

Memo

To: Dr. Carlotta Berry

From: Ander A Solorzano _____ and Ruffin White _____

Class: ECE425 – Mobile Robotics

Date: 1/23/2013

Title: Lab06 – Path Planning (Occupancy Grid and Topological)

PURPOSE

The purpose of this lab is to use topological and metric navigation to move the mobile robot (CEENBoT) from a start point to a desired goal location. Topological path planning is based upon landmarks in the world where distinctive places such as landmarks or *gateways* help the robot localize itself in the world and trigger actions that might change its direction. Metric path planning involves the use of a *wavefront* algorithm on a priori map to create a path from the robot's start position to the goal location. The final task of the robot is to demonstrate world navigation from a start point to and end point using *wavefront* expansion (i.e. metric navigation) on a topological map of the world's salient features (topological navigation). The list of robot commands and/or generated *wavefront* should be shown on the LCD to make it clear about the robot's current state and next state.

PROCEDURES AND STRATEGY

Before we began coding the robot to handle topological and metric navigation, we acquired some IR readings about different landmarks and gateways. With these readings, we were able to set some threshold values that would check each of the gateways and allow the robot to trigger a necessary change in direction.

Landmark	Front IR	Left IR	Right IR	Back IR
Corner in Front	15.3	23.4	22.7	50
Corner on left	7.8	10.3	50	50
Corner on right	7.8	50	10.5	50
Hallway on both sides	50	10.2	9.8	50
Hallway on left	50	50	10.1	50
Hallway on right	50	10.4	50	50
T-junction	8.1	50	50	50
Dead end	8.2	10.8	10.2	50

Table 1: IR sensor readings for topological landmarks present in the robot world. A reading of 50 was given to values that were too large and too noisy to acquire a clear reading. The readings are in centimeters and NOT in inches.

Now that we have acquired landmark readings of the robot world, we proceeded to create the topological and metric path planning behaviors of the robot. Some values might vary due to systematic errors from the sensors and nonsystematic errors from the environment.

Topological Behavior:

We started by implementing a user interface where the user will be able to input *turn left*, *go straight*, and/or *turn right*. The reason for doing this is so that we can tell the robot to “turn on the second left” or “turn on the third right” for example. Rather than telling the robot to “turn on the first available right” or “turn on the first available left,” we decided that this method is more user-intuitive since it would relate to telling instructions to a person. For example, these instructions would include: turn left at the corner, continue walking down the hallway until the third right, turn right and walk down until you reach your destination. After we coded the user inputs, we incorporated to implement our landmarks into the robot map. This way the robot would always be aware of what obstacles to expect on the current cell and on the next cell. A side feature that we included to our topological behavior consisted of a starting orientation setting. We would be able to indicate whether the robot starts by facing North, East, South, or West and the robot would correctly orientate itself to start moving in the map.

Metric Behavior:

We incorporated the wavefront diagram of the world using 4 bits for each of the 16 cells. The MSB correlates the front IR sensor, the right IR sensor correlates to the 2nd MSB, the back IR sensor correlates to the 3rd MSB, and the left IR sensor correlates to the LSB. In other words, a sensor reading of 0b0101, or 5 in decimal, would indicate that the robot is on a hallway with walls on both sides. The main feature about our wavefront, is that we are able to input a initial starting point and the final goal location. The robot would then traverse along a path using the map of the world to traverse fix distances.

Metric Behavior with a Topological Expansion:

Once the two basic behaviors were coded and implemented, we then proceeded to create a more complex path planning behavior that incorporates metric and topological path planning features. Before launching the robot, we have the enter the starting and end positions for the the *wavefront map*. However, we must also put in the desired user move commands such as turn left, turn right, go forward, and the starting orientation of the robot. The most difficult aspect of this behavior is that we must be able to always store and distort the wavefront map when the robot turns. This is because when the robot turns, the wavefront map also changes according to the robot's from view. This is similar to a GPS since the most popular types of GPS have the car marker always face forward while the world rotates around it. To implement this difficult yet unique feature into our code, we used a barrel shifter.

QUESTIONS

1. What was the strategy for implementing the algorithm?

In order for the robot to navigate from its starting location to a set goal point, the robot must use the given mass of the world along with the directions to generate a path. At first, the robot orientates itself within the virtual grid given the starting orientation and iteratively moves through, cell to cell, storing the specific gateways in the robot's own perspective as it would observe when traversing through the physical world and reality. It then stores up this wavefront to later use as a guide to verify image navigation that is indefinitely on the correct path. In this way, we integrate the best of both features in metric and topological navigation.

This allows us to provide more specificity in directions than pure topological abstracts can provide while simultaneously correcting for virtual and actual world mismatches. Take the example with the robot is told to go down the corridor and turn left. In turn the robot is told to either go one unit or three units forward and turn left into separate scenarios. However the actual world has 2 turn; one is two units forward and the second is three units forward. With our algorithm, if the robot is told to turn after the first unit, the robot will sense the environment and realize that such a turn is not possible. In fact that the user most likely meant to take the first turn on the left, as a prior possible turn was not detected, and will proceed to take the next ‘detoured’ turn.

However if the robot is told to traverse three units and then turn, the robot will **not be fooled** by the first possible turn after the second unit and will proceed to investigate if it turns possible after it's directed third unit. This gives the robot a better

intuition with respect to the cost and benefits of following a map versus the road similar to an actual human behavior, thus allowing for a more natural method of specifying how to proceed to a goal.

2. Were there any points during the navigation when the robot got stuck? If so, how did you extract the robot from that situation?

In order to account for odometry error, we used our previous wall following code to adjust the position of the robot as it traverses between walls. However this requires some small modifications. Left unmodified, our wall following code would persuade the robot to follow walls around corners, essentially skipping important decision gateways but helping the robot guide it to its ultimate path. We then tighten the thresholds as well as disabled the front IR's contribution to wall following and fine-tune the PID control parameters for the newer sampling rates and iteration times.

3. How long did it take for the robot to move from the start position to the goal?

We currently use our developed *arc* function which allows the robot to use its differential drive behavior to traverse any arc length of any radius, with all dimensions specified in centimeters. During testing and development, we simply left our arc function traverse a speed of 10 cm/s but is certainly capable of reaching higher speeds while safely maneuvering the course.

4. What type of algorithm did you use to select the most optimal or efficient path?

We coded the robot to emulate the behavior of an opportunistic human, in the respect that is orderly, methodical, and following the user input specifications, but will correct itself or take shortcuts should the opportunity present itself. In this way the robot will try its best to stay the course but will diverge from the original path in order to achieve its destination.

5. How did you represent the robot's start and goal position at run time?

During startup time, the user specifies the starting cell (4-bit input) and the starting orientation (2-bit input). This is done with two switches to reading create the binary placeholders or significant bits that make up the numerical index. For example, specifying the origin, or an index of zero, can be done by pressing the zero holder button four consecutive times, whereas specifying the cell across the map in the diagonal corner, index 15, can be done by pressing one four consecutive times, the first press being the most significant bit. Our robot is also capable of starting in any orientation, not simply just the North orientation of the map. This is due to our advance Gateway calculator that preemptively recalculates (specifically the cell orientations that encode the wall structures) the expected cell orientation while traversing its path.




6. Do you have any recommendations for improving the robot's navigation or wavefront algorithm?

If additional time was allotted, future improvements would include implementing our arc functions to gracefully and in a radial manner navigate through corners. This would allow for smoother transitions in between but always maintaining a center position between the walls. Another suggestion would be to **not** implement occupancy cell numerics, as this method of wavefront is severely crippled by “gravity wells”. This includes traversing through a corridor that is in fact close to the goal but any further progress to the goal would involve backtracking or proceeding away from the goal. This severely cripples a traditional wavefront algorithm and is what we sought to avoid within our project by using elements of **predesignated supervised guidance** and **adaptive obstacle** in past planning behavior.

CONCLUSION

In conclusion this lab allowed us to develop our own method using topological and metric path planning behaviors for path navigation and locomotion to migrate the robot from an initial starting location to an eventual goal point. In turn, we were able to use the best from both worlds to **develop a robust and intelligent mission control and planning interface** for our autonomous robot.

Memo

To: Dr. Carlotta Berry 
From: Ander A Solorzano  and Ruffin White 
Class: ECE425 – Mobile Robotics
Date: 1/23/2013
Title: Lab06 – Path Planning (Occupancy Grid and Topological)

PURPOSE

The purpose of this lab is to use topological and metric navigation to move the mobile robot (CEENBoT) from a start point to a desired goal location. Topological path planning is based upon landmarks in the world where distinctive places such as landmarks or *gateways* help the robot localize itself in the world and trigger actions that might change its direction. Metric path planning involves the use of a *wavefront* algorithm on a priori map to create a path from the robot's start position to the goal location. The final task of the robot is to demonstrate world navigation from a start point to and end point using *wavefront* expansion (i.e. metric navigation) on a topological map of the world's salient features (topological navigation). The list of robot commands and/or generated *wavefront* should be shown on the LCD to make it clear about the robot's current state and next state.

PROCEDURES AND STRATEGY

Before we began coding the robot to handle topological and metric navigation, we acquired some IR readings about different landmarks and gateways. With these readings, we were able to set some threshold values that would check each of the gateways and allow the robot to trigger a necessary change in direction.

Landmark	Front IR	Left IR	Right IR	Back IR
Corner in Front	15.3	23.4	22.7	50
Corner on left	7.8	10.3	50	50
Corner on right	7.8	50	10.5	50
Hallway on both sides	50	10.2	9.8	50
Hallway on left	50	50	10.1	50
Hallway on right	50	10.4	50	50
T-junction	8.1	50	50	50
Dead end	8.2	10.8	10.2	50

Table 1: IR sensor readings for topological landmarks present in the robot world. A reading of 50 was given to values that were too large and too noisy to acquire a clear reading. The readings are in centimeters and NOT in inches.