IEEE NITK CHAPTER

PROJECT – ROS Simulation of Drone Flight

|  |  |
| --- | --- |
| Tip icon | **ABSTRACT**  *The project aims at implementing the Autonomous Drone Navigation Stack in ROS. Drone Navigation in 2D would be simulated using Gazebo and Rviz.*  *Project Members: 1. Shobuj Paul - IEEE, Diode 2. Pranshu Shukla - IEEE, Piston* |

# Section 1 – project overview

|  |  |
| --- | --- |
| Tip icon | *The proposal is divided into two sections namely – Project Overview and Literature Review. The first section will cover the motivation, objectives, methodology, timeline, budget and deliverables of the project. The second section contains the literature review regarding the different phases of the project.* |

## The Motivation

Autonomous Navigation has become a very critical aspect of robotics as the world moves into a world with minimum human interaction to attain tasks. Unlike ground robots, Drones face a bigger challenge in autonomous navigation for it has to account for a lot of stability as well as goal oriented behaviour to attain a certain task. This project aims to serve as a starting point for the development of complicated autonomous navigation systems to be deployed in drones. It also aims to explore various path planning algorithms as well as introducing the juniors into the robotics framework using ROS

## The Opportunity

Autonomous Drones and UAVs have a lot of potential for both application and development in the robotics and automation field. They can be used for a wide variety of applications such as delivery and surveillance and can also be implemented in already existing technology to make better supersonic airplanes and fighter jets.

Robotic Operating System (ROS) is a robotics middleware which can be used for simulation, control and package management for robotics with a wide variety of in-built tools. ROS is becoming extremely essential to anyone working in the field of robotics and automation.

These are topics which are dominating both research and industry, and it would be great exposure to juniors who will be new to engineering or anyone who wishes to work in the field of robotics in the future.

## The Objective

The objective would be to teach juniors about basics of robotics and introduce them to simulation.in ROS and have a working simulation of a drone navigating in 2D by the end of the project.

## Project Approach

The project approach will be divided into the following phases:

1. **Phase 1 - Getting to know the ROS ecosystem and the Github Environment**In this phase, the main objective will be to allow the juniors to get an idea of the ROS ecosystem. We’ll bring about various features of ROS and guide them on making a publisher and subscribers to do various operations in turtleSim. We will slowly introduce Gazebo, Rviz, launch files and URDF Modeling. Each individual will be asked to maintain everything on a github handle.
2. **Phase 2 - Introduction to AMCL and Autonomous Navigation using a pre-existing ground robot.**The aim of this phase will be to introduce and allow the juniors to explore autonomous navigation methodologies in ROS. We will introduce them to various algorithms and functions used and how to implement them. We will end this phase by assigning the juniors to move a bot in a given environment to a location by developing their own Gmap and Navigation Script.
3. **Phase 3 - Understanding Drones and Implementing 3D AMCL**As of this phase, we’ll bring forward drones and how to launch and fly them around in ROS. We’ll understand the ROS Teleops requirements and pose requirements on a drone. Finally we’ll move on to employing ACML on the drone.

## Project Deliverables

The following are the full list of deliverables which will be achieved through the entire project timeline:

|  |  |
| --- | --- |
| Deliverable | Description |
| **Deliverable #1 -** Proficiency in ROS | Student members, after this envision project, will be able to efficiently develop, code and utilize basic features of ROS effectively |
| **Deliverable #2 -**  Proficiency in Path Planning Algorithms | As a key part in robotics, it becomes very important for one to understand the working of different planning algorithms. This will strengthen their roots in Robotics. |
| **Deliverable #3 -**  Understanding various filters used in localization | This will key an insight on how localization occurs in robotics such as using kalman filters, Bayes filter or particle filter. |
| **Deliverable #4 -** Understanding Drone flight | Obviously as we are dealing with drones, the members will become proficient in developing and understanding the workings of a Drone |

We will be using this as a learning curve project for juniors to embed them into the world of robotics using hands-on experience. They will be able to grasp and understand most important concepts to a good extent by the end of this timeline.

## Timeline for Execution

Key project dates are outlined below. Dates are best-guess estimates and are subject to change until a contract is executed.

|  |  |
| --- | --- |
| Tip icon | *In the Table below, the dates and duration of the project are tentative and subjected to change as it depends on the availability of resources to accomplish some tasks.* |

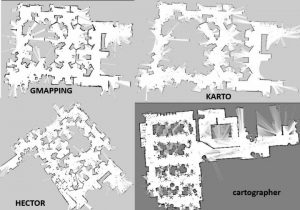
|  |  |  |  |
| --- | --- | --- | --- |
| Description | Start Date | End Date | Duration |
| Project Phase 1 | | | |
| Milestone 1: Installing and getting Started with ROS. Introducing to file structure and creating a publisher/subscriber. Also getting them started with Github and Ubuntu. | 15th February | 24th February | 10 Days |
| Milestone 2: Introducing Gazebo and Rviz. Explaining launch files and scripts. | 25th February | 6th March | 10 Days |
| Milestone 3: (Intermediate Project) Launching a given robot in Gazebo and Rviz. Trying to move it around using scripts. | 7th March | 16th March | 10 Days |
| Project Phase 2 | | | |
| Milestone 4: Introducing them theoretically to localization, AMCL and Path Planning operations. | 17th March | 26th March | 10 Days |
| Milestone 5: Implementing Gmapping and AMCL in ROS. Going through various parameters in ROS and getting them to understand how it works. | 27th March | 5th April | 10 Days |
| Milestone 6: (Intermediate Project) Implementing Gmapping and AMCL in ROS and sending goal commands using python scripts. Making sure bot reaches desired destinations | 6th April | 20th April | 15 Days |
| Project Phase 3 | | | |
| Milestone 5: Launching and working around with Drones and understanding their behaviours and commands. | 21th April | 30th April | 10 Days |
| Milestone 6: Looking into various 2D Mapping Algorithms. Implementing AMCL on Drones | 1st May | 15th May | 15 Days |
| Milestone 7: Writing scripts in Python to generate commands for drones to go to a certain location | 16th May | 30th May | 15 Days |
| Project End | | | |

## References

1. <https://www.youtube.com/watch?v=dND4oCMqmRs>
2. <https://www.youtube.com/watch?v=n6RjVbh3Vgc>
3. <https://www.youtube.com/watch?v=JZqVPgu0KIw>
4. <https://bitbucket.org/theconstructcore/hector_quadrotor_sim/src/master/>
5. <http://wiki.ros.org/depthimage_to_laserscan>

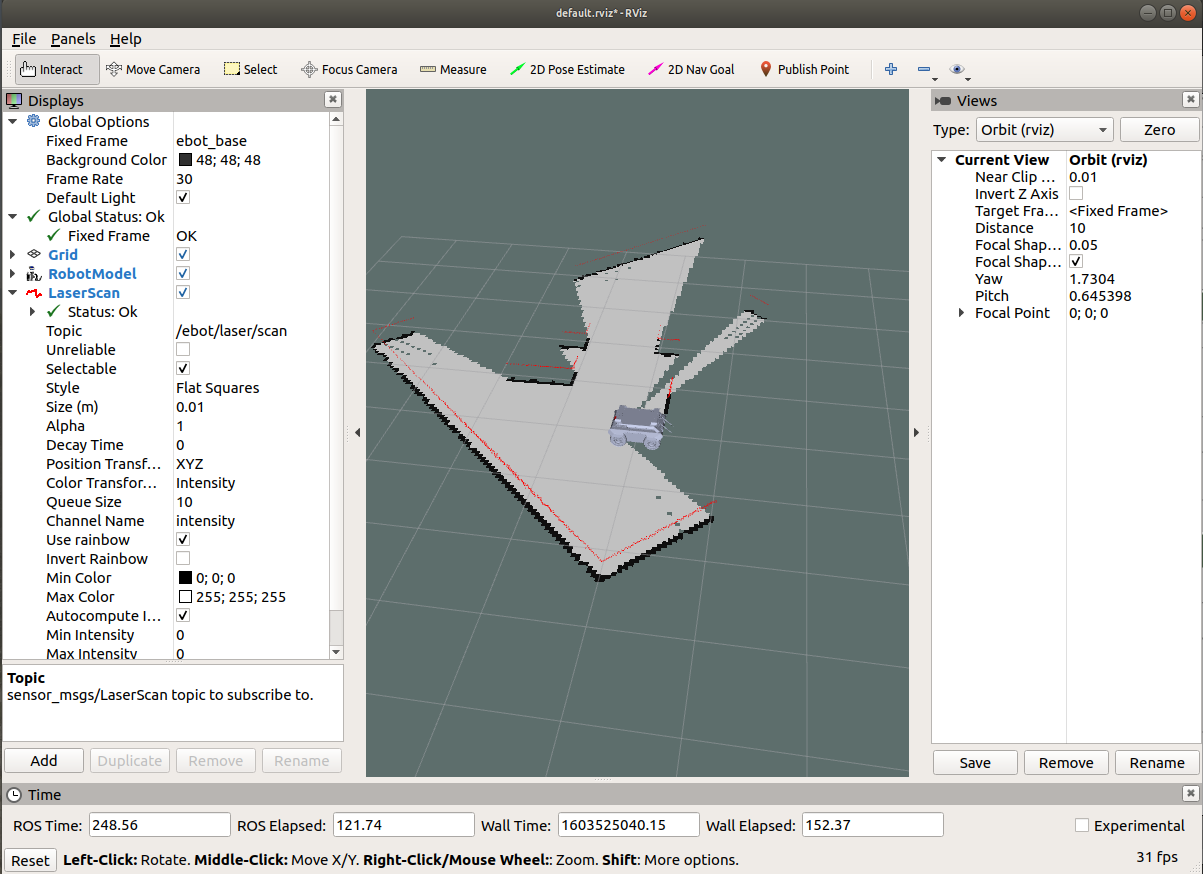
# SECTION 2 – LITERATURE REVIEW

## SLAM

A map is a representation of the environment where the robot is operating. It should contain enough information to accomplish a task of interest.In order to create a map it is necessary to merge the measurements from previous positions and also keep a track of current measurements and the pose of the bot as they are changing.

To overcome this problem techniques called Simultaneous Localization and Mapping(SLAM) are introduced. It estimates the map of the environment and the trajectory of a moving device using a sequence of sensor measurements.Some prominent techniques include: Gmapping, Hector mapping, Cartogrpaher, Rtab mapping etc.

**Gmapping**

The gmapping is a laser-based Simultaneous Localization and Mapping algorithm.It allows one to create a 2-D occupancy grid map from lidar/laser and pose data collected by a mobile robot. 

The working of Gmapping is as explained: First, it takes the measurement from odometry and laser scan and tries to localize the bot along with the laser scan matching and then using Extended Kalman Filter (EKF) it estimates the output by fusing odometry and laser scan matching values. Along with this, it uses a particle filter for localization.

**Localization**

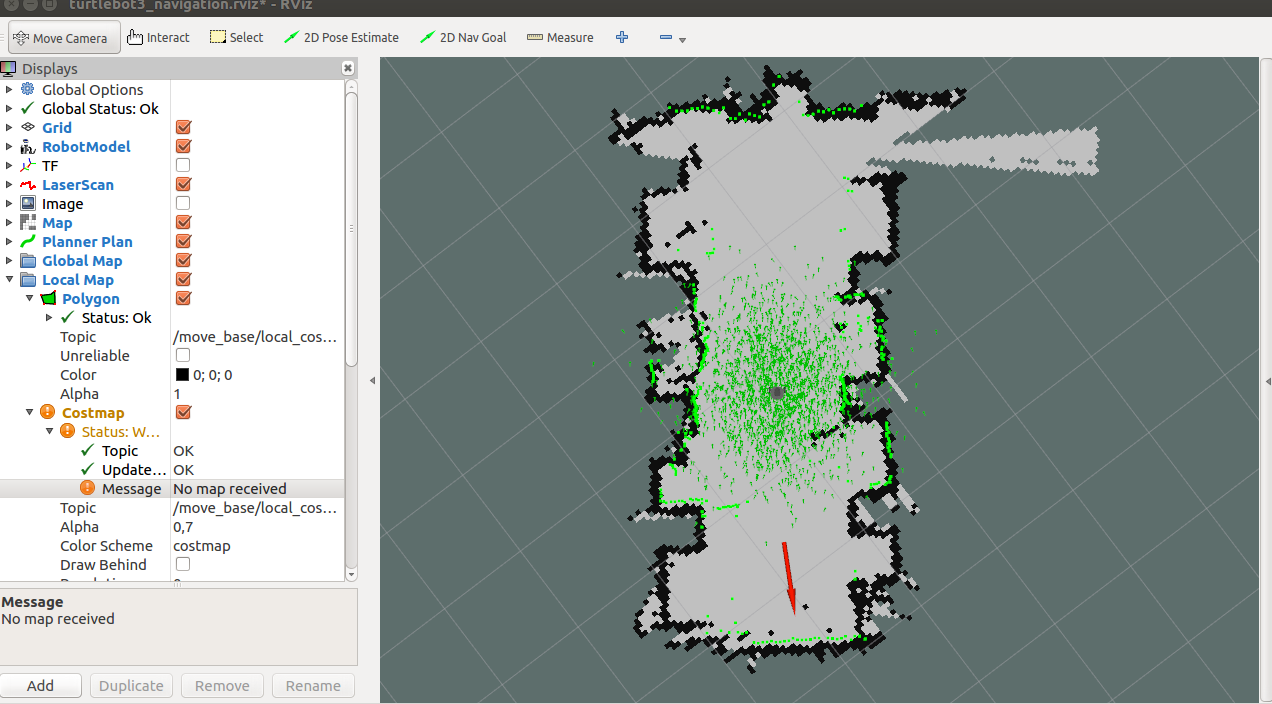
Robot localization is the process of determining where a mobile robot is located with respect to its environment. Localization involves estimating position and orientation of vehicle i.e. the pose of the robot as it moves and senses the environment. One way of knowing the position of the robot is tracking from its initial position. By measuring the wheel odometry reading of the robot we can calculate the distance travelled from the initial position and predict with certainty the cars pose on the map.

The method of odometry measurement is not accurate. Some of the issues that we see in localization include mistakes committed by odometry and lack of solutions to compensate for lack of initial position.

One very notable solution to this issue is the Monte Carlo Localization which uses particle filters. In our project, we will be using the Adaptive Monte Carlo Localization technique to localize using ROS in the next section.

A few other alternative options include Kalman Filter, Topological Markov Localization

**Adaptive Monte Carlo Localization (AMCL)**

Amcl is a probabilistic localization system for a robot moving in 2D. Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves and senses the environment. The algorithm uses a known map of the environment, range sensor data, and odometry sensor data. To localize the robot, the MCL algorithm uses a particle filter to estimate its position. The particles represent the distribution of the likely states for the robot. Each particle represents the state of the robot.

At start the robot has a map of a room and the probability of location and heading are randomly assigned using discrete particles(or poses). Each of this particle is saved in the filter as the possible estimate of the state. The filter can go through each of the particles and compare it with what the Lidar sensor output would have returned if that pose was actual one. Some of the wrong guesses are removed.

Now, It will compare the scene seen from the particle pose to the actual robot pose, the closer the scenes match higher the probability. Higher probability particles are given more weights. The particles are re-sampled by removing low probability particles. As the robot moves the estimated motion is applied to every particle predicting each of the poses whether it is the same as the current position of the robot.

Again step 2 is repeated and the particles get concentrated with very few distributions and aligns the actual robot pose. The algorithm recalculates the number of particles required for a new batch after each generation of poses, as distributions become narrow. Sampling method called KLD (Kullback–Leibler divergence)generates particles based on the difference in odometry reading and estimated pose of particles .(i.e. smaller sample size when particles get converged) hence called Adaptive Monte Carlo Localization.