to execute an ellipse or spiral when run on a slight grade. Carpeting can cause drastic changes in performance characteristics. Even rolling from scuffed to freshly waxed tiles can cause noticeable differences in a robot's behaviour.

If the robot is equipped to recognize the type of surface it is traversing and modify its behaviour accordingly, it must still periodically check its position in order to calibrate that behaviour. Many surfaces such as thick carpeting and polished floors have performance characteristics which change with time and use, making surface dependent behaviour inaccurate.

The system presented above helps to make positioning by rangefinder an efficient process. It also annotates route maps in a way that will aid a planning system in deciding when to check the robot's position and how much information it can expect to get from such a check. Since the regionalization and Voronoi diagram of each region can be computed in advance, route planning and positioning can be done efficiently at runtime.

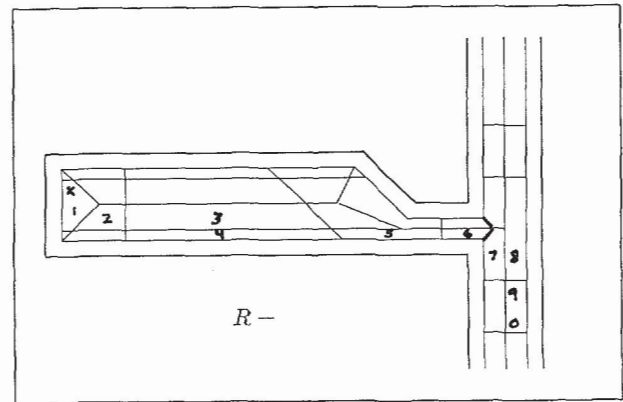


Figure 10: Finding a Navigable Conduit

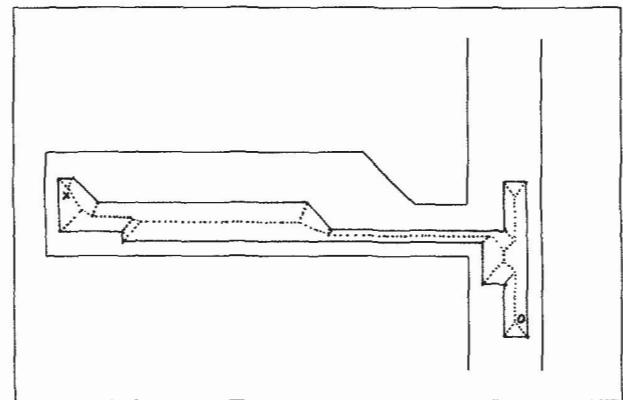


Figure 11: The Easiest to Navigate Route

6 Acknowledgements

The author wishes to thank Dan Gusfield, Joseph O'rourke, and Drew McDermott for their suggestions and ideas concerning this research. Thanks are also given to Thomas Dean and Yoav Shoham for their comments on earlier drafts of this paper.

This work was supported in part by the Advanced Research Projects Agency of the Department of Defence and monitored under the Office of Naval Research under contract N00014-83-K-0281.

Bibliography

- [1] Davis, E., *Representing and Acquiring Geographic Knowledge*, Technical Report 292, Yale University Computer Science Department, 1984.
- [2] Miller, David, Two Dimensional Mobile Robot Positioning, *Proceedings of the Ninth William T. Pecora Memorial Remote Sensing Symposium*, IEEE, USGS, NASA, ASP, Sioux Falls, South Dakota, October 1984, pp. 362-369.
- [3] Shamos, M., Hoey, D., Closest Point Problems, *Proceedings of the 16th Symposium on Foundations of Computer Science*, 1975, pp. 151-162.
- [4] Kirkpatrick, D., Efficient Computation of Continuous Skeletons, *Proceedings of the 20th Symposium on Foundations of Computer Science*, 1979, pp. 18-27.
- [5] O'Rourke, Joseph, Convex Hulls, Voronoi Diagrams, and Terrain Navigation, *Proceedings of the Ninth William T. Pecora Memorial Remote Sensing Symposium*, IEEE, USGS, NASA, ASP, Sioux Falls, South Dakota, October 1984, pp. 358-361.
- [6] Brooks, Rodney, A., Solving the Find-Path Problem by Good Representation of Freespace, *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Pittsburgh, Pennsylvania, August 1982, pp. 381-386.
- [7] Rowat, P.F., *Representing Spatial Experience and Solving Spatial Problems in a Simulated Robot Environment*, Technical Report, University Of British Columbia, 1979.
- [8] O'Dunlaing, C., Sharir, M., Yap, C., Retraction: A New Approach to Motion Planning, *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, ACM, Boston, Massachusetts, April 1983, pp. 207-220.
- [9] McDermott, Drew V. and Davis, Ernest, *Planning routes through uncertain territory*, *Artificial Intelligence*, 22 (1984), pp. 107-156.