METR4202 Group 5 Stage 0 Report

Rachel Chiong (s4698038)
Sara Alaei (47041437)
Grace Margaretha (47260425)
Mingxuan Zhou (47038400)

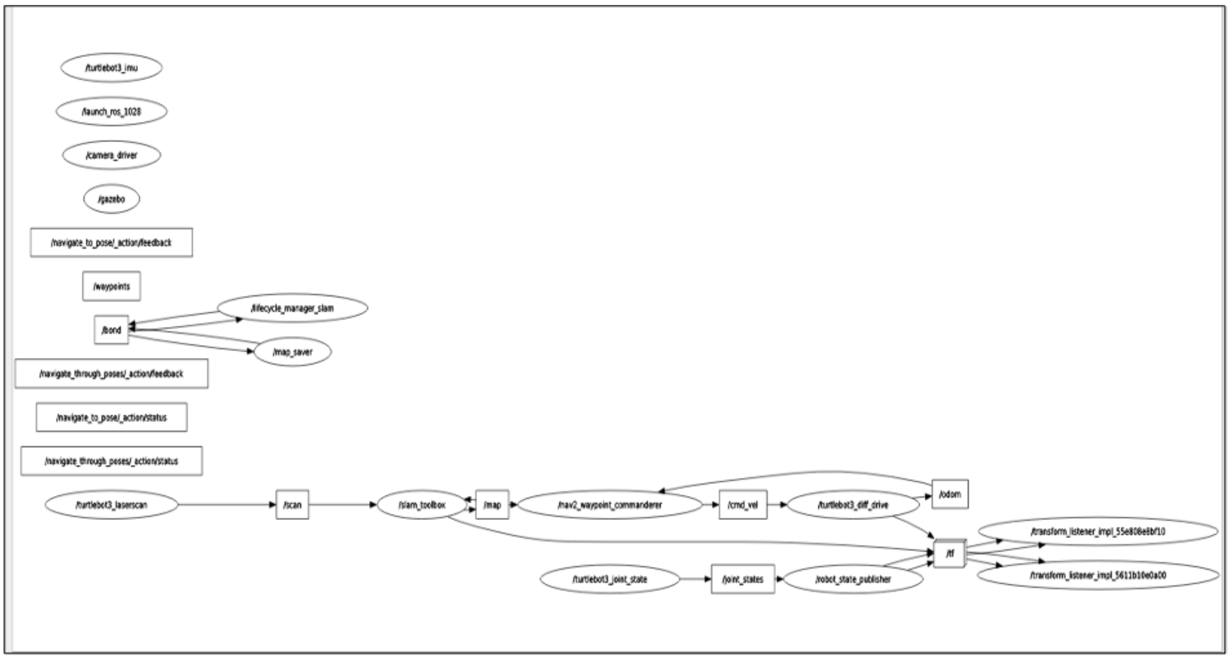
FronTEAR_commander Repo (bananas branch):
https://github.com/RachelChiong/METR4202.git

1 Introduction

This report will cover the topics, components and methods used to develop an exploration strategy for the TurtleBot3 to enable it to construct a 2D occupancy grid map. The strategy aims to explore frontiers of an unknown map effectively, leaving no areas unexplored.

2 System Diagram

Figure 1 is a system diagram displaying the interaction between the topics of the system. The main node for the frontier commander is the `nav2_waypoint_commanderer` which makes use of the `map` topic retrieved from the SLAM toolbox using the laser scan. The `nav2_waypoint_commanderer` also uses the feedback from the `odom` topic to localize the robot in the map to be able to determine the best next move.



/turtlebot3_imu
/launch_ros_1028
/camera_driver
/gazebo
/navigate_to_pose/_action/feedback
/waypoints
/lifecycle_manager_slam
/bond
/map_saver
/navigate_through_poses/_action/feedback
/navigate_to_pose/_action/status
/navigate_through_poses/_action/status
/odom
/turtlebot3_laserscan
/scan
/slam_toolbox
/map
/nav2_waypoint_commanderer
/cmd_vel
/turtlebot3_diff_drive
/transform_listener_impl_55e808e8bf10
/tf
/transform_listener_impl_5611b10e0a00
/turtlebot3_joint_state
/joint_states
/robot_state_publisher

Figure 1: System Diagram

3 Main Libraries and Referenced Code

The `FronTEAR_Commander` uses the following libraries and resources.

3.1 ROS Client Library for Python (RCLpy)

RCLpy is a python client library for ROS2, used to interact with the robot. Various features of RCLpy are imported into the code and utilised within the written functions. Nodes can be created which have the purpose of sending and receiving data from other nodes. The information can be accessed through topics. Publishers such as pose stamped and MarkerArray send sensor data to the subscribers: behaviour tree, cost map and occupancy grid and odometry.

3.1.1 BehaviourTreeLog

Behaviour Trees Log trees provide an event log. In this case, it used to check the current status of the robot: `IDLE, RUNNING, SUCCESS or FAILURE`. In the `FronTEAR_Commander`, the `BehaviorTreeLog` also acts as the main controller and initiates the frontier exploration when `IDLE`.

3.1.2 Occupancy Grid

The occupancy grid, retrieved from the `/map` topic, represents the environment that is broken down into a grid where each cell represents a location that is in the world. It assigns a tile cost of `0` for a known free space, `100` for a known occupied space (wall or obstacle) and `-1` for an unknown space.

The occupancy grid retrieved from the `/map` subscription is stored in the custom `OccupancyGrid2d` class, which is then used to determine the frontier.

3.1.3 Odometry

Odometry is used by the TurtleBot to estimate the position and orientation of the robot relative to starting position, in terms of (x, y) and its orientation (yaw). It covers both publishing the `nav_msgs, Odometry` messages over ROS, and a transform from an "odom" coordinate frame to a "base_link" coordinate frame over tf.

3.1.4 Pose Stamped and PoseWithCovarianceStamped

Pose Stamped contains the header and represents the position and quaternion position of the robot. The waypoint can be published to this topic, and it will effectively change the position of the robot. PoseWithCovarianceStamped is similar but the waypoint is not published rather a current position is set with it.

3.2 NumPy

The NumPy package was used to determine the centroid of a cluster of points by re-shaping the 2D array.

3.3 nav2_wavefront_exploration repo (source)

The custom OccupancyGrid2d class and related methods for converting between occupancy grid and and world locations were used Nav2_Wavefront_Exploration Repo by SeanReg. This was to simplify the map update and retrieval process for determining frontiers.

4 Components

4.1 Class: OccupancyGrid2d

This class is used to create an instance of the occupancy grid map. It contains methods like getCost, getSize and mapToWorld.

4.2 Centroid

This function determines the coordinates of the centre of the inputted array and returns an (x, y) tuple coordinate.

4.3 Class: FronTEARCommander

4.3.1 Call-back functions: `bt_log_callback`, `occupancyGridCallback` & `poseCallback`

- `Bt_log_callback` checks whether the robot is in `IDLE` and `NavigateRecovery` state and then moves to frontiers.
- `OccupancyGridCallback` constructs the occupancy grid (-1, 0, 100) values of the global map and;
- `poseCallback` updates the current position.

4.3.2 Near_unknown

This method checks whether at least one point near (x, y) is an unknown point, it returns True if near an unknown square, i.e., a frontier, and returns False otherwise.

4.3.3 get_Cluster

The `get_Cluster` method constructs a cluster of good waypoints starting at a particular given point. If a waypoint is added to a cluster, it is removed from the good waypoints list.

4.3.4 Seggregate_frontiers

This method handles the frontier points and classifies them as either good or bad waypoints where a "good point" is defined as a known free space which is adjacent to an unknown space, and a "bad point" is defined as a known occupied space adjacent to an unknown space.

Then, for each good waypoint, it generates the clusters and adds them to a priority queue where the largest cluster has the highest priority, until there are no more un-clustered waypoints.

4.3.5 move_to_frontier

This method is called on once the `bt_log_callback` reaches its IDLE state, and it handles moving to the next frontier. It by retrieves the priority queue of clusters from the `seggregate_frontiers` method and determines the centroid of the clustered occupancy grid points, then converts it to a world Pose that the robot can move to and adds the waypoint to the set of visited waypoints. If the new pose is already a visited, this waypoint is skipped and the centroid of the next largest cluster is determined and sent, and so forth.

If there are no more clusters left in the queue (i.e. all waypoints are explored), this function sets the `is_complete` flag which signals to the `rclpy` instance that exploration is complete, terminating the node.

4.3.6 setInitialPose and setWaypoints

The `setInitialPose` method sets the initial pose and `setWaypoint` converts the waypoint in occupancy grid map coordinates to a PoseStamped instance to be published.

5 Conclusion

The system uses SLAM to produce a map of the world and the robot navigates through the world via the goal waypoints that is determined and sent to the robot by the exploration algorithm. The algorithm determines unexplored areas and directs the robot to reach that goal position. A navigation strategy when the exploration strategy fails has also been implemented, by checking if the robot's location has changed for the past four iterations.