CSE 180 - Robotics
Final Project

Team Alpha

Christine Breckenridge, Aleksandr Brodskiy, Carlos Martinez

May 4, 2018

Outline

Documentation

In order to develop a complete and efficient solution of detecting treasures within an arbitrary map of consistent dimensions, an approach was devised such that the computations performed by the various, specialized algorithms would be distributed. In this manner, the implementation of this approach involved the creation of three nodes, ***nav.cpp***, ***planner.cpp***, and ***treasures.cpp***.

see the **Design Decisions** *section for further detail regarding optimizations.*

The primary data structures utilized in the formulization of this distributed approach were a one dimensional array as well as a map with a ***string*** type as the key and a ***pose*** object as the value. The simplification of storing a ***pose*** object within the map was predominantly related to its dependance on the ROS structs ***point*** and ***quaternion*** for the computing of the translation and rotation of the husky robot, respectively. The instantiation of the map constructor is demonstrated below with the ***treasures*** label:

STD::MAP< CHAR[], POSE > ***treasures***

Furthermore, by subscribing to the *map* topic it became possible to retrieve data embedded in the ROS message:

NAV_MSGS/OCCUPANCYGRID

This message contains the HEADER HEADER, MAPMETADATA INFO , and INT8_T[] DATA fields. The MAPMETADATA INFO field provides information regarding the height, width, and resolution of the environment map. The height, width, and resolution are defined as follows:

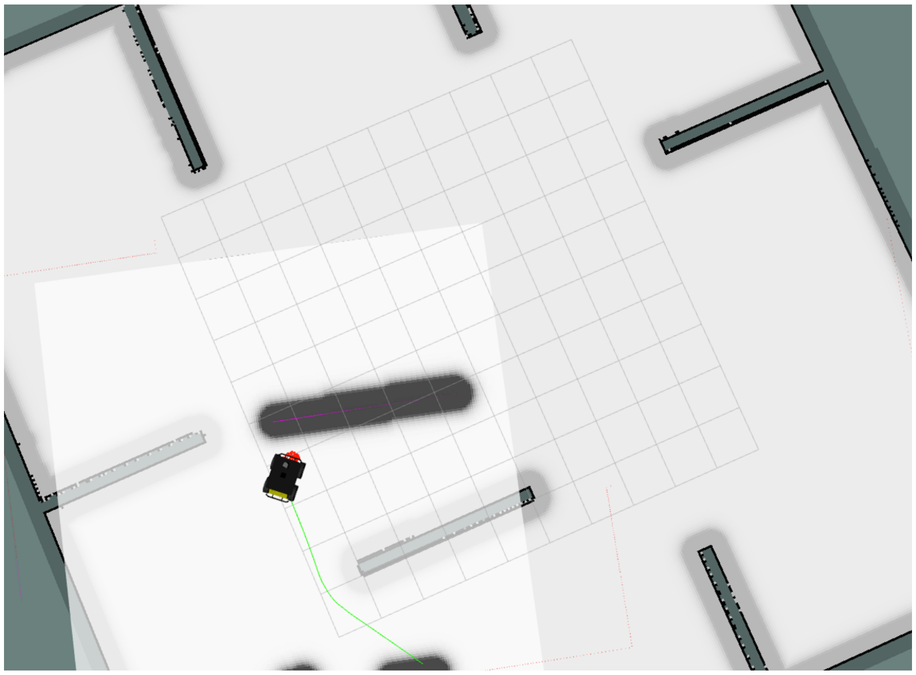
UNSIGNED INT HEIGHT
UNSIGNED INT WIDTH
FLOAT RESOLUTION

With these objects and data structures, the launch file created for the execution of the three nodes provided the husky robot with the ability to concurrently plan a path while scanning for treasures and traversing the environment. The latter two operations in tandem are the most computationally intensive programs of this endeavor.

In contrast to the computational intensity present in traversing and scanning of the environment, the planning algorithm devised for an efficient, and naively, optimal traversal of the environment. Likewise, the planner node was implemented to select points such that the husky robot moves back and forth laterally, from end of the map to the other at increasing distances from the top.

Design Decisions

the following screenshot exemplifies the husky robot's inability to plan a navigable path when there is considerable error in the localization.



as is observable from the screenshot, the husky robot is unable to differentiate between its estimated position and that of the physical barrier. This causes rotational discrepancies which lead to collisions with the environment.

In order to ameliorate the inconsistent accuracy of the husky robot's localization estimations, the inflation radius was increased. The utilization of inflation radius within the cost map enabled the avoidance of collisions with the environment . Essentially, this was the most prominent design decision as it was carefully speculated to prevent the husky robot from, as previously stated, colliding with obstacles present within the environment.

Testing/QA

<u>Treasures</u>

Global coordinates of the treasures dispersed about the map were computed with a transformation by utilizing the ***tf*** 2 transform listener. By looking up the transformation between the ***map*** and the LOGICAL_CAMERA_LINK frame the global coordinates of the treasure were calculated. The aforementioned MAP data structure was updated with each newly, non−redundant, found treasure. The returned pose was compared to the predefined pose to ensure validity of the computed data. As such, the mathematical integrity of the solutions implemented in ***treasures.cpp*** were verified and consequently published to the respective listeners.

<u>Navigation</u>

During the process of developing an efficient ***nav.cpp***, a timeout waiting period of 15 and 45 seconds was introduced in order to delay the sending of the next goal point to the husky robot. This was done in order to prevent the husky robot from prematurely traversing the map before arriving at the corner−most location of the map. In this manner the quality control aspect of this node was primarily ensured with a trial−and−error method of arriving at a reasonable timeout waiting period. Therefore the husky robot is able to begin traversing the environment from the corner−most location of the map.

<u>Planner</u>

The ***x_dist***, ***y_dist***, and ***obstacle_dist*** variables were utilized to demarcate distances along the x-axis and y-axis between points, respectively, as well as the minimum distance from obstacles. In testing, the values of these parameters were refined in order to obtain the desired behavior. The goal was to have the husky robot traverse the entire map to find the treasures without getting too close to obstacles so as to avoid collisions.

Conclusion

Upon completing the endeavor of traversing a treasure−*filled* map, a direct solution was implemented with respect to the planning, navigation, and treasure detection of the husky robot. Along with this, an indirect solution was implemented to mitigate the inaccuracy of the AMCL algorithm.