The Path Planning Component of an Architecture for Autonomous Vehicles

Richard Fox, Antonio Garcia Jr. and Michael L. Nelson
Department of Computer Science
The University of Texas Pan American
Edinburg, TX 78539, USA
Phone: (956) 381-3635
Email: fox@cs.panam.edu
tonyg@hiline.net
nelson@cs.panam.edu

Abstract

Path planning for an autonomous vehicle can occur at two different times. First, path planning might occur at mission specification time when the vehicle's initial path is determined and used to specify other mission factors. This task makes use of some model of the environment in planning a path that will avoid obstacles and hazardous areas. A second type of path planning might occur while the vehicle is underway to avoid unexpected or previously unknown obstacles and hazardous areas. This is path re-planning. This paper concentrates on path planning and path re-planning in both two-dimensional and three-dimensional environments as used in the STESCA general control architecture for autonomous vehicles.

Introduction

Path planning is a fundamental task of autonomous vehicles. A wide variety of algorithms exist that address different factors of path planning (for instance (Kavanaugh & Werner 1995)). These factors include whether the path is pre-generated (prior to run-time), reactive (generated at run-time) or some combination, whether the path is in a two- or three-dimensional space, what form and importance obstacles have, whether the space is static or dynamic and in dynamic environments, how path re-planning might occur, whether characteristics of the terrain such as slope or texture come into play, and how efficient the search for a path is. This paper presents algorithms for performing path planning and path re-planning for autonomous vehicles in either two-dimensional or three-dimensional space with either static or dynamic obstacles.

The path planning algorithm described herein is based primarily on geometric equations dealing with lines and intersections with objects. The path planning algorithm consists of three parts: a line generator which creates a path between two points, a collision checker which determines if the path intersects with any of the obstacles in the space, and a collision processor which alters the current path to go around the obstacle which is causing the collision. The collision checker may also use a heuristic function to evaluate how safe the path segment is by considering whether the path segment enters any hazardous areas which might include steep slopes or unacceptably rough terrains. A path selector then chooses the path which best fulfills the mission specifications.

This research is one component of a larger project investigating the feasibility of the STESCA control strategy. STESCA (Strategic-Tactical-Execution Software Control Architecture) is an approach to providing a general-purpose control architecture for any form of autonomous vehicle. STESCA is currently being applied to an autonomous underwater vehicle (Nelson 1998) and will be applied to a land-based wheeled vehicle.

This paper will offer a brief description of STESCA, followed by an examination of the path planning and path re-planning algorithms. The paper will then describe examples in two-dimensional and three-dimensional environments. The paper will conclude with an analysis of how the algorithm will be enhanced for actual usage by various robotic systems.

STESCA

STESCA (Nelson 1998; Nelson & Garcia 1997; Nelson & Rohn 1996), the Strategic-Tactical-Execution Software Control Architecture combines three distinct levels of control. The top Strategic level is used for mission specification. The middle Tactical level decides how to carry out those specifications. The bottom Execution level controls the actual hardware (i.e., sensors and control systems) of the vehicle. The Tactical level can be thought of as a collection of agents, contributing to solve the overall mission as presented. All interactions between the user and the vehicle occur at the Strategic level. Interactions between vehicle components are determined at the Tactical level. An overview of STESCA is provided in figure 1.

In STESCA, initial path planning is a part of the Strategic level. Along with various mission specifica-

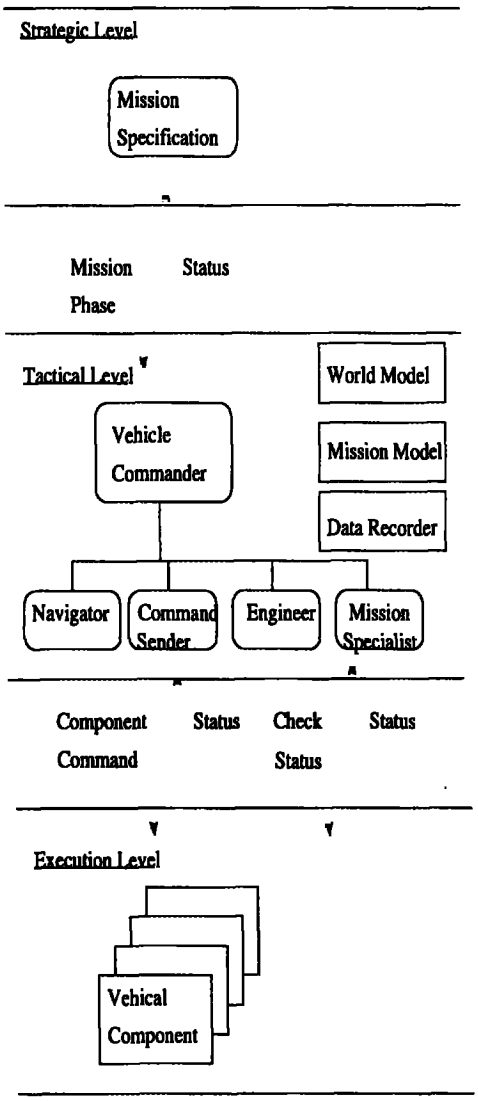

Figure 1: The Components of STESCA

tions, the user provides both the starting and the destination locations of the vehicle. The path planning algorithm then generates a path, using a world model which represents the environment of the vehicle including any known obstacles or hazardous areas.

Once the vehicle is under way, the Tactical level oversees the running of the vehicle. If unforeseen circumstances arise, the Tactical level must decide whether to continue with the given path. Choices include continuing with the current path, aborting the mission, altering the path by using an avoidance algorithm, or performing path re-planning by calling upon the path planner at the Strategic level.

The Tactical level uses a collection of software components to carry out its responsibilities. The vehicle commander coordinates the activities of all the other components of this level. A navigator is responsible for determining the vehicle's current location. The command sender is responsible for creating the appropriate set of vehicle component commands to carry out the specified mission. An engineer maintains the status of vehicle components. A mission specialist is responsible for collecting data during the mission.

The Tactical level also maintains three data stores: the world model, the mission model, and the data recorder. The world model is pre-loaded with information about the area of operation including whatever obstacles are known to exist within this area. The world model is used for path planning and path re-planning (if necessary). During the course of the mission, sensors and other information gathering agents update the world model as necessary to have it properly reflect the environment. The mission model maintains a copy of the stated mission, along with the vehicle component commands which were generated to carry out the mission. This information is used for post-mission analysis. The data recorder is used by the mission specialist to store information collected during the mission.

Path Planning and Selection

A key element of any mission carried out by an autonomous vehicle is planning a path or sequence of steps to take the vehicle from its present location to its destination. The "path planner" is the component of the software system that determines this path or sequence of steps. It uses some representation of the environment to take into account such factors as obstacles, rough terrains, and risky areas in order to avoid them (Chen, Szczerba, & Jr. 1995; Kamon & Rivlin 1995). For STESCA, initial path planning takes place at the Strategic level. The user specifies the starting point of the vehicle and the destination. The path planner works in either a two-dimensional space, for instance a land-based robot, or a three-dimensional space, for instance an underwater vehicle. The path planner requires some representation of the environment space, which includes information about obstacles (e.g., buildings, hills, rocks), terrain (e.g., rough versus smooth terrain, steep versus shallow slope) and risky elements (e.g., a mine field). Additionally, in the three-dimensional model for an underwater environment, information about currents may be included.

Once the vehicle is under way, the Tactical level must decide, if unforeseen circumstances arise, whether to continue with the given path, alter the path by using an avoidance algorithm, or perform path re-planning by calling upon the path planner at the Strategic level. Figure 2 shows the three primary levels of STESCA and how path planning and path re-planning interact.

The path generated by the path planner is composed of a series of line segments that are described by their endpoints in Cartesian two-dimensional or three-dimensional space. Each line segment is a path that is free of obstacles although it may enter an area of rough

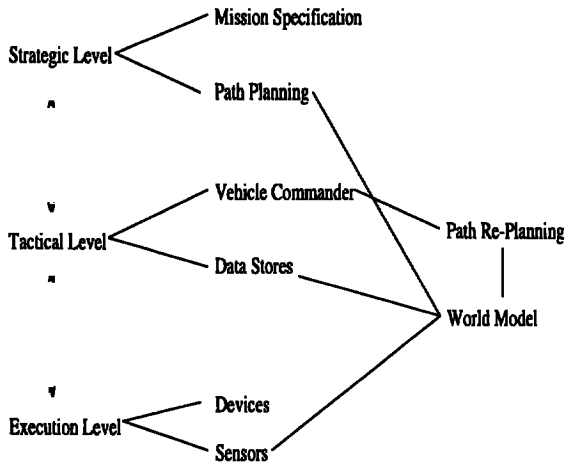

Figure 2: Path Planning and Path Re-planning in STESCA

or risky terrain. The obstacles are denoted, at present, as circles, spheres or cylinders, with a center point, radius, and a height (for cylinders). Further, a heuristic function can be applied, using the world model, to a line segment to determine if it enters a hazardous area.

Generating a path is divided into three steps. The first step is to generate the straight line between the source and destination points. The second step is to search through the list of known obstacles in the world model to determine if the given line intercepts any of them, and to use heuristics to determine if the path ventures into a hazardous area. If a collision is detected or an unsafe area is discovered, the third step, obstacle avoidance, occurs. Obstacle avoidance is processed by first generating a point outside of the obstacle or hazardous area, and then generating two new line segments, one from the source node to the new point and the other from the new point to the destination node. The two new line segments are then checked for collisions and hazardous areas.

Collisions between the line segment and any obstacles in the environment are determined by examining each obstacle in turn, and seeing if the line segments intercept that object at any point. Computing a collision is a simple matter of comparing the closest point of the line segment to the center of the circle (or sphere) and seeing if this distance is less than the radius plus a safety margin. If the distance is less, then the line segment comes too close or collides with the obstacle and the path must be rerouted around the obstacle.

Hazardous areas are detected by applying a heuristic function to each line segment. The function determines if the line intersects any area deemed hazardous. The function returns a value which is the degree that the area was deemed unsafe. A safety factor (provided at mission specification time) is used to determine whether the line segment is acceptable or not. Unsafe areas occur due to rough terrain, steep slopes, close proximity to dangerous obstacles or items, and so forth. If a line segment enters an unsafe area, the entire segment is deemed unsafe.

Obstacle avoidance (including hazardous area avoidance) is performed by selecting a point outside of the obstacle and altering the path to go from the source node to the new point and then on to the destination point. That is, the collision is avoided by going around the obstacle. There are numerous points to select from. Figure 3 demonstrates two possible points to select for obstacle avoidance, each on opposite sides of the obstacle. This figure also shows the two new paths generated from these points.

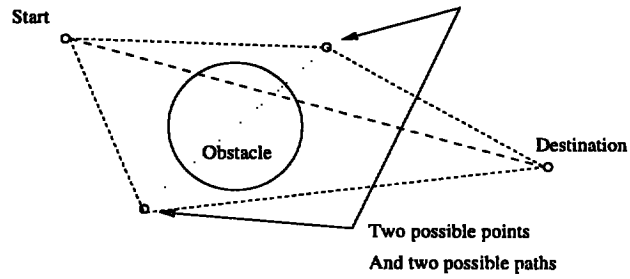

Figure 3: Obstacle Avoidance in STESCA

The path planner can operate in two different modes. In mode one, the point selected to avoid an obstacle is the point which is closer to the position of the collision. However, in mode two, the path planner will generate both points and retain both new paths as possible choices. The path planner operating in mode one will generate a single path composed of line segments. The path planner operating in mode two will recursively generate all possible paths using the two points around each obstacle. In mode two, the path planner then selects a path by using the following criteria. Which path is optimally shortest? Which path has the fewest line segments (i.e., the fewest course alterations)? Which path is deemed the safest in terms of hazardous terrain? Based on the mission specifications, one path will be selected.

It should be apparent that the algorithm in mode two could generate an exponential number of paths depending on the number of obstacle collisions. Therefore, the path planner can run in mode one if path efficiency is less important than generating a path rapidly and having the vehicle get underway. In mode one, the first safe path generated is selected no matter the distance. This version of the algorithm will use heuristics to ensure safety, but will not recursively generate more than a single path.

Path Re-planning

Because the path planner generates a path prior to the start of the mission, unforeseen factors may arise that must be taken into account. For instance, the vehicle may encounter an obstacle not represented in the

original world model or come across an unsafe area. In an underwater environment, unsafe areas might include heavy currents or shallow water. In a land-based environment, unsafe areas might include steep slopes or rough terrains. In both environments, uncharted mines may be discovered. During the execution of the mission plan, sensors are used to detect obstacles and identify hazardous areas. If such obstacles or areas are found by the vehicle sensors, then the world model is updated. The Tactical level of the control architecture includes a vehicle commander, whose task is to ensure that the mission is being carried out appropriately. As the world model is updated, the vehicle commander determines if the new information will in any way affect the current mission (for instance, obstacles found to be in the current path may require avoiding). The Tactical level also contains a path re-planner, which has the task of determining how the current path should be altered in such a situation.

If path re-planning is required, there are several possible options. First, the vehicle commander may decide to continue along the current path assuming that the changes to the world model are immaterial or non-threatening. This decision can be made if the heuristics applied to evaluating the current path using the updated world model indicate no change to vehicle safety.

Second, path planning can start anew from the current location using the updated world model and using the same destination point. This makes sense if there is enough time to perform a new path planning session and if the world model has changed sufficiently to warrant the amount of time it might take. In such a situation, the Tactical level hands command back to the Strategic level, which then generates a new mission plan based on the updated world model. The new mission plan might require additional or different commands at the Tactical level. Once the new mission plan, including the new path, is computed, control is returned to the Tactical level which then resumes the mission at the new point.

A third option is to backtrack to a previous position and simply select an alternative path around the obstacle, meeting up with the previous path at some later point. A fourth option is to attempt to "patch" up the path by going back to the original list from the initial path planning, and selecting a new route (if available) from the current point. Note that this option is only available if the original path was generated using mode two.

A fifth option is to abort the mission entirely. This last possibility was implemented in the NPS Autonomous Underwater Vehicle (AUV), which was programmed to surface and circle if the decision to abort was made (Nelson & Rohn 1996).

Examples

Two examples are demonstrated here. The first shows how the path planner charts a course in a two-dimensional space around circular objects using mode two. The second shows how the path planner charts a course in a three-dimensional space around spherical or cylindrical objects and hazardous areas. The second example, which is simplified for space, also demonstrates the need for path re-planning.

The two-dimensional example shows, in figure 4, the path generated for a land-based autonomous vehicle around a series of obstacles. The course of the vehicle is to start at point (1, 1) and navigate to point (15, 15). The initial path is shown as a hyphenated line. This path must be altered to avoid a collision with the obstacle located at position (6, 5). Two points are generated around the first obstacle, at (3, 8) and at (9, 2). Two new path segments are generated to get around the first obstacle. The segment that goes from (1, 1) to (3, 8) also collides with the obstacle at (2, 5) and two new points are generated to avoid this obstacle, at (0, 6) and at (4, 3). The segment that goes from (3, 8) to (15, 15) also collides with an obstacle, at (10, 13). Two additional points are generated to avoid this obstacle, at (7, 16) and (13, 10). The path from (1, 1) to (9, 2) to (15, 15) does not have any further collisions.

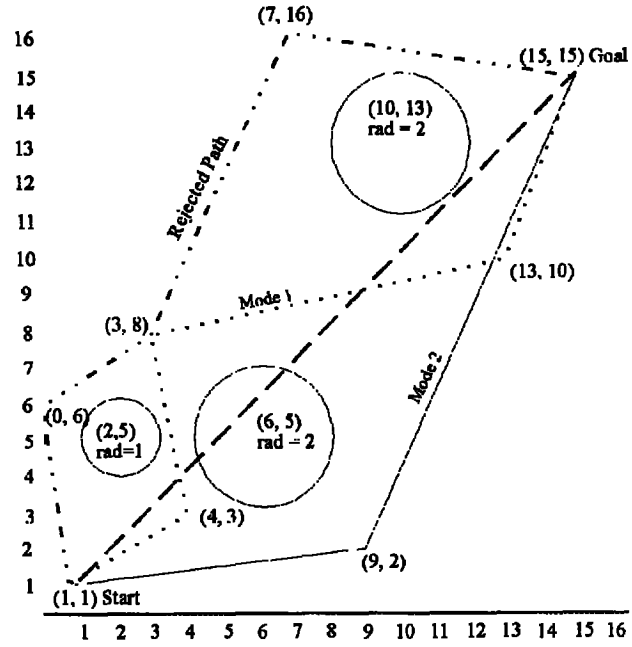

Figure 4: Two-Dimensional Land-Based Example

The result of using mode two of the path planning algorithm is a set of all generated paths, as shown in figure 4. Based on the mission specification, the selected path may be any of these generated paths. In this example, the shortest path is chosen (represented in the figure as a solid line in the figure). Another path, denoted with dotted and dashed lines is rejected because it is longer. In mode one, the algorithm would

only generated a single path, shown in the figure using a dotted line. This line was also generated in mode two and rejected because it is not the shortest. It should be noted that a mission specification might be "the fewest path alterations possible" meaning the fewest number of path segments. The path chosen here as shortest would also qualify as the one with the fewest path segments. It should also be noted that this example does not include any hazardous regions.

The second example demonstrates an underwater environment and is shown in figure 5. The original path takes the vehicle from a position high in the water to a lower position. The original path collides with a single obstacle (represented as a cylinder from the sea floor) and therefore two alternative paths are created that go around the obstacle. The shorter path is selected. Once under way to the destination, a strong current is detected by the vehicle's sensors. The vehicle commander notes that this is a hazard and orders path re-planning. Path re-planner may select one of several choices depending upon the necessity. First, the current may be ignored (for instance, if it were not very strong or threatening). Second, the mission might be aborted having the vehicle surface. Third, the current path could be altered to go around the current. A fourth option is to backtrack around the current prior to engaging any of the prior options so that the vehicle does not get pushed off course. Again, the mission specifications combined with the strength of the current will provide the information necessary to make the decision.

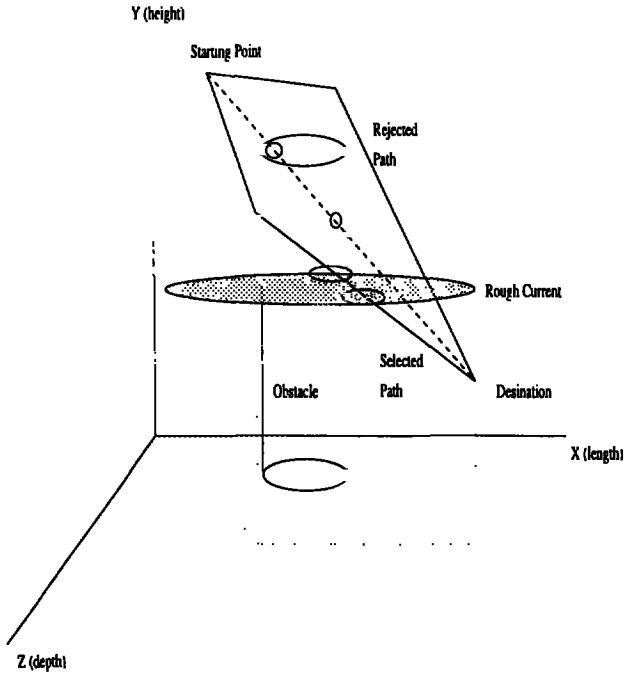

Figure 5: Three Dimensional Underwater Example

Future Work and Conclusions

The STESCA architecture is currently being tested on an autonomous underwater vehicle which had previously been programmed using a different control architecture. The intent is to reprogram it using STESCA, which should prove to be a more general control architecture. All work has been done on a simulator. The architecture is currently being implemented on a land-based robot. The path planning algorithm, described in this paper, has already been tested on a number of cases using both two-dimensional and three-dimensional world models. The algorithm is currently being modified to use heuristics in order to evaluate the safety of each generated path. There are also plans to include a greater variety of obstacle representations rather than circular or spherical. To date, all research has been very encouraging.

Acknowledgments

This work was supported in part by a Faculty Award for Research (FAR) from NASA. Many faculty and students at the University of Texas - Pan American have assisted in the overall creation and testing of STESCA.

References

Chen, D. Z.; Szczerba, R. J.; and Jr., J. J. U. 1995. Planning conditional shortest paths through an unknown environment: A framed-quadtree approach. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and System Human Interaction and Cooperation*, volume 3, 33–38. IEEE Press.

Kamon, I., and Rivlin, E. 1995. Sesnory based motion planning with global proofs. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and System Human Interaction and Cooperation*, volume 2, 435–440. IEEE Press.

Kavanaugh, M. E., and Werner, B., eds. 1995. *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, Volumes 1-3.* IEEE Computer Society Press.

Nelson, M. L., and Garcia, V. 1997. An object-oriented approach to autonomous underwater vehicle control. In *Proceedings of the 10th International Symposium on Unmanned Untethered Submersible Technology*, 385–393.

Nelson, M. L., and Rohn, V. 1996. Mission specification for autonomous underwater vehicles. In *Proceedings of Oceans '96*, 407–410. MTS/IEEE Press.

Nelson, M. L. 1998. A software control architecture for autonomous vehicles. In *Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31)*, volume III, 226–232.