Autonomous Navigation with Collision Avoidance using ROS

Akash U[1*], Ananya Naik[1], Sandeep S[1], Deepti Raj[2]
Student[1], Assistant Professor[2]
Department of Telecommunication, Dayananda Sagar College of Engineering, Bangalore, Karnataka, India
Email: *akashu97@gmail.com
DOI: http://doi.org/10.5281/zenodo.2670059

Abstract
Simultaneous navigation and mapping is a modern mapping technique. The aim of SLAM is to develop 2D environment of a location while tracking the robot's position. This paper aims to develop ROS enabled robot with SLAM features in order to avoid collisions and navigate autonomously. A world is simulated using Gazebo and visualized using a tool called Rviz. Autonomous navigation is achieved by mapping the environment and plotting the odometry. Particle filtering is the algorithm on which SLAM works. This helps in using the odometry values to find the probable path for the robot to move whilst avoiding collision.

Keywords: *Simultaneous Navigation and Mapping, ROS, Gazebo, Rviz, Odometry, Particle filtering.*

INTRODUCTION
In this present world where the population is growing day by day and an increasing faster aging society, the older people of the society are facing a number of difficulties in various phases of their lives. The older people will have difficulties in movement of their limbs, hands and various nerve related issues. In order to help the aging society potential partial replacement is required which would serve as an assistant to the existing system which is composed of doctors, physiotherapist, etc. Various researches have been made in order to develop potential assistant in the form of robotic guides or robots. This robotic guides or robots they not only help aged people but also they are boon for physically challenged people [1].

The aim of SLAM is to develop 2D environment of a location while tracking the robot's position. As the name suggest it is nothing but a combination of localization and mapping problem, several algorithms exists to solve this problem, at least approximately in certain period of time under certain environment. Popular approximate solutions include particle filter, extended Kalman filter and Graph SLAM.

SLAM
It was first introduced by R.C. Smith and P. Cheeseman in 1986. Further it was improved by research group of Hugh F. Durrant-Whyte in the early 1990's they showed that solution to SLAM problems exist in the infinite data limit.

Robot is placed in an unknown environment and it doesn't have any knowledge of its current position, the robot performs the SLAM for estimating the position of landmarks and also its own location by moving along the unknown environment. The success of SLAM is determined by the fact that more the robot explores the environment the better will be the quality of the map and its current position [2].

GMapping Tool
This tool is used for the implementation of SLAM algorithm. It is one of the most well established laser scanner based SLAM algorithm that was proposed by Grisetti 2006. This algorithm is based on

RBPF as an effective means to solve SLAM problems and used easily in ROS. GMapping was shown to be robust in their experiments when compared to other 2D SLAM approaches. It needs odometry information for position and heading direction estimation. Odometry can be improved by using the left and right wheel encoder equipped on robot and accuracy will be improved by using a single access Gyroscope [3].

Robot Operating System (ROS)
It is robotics middleware (i.e. collection of software frame works for robot software development). All though ROS is not an operating system, it provides services designed for heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message passing between processes and package management. ROS based processes are represented in the form of graph architecture where as processing takes place in nodes that may receive post, control, state, planning, actuator and other messages. ROS itself is not a real-time OS, though it is possible to integrate ROS with real-time code [4].

GAZEBO
Gazebo is a 3D Simulator which can efficiently simulate complex environments. The main components of a Gazebo are models, GZ Sever, GZ Client and GZ Web. Gazebo helps in visualizing parameters such as Inertia, Forces and Sensor Information. It is a robust engine with programmable graphic Interface. Features of Gazebo are, dynamics simulation, advance 3D graphics, built in robot models, cloud simulation, command line tools and TCP IP transport [5].

RVIZ
RVIZ is installed using ROS dep package. RVIZ provides visualization of the current configuration on a virtual model of the robot. It can display live sensor values such as camera data and other odometric measurements. It also has troubleshooting tools and can run on a virtual machine [6].

METHODOLOGY
The idea proposed in this paper consists of the following steps:

Creation of the robot world.
The simulator used here is Gazebo where a world is created using components provided such as walls, shapes and other objects. Roslaunch command is used for any package linked with ROS. The reference bot used here is a Turtlebot and the world is depicted.Using this map the robot plots the odometry of the map.

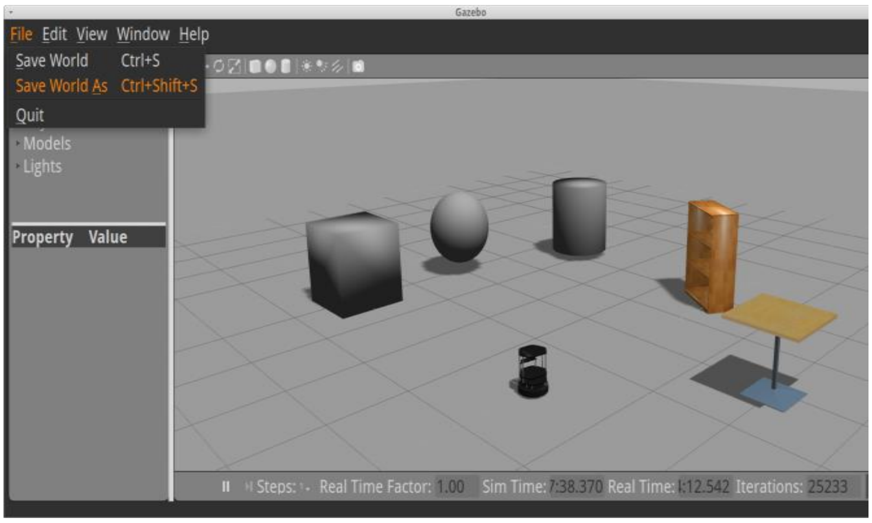

Gazebo
File Edit View Window Help
Save World Ctrl+S
Save World As Ctrl+Shift+S
Quit
Models
Lights
Property Value
Steps: 1 Real Time Factor: 1.00 Sim Time: 7:38.370 Real Time: 1:12.542 Iterations: 25233

Figure 1: *Robot World*

Updating the odometry values.
Using the world shown above the robot plots the odometry parameters so that a global plan can be made to reach the destination. These values are fed to the robot's navigation stack [7].

Running the navigation stack.
The navigation stack of the robot consists of URDF codes for each joint of the robot. These codes are initiated after the odometry values reach the navigation stack. The cmd_vel package is responsible for the motion of the robots autonomously.

Flow Chart

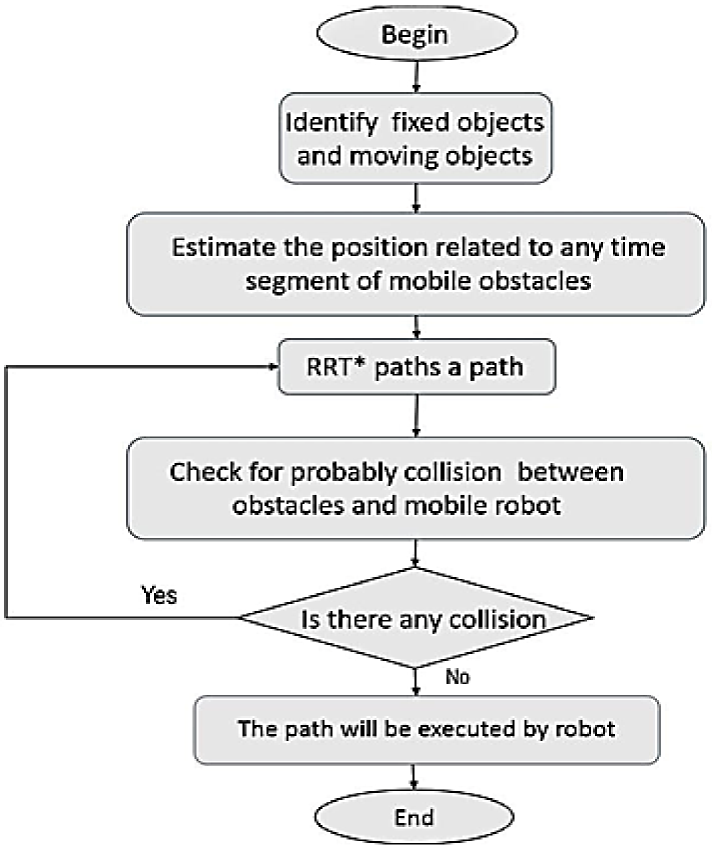
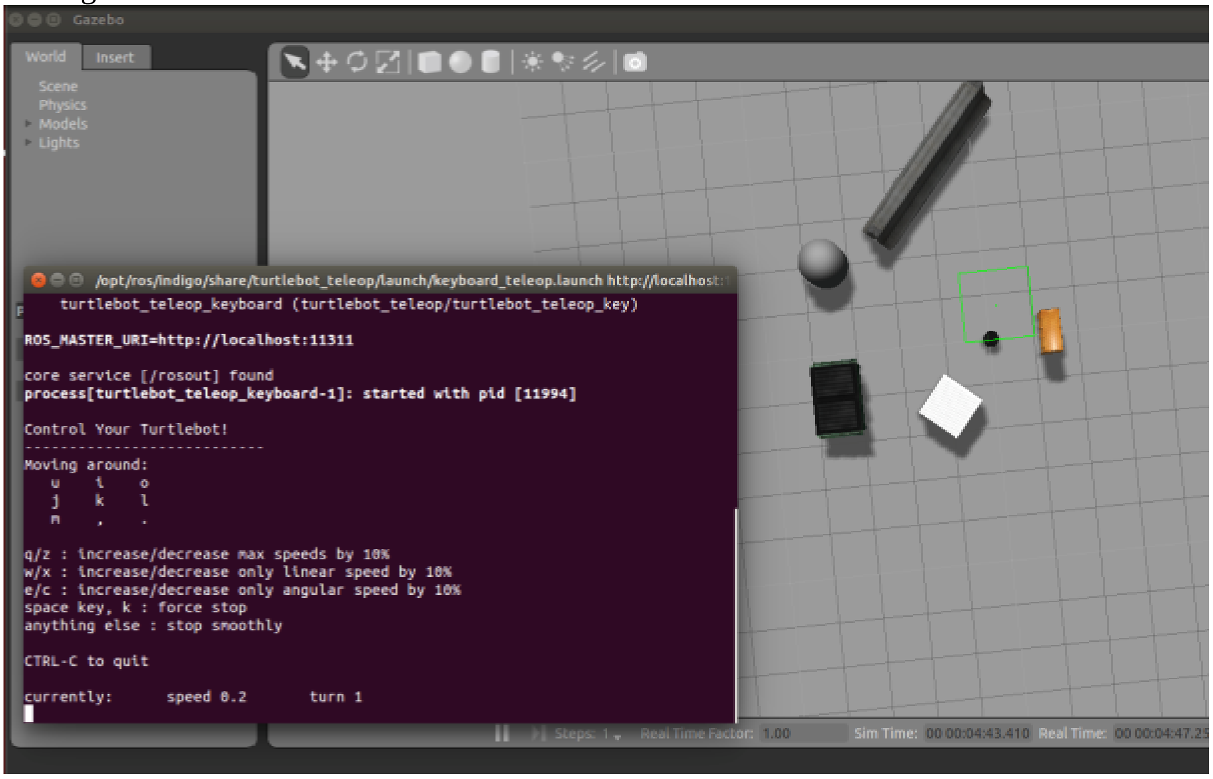

Begin
Identify fixed objects and moving objects
Estimate the position related to any time segment of mobile obstacles
RRT* paths a path
Check for probably collision between obstacles and mobile robot
Yes
Is there any collision
No
The path will be executed by robot
End

Figure 2: *Flow chart of robot avoiding collisions and navigating autonomously*

The above figure shows flow chart of autonomous navigation and collision avoidance. Robot uses Simultaneous Localization and Mapping technology and creates a 2D image of the environment, the robot will be specified its destination and once the destination is provided, robot will estimate the position of fixed objects and dynamic objects. RRT (Rapidly Exploring Random Tree) will generate a best possible path to the provided destination, RRt is an algorithm which is used to provide the best possible path to the destination by building a space filling tree and the tree is constructed from the samples that are drawn randomly from search spaces. RRT have been widely used in autonomous robot motion planning because of its ability to easily handle problems related to obstacles, and once the path is created robot will check probability of collision between the obstacle and itself and moves accordingly. If there is any collision detected then RRT algorithm will create the best possible path to reach the destination and the robot will travel according to that path and reaches its destination. If there is no collision detected then the robot uses the same path to reach its destination [8].

RESULTS

The figure below shows simulation in Gazebo



```
/opt/ros/indigo/share/turtlebot_teleop/launch/keyboard_teleop.launch http://localhost:
    turtlebot_teleop_keyboard (turtlebot_teleop/turtlebot_teleop_key)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[turtlebot_teleop_keyboard-1]: started with pid [11994]

Control Your Turtlebot!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1
```

Figure 3: *Simulation in gazebo*

The figure below shows simulation of robot in gazebo wherein the initial trail was to navigate the robot with the help of teleop package. The teleop package was coded in URDF in order to move the robot without simultaneous navigation and mapping. The aim was to calibrate the wheels of the robot. Figure below shows simulation using Rviz:

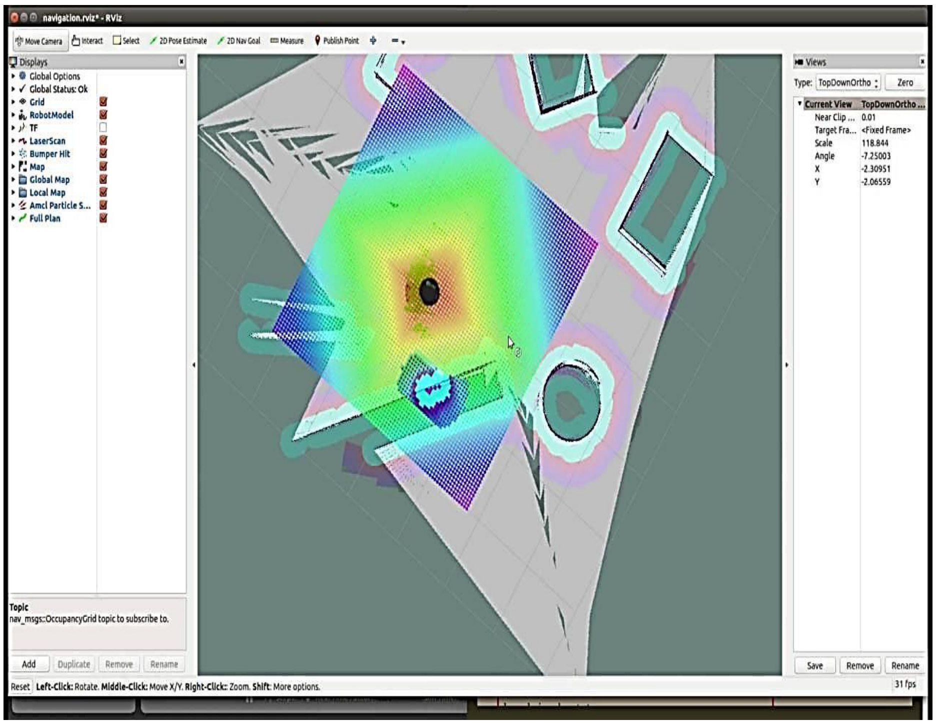


Figure 4: *Simulation using Rviz*

The map co-ordinates are fed to the Rviz simulator using the amcl package. The robot was given a global path using the 2D navigation goal feature in Rviz. Using the given global plan the robot uses local planner which is basically particle filtering to find the best probable path to achieve the global plan. The multicolored radius of the robot does not coincide with the object edge thereby enabling the robot to avoid collisions. The parameters achieved from the Rviz simulations are as follows:

- max_vel_x: 0.22 m/s.
- min_vel_x: 0.02 m/s.
- max_rotational_vel: 2.84 rad/sec.
- Map update interval: 2.0 – 20.0.

These parameters meet the reference bot's default configuration showing precision of the simulation [9-10].

CONCLUSION AND FUTURE SCOPE

Previous research focused on using the concept of SLAM on robots. There were some disadvantages regarding usage of LIDAR. The concept of localization and mapping was being scaled at various operations but the work here focuses on using SLAM on ROS to autonomously navigate and simultaneously avoid any collisions. Furthermore, the objective is to improve this method by embedding it with an arm to lift objects using image recognition which can be of great help to the blind.

ACKNOWLEDGEMENT

We would like to express our gratitude to our project guide Mrs. Deepti Raj, Assistant Professor, DSCE, Bangalore for her constant support throughout this endeavor.

REFERNCES

1. Nupur Choudhury; Rupesh Mandal "Human Robot Interaction Using Android and Point Bug Algorithm". Published in 2016 under IEEE.
2. Manuel Gonzalez Ocando, Novel Certad, Said Alvarado and Angel Terrones; "Autonomous 2D SLAM and 3D Mapping Of an Environment Using a Single 2D LIDAR and ROS". in 2017.
3. Adrian Lendinez Ibanez, Renxi Qiu and Dayou Li; "Implementation of SLAM Using ROS and Arduino". Published in 2017 on IEEE conference paper.
4. Joao Machado Santos, David Portugal, Rui Pedro Rocha;"An evaluation of 2D SLAM Techniques available in Robot Operating System" Published in 2013 on IEEE journal.
5. Seokju Lee, Girma Tewolde, Jongil Lim and Jaerock Kwon; "2D SLAM solution for low cost mobile robot based on embedded single board computer "; Published in 1st September 2017.
6. Ankur Bhargava, Anjani Kumar ;"Arduino controlled robotic arm". in 2017.
7. Chien-Wei Chen, Rui-Ming Hong, Hung-Yu Wang;"Design of a controlled robotic arm" in 2017.
8. "Robotics motion planning and navigation bug algorithm"Website:https://www.cs.cmu.edu"
9. Making of a robotic arm. Website: https://www.electronicshub.org/robotic-arm/.
10. Slam on ROS on Wikipedia .Website: http://wiki.ros.org/gmapping.