CSE-180 Final Project
Detecting Dynamic Obstacles in a Robot's Environment

This solution outlines an approach to detecting dynamic obstacles or posts in a robot's environment by comparing a series of occupancy grid maps.

Method:

The code provided (myTestNavigator.cpp) subscribes to the /global_costmap/costmap topic, where it receives nav_msgs::msg::OccupancyGrid messages representing the robot's perception of its environment. These messages are compared with a stored initial occupancy grid map to identify inconsistencies between the two, which are indicative of changes in the environment, such as the presence of dynamic obstacles.

Process Overview

Store the first occupancy grid map as the initial state (using firstGlobalCostMapMsg). Continuously listen for updates to the occupancy grid map. Check for updates every 2 seconds (update_threshold). If the map is updated, compare the latest occupancy grid map with the initial state using the CompareCostmaps function. If inconsistencies are found, convert their indices to poses using the GlobalCostmapIndicesToPose function and log their positions.

CompareCostmaps Function

This function calculates the difference between cell values in the latest occupancy grid map and the initial state. It ignores cells with a value of -1 (unknown or unmapped regions). If the absolute difference between cell values is greater than a predefined threshold (8 in this case) and the new cell value is greater than the old, with the new cell value also being 100, it marks the cell as inconsistent, suggesting that there is a dynamic obstacle at that location. If 5 or more inconsistent cells are found, the function stops processing further cells and returns the list of inconsistent cell positions.

GlobalCostmapIndicesToPose Function

This function converts the row and column indices of an inconsistent cell in the occupancy grid map into a geometry_msgs::msg::PoseStamped object. This enables the robot to use the position of dynamic obstacles in its navigation system.

Implementation

The implementation provided is a complete example that integrates the dynamic obstacle detection into a navigation system. It starts by initializing the robot's pose, followed by setting a series of waypoints for the robot to navigate. It then executes a spin in place and proceeds to navigate through the defined waypoints, while continuously checking for dynamic obstacles using the described method.