|  |
| --- |
| **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 a 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 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 surface friction and random noise detected by the IR sensors. 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

## 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 with 18”x18” obstacles. The world is further divided into a 4 x 4 array that will contain several gateways.

The robot will have a priori 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 and gateways that the robot traverses. After the robot localizes itself, it 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.

## THEORY

## 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, executed movements our robot uses a brute force algorithm to deduce possible origins and current locations. Upon initialization, the robot is already given a complete map depicting every cell, and every sells specific gateway (specifically the orientation of every possible wall with reference to a northern orientation). The robot initializes the tree by acquiring its root seed, or otherwise known as a starting location. This is not entirely accurate in referring to the starting location as always the root seed, as our tree resembles a dynamic queue with regard to the possibility of queue data flow, with the oldest root elements popped out and newer ones placed on top of the queue. In that respect the oldest element of the queue always serves as the root seed for the tree.

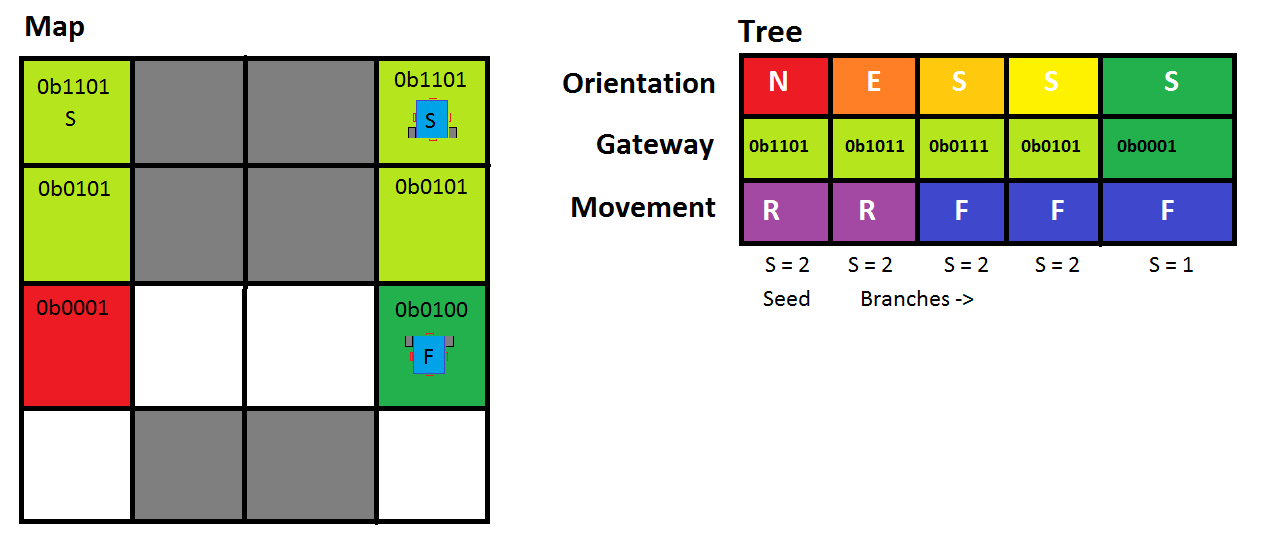
By inspecting figure @@@@@@@@@, one can observe a test case scenario where the robot is placed within a symmetrical environment in the top right corner. The robot is first placed within its environment in a front facing North orientation, it is possible to use the same localization process to localize from any starting orientation and will further be explored later in this document. The robot continues the localization process in a circular fashion of sensor acquisition, data processing, and motion execution. Once the tree has been initialized with the 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 used 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 as there is only one solution for our growing tree. Multiple seeds would entail that multiple solutions are still possible and that not enough branches have been gathered or that our tree does not contain enough uniquely identifiable features. While lastly a single seed well entail that no solution has been found and that either our map data is incorrect and we have stumbled across uncharted territory or that in fact are recorded gateway data has succumbed to some sensor error.

Continuing with the test case scenario as shown within the figure, two potential seeds still exist, and as only one branch has been constructed, 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; if the left wall is open, proceed to make a left turn, otherwise proceed to check the front wall if for if forward motion if possible, and should neither be the case, proceed to make a right spin in place. As shown in the figure, specifically the growing branches within the tree, the robot proceeds to spin right twice, then move forward twice. Up to this point, the two potential seeds that have been found and remain suitable solutions for our current tree. It is only until we reached the fifth branch within out tree do 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.

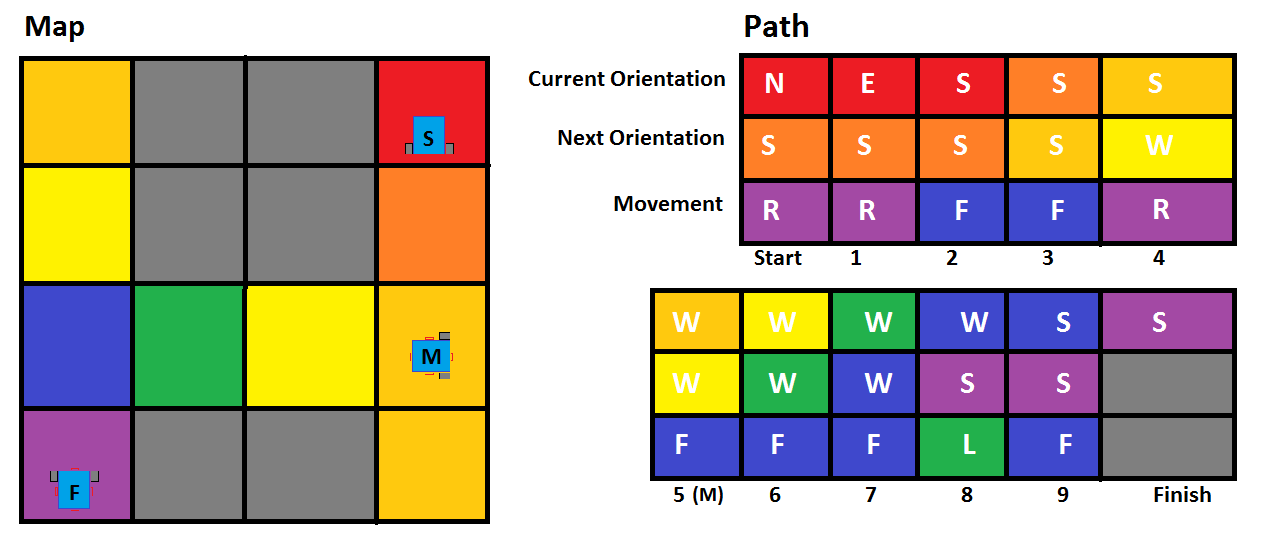
In this specific example, I have chosen one of the starting points in orientations that would result in the longest path localization, thus proving in this particular instance, that the minimum number of necessary branches for immediate localization is five. However this algorithm is quite suited for larger or more symmetric maps and is the case where we maintain this parameter configurable upon compile time. Another interesting note is our algorithms capacity to localize under fewer than five iterations. Given 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 and the 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 is only one potential seed that will serve as a solution for our tree. And lastly, with credit 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 it's 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 given worst-case scenario, should a corrupted sensor reading occur on our fifth iteration that would result in our localization, only five more correctly recorded gateways would be necessary to repair our trees integrity, and thereby 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 optometry occur when traversing turns.

The particular details of this algorithm can be described as follows: upon every reiteration our localization subroutine we sequentially and 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 function applies the tree to the given cell within the map by reconstructing the past that the robot has observed with the starting origin containing the seating question. During the path reconstruction, should a branch become invalid by either of two ways; the first a discontinuity and observed gateway and the gateway of the map result in a disagreement, or second should the recorded movements result in an out of bounds error, thereby inferring that the robot would have had to venture outside of the discrete representation of the map. Should the potential seed past the scrutinized station of the current tree, the numeric seed index is then 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 trick of using the global variables as local placeholders within the branch calculations, after a successful localization iteration, these global variables are left assigned the last location as dictated by the tree using the single found seed, thereby already reconstructing our current location without further computation.

In order to achieve omni-orientational localization, we merely expand our search by either using additional rotated versions of our current map, or simply rotating only 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 of when achieving only one remaining seed among all the possible orientations. When we are found to be left with one remaining seed, we simply use that orientational offset that the solution is derived from to accommodate and correct our orientation with respect to the northern reference of the map. This does however subtract in the regard that it does require more computations and the possibility of even further localization attempts on evenly the most geographically unique gateways. 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 achievement 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 I navigation algorithm proceeds to explore a path of least resistance. As opposed to our previous implementation of metric locomotion, specifically in regards to generating a pre-deterministic path and relying on its and fallible execution, our robot instead follows its nose. What I mean by that phrase is that in order to encompass a less (missing something?)

## RESULTS

WRITE HERE RUFFIN

## CONCLUSION AND RECOMMENDATIONS

WRITE HERE RUFFIN