Experiments in Real-time Path Planning for Riverine Environments

Caleb M. Reed

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Daniel J. Stilwell, Co-Chair
Andrew J. Kurdila, Co-Chair
Christopher L. Wyatt, Member

April 22, 2008
Blacksburg, Virginia

Keywords: autonomous surface vessel, path planning, obstacle avoidance, A*, occupancy map

Experiments in Real-time Path Planning for Riverine Environments

Caleb M. Reed

(ABSTRACT)

This work focuses on the development and implementation of an autonomous path planning and obstacle avoidance algorithm for an autonomous surface vehicle (ASV) in a riverine environment. The algorithm effectively handles trap situations, which occur when the river bends away from the destination. In addition, the algorithm uses real-time sensor feedback to avoid obstacles.

A general global route is proposed based on an *a priori* shoreline map. Then, local paths are calculated considering both the *a priori* data and measurements received from an obstacle sensor. These paths roughly follow the global path. The algorithm was tested on an ASV equipped with basic navigational sensors and an omnidirectional camera for obstacle detection, and experimentation verified its effectiveness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The principal goal of this research is to develop a path planning algorithm that enables an autonomous surface vehicle (ASV) to autonomously navigate in an unstructured riverine environment. The complex shape of a river and frequent presence of obstacles make such a task challenging. To account for these conditions, the algorithm performs two levels of planning. A global path plan is first developed, using an *a priori* map, that guides the ASV to follow the overall shape of the river. Then, short paths are repeatedly planned in order to avoid detected obstacles. Each of these plans is the result of an A* search conducted on an occupancy map generated using available vision-based range estimation data.

1.1 ASV Background

ASVs have been a topic of research for at least 15 years [1]. As a result, numerous ASVs have been developed and successfully implemented [1, 2, 3, 4]. Some ASVs are designed with specific capabilities, including acoustic signal tracking [5], environmental survey [6], and position verification for autonomous underwater vehicles (AUVs) [7]. However, none of these vehicles are capable of real-time autonomous obstacle avoidance.

1.2 System Setup

The ASV used in this research is shown in Figure 1.1. It is a product of the Autonomous Systems and Controls Lab (ASCL) at Virginia Tech, specifically designed to operate in a shallow-water environment. It is useful as a platform on which different concepts and algorithms can be developed and tested [8].

An inflatable pontoon hull provides a stable platform with good buoyancy characteristics.

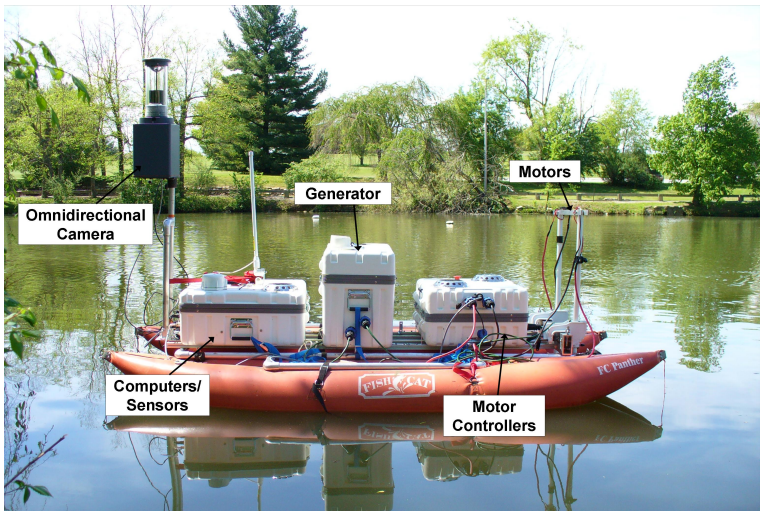

Figure 1.1: Configuration of the ASV

Table 1.1: Dimensions and Significant Properties

Dimensions	2.7 m x 1.5 m
Weight	125 kg (275 lb)
Operating Speed	1.5 m/s

It is propelled and steered by two electric thrusters powered by a gasoline generator. This generator also provides power to the electronics. Table 1.1 shows the applicable properties of this vehicle.

The ASV is equipped with a GPS receiver and a three-axis gyro for navigational feedback. Two laptops perform the computational tasks required by the ASV. All control algorithms are implemented on one laptop, while all machine vision and navigation algorithms are implemented on the other. In this work, they will be addressed as the control laptop and vision laptop, respectively. All water-sensitive equipment is housed inside waterproof boxes. Communication with a remote base station is accomplished through a mesh network, enabling an operator to monitor the system performance and have remote control of the ASV if necessary.

An omnidirectional camera on the ASV determines the position of points on the shoreline and other fixed obstacles from features in the camera image. The camera captures images at a rate of 5 Hz. Localization of features in the camera image is accomplished by stereo vision as the ASV moves. Since the baseline between images is known, the location of a feature can be estimated by triangulation of its location in successive images. Due to this procedure, features directly in-line with the velocity of the ASV are not observable, and significant error is introduced for features near this line. Therefore, tracking only occurs for features that are close to abeam of the ASV [9]. Also, the camera is most accurate for features between 10 m

and 50 m away. Position estimates for features within this range are generated, and a list of GPS coordinates for each tracked feature is produced [10]. In this way, the ASV is capable of providing the navigation system with the necessary feedback for obstacle avoidance.

1.3 Problem Description

Using this platform, the objective is to implement an autonomous navigation and obstacle avoidance system. Operation in a riverine environment is particularly interesting because of the navigational challenges provided by frequent bends in the river and the high density of artificial structures. All path planning and control decisions are to be made on-board the ASV. It is assumed that the ASV is operating in an area with some sort of map or satellite imagery available so that an *a priori* map is available, although the map may be inaccurate and will not indicate all possible obstacles. Success is measured by the ASV's ability to traverse a section of river without external navigational input. Testing is conducted on Peak Creek, a tributary of Claytor Lake in Pulaski County, Virginia. The operating area is shown in Figure 1.2.

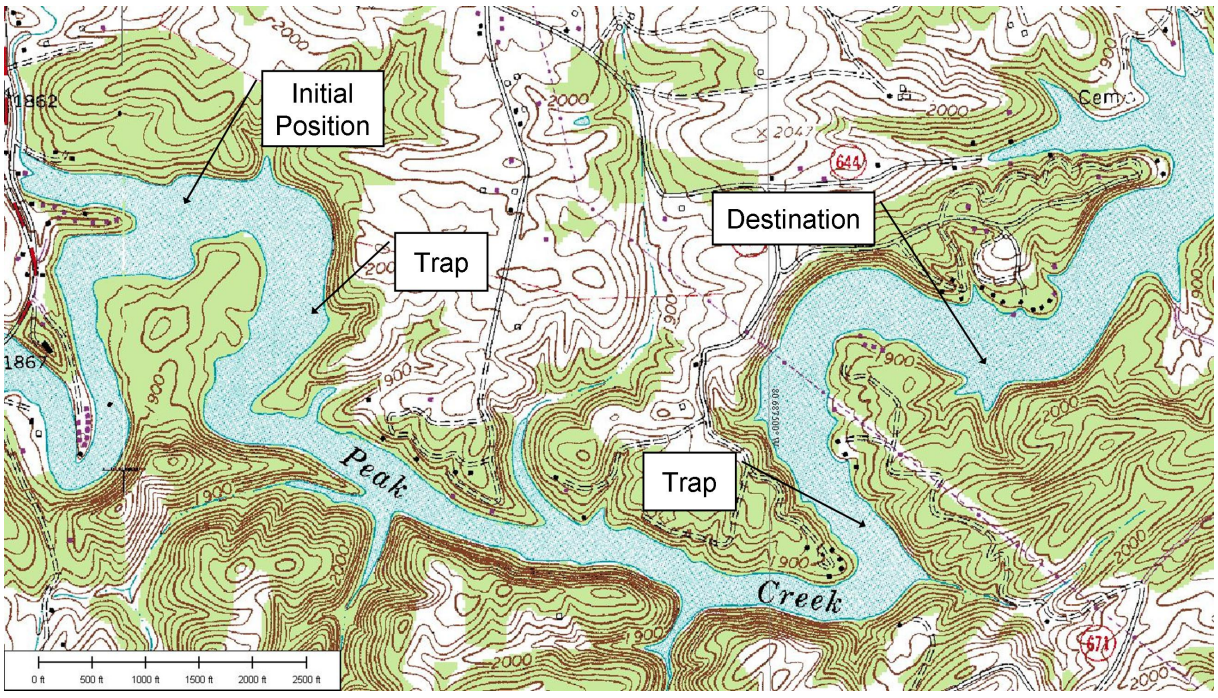

Figure 1.2: The operating area for ASV testing

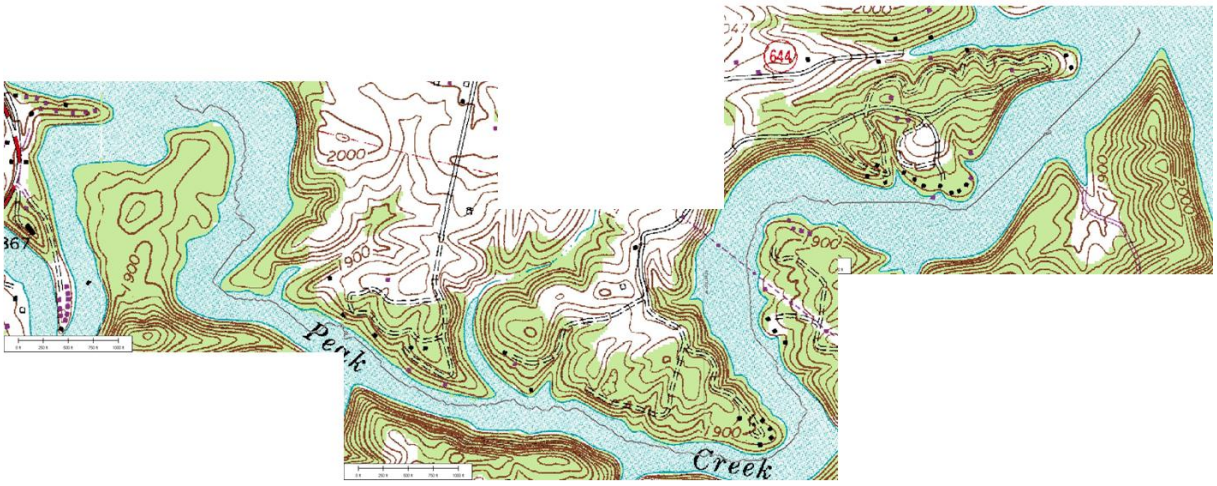

Figure 1.3: Complete path of ASV navigation experiment

1.4 Solution

The product of this research is a functional, experimentally proven algorithm which enables an ASV to autonomously plan a route to a destination and avoid obstacles encountered along that path. The results of the experiments are summarized in Figure 1.3. An occupancy map of the operating area is constructed using *a priori* shoreline data. A global path plan is developed based on this map. Then, smaller high-resolution occupancy maps are developed that incorporate measured obstacle data with the *a priori* data. Elements of the global path serve as destinations on these local maps. This relationship is important to the effectiveness

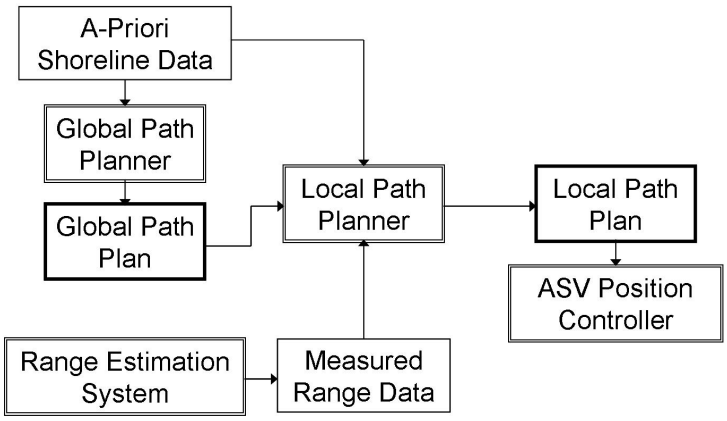

Figure 1.4: Summary of path planning algorithm

of this algorithm and it is described in detail in Section 3.3. The result of the interdependence

between the global and local planning steps is an algorithm that avoids traps, which occur when the river bends away from the destination as shown in Figure 1.2, while being flexible enough to avoid obstacles in real-time. The algorithm is illustrated by the flow chart in Figure 1.4.

Chapter 2

Exploration

An ASV with the ability to navigate in any environment without knowledge of that environment would be useful. This type of navigation is termed *exploration*. Experimentation using an algorithm intended to provide this functionality to the ASV produced modest results. In this work, the algorithm that does not rely on *a priori* data is called the *exploration algorithm*.

The exploration algorithm is simple. A map is generated of the area immediately surrounding the ASV, 40 m in each cardinal direction. The map is populated by data measured by the omnidirectional camera. This map generally does not contain the destination, so a path is planned that is close to a straight line between the current position of the ASV and the desired destination. Obstacles along that line cause the ASV to deviate from a straight path.

2.1 Experimental Results

To test this algorithm, the ASV is deployed in Peak Creek. Various destinations are tested that require different degrees of exploration. The initial tests are simple cases with no obstructions between the initial position and the destination, such as the one shown in Figure 2.1. This run is 340 m long.

Gradually, the difficulty of the exploration problem is increased. Another situation requires the ASV to travel a longer distance and avoid the shoreline. As Figure 2.2 shows, the ASV is able to complete this task. The total length for this trial is approximately 1470 m. Some practical performance characteristics of the omnidirectional camera are illuminated by this experiment. Due to the artificial stereo geometry construction, the camera performs best when it is travelling with constant velocity. Therefore, when the ASV performs a significant maneuver to avoid a detected obstacle, the number of false detections drastically increases. This is detrimental to the performance of the path planning algorithm because it introduces

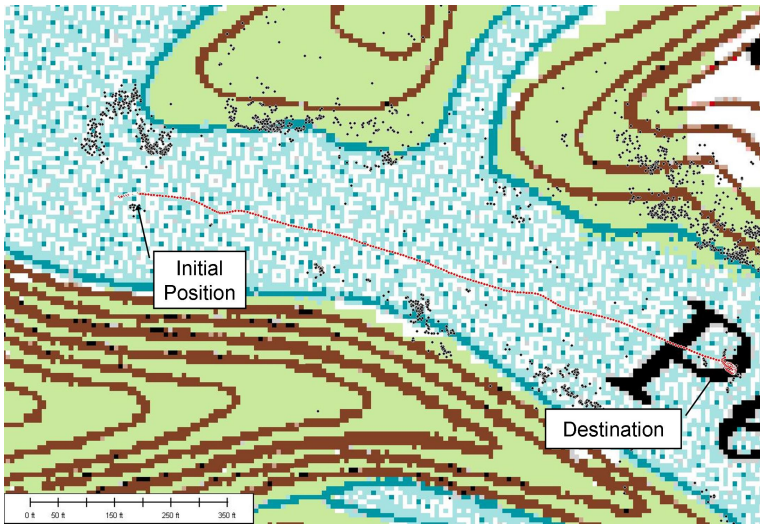

Figure 2.1: The ASV conducts a short exploration experiment

a large number of false obstacles that are subsequently avoided. The ASV still reaches the destination, but it follows an inefficient path.

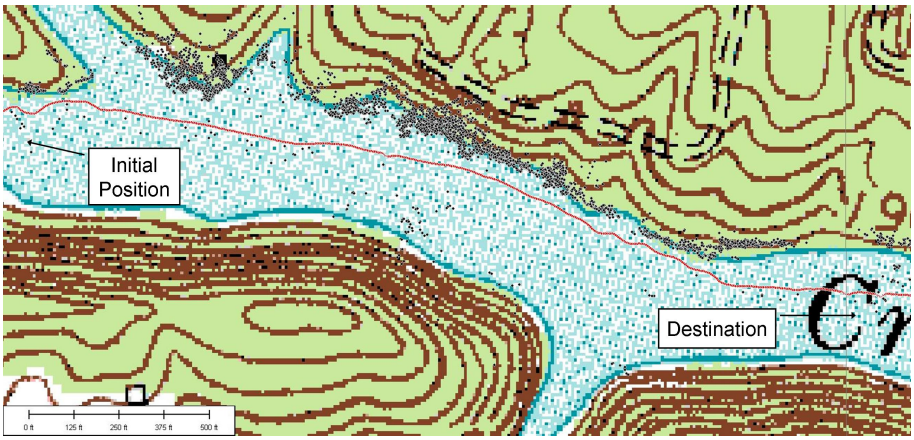

Figure 2.2: The ASV conducts a long exploration experiment

The last set of successful tests requires the ASV to navigate around a bend in the creek. These are difficult exploration experiments because the ASV has to deviate significantly from the straight-line path and depend on updates from the camera to learn the shoreline location. Figure 2.3 shows the path of the ASV around this bend. After an initial user override, required because of the inability of the omnidirectional camera to detect features directly ahead of the ASV, the path planner guides the ASV safely around the bend to the desired destination, a distance of about 590 m. This is a significant accomplishment because the path planner produces this path without *a priori* information. All decisions are based on measurements taken by the camera as the route was traversed.

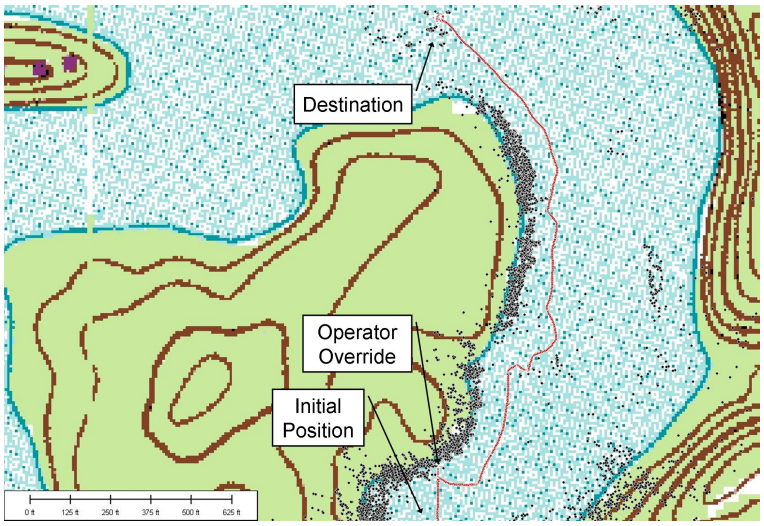

Figure 2.3: The ASV successfully navigates around a bend

2.2 Exploration Conclusions

This is the extent of the successful exploration experimentation. When there is a significant bend in the river away from the straight-line path between the current position of the ASV and the destination, local information that is used for planning the path is not sufficient for navigating around the bend. This condition is called a trap. Traps encountered during these experiments are shown in Figure 1.2.

Additional exploration algorithms were not attempted. Indeed, it has been shown in [11] that every exploration algorithm is inefficient for some potential environment. Specifically, that there is no upper bound for ratio of the distance covered due to a specific exploration algorithm to the length of the optimal path for a given environment. That means that every exploration algorithm is arbitrarily bad for some environment. The problem can be considered as a two-player game where one player chooses a search algorithm and then the second player sets up the environment. The second player is always able to set up a pathological environment that makes the particular search algorithm inefficient. Although the environment is fixed for these experiments, the theory states that there could be some environment that renders the path planning algorithm inefficient. Imperfect sensor readings further complicate the search task. Therefore, a completely different class of algorithms was selected to address the navigation challenge.

Chapter 3

A Priori Algorithm

The algorithm used to complete the navigation challenge depends on an *a priori* map and is called the *a priori algorithm* in this work. The *a priori* algorithm operates in two stages: a global planning stage and a recurrent local planning stage. The global path planner generates a route from the current position of the ASV to the desired destination using a low-resolution map populated with *a priori* data. Then, the local planner generates a route from the current position of the ASV to a nearby waypoint on the global route using a high-resolution map populated with *a priori* data and obstacles detected in real-time by the vision system. The local planner executes repetitively until the ASV reaches the desired destination.

The transition from an exploration algorithm to an *a priori* algorithm is supported by a number of factors. As Section 2.2 describes, exploration algorithms have no upper bound on efficiency. Experimentation showed the difficulty of such a task. Also, with the wide availability of maps and digital imagery, it is safe to assume that some *a priori* data is available for any area of interest. An *a priori* algorithm would eliminate the inefficiencies particular to exploration algorithms.

The *a priori* approach is particularly useful for long-range path planning in complex environments. In [12], an autonomous ground vehicle (AGV) completes a 142 mile-long course with the combination of an *a priori* route and local obstacle avoidance. In [13], a formal multi-horizon path planning technique for an AGV is presented that includes short-range obstacle avoidance, a mid-range path planner, and a long-range planner based on prior information or accumulated sensor measurements. [14] introduces the idea of using *critical points* derived from *a priori* data as subgoals for navigation. These efforts provide a skeleton from which to flesh out a path planning algorithm for an ASV.

3.1 *A Priori* Data

An ASV in a riverine environment encounters a complex navigational situation. Its overall direction is strictly dictated by the shape of the river, but it has great flexibility within the bounds of the shore. Therefore, a rough sketch of the shape of the river in the form of an *a priori* map enables the ASV to complete a mission quickly and efficiently. Furthermore, this data is not difficult to generate due to the proliferation of maps and satellite imagery and the contrast between water and land displayed in those media.

The map in Figure 1.2 is used to generate the *a priori* data for this work. The water in the image is selected using a color selection tool from a photo editing program. The remainder of the image is removed and the color of the water area is changed to black. Using this binary image, shown in Figure 3.1, and an edge detection function, the pixel coordinates of the shoreline are computed. Based on the GPS coordinates of the edges of the image, those pixel coordinates are converted to GPS coordinates for use as the *a priori* data set. This process is summarized in Figure 3.2, which is pseudocode for the MATLAB program created to conduct these calculations.

Figure 3.1: Black and white image derived from original map

```
pixel_edge = bwboundaries(binary_image); // Produces a vector of pixels
for each element of pixel edge
   GPS_edge = lower_bound + pixel_edge(i)*GPS_per_pixel;
end
```

Figure 3.2: Pseudocode for edge detection and GPS shoreline calculation

3.2 Global Path Planner

One of the steps of the *a priori* algorithm is the generation of a global path plan from the initial position of the ASV to the desired destination. A low-resolution occupancy map is generated using the *a priori* data. If an element of the *a priori* data set falls within a cell on the map, it is considered occupied. Otherwise, it is empty. An example is shown in Figure 3.3. This occupancy map consists of 10 m by 10 m cells. An A* search is conducted on this *a priori* occupancy map and a global path, P_g, is generated. Figure 3.4 shows one such global path. The output of the A* search is a list of waypoints in GPS coordinates. This process is not conducted in real-time, which is acceptable since the mission has not yet begun.

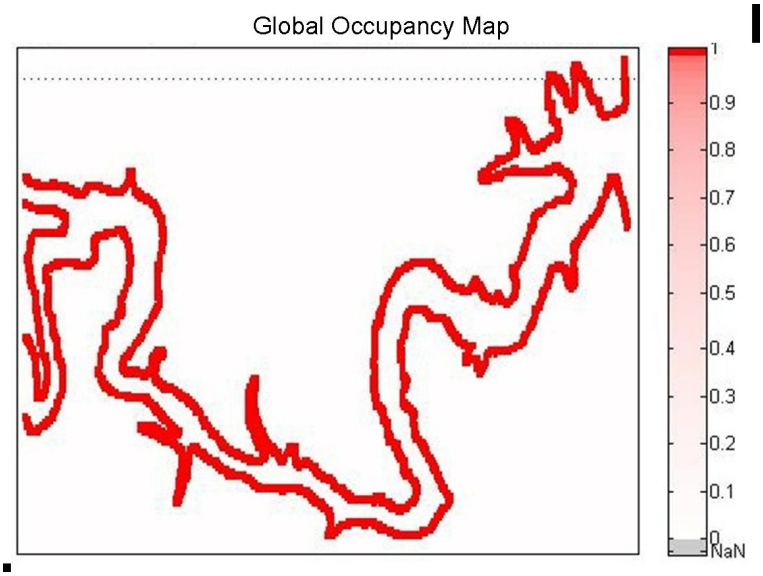

Figure 3.3: A global *a priori* occupancy map, each cell is 10 m by 10 m

3.2.1 A* Search

The A* search algorithm is a graph search that computes an optimal path from an initial position to a goal position in a global environment as long as a path exists [15, 16]. This guarantee is based on the condition that the heuristic function G used to calculate the cost of moving between cells is admissible. A heuristic is admissible if

$$\forall x \in X : 0 \leq G(x) \leq G^*(x) \tag{3.1}$$

where x are nodes within the graph X. G^* is the true cost function and G is an estimate.

As a brief description of the A* algorithm, the search begins at an initial node. The cost to come to each neighboring node, $C(x)$, and to go from that neighboring node to the

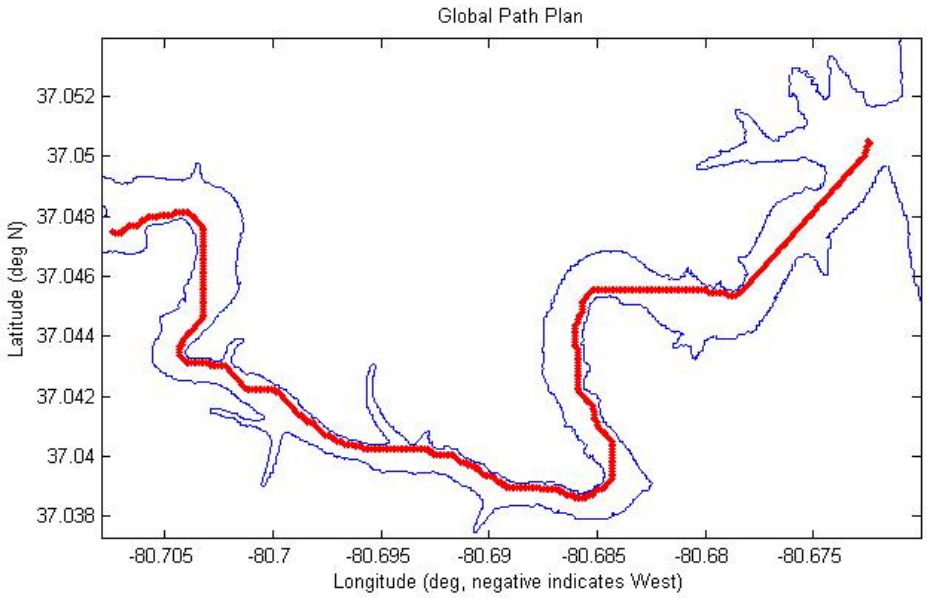

Figure 3.4: Results from a global A* search

destination, $G(x)$, is then calculated. $C(x)$ is calculated incrementally by adding the distance between the nodes with a scaled-up value of the occupancy map value and is assumed to be true, $C^*(x)$.

$$C(x) = \text{distance}(x_0, x) + \alpha \text{OccMap}(x)$$

where $\alpha > 1$. $G^*(x)$ is estimated as the Euclidean distance between the neighboring node and the destination [17]. Since the cost to travel between nodes is the distance between the nodes added to the scaled occupancy value, the minimum cost between two nodes would be the Euclidean distance.

$$G(x) = \text{distance}(x, x_d) + \alpha \cdot 0 \leq G^*(x)$$

where the true cost $G^*(x)$ includes the actual risk incurred by the proximity of obstacles. Therefore, using the Euclidean distance as the heuristic for estimation of the cost to go to the destination cannot be an overestimation and, as a result, renders the algorithm admissible.

The cost assigned to each node is determined by summing $C(x)$ and $G(x)$. The node with the lowest cost is chosen as the next node to explore. The remaining nodes are stored in an ordered list with the lowest-cost node at the top. The neighbors of this new node are opened and their cost is calculated and stored in the list and the process continues. If the search reaches a node that has already been explored and was stored with a lower cost, which would happen if the search had to backtrack out of a trap, that expensive path would be discarded and the search would resume from the low-cost node. Each node is given parent information so that the optimal course can be extracted by following the parent trail back to the initial position when the search reaches the destination. Pseudocode for the A* algorithm from [18]

is shown in Figure 3.5. This mature algorithm is explicitly defined in the refrences from this section.

3.3 Local Path Planner

The global path planner is not executed in real-time because it is operating on a large area and takes a significant amount of time to complete. In order to incorporate measured data and conduct planning in real-time at a useful resolution, it is necessary to include a local planning routine in the *a priori* algorithm. The critical aspect of this algorithm is the relationship between the global planning step and these repetitive local planning steps. The global path plan is developed, made up of GPS coordinates as waypoints for the route. However, instead of following this route directly, each local plan is developed from the position of the ASV to a waypoint on the global path. This local planner executes in real-time and incorporates measured obstacle data from the omnidirectional camera. In this way, real-time obstacle avoidance is achieved while still travelling in the most efficient general direction.

A high-resolution (1 m^2 per cell) local occupancy map is generated with contributions from both the *a priori* data set and measured data from the omnidirectional camera. Occupancy mapping is described in detail in Section 3.4, but in general, an occupancy map is a representation of the environment in terms of the probability that a specific area is occupied. The occupancy map is centered on the midpoint of a line connecting the position of the ASV with the temporary target from the global path. An A* search is conducted on this local map in order to plan the detailed path the ASV will follow. Elements of P_g, the global path, serve as local destinations within these maps. P_g, from Section 3.2, is a vector consisting of waypoints along the global path, each between 10 m and 15 m apart. The coordinate on the global path that is closest to the current position of the ASV, $P_g[c]$, is calculated. Then, the coordinate on the global plan four elements ahead of the closest coordinate to the current position is selected as the temporary target:

$$x_{tt} = P_g[c + 4]. \tag{3.2}$$

This process is illustrated in Figure 3.6.

It is possible that the temporary target will lie on an obstacle. In order to compensate for that phenomenon, the distance between $P_g[c + 4]$ and x is greater than the ASV can travel in one planning iteration. The ASV will begin to move in that direction, but the next local plan will be generated before it arrives. Therefore, if the local destination lies on an obstacle which was not part of the original data set, the next iteration will plan around that obstacle. An example from experimentation is shown in Figures 3.7 and 3.8.

This effect is limited to deviations from the *a priori* data set of 20 m. Greater deviations would require the global path planner to execute again, incorporating the measured data.

```
// initialize a start node
StartNode.Loc = StartLoc
StartNode.CostFromStart = 0
StartNode.CostToGoal = PathCostEstimate( StartLoc, GoalLoc, Agent)
StartNode.Parent = null
push StartNode on Open

// process the list until success or failure
while Open is not empty
 pop Node from Open // Node has lowest TotalCost

 // if at a goal, we're done
 if (Node is a goal node)
  construct a path backward from Node to StartLoc
  return success
 else
  for each successor NewNode of Node
   NewCost = Node.CostFromStart + TraverseCost( Node, NewNode, Agent)
   // ignore this node if exists and no improvement
   if (NewNode is in Open or Closed) and (NewNode.CostFromStart <= NewCost)
    continue
   else // store the new or improved
    // information
    NewNode.Parent = Node
    NewNode.CostFromStart = NewCost
    NewNode.CostToGoal = PathCostEstimate( NewNode.Loc, GoalLoc, Agent)
    NewNode.TotalCost = NewNode.CostFromStart + NewNode.CostToGoal
    if (NewNode is in Closed)
     remove NewNode from Closed
    if (NewNode is in Open)
     adjust NewNode's location in Open
    else
     push NewNode onto Open
    end // store information
   end // else:  Node is new or improved
  end // for each successor NewNode
 end // else:  Node is not a goal
 push Node onto Closed
end // while Open is not empty
return failure
```

Figure 3.5: Pseudocode for the A* search algorithm

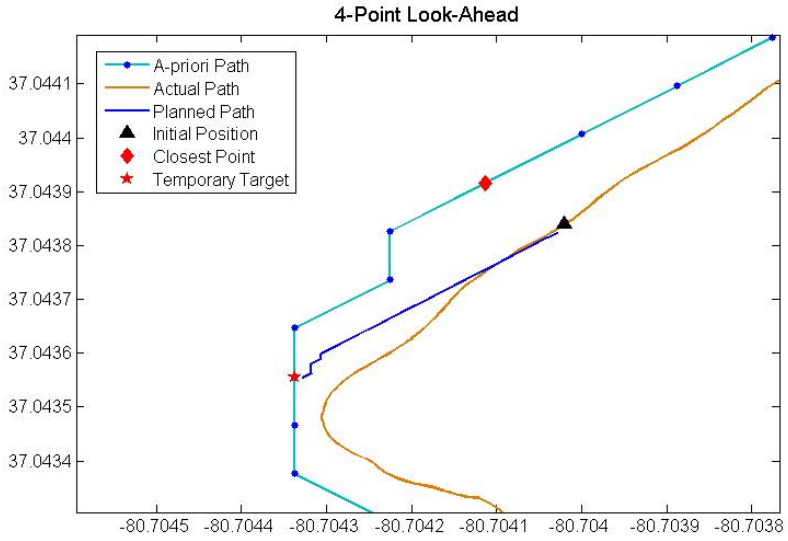

Figure 3.6: The temporary target is four points ahead of the closest point

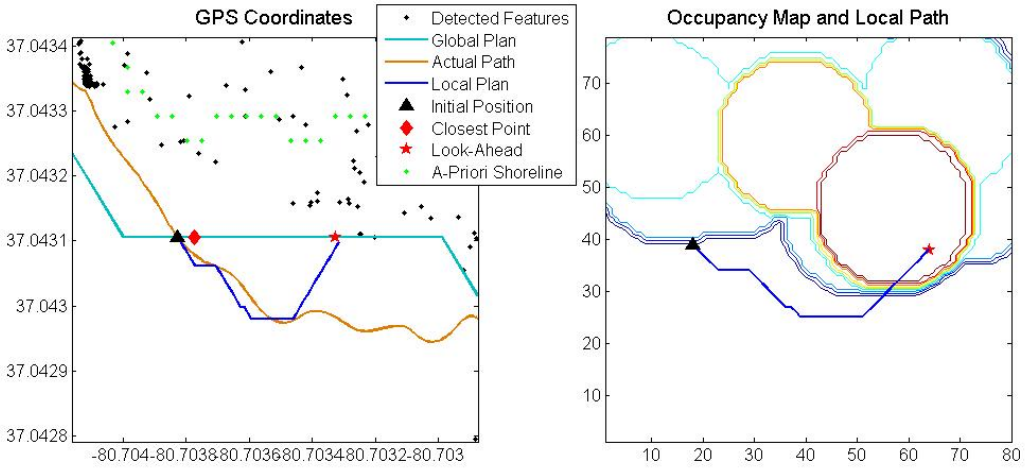

Figure 3.7: The temporary target is near a detected obstacle

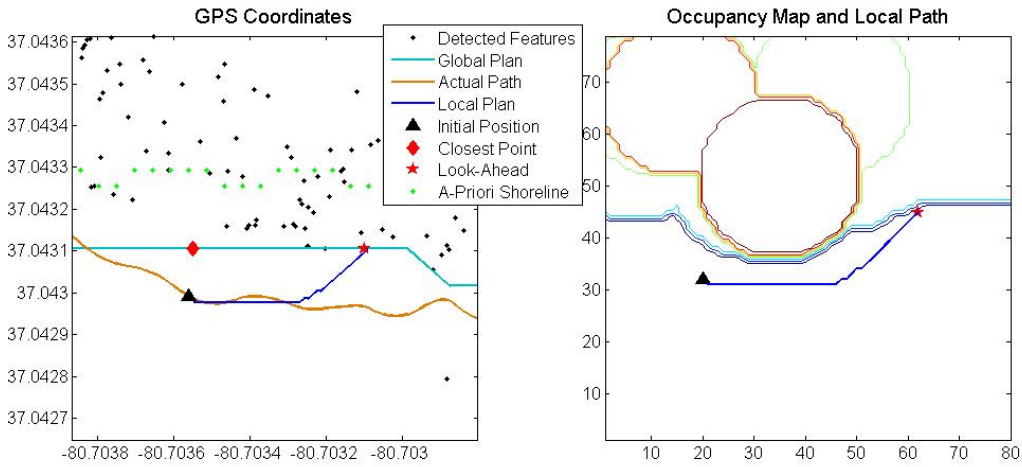

Figure 3.8: After moving forward, the new temporary target is in a clear area

The ASV would pause during this process. Once a new global plan is generated, the ASV could resume navigation.

3.4 Occupancy Map

An occupancy map represents space as a collection of probabilities which reflect the user's expectation that the corresponding geographical area is occupied. A low value ($P_i \rightarrow 0$) suggests that the area is empty or safe, while a high value ($P_i \rightarrow 1$) suggests the opposite. This is an effective method of representing an environment, as evidenced by its use in autonomous vehicle applications [19, 20, 21, 22].

3.4.1 Bayesian Principles

The probabilities assigned to the occupancy map are generated by an application of Bayes' rule. Generically, let m_k represent a single measurement and $m_{1:k}$ represent measurements 1 through k. Let $P(O_i|m_k)$ be the probability that the cell i is occupied conditioned on the current measurement. Bayes' rule for cell i is

$$P(O_i|m_k) = \frac{P(m_k|O_i)P(O_i|m_{1:k-1})}{P(m_{1:k})}. \tag{3.3}$$

where $P(m_k|O_i)$ is the probability that the sensor measurement m_k indicates that cell i is occupied given that cell i actually is occupied. This probability is determined from the sensor model. $P(O_i|m_{1:k-1})$ is the prior information about the state of cell i. It is determined either

from an *a priori* map or previous measurements. $P(m_{1:k})$ is a scaling factor which can be computed as

$$P(m_{1:k}) = \sum_{j=1}^{2} P(m_k|O_j)P(O_j|m_{1:k}) \tag{3.4}$$

where if $j = 1$, O_j represents occupied and if $j = 2$, O_j represents empty [21, 23, 24]. The details of this equation as it relates to implementation for this research are discussed in Section 3.4.2.

An effect of this update scheme is that the probability values in cells with detected obstacles increase while those in cells without detections decrease. Erroneous detections are mitigated because previous and subsequent measurements correctly identify those cells as empty. This effect is illustrated by simulation results shown in Figures 3.9, 3.10, and 3.11. An *a priori* map is established as a straight line. A series of measurements supports that feature and adds a protrusion, simulating the detection of a dock. A few measurements are seperated from this feature, but since they are only measured once, they are relatively insignificant. This effect is further demonstrated by post-processed experimental data shown in Figure 4.10.

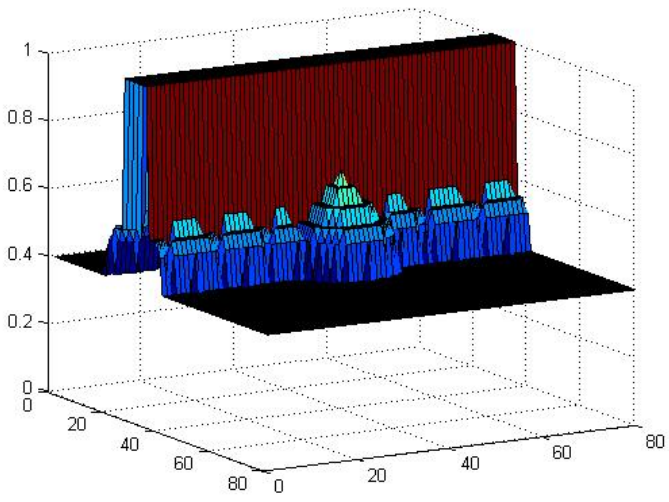

Figure 3.9: Simulated initial update showing measurements around *a priori* feature

3.4.2 Proposed Implementation

The Bayesian update scheme is adopted in the occupancy mapping algorithm.

$$P(i,j|m_k) = \frac{p * P(i,j|m_{1:k-1})}{p * P(i,j|m_{1:k-1}) + (1-p) * (1 - P(i,j|m_{1:k-1}))} \tag{3.5}$$

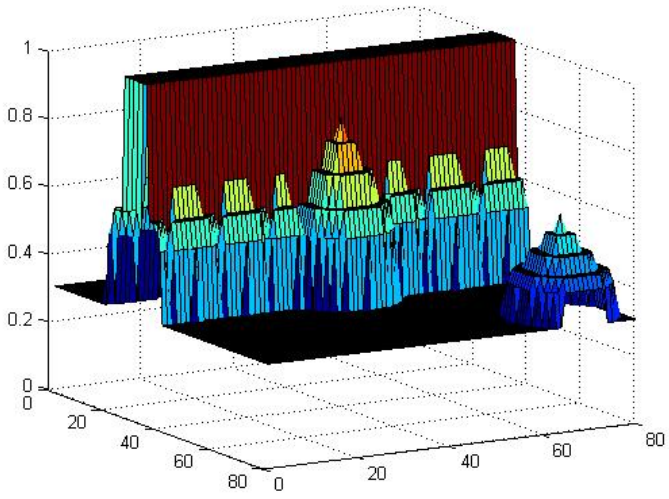

Figure 3.10: Simulated first false detection

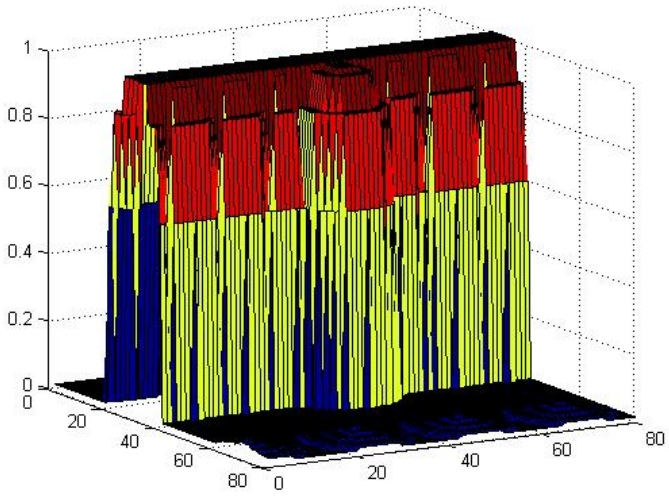

Figure 3.11: The false detections are insignificant by the end of the simulation

is the adoption of Equation 3.3 for this work. p is tabulated by range based on a Gaussian error model for the sensor and prior(i, j) is taken directly from the existing occupancy map. Each time the range estimation routine executes, a list of GPS coordinates of obstacles forms the new measurement. Each of the cells that contain the estimated location of the obstacles are the center of individual distributions. The value of p for each of the neighboring cells is extracted from the table according to the distance of that neighboring cell from the center of each distribution. When a cell is outside the effective range of the detected obstacles, it is considered empty and a constant value is assigned for p indicating probability that the sensor would measure that cell as empty given that it were actually occupied.

Due to the large size of the desired operating area, the space was divided in order to increase the speed of the mapping algorithm. The operating area as a whole is represented by a single map R. R is composed of many smaller maps, r_i, each square with 100 m edges. These sub-maps contain the probability values for the points in the environment that they represent. At the beginning of the mission, the *a priori* data is applied to these maps. Then, to perform an update, the position of the ASV is compared to R in order to select the necessary sub-maps. Sub-maps are selected so that the distance between the position of the ASV and the edge of the map is greater than the range of the camera, 50 m. When the ASV is within 50 m of the edge of a sub-map, the adjacent sub-map is also selected. Up to four sub-maps may be selected, and these maps are connected at their adjacent edges to create a single, larger map, a process called splicing.

This formulation is a compromise which supports a number of goals. The space required to store the maps is fixed according to the size of the operating area. The only changes that take place during a mission are to the values of the map elements. New maps are not created. As a result, there is no limit to the number of obstacles that can be processed over the course of a mission and therefore no computational limit to the length of a mission placed by the occupancy mapping routine. Dividing the operating area into a series of small maps enables the map update routine to execute quickly. Also, the system can operate at a much higher resolution than if it was necessary to represent the entire area with a single large map.

3.4.3 Resolution Changes

Certain applications require the ability to reduce the resolution of these maps. The approach adopted here is to consider sets of four adjacent cells that form a 2 m by 2 m square. These four cells are labeled A, B, C, and D. Let P_i signify the probability that cell i is occupied. The cell formed by the combination of these four cells is called E. Figure 3.12 shows this setup. Based on [24], these cells are combined by

$$\begin{aligned} P_A \cup P_B \cup P_C \cup P_D &= P_E \\ = P_A + P_B + P_C + P_D - P_A P_B - P_A P_C - P_A P_D - P_B P_C - P_B P_D - P_C P_D & \\ + P_A P_B P_C + P_A P_B P_D + P_A P_C P_D + P_B P_C P_D - P_A P_B P_C P_D. & \end{aligned} \tag{3.6}$$

This process can be repeated until the desired resolution is reached.

In order to increase the resolution of a map, equivalent to dividing one parent cell into four smaller child cells, it is assumed that the occupancy probabilities of the four child cells are equal. If $P_A = P_B = P_C = P_D = P_X$, and P_E is the given value of the parent cell,

$$P_X = 1 - \sqrt[4]{1 - P_E}. \tag{3.7}$$

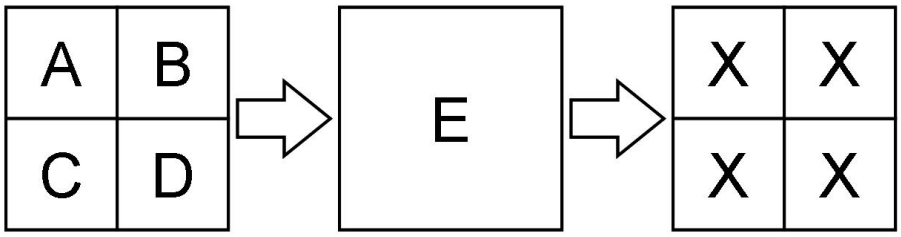

Figure 3.12: An illustration of the resolution changing process

Equation 3.7 is the result of solving equation 3.6 for P_E assuming $P_A = P_B = P_C = P_D = P$:

$$P^4 - 4P^3 + 6P^2 - 4P + P_E = 0. \tag{3.8}$$

In order to solve this equation, let $a = 1$, $b = -4$, $c = 6$, $d = -4$, and $e = P_E$. Define

$$f = c - \frac{3b^2}{8} = 6 - \frac{3(-4)^2}{8} = 0$$

$$g = d + \frac{b^3}{8} - \frac{bc}{2} = -4 + \frac{(-4)^3}{8} - \frac{(-4)6}{2} = 0$$

and

$$h = e - \frac{3b^4}{256} + \frac{b^2c}{16} - \frac{bd}{4} = P_E - \frac{3(-4)^4}{256} + \frac{(-4)^2 6}{4} - \frac{(-4)(-4)}{4} = P_E - 1.$$

Notice that if $P_E \leq 1$ then $h \leq 0$. These definitions allow equation 3.8 to be converted to a cubic equation according to

$$Y^3 + \frac{f}{2}Y^2 + \frac{f^2 - 4h}{16}Y - \frac{g^2}{64} = 0$$

so

$$Y^3 + \frac{0}{2}Y^2 + \frac{0 - 4h}{16}Y - \frac{0}{64} = Y^3 - \frac{h}{4}Y = Y(Y^2 - \frac{h}{4}) = 0.$$

Eliminating the root $Y = 0$ yields

$$Y^2 - \frac{h}{4} = 0,$$

which can be solved using the quadratic equation. If $h = (-1)m$,

$$Y_{2,3} = \pm\frac{\sqrt{(-1)m}}{2} = \frac{\sqrt{m}}{2}j, -\frac{\sqrt{m}}{2}j.$$

Define

$$p = \sqrt{Y_2} = \sqrt{\frac{\sqrt{m}}{2}j},$$
$$q = \sqrt{Y_3} = \sqrt{-1\frac{\sqrt{m}}{2}j},$$
$$r = -\frac{g}{8pq} = 0,$$

and

$$s = \frac{b}{4a} = \frac{-4}{4} = -1.$$

The roots of equation 3.8 are now

$$\begin{aligned} P_1 &= p + q + r - s \\ P_2 &= p - q - r - s \\ P_3 &= -p + q - r - s \\ P_4 &= -p - q + r - s. \end{aligned} \tag{3.9}$$

Individually,

$$P_1 = \sqrt{\frac{\sqrt{m}}{2}j} + \sqrt{(-1)\frac{\sqrt{m}}{2}j} + 1 > 1, \Re,$$
$$P_2 = \sqrt{\frac{\sqrt{m}}{2}j} - \sqrt{(-1)\frac{\sqrt{m}}{2}j} + 1 \to \Im,$$
$$P_3 = -\sqrt{\frac{\sqrt{m}}{2}j} + \sqrt{(-1)\frac{\sqrt{m}}{2}j} + 1 \to \Im,$$
$$P_4 = -\sqrt{\frac{\sqrt{m}}{2}j} - \sqrt{(-1)\frac{\sqrt{m}}{2}j} + 1 < 1, > 0, \Re$$

so P_4 is the root of interest.

$$\begin{aligned} P_4 &= -\sqrt{\frac{\sqrt{m}}{2}}\sqrt{j} - \sqrt{(-1)}\sqrt{\frac{\sqrt{m}}{2}}\sqrt{j} + 1 \\ &= -\sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}j) - \sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}j) + 1 \\ &= -\sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}) - \sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}j) - \sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}) + \sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}j) + 1 \\ &= -\sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}) - \sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}) + 1 \\ &= -2\sqrt{\frac{\sqrt{m}}{2}}(\frac{1}{\sqrt{2}}) + 1 = -2\sqrt{\frac{\sqrt{m}}{4}} + 1 = -\sqrt{\sqrt{m}} + 1. \end{aligned} \tag{3.10}$$

Since $m = 1 - P_E$,

$$P_4 = P = 1 - \sqrt[4]{1 - P_E}.$$

Chapter 4

Experimental Results

To evaluate the path planning and obstacle avoidance algorithm, the ASV was deployed in Peak Creek with a destination 3 miles away. The map in Figure 1.2 shows an overview of the navigation problem. The docks along the creek provided deviations from the *a priori* data so that the real-time obstacle avoidance system would be verified. Figure 4.1 provides an example of the type of shoreline commonly encountered during these experiments.

Figure 4.1: Type of shoreline commonly encountered by ASV on Peak Creek

For these experiments, the occupancy mapping procedure was modified from that described in section 3.4. Rather than store a series of maps, each obstacle location is stored on a list. Each time a local occupancy map is generated, the entire obstacle list is referenced. It is as if each new measurement is actually all the previous measurements combined with the most recent set. This means that, according to Equation 3.3,

$$m_k \equiv m_{1:k-1} + m_k.$$

Also, $P(O_i|m_{1:k-1})$ is set to 0.5 for all cells.

Once an occupancy map is generated, it is modified in order to account for the minimum range requirement of the omnidirectional camera (10 m). Each time there is a cell with a higher probability value than its neighbors, that high probability value is copied into all of

```
for all cells i in the occupancy map
  for all cells j within radius of current cell i
    if new cell j has probability less than current cell i
      reset the probability of cell j to equal the current cell i
    end
  end
end
```

Figure 4.2: Pseudocode for probability expansion

the neighboring cells up to 10 m away. This process is summarized by to the pseudocode shown in Figure 4.2. Although the modifications occur on the occupancy map, the effect is to generate a safety radius around the ASV that prevents it from going too close to an obstacle for accurate tracking.

4.1 *A Priori* Results

The results of these experiments are summarized in Figures 4.3, 4.4, and 4.5, which represent different sections of one continuous run. As illustrated, the ASV was able to reach the intended destination. This trial was 3.02 miles long and took the ASV approximately 100 minutes to complete.

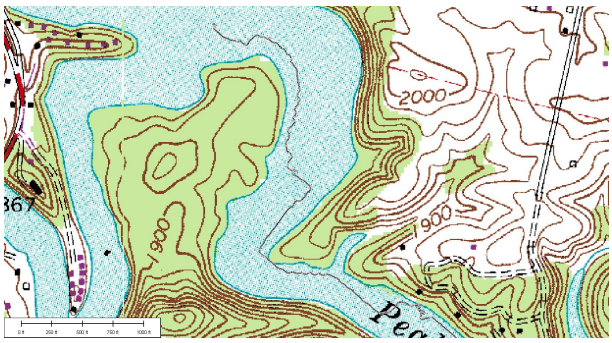

Figure 4.3: First section of successful path planning experiment

This experiment did not execute perfectly. A software malfunction occurred that required the system to be reset. Once it was running again, the path planner continued from its current location. Also, there were two cases where the ASV was not able to detect a dock due to the inability of the omnidirectional camera to detect objects directly ahead of the vehicle. Current work to develop a fixed stereo camera system will eliminate this limitation.

Neither of these occurrences detracts from the effectiveness of the algorithm. All course commands were issued by the path planner and all detected obstacles were successfully avoided. Figure 4.6 shows two cases where the ASV avoided docks that were in the way of

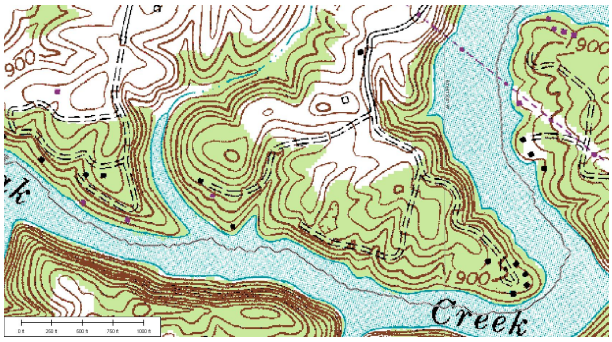

Figure 4.4: Second section of successful path planning experiment

Figure 4.5: Third section of successful path planning experiment

the projected global path.

The first avoidance is a good example of the effectiveness of the Bayesian update scheme. As the ASV approaches the dock, shown in Figure 4.7, it is able to measure the dock's position. Figures 4.8, 4.9, and 4.10 are selected occupancy maps from a sequence generated by post-processing the data from this experimens. Figure 4.8 shows the state of the environment as the ASV approaches the dock. In Figure 4.9, the dock has been detected. It is also significant that a feature shown in Figure 4.8 has been eliminated in this figure since it was a false detection. Finally, Figure 4.10 shows how repeated measurements of the dock increase its significance in the occupancy map.

These results verify the effectiveness of the proposed algorithm. The goal of implementing an autonomous path planning system for the ASV is achieved.

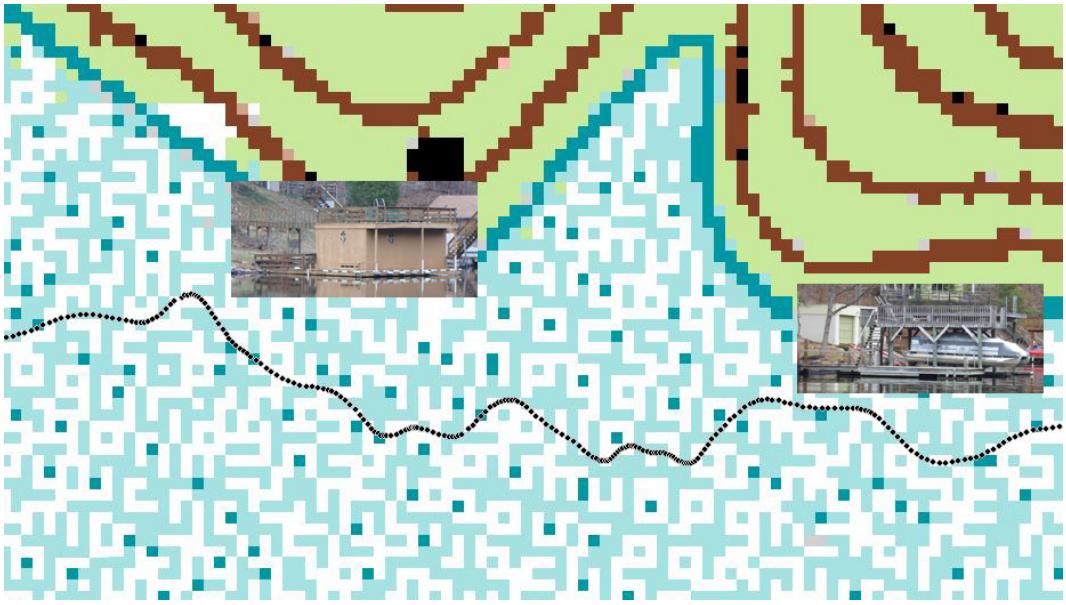

Figure 4.6: The ASV's path avoids obstacles

Figure 4.7: View of a dock from the ASV

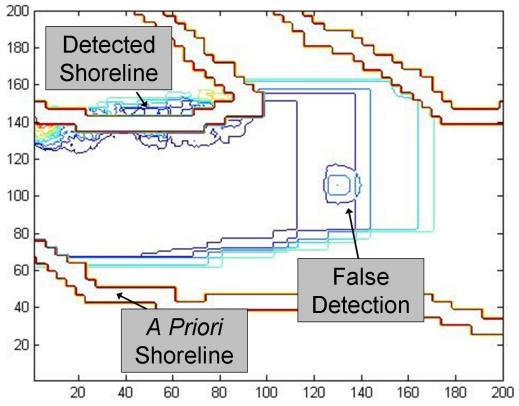

Figure 4.8: The ASV approaches the dock

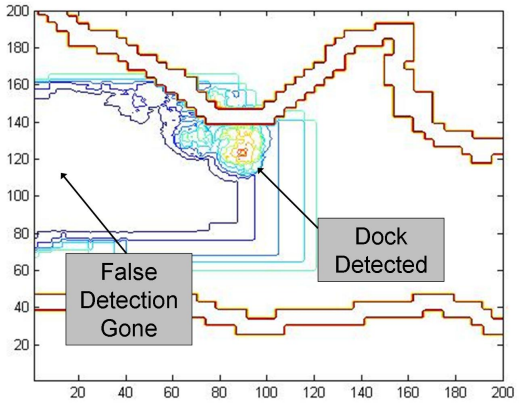

Figure 4.9: The ASV detects the dock and processes out the previous false detection

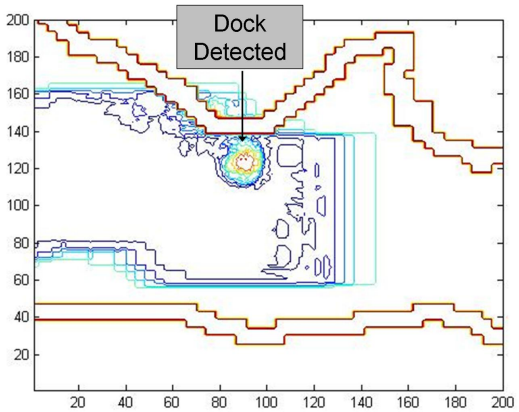

Figure 4.10: The dock is strongly represented

Chapter 5

Conclusions and Future Work

The primary contribution of this research is a real-time autonomous path planning and obstacle avoidance algorithm for an ASV in a riverine environment. The goal of a successful navigation experiment is met. The algorithm uses an *a priori* data set to generate a global path to an objective. Then, using elements of this path as local destinations, the algorithm iteratively plans local paths using occupancy maps generated from the *a priori* data and sensor measurements in order to avoid obstacles in real-time.

This algorithm is effective because it incorporates the desirable aspects of local and global planning procedures. The global path plan allows the ASV to efficiently avoid trapping situations, while the local planning provides real-time obstacle avoidance functionality. Experimental results verify the effectiveness of this algorithm.

5.1 Future Work

Future research could be conducted to improve the performance of this system. Research is ongoing to develop a fixed stereo camera system that will eliminate the blind spot directly in front of the ASV, thereby improving the performance of the obstacle avoidance system. Also, detailed comparisons of different search algorithms could be conducted in order to implement the most efficient *a priori* path planning algorithm. Implementing the proposed occupancy mapping algorithm would verify the results from the post-processed data.

The sensor model used for the Bayesian update scheme is basic. Future research would produce an accurate model that accounts for many of the sensor properties that were neglected here.

Bibliography

[1] J. Manley, A. Marsh, W. Cornforth, and C. Wiseman, "Evolution of the autonomous surface craft *AutoCat*," in *Proc. of OCEANS 2000*, vol. 1, 2000, pp. 403-408.

[2] J. Manley, "Evolution of the autonomous surface craft "ACES"," in *Proc. of OCEANS 1997*, vol. 2, 1997, pp. 827-832.

[3] A. Pascoal *et al.*, "Robotic ocean vehicles for marine science applications: the European ASIMOV project" in *Proc. of OCEANS 2000*, 2000, pp. 409-413.

[4] C. Reed, B. Bishop, and J. Waters, "Hardware selection and modeling for a small autonomous surface vessel," in *Proc. of 38th SSST*, 2006, pp. 196-200.

[5] L. Cooney *et al.*, "Design of an acoustic-homing autonomous surface vessel," in *Proc. of OCEANS 2006*, 2006.

[6] M. Caccia *et al.*, "Design and exploitation of an autonomous surface vessel for the study of sea-air interactions," in *Proc. of 2005 IEEE ICRA*, 2005, pp. 3582-3587.

[7] A. Leonessa, J. Mandello, Y. Morel, and M. Vidal, "Design of a small, multi-purpose, autonomous surface vessel," in *Proc. of OCEANS 2003*, 2003, pp. 544-550.

[8] A. Subramanian, X. Gong, J. Riggins, D. Stilwell, and C. Wyatt, "Shoreline mapping using an omni-directional camera for autonomous surface vehicle applications," in *Proc. of OCEANS 2006*, 2006.

[9] B. Xu, D. Stilwell, A. Gadre, and A. Kurdila, "Analysis of local observability for feature localization in a maritime environment using an omnidirectional camera," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3666-3671.

[10] X. Gong, A. Subramanian, C. Wyatt, and D. Stilwell, "Performance analysis and validation of a paracatadioptric omnistereo system" in *Proc. of IEEE 11th International Conference on Computer Vision*, 2007.

[11] C. Papadimitriou and M. Yannakakis, "Shortest paths without a map," in *Theoretical Computer Science*, vol. 84, ed. Elsevier, 1991, pp. 127-150.

[12] S. Thrun *et al.*, "Stanley: the robot that won the DARPA grand challenge," Journal of Field Robotics, 2006, pp. 661-692.

[13] K. Kluge and M. Morgenthaler, "Multi-horizon reactive and deliberative path planning for autonomous cross-country navigation," in *Proc. of SPIE*, vol. 5422, 2004, pp. 461-472.

[14] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proc. of 1986 IEEE ICRA*, 1986, pp. 1664-1669.

[15] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, Inc, 2003, pp. 97-104.

[16] S. LaValle, *Planning Algorithms*, Cambridge University Press, 2006, pp. 37,38.

[17] J. Latombe, *Robot Motion Planning*, Klewer Academic Publishers, 1991, pp. 604-608.

[18] B. Stout, "The Basics of A* for Path Planning," in *Game Programming Gems*, Ed. DeLoura, Charles River Media, Inc, 2000, pp. 254-263.

[19] M. Jun and R. D'Andrea, "Probability map building of uncertain dynamic environments with indistinguishable obstacles," in *Proc. of the American Control Conference*, 2003, pp. 3417-3422.

[20] M. Jun, A. Chaudhry, and R. D'Andrea, "The navigation of autonomous vehicles in uncertain dynamic environments: a case study," in *Proc. of the 41st IEEE Conference on Decision and Control*, vol. 4, 2002, pp. 3770-3775.

[21] A. Tirumalai, B. Schunck, and R. Jain, "Evidential reasoning for building environment maps," IEEE Transactions on Systems, Man, and Cybernetics, vol. 25, 1995, pp. 10-20.

[22] B. Merhy, P. Payeur, and E. Petriu, "Application of segmented 2D probabilistic occupancy maps for mobile robot sensing and navigation," in *Proc. of IMTC 2006*, 2006, pp. 2342-2347.

[23] D. Pagac, E. Nebot, and H. Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles," IEEE Transactions on Robotics and Automation, vol. 14, 1998, pp. 623-629.

[24] A. Leon-Garcia, *Probability and random processes for electrical engineering: second edition*, Addison-Wesley Publishing Co, Inc, New York, 1994, pp. 32-34, 53-55.