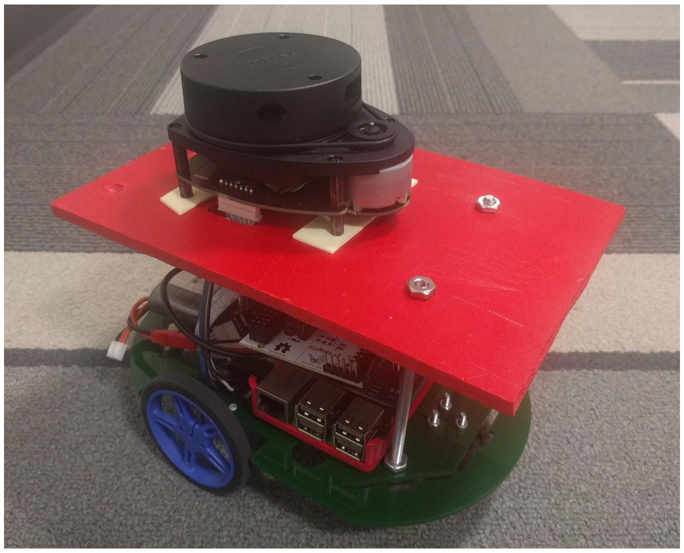
Final Project Report for group 11

Group members: Kieran Nichols, Alex Becker, Leo Liu

Intro:

In a world of navigation ruled by sound radar, there comes a time when technology must improve. Lidar is here to achieve that purpose. Our group chose to use the Robotic Operating System (ROS) to move a mobile robot and to use LIght Detection And Ranging (LIDAR) device to build a point cloud map of the room. We chose this project to gain an understanding of the high level operations of ROS in combining the operations of wheel motors, LIDAR data, and mapping. We also wanted to explore the use of LIDAR in efficient mapping of spaces and test its ability to predict its location tracking without the use of odometry.

Design:



The hardware design was built on the developed moving robot base. A new plastic shaft supported by three screwed bars were designed and fabricated to mount and support the lidar on the top of the robot. The shaft is carefully leveled horizontally to guarantee that the lidar scan plane is horizontal and prevent inclined laser beam from hitting the floor and causing incorrect detection of the floor as barrier.

The software design is separated into two parts: robot control and data collection. The Robot control part was built based on the previous code for remote controlling the robot with keyboard command. The structure was kept the same, with adjustance of scaling factors of keyboard velocity command values to assure smooth and slow reaction and provide a stable platform for the data collection.

For the data collection part, the rpLidar driver collected the information received by the laser sensor and process it into a digital signal. The signal was then broadcasted by ROS node through “/scan” topic. The ROSBag function was also activated and save the messages in the “/scan” topic on the SD card. After the whole data collection process finished, the saved messages was downloaded to the computed and processed through “hector mapping‘ algorithm to generate the map.

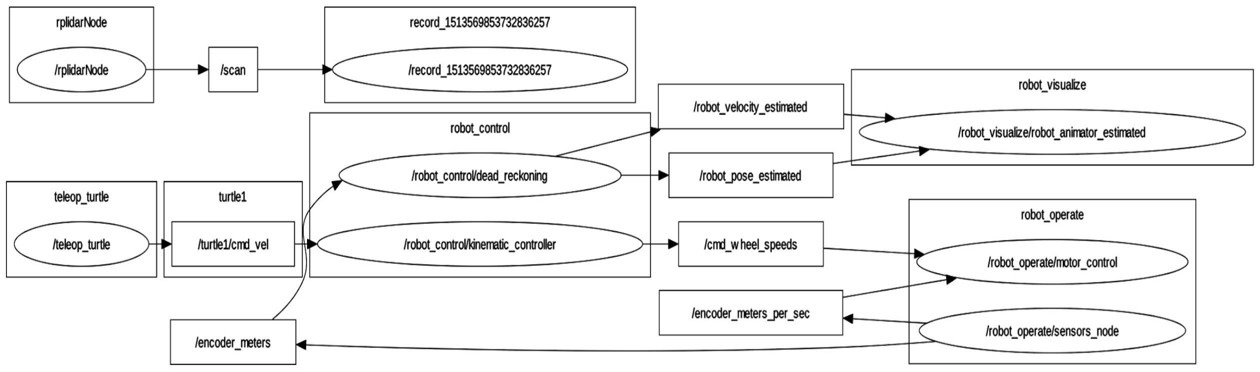

rplidarNode
/rplidarNode
/scan
record_1513569853732836257
/record_1513569853732836257
robot_control
/robot_control/dead_reckoning
/robot_control/kinematic_controller
/robot_velocity_estimated
/robot_pose_estimated
robot_visualize
/robot_visualize/robot_animator_estimated
teleop_turtle
/teleop_turtle
turtle1
/turtle1/cmd_vel
/cmd_wheel_speeds
robot_operate
/robot_operate/motor_control
/robot_operate/sensors_node
/encoder_meters_per_sec
/encoder_meters

Figure 1 shows the rqt_graph of the node/topic system for ROS on the mobile robot with lidar system. The /robot_controlled_reckoning was controlled using the code supplied by Professor Adamczyk (Mobile_Robot_For_Fair).

The can data was transferred to Virtual Machine (Ubuntu) to process the LIDAR data. The raspberry pi that collected the LIDAR data and control the wheel motor was not able to plot the mapping data to another computer was used. The rosbag was read and played on the Ubuntu using the hector_mapping node to display the point cloud progressed through time as the mobile robot moved through the hallway at Wendt Commons.

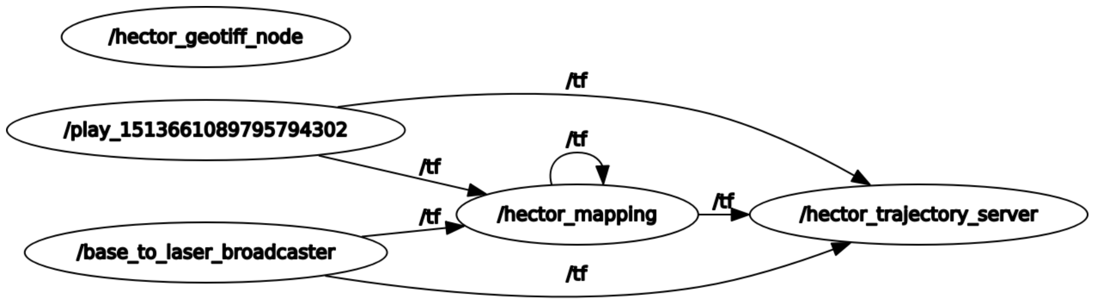

/hector_geotiff_node
/play_1513661089795794302
/base_to_laser_broadcaster
/hector_mapping
/hector_trajectory_server
/tf
/tf
/tf
/tf
/tf
/tf

Figure 2 shows the rqt_graph of the node/topic system for ROS on the Ubuntu that mapped the LIDAR data.

There are several worth-noting points in this design. First, the only the “/scan” topic was saved in ROSbag and submitted to mapping algorithm. This method was selected to prevent the overload of the processor on the raspberry pi, which is a possible reason for the buffer overflow that may cause the incorrect process of control command. Second, it is noticeable that the encoder data was not collected and processed by the mapping algorithm. This could be the main reason for the error in map generation that is discussed below in the “Result” section. The reason for this missing data is that first the code used for processing encoder signal did not broadcast messages in the format recognizable to the mapping algorithm, and second the transformation matrix information that indicates the location of encoders relative to the lidar was not broadcasted. Therefore considering the complexity of the required work, the encoder data was not collected and therefore the map is generated purely based on the collected lidar sensor data.

Results:

Our results obtained through this project came at two distinct times while working on this project. The first breakthrough was when we were able to run the LIDAR software using ros on our laptops. The second was when we were able to run the LIDAR on our mobile robot using a raspberry pi, and then reading the data created by our LIDAR on our laptops. Our finished product is a mobile robot and LIDAR system that is being run and controlled completely through ros.

Our first result can be seen in figure 1, this is a clean 2D map created by holding the lidar parallel to the ground in our hand, and running it on our laptop using ros. In order to run the lidar and collect data to create a map we downloaded software called rplidar_ros and hector_slam_ros. Hector_slam_ros aloud us to use rviz, and the rviz used the scan topic created by the LIDAR to create a map. The gray space in the map is empty space with no obstacles, and the black lines are the walls, or obstacles the LIDAR is picking up.

Our second result can be seen in figure 2, this map is a 2D map of the same clean map shown in figure 1, the only difference being figure 2 was taken from the mobile robot. The map was taken the same way for figure 2, the only difference being we bagged the topic being sent out of the rplidar_node/ called scan/, and we replayed the bag on one of our laptops where we could run rviz without any problems. We think that figure 2 is much less clean compared to figure one, because of the fast turns that the mobile robot takes. Our mobile robot was run by using the launch file robot_operate.launch and was controlled by the turtle_teleop_key/ node. As you can see from out videos, the mobile robot turned at a high velocity. Earlier in the project we learned that moving the LIDAR at high velocities, and tilting the LIDAR created bad readings. After replaying the videos and the mapping at the same time we realized that everytime we turned the robot, the map would get distorted and create false data.

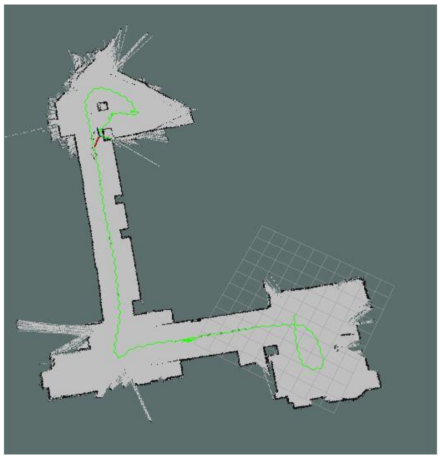


Figure 3: Lidar map created by walking with laptop

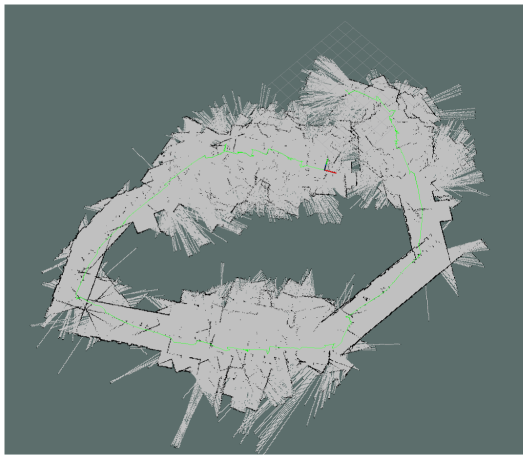


Figure 2: Lidar map created by mobile robot

Discussion:

To aid in another group being able to continue or alter our project, we will list the major steps we took to get the mobile robot, ROS, and LIDAR working. Due to the aforementioned memory limitations of ROS on the raspberry pi, we used the raspberry pi for the data collection and our virtual linux machine (ubuntu) for the mapping. The groups should be able to use the code (that is attached in the zip files) to at least replicate our work.

To run the mobile robot, we used the last folder sent to us from Professor Adamczyk and Josh, and folder name was me439robot. In this folder it had a launch file named robot_operate.launch, and python files, the most important being motor_control.py, sensors_nodes.py, and open_loop_stage_manager.py. The launch file ran a stage_setting course written out in open_loop_stage_manager.py, and then once the stage settings ran its course, the mobile robot waiting for inputs from the user using the turtle_teleop_key/ node. We set the stage_settings to (0, 0, 0), which was (duration, vc, omega), this meant that once the launch file was launched using roslaunch the mobile robot would sit idle waiting for inputs form the user. In order to run the launch file and have the robot run smoothly we change settings in motor_control.py and sensors_nodes.py. This is done earlier in the course, make sure you write down the values set to these parameters, so just incase the files get lost, the python code can be changed quickly without any hassle.

To run the mapping software, the catkin_ws directory can be utilized. Assuming you have ROS, you run the command "source devel/setup.bash" within the catkin_ws directory. Then you run "roslaunch hector_mapping_launch turtorial.launch". Unfortunately you may get an error that one of the launch files may not exist; you can repeat the previous step a few times to remedy this problem. The roslaunch step will open 'rviz' which is the graphical software that will display the mapping information. You will need to obtain a rosbag with the /slam topic that will contain the lidar information to create a map. Open another terminal and use the command "rosbag play --clock <file location>". After a few seconds, rviz will start to display in real time how the mapping information is build as the robot moves through the space (e.g. room/hallway).

The design in the software level is highly limited by the computing capability of raspberry pi. We assume that an improved result could be gained by using a more powerful processor or by successfully implementing shared roscore to perform distributed computation instantly. It has been proved in the discussion of our results that the odometry data is necessary to perform SLAM in complicated indoor environment. Therefore finishing the node for static transformation matrix broadcast of the encoder positions and the node that broadcast "/odometry" topic matching the mapping algorithm is highly suggested.

All in all, this project has provided us with knowledge of the capabilities and limitations of ROS. We were able to manually control the mobile robot with our directional keys through the teleoperation node, collect LIDAR data using the rplidar and rosbag nodes, and map a hallway using the SLAM node. Future projects could focus on ensuring smoother and slower turns so that there will be no jerky movements that affect the LIDAR data collection. We hope that future project could implement a obstacle avoidance algorithm using the point cloud data from the LIDAR. The new technology of the LIDAR has showed us its abilities (though currently limited) in mapping a hallway and we hope that people could continue its implementation in creating internal maps of the building and obstacle avoidance.