Autonomous Vehicle Using Simultaneous Localization and Mapping

Maninder Bir Singh Gulshan[1], Manasvi Grover[2], Priyansha Chichra[3] and Rajiv Sharma[4]

Department of Electronics & Communication Engineering

Dr. Akhilesh Das Gupta Institute of Technology and Management, Delhi, India

E-mail:- *maninderbirgulshan@gmail.com[1], manasvi.grover22@gmail.com[2], priyansha9998@gmail.com[3], rajiv.sharma@adgitmdelhi.ac.in[4]*

DOI: - https://doi.org/10.47531/MANTECH/ECC.2021.26

Abstract

Building a map and localization of mobile robots is an elementary and key complication in research on robotics. This paper is a thorough implementation of PyRobotics algorithms for simultaneous localization and mapping (SLAM) robots under numerous geographical surroundings. The basic idea and attributes of SLAM, along with the features and categorization of map representation without any human interference locating itself based on probability theory using PyRobotics algorithm, are analyzed in the paper. This paper provides the recent development, features, implementation, recent issues and algorithms of SLAM. Finally, these PyRobotics algorithms are implemented on the Hardware Setup using Python Integrated Development and Learning Environment (IDLE) and Robot Operating System (ROS) - kinetic.

Keywords:- *SLAM, PyRobotics, LiDAR, KF, EKF, orchestrating.*

INTRODUCTION

It is the process of engendering a map utilizing a robot or unmanned conveyance that navigates that environment while utilizing the map it engenders. Mapping is rudimentary determining the location of objects in the environment, and localization is establishing the robot's position with reverence to these objects.

Mobile robots are expected to perform perplexed tasks that require navigation in involute and dynamic indoor and alfresco environments without any human input. In order to autonomously navigate, path plan, and perform these tasks efficiently and safely, the robot needs to be able to localize itself in its environment predicated on the constructed maps.[2]

Mapping the spatial information of the environment is done online with no prior erudition of the robot's location; the built map is subsequently utilized by the robot for navigation. In navigation, robotic mapping and odometry for virtual authenticity or augmented authenticity, simultaneous localization and mapping (SLAM) is the computational quandary of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

While this initially appears to be a chicken-and-egg quandary, there are several algorithms kenned for solving it, at least approximately, intractable time for certain environments. Popular approximate solution methods include the particle filter, elongated Kalman filter, Covariance intersection, and Graph SLAM.[2]

The system requires a wide range of technologies: It requires token where it is (localization), where it is safe (mapping), where and how to move (path orchestrating), and how to control its kinetics (path following). The autonomous system would not work correctly if any of these technologies is missing.[6] SLAM is a key component in self-driving conveyances and other autonomous robots, enabling cognizance of where they are and the best routes to where they are peregrinated by engendering its own maps; SLAM enables more expeditious, more autonomous and adaptable replication than pre-programmed routes.[8]

- Autonomous conveyances
- Unnamed Aerial Vehicles (UAVs)
- Autonomous Submersed conveyances

- Planetary rovers
- Domestic robots like Roomba
- Self-driving cars
- Medicine
- Planetary, aerial, terrestrial and oceanic exploration
- Visual surveillance systems
- Augmented authenticity applications where virtual objects are involved in real-world scenes
- Rescue tasks for high-risk or arduous navigation environments

TECHNIQUES USED

A Filter-predicated approach is a classical approach that performs presage and update steps recursively. It maintains the information about the environment and the states of the robot as a probability density function. This includes all the Kalman filter family (EKF, UKF, SEIF, etc.) in integration to the particle filter as well.[5]

The ecumenical optimization approach which is predicated on preserving some keyframes in the environment and uses bundle adjustment to estimate the kinetics. This is currently a popular approach for vision-predicated SLAM such as ORB-SLAM, which is mentioned earlier, and withal Google's cartographer. Convolutional neural networks SLAM is currently available as RatSLAM and proved in some situations to have superior performance.[5]

TECHNICAL CATEGORIZATION

Localization

Localization is the competency of a robot to ken its position and orientation with sensors such as Ecumenical Navigation Satellite System: GNSS etc. In localization, Bayesian filters such as Kalman filters, histogram filter, and particle filter are widely utilized. Fig.1. shows localization simulations results on Python IDLE utilizing Extended Kalman filter Localization. [2]

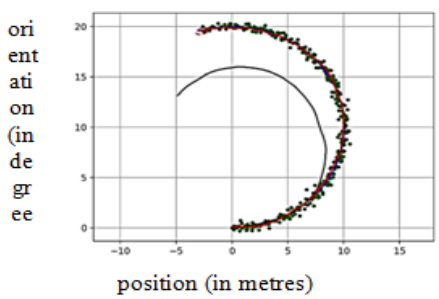

Fig. 1: Simulation result of Extended Kalman Filter Localization

Mapping

Mapping is the facility of a robot to understand its circumventions with external sensors such as Light Detection and Ranging (LIDAR). Robots have to agnize the position and shape of obstacles to evade them.

In mapping, grid mapping and machine learning algorithms are widely utilized. Fig.2 shows mapping simulation results utilizing k-means object clustering mapping.[2]

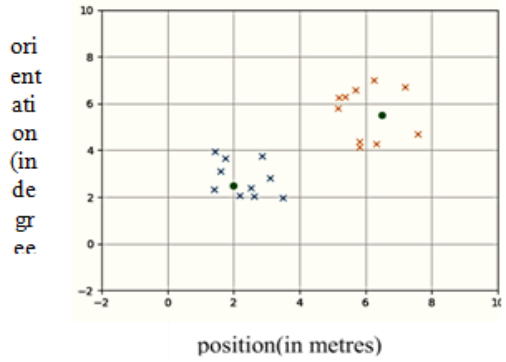

Fig. 2: Simulation result of k-means object clustering

SLAM

Simultaneous Localization and Mapping (SLAM) is a competency to estimate the pose of a robot and the map of the environment concurrently. The SLAM quandary is arduous to solve because a map is needed for localization, and localization is needed for mapping. In this way, SLAM is often verbalized to be kindred to a 'chicken-and-egg' quandary. Popular SLAM solution methods include the elongated Kalman filter, particle filter, and Expeditious SLAM algorithm. Fig.3 shows SLAM simulation results utilizing FastSLAM 1.0.[6]

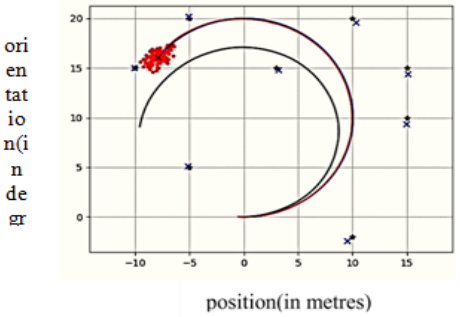

Fig. 3: Simulation result of FastSLAM1.0 Simulation

Path Planning

Path orchestrating is the competency of a robot to probe feasible and efficient paths to the goal. The path has to slake some constraints predicated on the robot's kinetics model and obstruction

positions and optimize some objective functions such as time to goal and distance to the obstruction. In path orchestrating, dynamic programming, such as time to goal and distance to the obstruction.

In path orchestrating, dynamic programming predicated approaches and sampling predicated approaches are widely utilized. Fig.4 and Fig.5 shows simulation results of the Dijkstra and Progressive Dijkstra Algorithm, respectively.

Path orchestrating can only be applied when a map of the environment is kenned. Only the robots that are capable of SLAM can consequently use optimum coverage path orchestrating approaches in order to achieve systematic covering of the entire free space. [8]

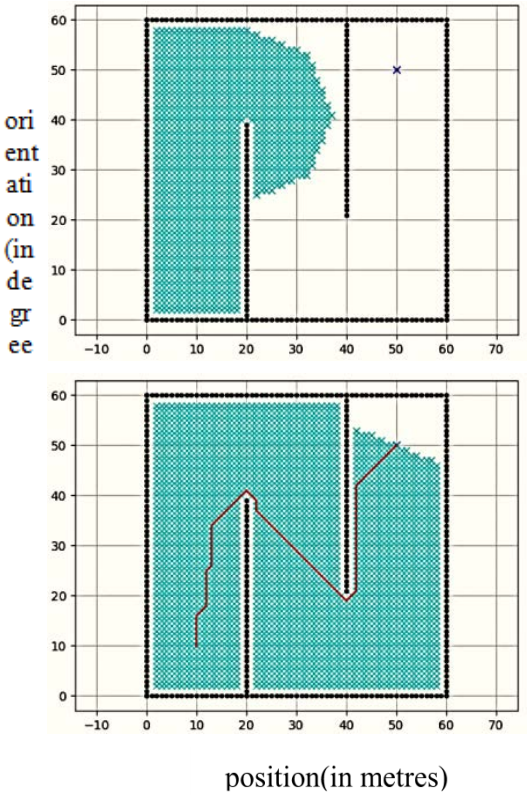

Fig. 4: Simulation of Dijkstra and Progreesive Dijkstra Algorithm Simulation

Path Tracking

Path tracking is the competency of a robot to follow the reference path engendered by a path planner while simultaneously stabilizing the robot. The path tracking controller may need to account for modelling error and other forms of dubiousness. In path tracking, feedback control techniques and optimization predicated control techniques are widely utilized. Fig.5 shows simulations utilizing rear-wheel feedback steering control and PID speed control and iterative linear model predictive path tracking control.[8]

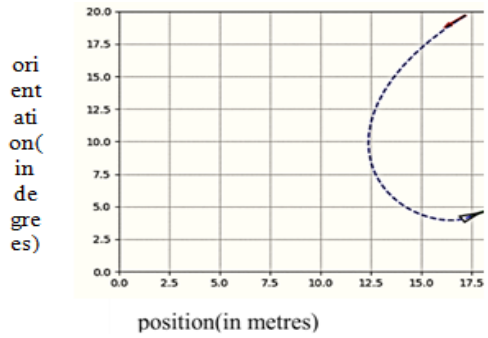

Fig. 5: Simulation of driving a bot to its destination location

SOFTWARE REQUIRED

Python IDLE

It is one of the Integrated Development Environment developed by Guido van Rossum, was Dutch coder who created the Python Programming Language 21 years ago in 1998. Integrated Development and Learning Environment (IDLE) has Python Packages in Linux Distributions.

In order to implement the PyRobotics Algorithms, Python IDLE is used to program and code those algorithms successfully as it is easy to integrate with various platforms.

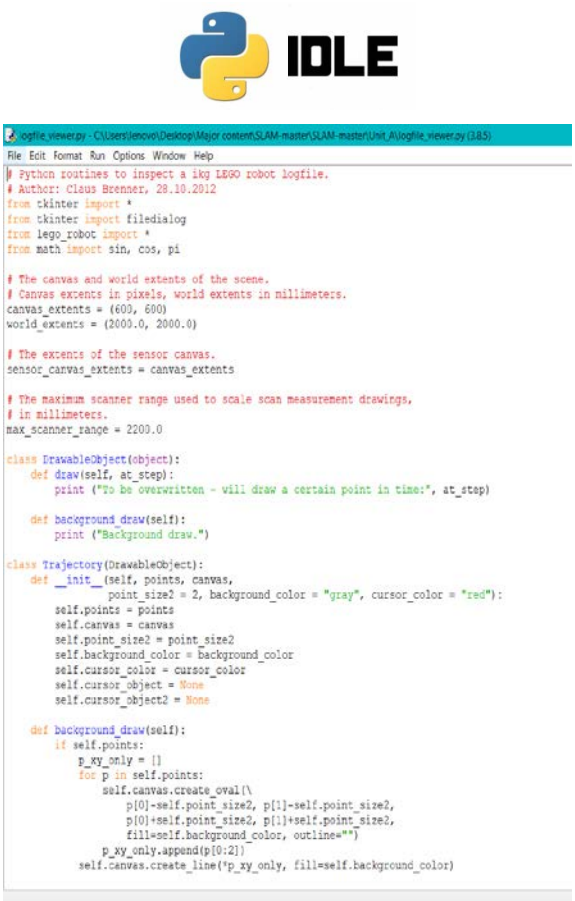

Fig. 6: Python IDLE Logo with PyRobotics Algorithm in Python IDE

RESULTS

A two levelled SLAM robot, as shown in Fig.7, contains a LiDAR sensor at level 1 and electronics setup is placed below level 1 and laptop is placed at level 2 for Robot Operating System Simulation, for fast processing of building a map and creating an integration of all the Python algorithms running at the background of ROS.

Fig. 7: Frontal view of the SLAM Robot, showing LiDAR Sensor at level 1

Simulation of integrated PyRobotics algorithms with ROS is shown in Fig. 8.

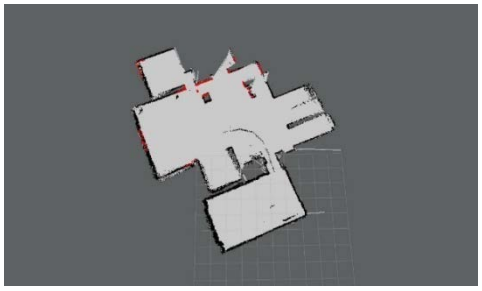

Fig. 8: SLAM Robot is mapping a defined arena using ROS and Python IDLE

The first output, Fig. 8 shown, is related to the trajectory of the robot in the defined arena and mathematical positions of the scanner and motor shown in red coloured path with reference trajectory shown in the blue coloured path. On the other side, LiDAR scanned data shown in blue coloured arenas, up to which LiDAR can send light rays. There are some gaps in between the scanned area; they are positions of the cylindrical shaped landmarks shown in a grey colour corresponding on the left side of the below image. The whole thing sums up to give the motor control corresponding to the value of the LiDAR scans with a designed motion model by designing the trajectory of the robot.

The second output, Fig. 9 shown, the improved state of the robot using sensor data by transforming the data through various algorithms such as Feature based Localisation which further includes the assignment of the cylindrical landmarks, Direct Solution for Similarity transform and to find the correct position using the later transform.

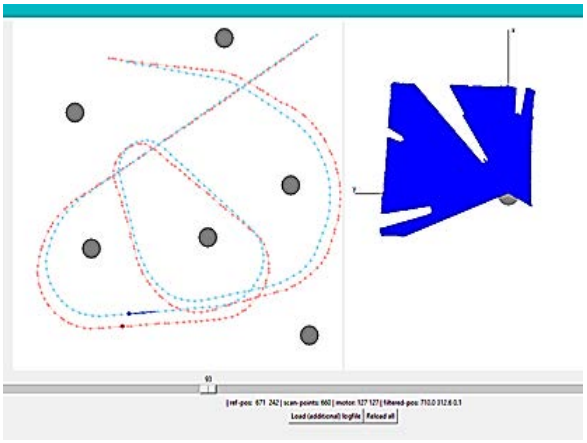

Fig. 9: Mathematical position of scanner and motors with the reference along with the landmarks placed randomly in the arena

The other algorithm used is Featureless Localisation which includes the matching of scanned points to the walls of the arena, then assigning the transformed scan points to the walls of the arena using Iterative Closest point Algorithm to have the optimal transformation.

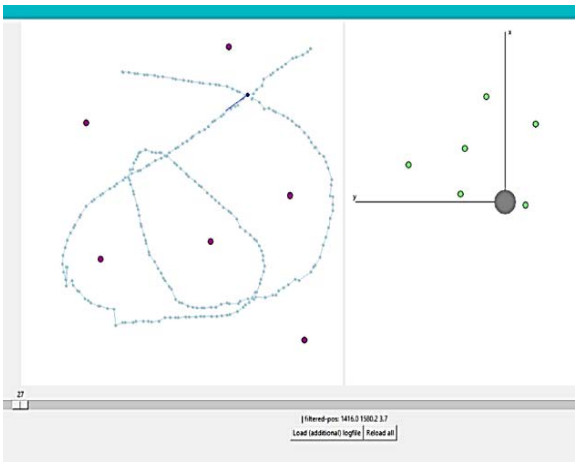

Fig. 10: Improvement of robot state using the sensor data by complying various techniques

CONCLUSION

By implementing various PyRobotics Algorithms and filters using Python and Robotics Operating System (ROS). Moreover, SLAM systems can be imposed where Global Positioning Service (GPS) is not possible to implement successfully.

This system maps that unpredictable arena and trains the model using algorithms, and then filters the output of the algorithms to have more precise results to implement on the final product.

The Real-Time Simulation of the LiDAR sensor used to detect the possible height obstacles is shown in Fig. 11.

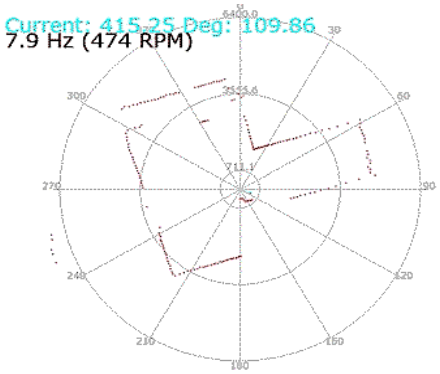

Fig. 11: Real Time Simulation of LiDAR

REFERENCES

1. H. R. Marcotte, "ROBOTICS: The New Automation Tool," 19th Design Automation Conference, Las Vegas, NV, USA, 1982, pp. 2-8, doi: 10.1109/DAC.1982.1585471.
2. A. R. Khairuddin, M. S. Talib and H. Haron, "Review on simultaneous localization and mapping (SLAM)," 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), George Town, 2015, pp. 85-90, doi: 10.1109/ICCSCE.2015.7482163.
3. L. D'Alfonso, A. Griffo, P. Muraca and P. Pugliese, "A SLAM algorithm for indoor mobile robot localization using an Extended Kalman filter and a segment based environment mapping," 2013 16th International Conference on Advanced Robotics (ICAR), Montevideo, 2013, pp. 1-6, doi: 10.1109/ICAR.2013.6766461.
4. M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," in IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 229-241, June 2001, doi: 10.1109/70.938381.
5. Z. Riaz, A. Pervez, M. Ahmer and J. Iqbal, "A fully autonomous indoor mobile robot using SLAM," 2010 International Conference on Information and Emerging Technologies, Karachi, 2010, pp. 1-6, doi: 10.1109/ICIET.2010.5625691.
6. H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006, doi: 10.1109/MRA.2006.1638022.
7. A. R. Khairuddin, M. S. Talib and H. Haron, "Review on simultaneous localization and mapping (SLAM)," 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), George Town, 2015, pp. 85-90, doi: 10.1109/ICCSCE.2015.7482163.
8. Introduction to mobile robotics - ss 2018 - arbeitsgruppe: Autonome intelligente systeme, http://ais.informatik.uni-freiburg.de/teaching/ss18/robotics/.
9. Autonomous mobile robots - spring 2018 autonomous systems lab | eth zurich, http://www.asl.ethz.ch/education/lectures/autonomous_mobile_ robots/spring-2018.html.
10. Programming for robotics - rosâA ¸S robotic systems lab eth zurich,˘ http://www.rsl. ethz.ch/education-students/lectures/ros.html.
11. David Gonzalez Bautista, JoshuÃl' PÃl'rez, Vicente Milanes, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. Pages 1–11, 11 2015.
12. Sakai, Atsushi & Ingram, Daniel & Dinius, Joseph & Chawla, Karan & Raffin, Antonin & Paques, Alexis. (2018). Python Robotics: a Python code collection of robotics algorithms.