|  |
| --- |
| **Rose-Hulman Institute of Technology** |
| **Arkin Final Report – Localization and Search** |
| **ECE425-Mobile Robotics** |
|  |
| Ander Solorzano & Ruffin White |
|  |

Rose Hulman Robotics Team CM 5000

5500 Wabash Avenue

Terre Haute, IN 47803

# ABSTRACT

The purpose of this final project is to demonstrate the integration of some of concepts learned in this quarter by creating localization and search algorithms for the CEENBoT platform. The localization task involves using sensor feedback with a navigation routine to determine the location of a lost robot in the world, the location of a fire (heat) source, and then rescue it by moving it to its home location. The CEENBoT robot uses a total of four IR range sensors, with a single IR sensor attached to each side of the robot, for detecting walls and/or close proximity obstacles. The robot uses these sensors to map the world, localize itself, and verify that it has entered or left a discrete cell in the world. Located in the front of the robot, the CEENBoT also uses a heat sensor to detect heat sources and move towards or away from them. Using a very efficient mapping and localization algorithm, our robot managed to constantly locate itself in the correct cell and orientation within 3-5 moves or turns. Rarely did our robot get lost while exploring due to non-systematic errors such as odometry or IR sensors errors. Using our methods described in this report, our CEENBoT robotic platform, named Arkin, managed to come up in second place during the competition.

Table of Contents

[ABSTRACT 1](#_Toc348885237)

[OBJECTIVE 3](#_Toc348885238)

[THEORY 3](#_Toc348885239)

[METHODS 4](#_Toc348885240)

[RESULTS 10](#_Toc348885243)

[CONCLUSION AND RECOMMENDATIONS 10](#_Toc348885244)

# OBJECTIVE

In this final project, the CEENBoT robotic platform will use localization, mapping, exploration, and control technique learned and discussed in the class lectures. The robot world consists of a 6’ x 6’ boxed world consisting of 18”x18” cells. The world is therefore divided into a 4 x 4 array that we will refer to as gateways.

The robot will have a prior knowledge of the world and its starting orientation but it will not know where it is starting or where a potential heat source (i.e. the fire) is located. Once a user presses GO, the robot should start moving around the world and localize itself within 3 to 5 moves or turns. The length of time that it will take for the robot to localize itself depends on the uniqueness of gateways that the robot traverses. Once localized, the robot will then proceed to explore the world until it finds the heat source. Once the robot detects a heat source, it will finally proceed to its user-designated home position where it will report the location of the heat source.

|  |  |
| --- | --- |
|  |  |
| Figure 1: Example of a robot world and gateways – The figure on the left represent an example of robot world with topological encounters such as dead end (D) and T-junction (T). Gateways are unique locations in the world where the robot can change its direction based on the location. | |

# THEORY

In order to achieve our objective, we require the use of some mapping techniques, localization concepts, obstacle tracking behavior, and path planning algorithms.

### Mapping

Mapping refers to the process of using sensor data and the robot’s movement to generate a model of the world [1]. Successful mapping of a world allows the robot to achieve successful navigation. The map must also be compatible with the robot sensors in order for the robot to behave properly. Since each of the 4 IR sensors is placed along each side of the robot, it makes sense for the world and its cell resolution to have a rectangular like shape. If the world was circular, an entirely different behavior might have been observed.

### Localization

Due to the topological locations of our world and its unique landmark features, feature-based localization was best suited to help the robot find itself in the world [1]. Using Markov probabilistic decision process, this type of localization uses connections among landmarks and the frame of reference that is absolute in this case [1]. In other words, the cells in our world consisted of 18” in height and width thus the use of discrete or absolute steps for localization can help the robot check and move from cell to cell. Unique landmarks and gateways such as T-junctions, dead ends, middle of a hallway, can help build a chain of visited landmarks that will help the robot localize itself.

### Obstacle Tracking

Using the wall following behavior developed in a previous lab and only using a proportional (P) controller, the robot was able to track walls and center itself on the middle of a hallway when traversing the world. Using potential field theory, obstacles and/or walls in front of the robot serve as repulsive fields which the front IR managed to detect and allowed the robot to perform the necessary action to avoid collision. The goal location, on the other hand, serves as an attractive field which allowed the robot to move closer and closer to its cell location.

### Path Planning

Metric and wavefront-type navigation were used in this lab to help the robot navigate from one point to a user-designated goal location. Cells farther away from the robot had relatively high discrete numbers such as a value of 20. Obstacle cells had a value of 99 in order to prevent the robot from trying to go into that cell. Finally, the goal location had a value of 0 and once the robot detected that it had reached its goal location, it would then stop all motion. Using a four-neighborhood search algorithm, the robot would scan all of the neighboring cells and proceed to face and move to the cell with the lowest value. Thus this would eventually allow the robot to travel from a random location to the user-designated goal location.

# METHODS

## Localization

The localization method implemented within our robot utilizes a method of deductive calculations and deterministic expansions of possible candidate locations rather than by discrete probabilistic means. Using a tree structure to hold historical orientations, observed gateways, and executed movements, our robot uses a brute force algorithm to deduce possible origins and current location. Upon initialization, the robot is already given a complete map depicting every cell, and every cells-specific-gateway (specifically the position of all four possible walls of a cell with reference to a northern orientation). The robot initializes the tree by acquiring its root seed, or otherwise known as a starting location. It is not entirely accurate in referring to the starting location as always the root seed, as our tree resembles a dynamic queue data structure, where the oldest root elements are popped out and replaced by newer ones. Using this methodology, the oldest element of the queue always serves as the root seed for the tree.

|  |
| --- |
| C:\Users\whitemrj\Documents\GitHub\ECE425\Final Project\Map and Tree.png |
| Figure 2: Localization Process – This figure shows the propagation moment of a kidnaped robot, the robots possible locations within the map, as well as is growing tree structure used to solve for its location from start to finish. |

In Figure 2 is a test case scenario where the robot is placed within a mirror symmetrical environment in the top right corner. The robot is placed a front facing North orientation; it is possible to use the same localization process to localize from any starting orientation which will be discussed later in this report. The robot continues the localization process in a circular fashion consisting of sensor acquisition, data processing, and motion execution. Once the tree has been initialized with a measured root seed using the sensor data, that specific gateway is then used as an ideal template to search the map for possible matches or potential seeds. In this case the robot will find potential seeds that possess identical gateways. We use the number of resulting potential seeds as a method of determining our state of confusion; a single seed would entail that our exact position has been found and that there is only one solution for our growing tree that remains. Multiple seeds would entail that multiple solutions are still possible and that either not enough branches have been gathered or that our tree does not contain enough uniquely identifiable features. Lastly, zero seeds would entail that no solution has been found and that either our map data is incorrect or the robot has succumbed to some sensor error.

Continuing with the test case scenario as shown within Figure 2, since two potential seeds still exist, further measurements are still necessary for complete localization. Using our standard maze navigation algorithm as developed within our exploratory mapping algorithm, the robot continues to explore on the conditional premise of the behavior that follows:

1. If the left wall is open, proceed to make a left turn.
2. Otherwise check to see if the front area is clear of obstacles and walls. If it is, move forward.
3. If state 1 or state 2 cannot be performed, the robot is therefore in a dead-end with walls enclosing the front, and left sensors. If this is the case, the robot would then proceed to make a right turn until its front sensor is clear.

As shown in the Figure 2, the robot proceeds to spin right twice, then move forward twice. Up to this point, the two potential seeds that have been found remain suitable solutions for our current tree. It is only until we reached the fifth branch within out tree that we encounter a uniquely identifiable feature that is the deciding factor for this case scenario. Due to this specific mirrored symmetry of the particular gateway we currently observe, as well as the sequential pattern and history of our orientation and movement, we can thereby determine that there remains only one possible solution for our elongated tree.

In this specific example, we have chosen one of the starting points in orientations that would result in the longest path to localization, thus proving in this particular instance, that the minimum number of necessary branches for immediate localization, or localization as soon as possible, is five. However this algorithm is quite suited for larger or more symmetric maps. Another interesting note is our algorithms capacity to localize under fewer than five iterations. Take the case where the robot is placed within the middle of one of the T intersections, remaining of course is the assumption our robot has been initialized in a front facing North orientation. Due to the uniqueness that this specific gateway serves, our robot will be able to initialize its current location and a single observation of the surrounding gateway. Such is the case that where there spawns only one potential seed that will serve as a solution for our tree. Lastly, with credit to using a queue-based structure, should a sensor reading corrupt our tree, preventing immediate localization using the encumbered data, our algorithm will fail gracefully and places the robot in a temporarily prolonged state of confusion.

In the case where all five branches have been grown and a single seed solution has not been found, a robot will continue to map and record its environment by replacing its oldest seed with the second oldest. In a worst-case scenario, should a corrupted sensor reading occur during the construction of our fifth branch, only five more correctly recorded gateways would be necessary to repair our trees integrity, and thereby find a singular seed solution. With the inclusion of omni-orientational localization, a similar graceful recovery from orientation confusion would also be possible should an error in rotational odometry occur when traversing turns.

The particular details of this algorithm can be described as follows: upon every reiteration of our localization subroutine, we sequentially search every cell for a bit by bit similarity with the gateway of our current root seed. Should the potential seed be found, the seed is investigated and scrutinized by reconstructing our tree on top of the cell. This separate sub-function applies the tree to the given cell within the map by reconstructing the past using what the robot has observed with the starting origin transposed onto the seed in question. After the potential seed passes the scrutinizing process of the current tree, the numeric seed index is incremented. After all potential seeds have been investigated, and that the seed index has retained a value of one, we are thereby localized. Using the global variables as local placeholders within the branch calculations, after successful localization iteration, these global variables are assigned the last location as dictated by the tree using the single seed previously found, thus already reconstructing our current location without further computation.

In order to achieve localization, we merely expand our search by either using additional rotated versions of our current map or by simply rotating the orientation values within our own tree. By rotating our entire tree and searching again for potential seeds using our root seed as an ideal template, we can abstract our state of confusion to assume a state of successful localization upon the event when we have achieved only one remaining seed among all possible orientations. When we are found to be left with one remaining seed, we simply use that orientation to correct our robot orientation with respect to the northern reference of the map. However, this could be regarded as a potential drawback since it requires additional computation and the possibility of even further localization attempts on evenly the most geographically unique gateways, should any duplicates of such exist. If initial orientation is unknown, even the T junctions as identified before will themselves will seem identical due to their rotational symmetry and will not remain distinguishable when using the previously exampled tree of a single branch.

## Navigation

Once the robot has been localized and its current cell location and orientation are then specified, our robots next task will be to autonomously navigate from its current waypoint to a predetermined goal. This goal could be specified on-the-fly by using our pre-existing function that calculates the deterministic metric map and cost values for all cells within the map given any arbitrary goal point. Once the metric cost map has been calculated, the robot proceeds to traverse the path of least resistance. As opposed to our previous implementation of metric locomotion, specifically in regards to generating an entire pre-deterministic path and relying on its and fallible execution, our robot instead follows its nose, or makes navigational base decisions on a continuous point by point basis.

There are several methods for generating cost maps; one would include assigning each cell the value of the calculated Euclidean distance to the goal cell. However, this method usually results in gravity wells: an instance where the robot is geographically close to its goal point but due to an obstacle or barrier, achieving that goal would require in moving in a direction that would momentarily result a further displacement to the goal. Such predicaments have been solved by developing a more elegant approach in path planning, such within the implementation of an A\* algorithm. Based upon your perspective of the initial conditions, there can be several ways in implementing an A\*algorithm. As part of one of the tasks, the robot is required to navigate back to its designated home after successfully finding and recording the location of the simulated fire. Due to the limited time constrain that the robot needs to achieve all tasks, we've chosen to use the pre-deterministic approach by solving for all point prior to launching the robot. Although the benefits in time savings due to our clock frequency and computation time are marginal at best, this helps reduce the number of points of failure should our robot ever attempt to plan the path starting from a non-valid coordinate or cell index.

In order to implement this path planning approach, thereby rendering a cost map that will serve as a path solution to the goal starting from any other location, we need to utilize a fundamental technique extended in any traditional A\*algorithm; that is recursion. By developing a recursive algorithm given only at a starting index and current accumulated distance, the robot uses those parameters to assign values to the global cost map and recursively call itself on neighboring cells. In practice, we simply begin by calling a recursive function upon the goal location as well as an initial accumulated distance of one. The recursive function will then assign the distance value it has received to the corresponding cell within the cost map to that of additional starting cell is received.

The algorithm will then perform a four-neighborhood search; first verifying that the neighboring cell is not out of bounds with respect to the street map and secondly that the value of that cell with regard to the cost map has not already been assigned a value. If the neighboring cell in question passes both cases, the recursive function is again called upon that specific location using an incremented value of the current accumulated distance. There's one additional step that we must include within our recursive function; that is to include the case for obstacles. In reality the very first thing our recursive function performs is to observe the gateway of its starting location using the predefined map. Because the recursive function has been called the first place, we can already assume that the cell location of the Gateway we wish to observe is a valid index and not out-of-bounds. Before assigning cost values to the cell, we check to see if the relevant gateway we are located in is bounded by walls on all sides. If the gateway we observe is completely enclosed, then that location is treated as an obstacle in thus assigned a uniquely high cost value of 99. This prevents our path planner traversing through obstacles and designating deceitful paths.

There are a few assumptions one must carefully consider when implementing such a method. First, that the uniquely high cost value you assign to cell consisting of objects; given that the objects are completely non-traversable, is such that the cost value should be greater than the value of the largest or longest conceivable accumulated distance within the map. This would prevent the robot attempting to traverse through a non-navigate able obstacle rather than proceeding into the next cell of a substantially lengthy path. Secondly, by discriminating only completely enclosed regions as obstacles, we neglect to observe the path between two navigable regions separated by a single wall as non-navigateable. In future implementations, it would be wise to have the algorithm account for the existence of walls separating boundary region. This however would also call for our cost navigation algorithm by which we navigate from cell to cell to also recognize this environmental blockade. An additional reason this was not implemented was out of the lack of necessity; given the environment within the competition, every traversable area within the map was only segmented by completely enclosed obstacles and not by single wall barriers. This allowed us to assume a path planning structure based on occupancy grid while simultaneously maintaining a descriptive feature method in order to record gateways used in our localization.

|  |
| --- |
| C:\Users\whitemrj\Documents\GitHub\ECE425\Final Project\Map and Metric.png |
| Figure 3: Cost-based Navigation - A rendition of the robots navigational movements from it starting location to its final goal as superimposed on top of a heat map representing the cost value within occupancy grid. |

.

Figure 3 shows an example of the robot traversing from a starting point in the top right corner to its end goal location at the bottom left of the map. As the robot begins already localized within this example, the robot is able to immediately proceed to its goal location using the navigational maze method described within the localization section. Along with the visualized occupancy grid, a table representing the robots attributes during each step in its path are also visualized with respect to its iterative process of achieving it sub-goal or the neighboring cell of the lowest cost.

# RESULTS

Using the methodology described above, our robot would perform all of the tasks (i.e. localize itself, find a source, and travel home) within 40 to 60 seconds depending on the distance needed to traverse. Overall, our robot placed in second and few odometry errors and non-systematic errors were observed.

# CONCLUSION AND RECOMMENDATIONS

The purpose of this final project is to demonstrate the integration of some of concepts learned in this quarter by creating localization and search algorithms for the CEENBoT platform. The robot was tasked with localizing itself on a previously known world, finding a heat source, and then traveling to the user-designated goal location. This project involves some major topics and concepts in mobile robotics such as localization, mapping, path planning, and obstacle tracking.

|  |
| --- |
| C:\Users\whitemrj\Documents\GitHub\ECE425\Final Project\Map and Tree Hard.png |
| Figure 4: Localization Process – A case scenario where the world is rotationally symmetric and whereby assuming the starting orientation to be unknown or arbitrary will render it impossible to completely localize due to the complete symmetry. |

Figure 4 depicts a troubling scenario where the navigable regions within the environment are rotationally symmetric. Given the localization methods described within the prior sections, the specific techniques of using a queue-based structure, and deterministically solving for one's location, this is a specific counterexample that renders what we have discussed quite helpless in the efforts of localization when one's initialized orientation is unknown. This map, along with any other rotationally symmetric environment (among which there are many given a 5x5 occupancy, such as the navigateable inverse of the map represented within Figure 3) is a suitable incentive for introducing additional environmental sensors. Such a sensor could include a digital compass or magnetometer which reads current orientation utilizing Earth's magnetic fields in order to localize in such symmetric or featureless environments.

# REFERENCES

[1] C. Berry. *Mobiile Robotics for Multidisciplinary Study.* Morgan & Claypool Publishers. Chapter 4.