PROJECT ALTAIR

Week 2

Hasin Mahtab Alvee
210042174
SWE'21

Contents

Link for Project Altair Week 2 Git Repo

Link for ROS Packages Git Repo

1 Python Programming and OpenCV

1.1 Python Programming

Logical 1. Progress Rover Instructions

References

1. Python Lists and Tuples
2. Dictionary in Python

1.2 OpenCV

Logical 2. OpenCV to Extract data

References

1. NumPy Tutorial
2. OpenCV Tutorial
3. Detecting Multiple Colors

2 ROS Part

2.1 Task 01

Practical 1. Turtle Sim

2.1.1 1.

2.1.2 2.

2.1.3 3.

References

1. Turtle BOT Guide
2. Bellman Ford Algorithm
3. Bellman Ford Algorithm

3 Theoretical Part

3.1 Task 01

Theoretical 1. LoRa Module for Communication

Communication to transfer data is very crucial between devices, such as IoT devices in remote rover deployments projects. Different types of wireless methods are used to receive data from devices, some of the most widely used are the Generic Radio Frequency module and the LoRa Module. Both have their advantages and disadvantages.

In a scenario, where a rover is deployed in a remote and vast area, the **LoRa connection module** would be more effective than a generic radio frequency module. Let's consider the use cases -

- **Data transmission Range :** For a vast area, the LoRa is better, as it is the best for long range connections. LoRa means, Long Range. It can be connected from 500 meters to all the way upto 10 km. Whereas, a generic RF module can only work correctly at maximum 25 meters.
- **Power Consumption :** Power consumption is lower in the generic RF module, which is only 2.9mA while receiving and only 0.3mA while sending. Compared to that, LoRa has a significant amount, which is 10mA both sending and receiving, which is still low than most other wireless communication methods.
- **Data transfer Rate :** LoRa sacrifices data transfer rate for the long range connectivity, though the generic RF module isn't any better than LoRa. The generic RF only has 4 kilobits per second transfer rate. Whereas, LoRa has a max data transfer rate of 37.5 kilobits per second which is great for transmitting sensor data, but video transmitting is not possible. It can transfer sensor data which can be used to measure environmental conditions such as temperature, sound, pollution levels, humidity and wind. Furthermore, the data transfer rate can be changed if wanted to increase the rate for a smaller cover range. Which makes it very adaptive to environments.
- **Connectivity in challenging environments. :** LoRa utilizes different types of gateways communication methods for faster and safer data transmission. Two architectures are - Gateway LoRa and Waspmote. Both of these make sure that data can be sent and received over larger areas. Which also makes it easier to maintain connectivity in challenging environments, which the generic RF module lacks.

References

1. LoRa Connectivity
2. NRF24 VS LoRa For Wireless Communication Between IoT Devices

3.2 Task 02

Theoretical 2. Pose Graph Optimization in SLAM

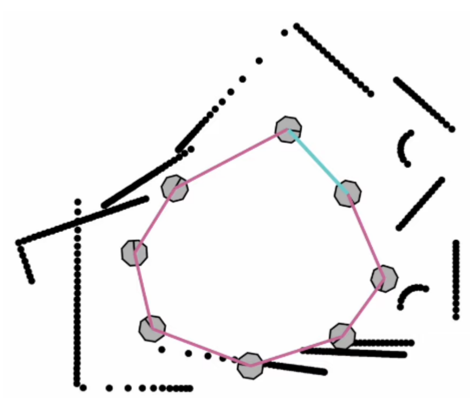
SLAM stands for Simultaneous Localization and Mapping, which means there will be no given data on the model of the environment. SLAM used two types of algorithm to map the environment,

- **Filtering :** Measures the latest movements on the go to update the environment map.
- **Smoothing :** Takes into account the complete set of robot trajectories to map the environment. And the recent factor standard framework in Pose graph optimization for this method.

In a scenario, a vehicle is equipped with LIDAR and odometry sensors to perceive its surroundings and estimate its position and orientation. The LIDAR helps to calculate the disrance between the vehicle and the obstacles, while the odometry uses movements of the encoder to determine how much the vehicle has moved from it's previous location.

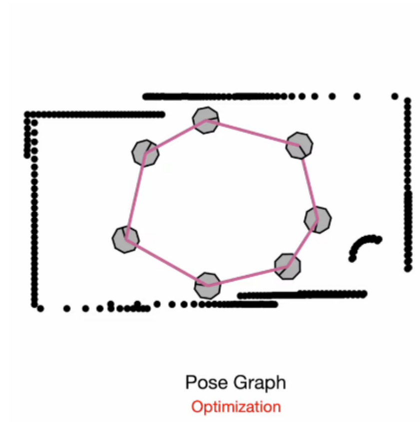
For the ideal conditions, where the LIDAR and odometry will give precise locations and data, which in return will give us the perfect position for the map in one loop. And we can map the environment without any error. But this mapping example is not realistic. Both the LIDAR measurements and odometry will have it's errors and those error will add up for each loop if they are not considered and fixed with the iteration of loops in mapping.

Pose graph optimization is a specific framework within SLAM that optimizes the estimated poses (positions and orientations) of the robot and the mapped environment based on sensor measurements. It involves creating a graph where robot poses are represented as nodes, and constraints between these poses are represented as edges. Using the pose graphing method, we can save the readings of each time the robots moves and measures the distance of the next obstacle or wall, we save that to a graph and continue to do this until the robot comes back to it's starting position. But we still can't do much with these readings and that's when the edges come in handy.



Pose Graph

As we have two positions of the robot with the exact same obstacle data, it means those two positions are the same, so any difference between the two points that are joined by the edge, can be dismissed and should be put onto each other, as this creates a tension among all the other connected edges and nodes as well, this pulls the graph closer and giving it a better shape which is similar to the actual map.



As we keep looping around the map and keep measuring data and the distances between nodes and keep track of the edges, we can get the almost exact map data only after a few more iterations of the whole environment. Even if some incorrect measurements are made, they can always be fixed using the other nodes as this Pose graph method uses absolute positioning instead of relative positioning.

Now that we have out Pose graph ready, we can turn this into an environment map model using different methods, such as Binary Occupancy Grid and Probabilistic Occupancy Grid. The binary method uses 0 and 1 to represent if a grid position is occupied or not and then builds a map around the occupied points. And the Probabilistic takes into account how much a point in the grid is occupied between 0 and 1, and then have black cells for occupied and white cells for un-occupied space, and everything else is in gray.

3.3 Task 03

Theoretical 3. Pixhawk

References

1. DC Motor with Encoder
2. Using Rotary Encoders with Arduino