Autonomous Exploration

PERCEPCIÓ I COGNICIÓ EN L'EXPLORACIÓ ROBÒTICA

Practice 3 - Optional Section

Author: Victor Escribano Garcia

Teacher: Angel Santamaria Navarro

Date: January 4, 2024

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
ETSEIB

1.1 LFE Method - Last Frontier Exploration

Q1. Explain new exploration method The original autonomous exploration algorithm computed the distance from the robot to frontiers in a map. However, this approach had a drawback: if the robot had already started planning a path to a frontier and a new, closer frontier appeared during the remapping, the robot would not adapt its path to reach the closer frontier. This led to suboptimal exploration and inefficient use of resources.

To address this issue, I made changes to the code in explore.h, explore.cpp, frontier-search.h, and frontier-search.cpp. The key idea was to shift the focus of the cost function from the robot's current position to the position of the last pursued frontier goal. This allowed the robot to continuously adapt its path to the nearest frontiers, providing a more efficient exploration strategy. So now if during a remmapping a new path needs to be computed it will be computed taking into account the **distance from the last frontier** and not the distance from the robot, making then that the following frontier chosen will be on the line of the exploration, not leading to jumps from one point to another on the map.

The only drawback of the method is the compute method for the distance, the original explore lite package used euclidean distance and this method does not work well on this situation as it can take a neighbour frontier that is on the other side of the wall. To solve this the nav2 path distance should be tacked into account instead of the euclidean distance (Not implemented).

Code modifications:

1. Explore.h: Added a new member variable last_frontier_goal_ of type geometry_msgs::msg::Point to the Explore class. This variable stores the position of the last frontier goal pursued by the robot.
2. Explore.cpp: In the constructor of the Explore class, I initialized last_frontier_goal_ with the robot's initial position. This initialization occurred within the if (return_to_init_) block, where the robot's initial pose was obtained. Modified the makePlan method to update last_frontier_goal_ each time a new goal was pursued. Additionally, I changed the call to search_.searchFrom to pass last_frontier_goal_ as an argument, which shifted the reference point for the cost function calculation.
3. Frontier-search.h: Updated the method signature of the searchFrom method in the FrontierSearch class to include an additional parameter geometry_msgs::msg::Point last_goal. This parameter was used for calculating the minimum distance from the last frontier goal, rather than the robot's current position.
4. Frontier-search.cpp: Altered the implementation of the searchFrom function to utilize the last_goal parameter as a reference point for finding and evaluating frontiers. Updated the buildNewFrontier method to accept geometry_msgs::msg::Point last_goal as a parameter, instead of an unsigned int. Inside this function, the distance to each frontier cell was calculated relative to last_goal, affecting the min_distance and subsequently the cost of each frontier.