Autonomous Exploration

PERCEPCIÓ I COGNICIÓ EN L'EXPLORACIÓ ROBÒTICA

Practice 3

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

Q1. What is the formula used in explore_lite node to compute the frontier cost? In the explore.cpp file of a ROS-based autonomous exploration system, the FrontierSearch class (defined in frontier_search.cpp) is utilized to identify exploration frontiers. The cost of each frontier is calculated using the frontierCost function, which employs the formula: Cost = $(potential\_scale\_ * frontier.min\_distance * costmap\_- > getResolution()) - (gain\_scale\_ * frontier.size * costmap\_- > getResolution())$; Here, potential_scale and gain_scale are weights for the frontier's distance from the robot and frontier's size. Costmap_− >getResolution() converts these into real-world physical units.

Q2. Enumerate or briefly describe modifications you could do to improve the frontier selection. In order to improve this formula we could penalize the cost of a big turning angle from the robot orientation to the frontier, penalizing the frontiers that are near the FOV of the current robot's pose. Also we could implement an historical of the robot's explored frontiers, not using it as a black list but using it to take further decisions based on the previous data, for example a change of strategy once we have explored enough big frontiers and we want to go deeper on the smallest.

Q3. Are the default parameters loaded in the explore.launch.py file good enough to explore the whole environment? The default parameters gives a weight of 3 to the frontier distance and a weight of 1 to the frontier size, nevertheless, running the code this leads to long *goes-and-backs* from one frontier to another without fully explore one path, making the exploration not-optimal. To solve that we can give bit more weight to the distance ($potential\_scale\_$) than to the frontier size ($gain\_scale\_$), making that the ones that are far away have more cost.

Q4. Do you find some quick tuning of the parametes (loaded in the explore.launch.py file) that can help either reducing the exploration time or with the quality of the created map? As the size of the frontiers on this scenario is very diverse I decided to go with the closest frontier approach, giving a weight of 20 to the distance ($potential\_scale\_$) and 0 to the frontier size ($gain\_scale\_$). Also I noticed that the distance is calculated from the robot to the frontier, not to the current frontier to the next one, for that I reduced the planner frequency to 0.05Hz to calculate the distance to the next frontier once we arrive to the goal, also to not waste time waiting for the planner y reduced the progress timeout parameter to 1 second in order to recalculate the new goal once the frontier is reached.

Q5. Add a print screen of the best map (rviz) resulting from an exploration after tuning a bit the parameters.

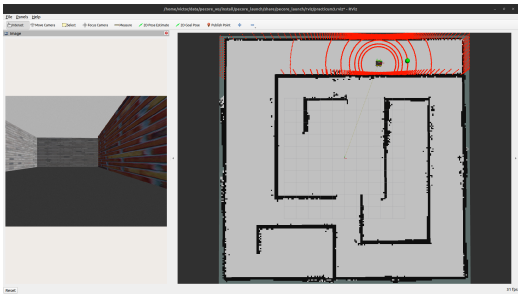

Figure 1.1: RVIZ2 visualization resulting from an exploration after tuning the parameters.