Search Algorithm

Luigi Vanacore
ZhuoXin Shi
Francis Weber
Ricardo Westerman
Rui Xia

December 15, 2024

1 Introduction

The autopilot node map exploration strategy combines two complementary approaches: a random search and a systematic evaluation.

The random search quickly identifies potential waypoints on the map's frontiers, allowing for rapid exploration of new areas. This method is particularly effective in the early stages of mapping where the map is smaller.

To complement this, a systematic evaluation periodically examines the entire known map, identifying areas with the highest potential for new information. This ensures thorough exploration, especially in complex environments.

By alternating between these methods, the system balances speed and thoroughness. The random search provides quick expansion of the explored area, while the systematic evaluation ensures no important details are missed. This combination results in an efficient and adaptable exploration strategy suitable for various environments.

This search strategy is interrupted when the position of an ArUco target is received by the autopilot node. At this stage a suitable pose close to the target is published in order to localise accurately the target.

2 Subscriptions and Publishers

2.1 Subscriptions

- `behavior_tree_log` (Type: BehaviorTreeLog)
  - Used to determine when the robot is ready for a new waypoint, or the previous point sent is not reachable
- `global_costmap/costmap` (Type: OccupancyGrid)
  - Provides current occupancy grid information
- `pose` (Type: PoseWithCovarianceStamped)
  - Provides current position of Turtlebot
- `aruco_map_position` (Type: PointStamped)
  - Provides ArUco marker position information

2.2 Publishers

- `goal_pose` (Type: PoseStamped)
  - Publishes next waypoint parameters
- `potential_point` (Type: PointStamped)
  - Publishes potential waypoints for visualisation
- `current_point` (Type: PointStamped)
  - Publishes current coordinate for visualisation

2.3 Random search

This is the default strategy of the search algorithm. It selects a waypoint when a new occupancy grid is received and the previous goal is achieved or discarded. It uses a random selection process to find suitable points in the costmap, applying the following conditions to ensure the selected point is appropriate for exploration.

- Not too close to an obstacle (cost value has to be lower than 75)
- Not unknown
- Not further than 3 meters and closer than 1 meter from the current position
- On the frontier (It has at least 20 uncertain cells in a $0.16m^2$ box around the point)

If a suitable point isn't found after 10,000 iterations, or if 50 good points are found but not within range, it switches to the new strategy.

2.4 Whole map analysis

In this strategy the entire occupancy grid is processed through the new_strategy() function, creating a sorted list of cells based on the number of unknown cells in a $0.36m^2$ box around them. It is called periodically to ensure a robust exploration of the map, and exceptionally, as mentioned before, when the random search is struggling.

It selects points that are within 5 meters of the current position or, every fourth calls, considers points beyond 5 meters to encourage exploration of distant areas.
The uncertainty around a potential point is checked only if the point is:

- Not too close to an obstacle
- Not unknown
- Not further than 5 meters (when the distance constraint is applied)

This ensures a quick processing of the entire cost map.

2.5 ArUco marker detection response

The algorithm implements a response to ArUco marker detection. When the position of a marker is received from the topic aruco_map_position, it calculates the distance between the robot and the marker.

If the marker is more than 5 meters away, the robot moves to the midpoint between its current position and the marker, this strategy avoids to send far points in potential unknown regions.
For distances between 1.5 and 5 meters, the robot moves to a point 1.5 meters away from the marker. If the robot is within 1.5 meters of the marker, it remains at the pose where it detected the target. In all cases, the robot orients itself towards the marker and waits 15 seconds. This adaptive approach ensures efficient navigation towards the marker while maintaining a safe distance for accurate localization.

2.6 Completed map behavior

The map is considered fully explored when the new_strategy() has been called for 5 consecutive times because the random search was not able to find a suitable point. At this stage, random points close to obstacles are sent to look for potential missed targets.