Implementation of a Robot Behaviour Learning Simulator

Kushagra Singh Bisen

Ecole des Mines de Saint Etienne

kushagrasingh.bisen@etu.emse.fr

July 5, 2021

Overview

1. Recap
   - Last Meeting.
2. An Introduction
   - What is ROS?
   - What is Gazebo?
   - What is a Turtlebot?
   - What is SLAM?
   - What is Navigation Stack?
3. Work Done.
   - Since Last Meeting.
   - ROS Graph
   - Goals : Next Meeting.

What is ROS?

ROS stands for Robot Operating System, which is not exactly an operating system but more like a middleware between the Robot and the Enginner controlling the Robot. It provides a great level of abstraction over the drivers that are needed to move the robot in real-time, also providing a prototype of its working in any simulator software.

What is Gazebo?

Gazebo is the most popular simulation software for ROS. It is the industry/research standard for the simulations. The accompanying physics engine is similar to real life in the terms of friction and collisions. The overall accuracy of the simulator to the real-time robot is around 85 percent.

Turtlebots

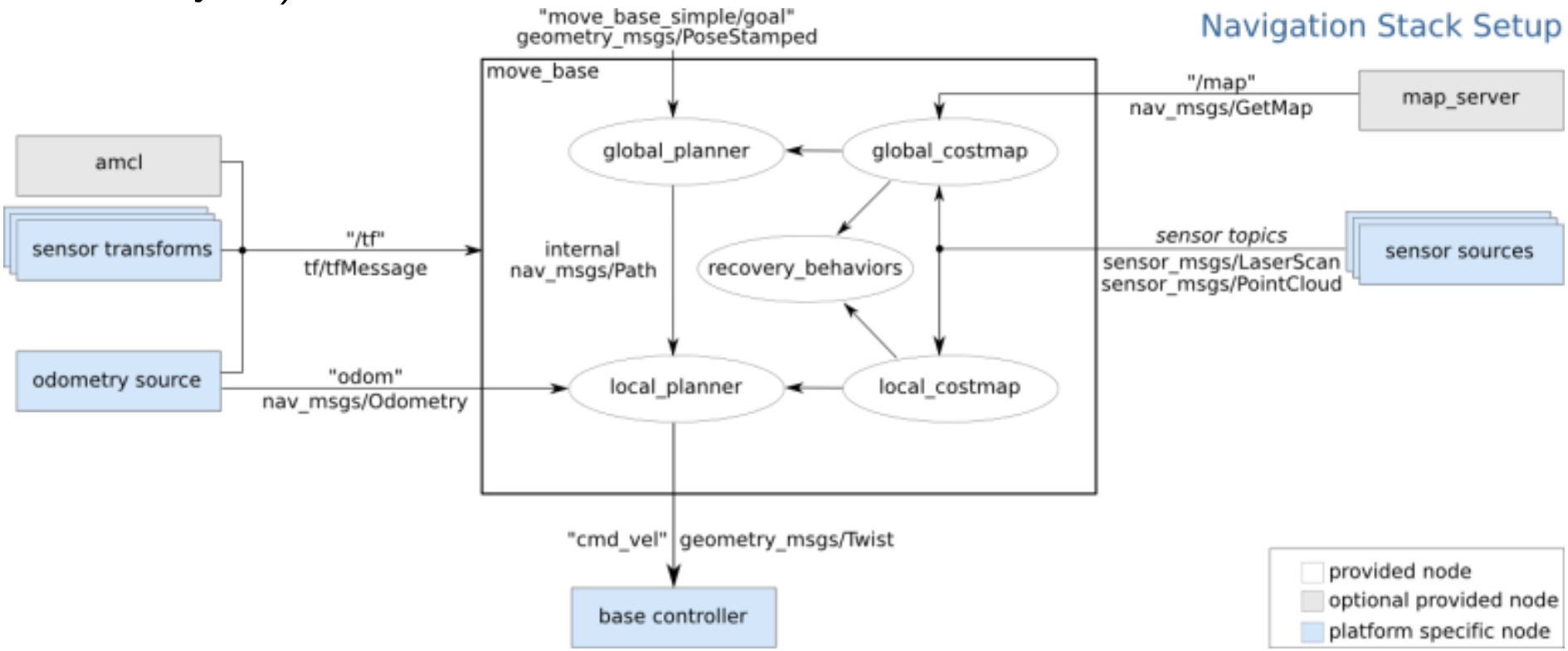
Turtlebot and Turtlebot3 are open-source robots with both hardware and software being open sourced. In the multi agent system paradigm, these both are nothing but a point object. They are equipped/can be equipped with various sensors and can be modified. In Gazebo, as we are not using any sort of real hardware/ labaratory environment even sky is not the limit, as people simulate mars rover robot in Gazebo.

What is SLAM?

SLAM stands for Simultaneous Localization and Mapping, which as the name explains localizes the robot in an environment and maps. For any sort of Navigation to happen, the robot should know where it stands in the map and should know how the map actually looks like. The Turtlebot used it's LIDAR sensor to detect the environment and then save it.

Navigation Stack

Navigation Stack deals with the tools/libraries required for the robot to navigate from a position to the goal (strategies of doing the motion can be defined by us)



The ways we tried to solve the problem earlier.

- As we wished to avoid the obstacle and go to the goal, I first tried to use the LIDAR sensor, and on contact with the obstacle it would change it's angular and linear velocity. It was arbitrary and not very useful. One could've tried to make it 'stop' if it reached a particular point, but as it did not know anything about it's environment the robot would not know if it is kept in desert or himalayas which is not a good approach.
- The next objective became to make the robot know wherever it is, and thus came SLAM to the equation, SLAM helped us to localize the robot in the environment and then map it. After the mapping was done, it was able to go from A to B, but the motion was the shortest path (with respect to the angular field of motion) as it had a 360 degree of motion. We wish the robot to move in 4 ways.
- The path planning in navigation came next, which we are using now. Here, we mapped an environment, made a map with blocks and tiles and walls, which is alterable to personalized preference in various scenarios.

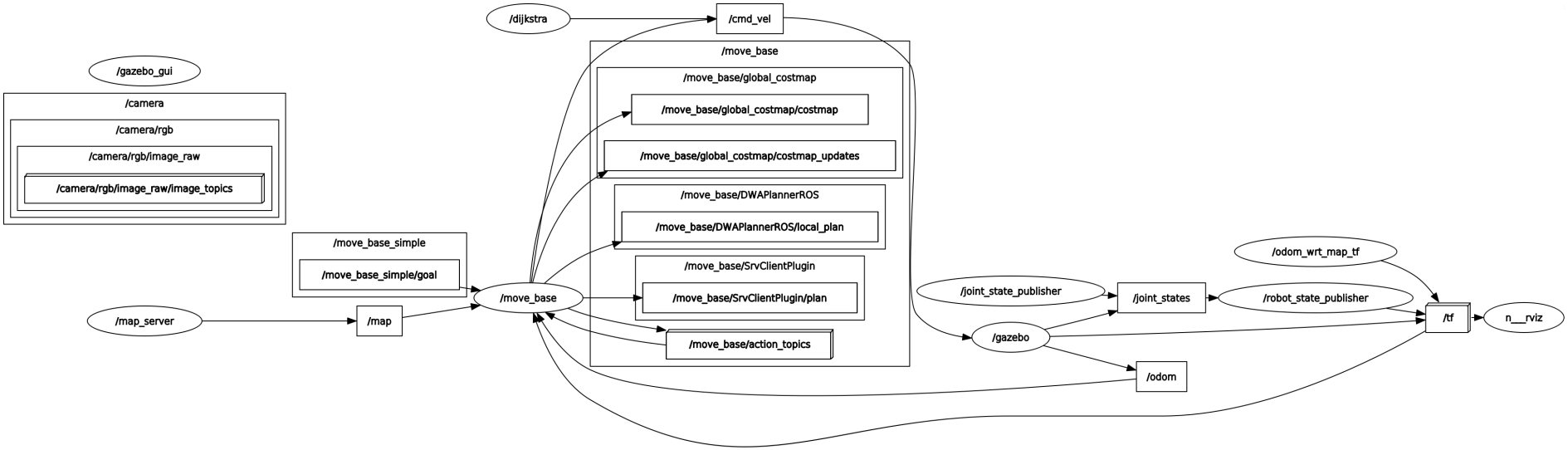
The ways we tried to solve the problem earlier.

- The map is then cut up into various smaller boxes considering the radius of the turtlebot. The costmap 2D thus divides the map into such parts. Note that the costmap2D and the planning we are initiating right now is only with the 'global planner'. I will add the local planner functionality in further steps (I am still reading on it).
- The local planner is mostly used for the collision avoidance, collision from another robot or an object which is mobile in the environment. I am not much read into the field but reading a few survey papers, I have a sense of understanding that the collision avoidance methods are still a good area of research. I read on an internet forum that probabalistic approach to the problem is the most robust way most of the time. I read a paper titled "Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Pattern" but was not able to understand much in 1 day. I have attached the paper in the mail.

Work : since last meeting.

- The GitLab repository's folder structure was organized.
- The schema to organize the main software's folder structure was worked on.
- Method to store the neighbour's position and state was done, conceptualized and found but has some bugs at the moment.
- I read about and saw some projects using the local planner's implementation.
- I found out that I can also program the move base node for it to move in a better deterministic way.

ROS Graph


/dijkstra
/cmd_vel
/move_base
/move_base/global_costmap
/move_base/global_costmap/costmap
/move_base/global_costmap/costmap_updates
/move_base/DWAPlannerROS
/move_base/DWAPlannerROS/local_plan
/move_base/SrvClientPlugin
/move_base/SrvClientPlugin/plan
/move_base/action_topics
/gazebo_gui
/camera
/camera/rgb
/camera/rgb/image_raw
/camera/rgb/image_raw/image_topics
/move_base_simple
/move_base_simple/goal
/map_server
/map
/move_base
/joint_state_publisher
/gazebo
/joint_states
/odom
/odom_wrt_map_tf
/robot_state_publisher
/tf
n__rviz

Goals : Next Meeting

For the next meeting,

- I propose for me to sort out the bug and put out the basic version of the log file.
- Rest other goals depend on you, and the next meeting date.

Thank you for your time.