TARGET BASED NAVIGATION USING ROS NAVIGATION STACK

PRESENTED BY:-

HITA GANGINENI

HIMAJA KESARI

NADHIYA GANESAN

UDHEEP GHANTA

INTRODUCTION

ROS Navigation Stack:

* This methodology is used to run the robot from an origin point to destination safely by navigating it through obstacle-free path using sensor information.

* We use the concept of navigation stack in this project and explore unknown environment with the use of **'GMAPPING'.** Through the exploration of GMAPPING, we create a map from the explored path for the robot to navigate through the environment.

* The 'Turtlebot3 Waffle Pi' robot is simulated using the Gazebo simulator.

* Algorithm being used for the implementation of the autonomous exploration is Rapidly Exploring Random Tree(RRT) Algorithm.

TECHNOLOGIES USED

- ROS Navigation Stack
- Gazebo
- rrt_exploration
- TurtleBot3 robot
- Gmapping
- RVIZ

SYSTEM IMPLEMENTATION

- We used the RRT Algorithm to autonomously explore the environment.
- The implementation of RRT Algorithm was from rrt_exploration package from ROS Navigation Stack.
- Our Implementation has three major steps:-
    - Setting the robot in the gazebo house environment
    - Generate a map for the unknown environment
    - Use a path planner to reach a target using the map generated.

PHASE1 : PLACE THE TURTLEBOT3 ROBOT IN HOUSE ENVIRONMENT

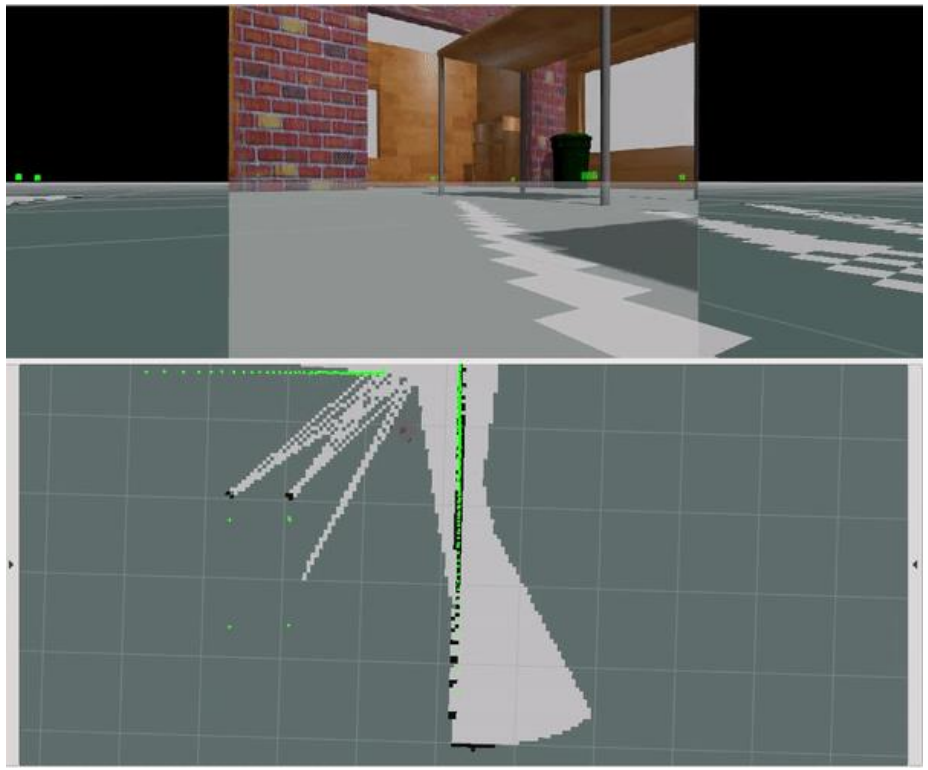
- We set the environment variable to the robot model to be used, which is waffle_pi in our project.
- We use the following commands to set our robot as waffle_pi
- Execute the given launch to open Gazebo with the given world file and place the robot Turtlebot3 Waffle pi model in it.
    - export TURTLEBOT3_MODEL=waffle_pi
      source ~/.bashrc
    - roslaunch ros_autonomous_slam turtlebot3_world.launch (ros_autonomous_slam is the name of our package).
    - We keep this process running and execute the other phase's commands in a different terminal.

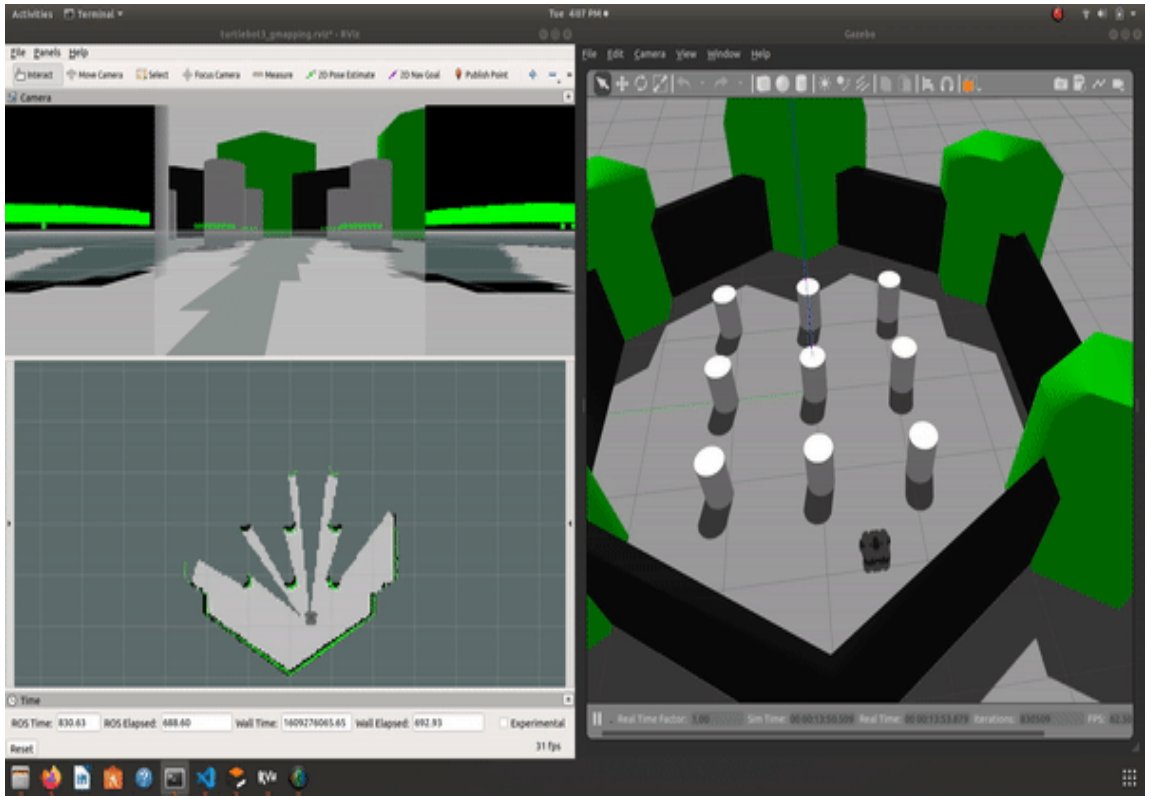
PHASE 2: GENERATING MAP OF OUR ENVIRONMENT

- We generate the map of a unknown environment autonomously without any human intervention
- We start a autonomous explorer which is a python based controller to move the robot exploring all the areas which help the slam node to complete the mapping.
- The algorithm we use for autonomous exploration is RRT(Rapidly exploring Random Trees).
- A Rapidly-exploring Random Tree (RRT) is a data structure and algorithm that is designed for efficiently searching nonconvex high-dimensional spaces.
- RRTs are particularly suited for path planning problems that involve obstacles.

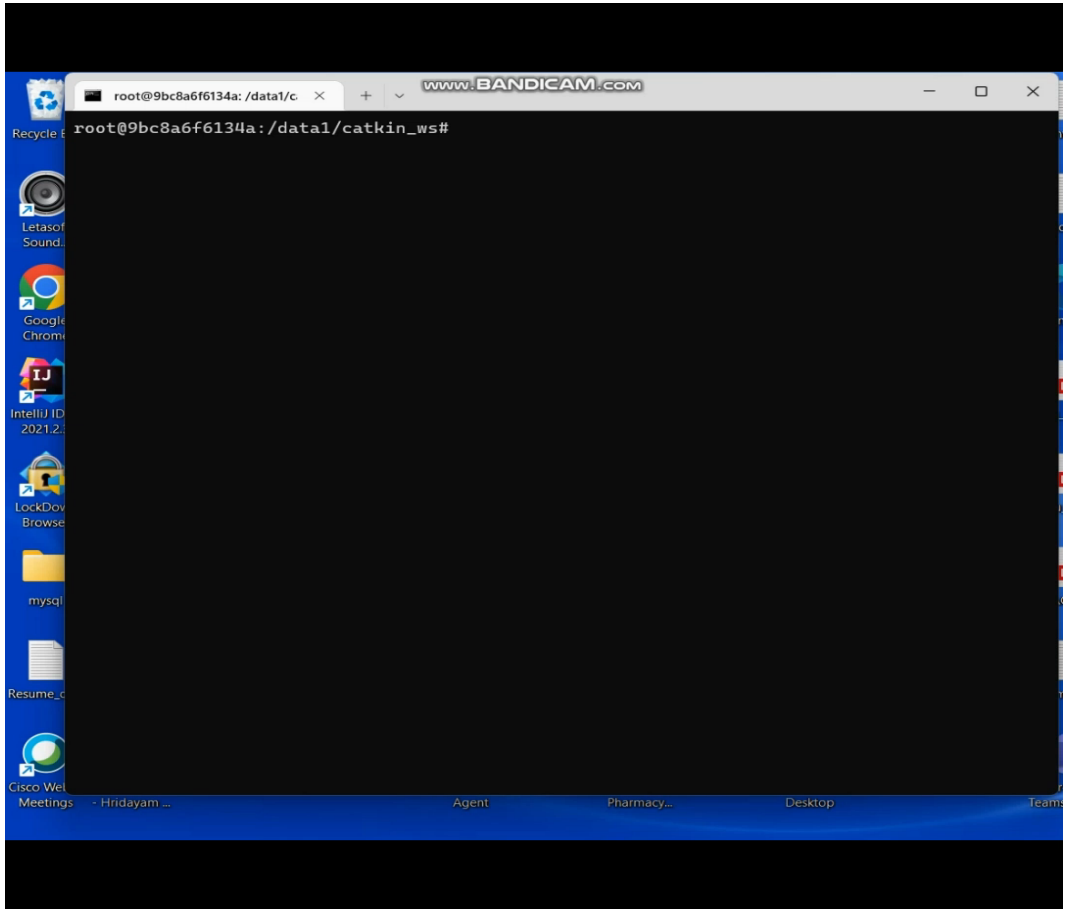
RRT WORKING

DEMO- MAP GENERATION





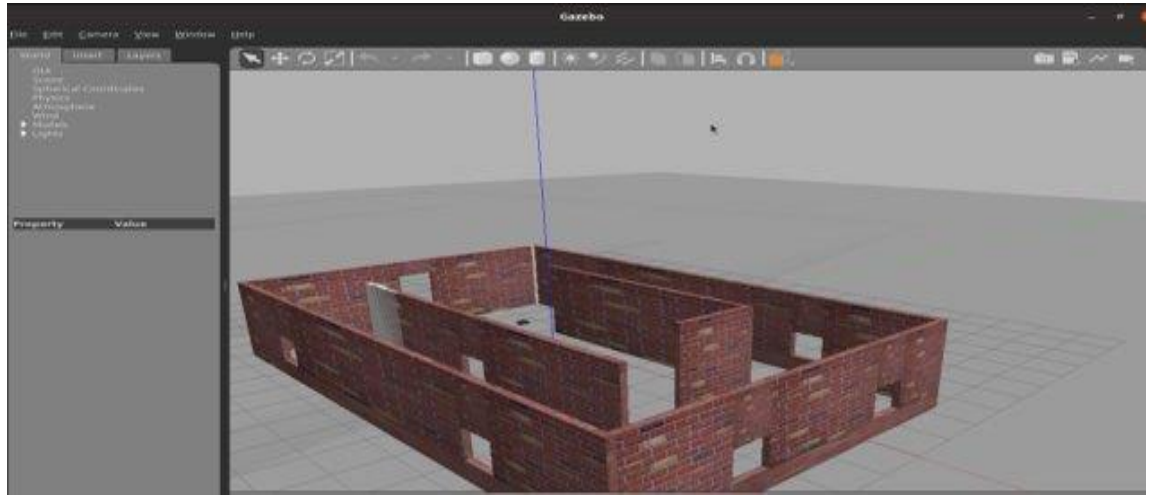
DEMO


root@9bc8a6f6134a: /data1/c
www.BANDICAM.com
root@9bc8a6f6134a:/data1/catkin_ws#
Recycle
Letasof
Sound...
Google
Chrom
IntelliJ ID
2021.2
LockDov
Browse
mysql
Resume_c
Cisco We
Meetings
- Hridayam ...
Agent
Pharmacy...
Desktop
Teams

NAVIGATION

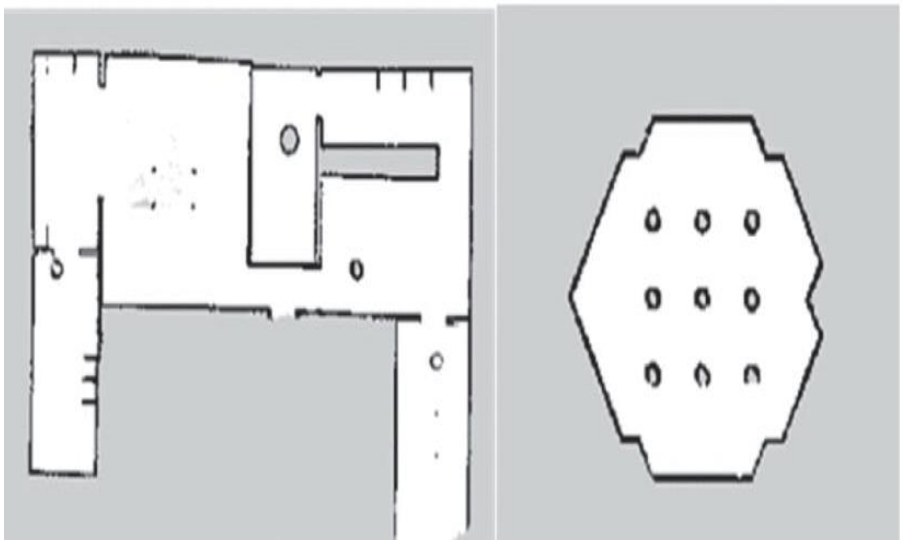
EXPERIMENTS AND RESULTS

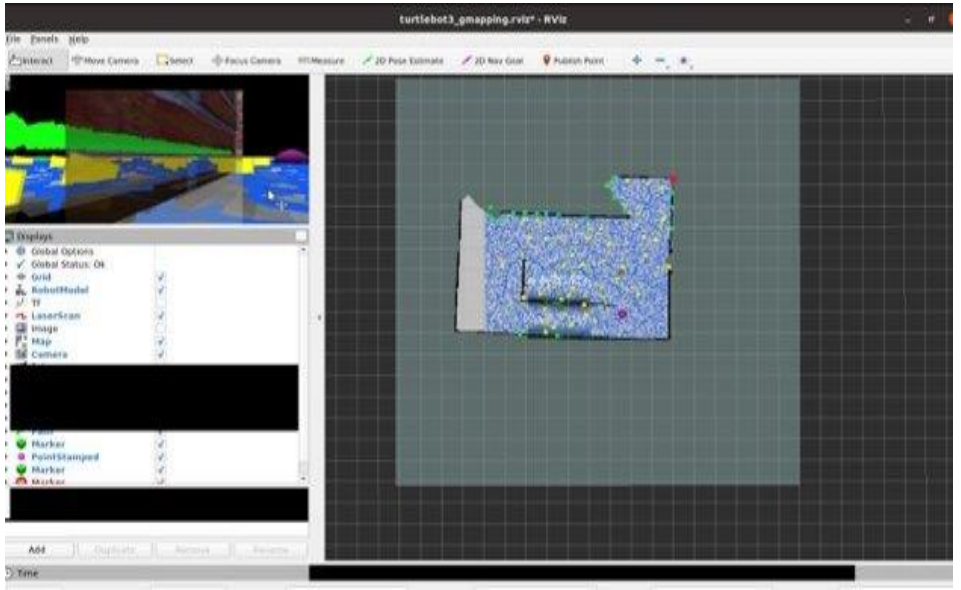
We performed SLAM (Simulataneous Localisation and Mapping) using RRT algorithm in different environments .We placed the turtlebot3 robot in the environments and it explored unknown environment and generated the map .



EXPERIMENTATIONS AND RESULTS

•Once we generated the map of the environment using GMapping ,we saved it as yaml file.
•We verified the map similar to the simulation environment .
• Generate map set as the reference frame and we specified 2D pose estimate of the robot and the target to perform goal based navigation using in-built ROS navigation stack.





EXPERIMENTS AND RESULTS

- To check the efficiency of the RRT algorithm ,criteria such as the map quality during mapping ,time taken for the robot to navigate and get to its destination for each environment ,were recorded .The time taken was recorded with five test runs and the average value was evaluated .

The average search time for the RRT algorithm is 52.44 secs
The average path length is 1490
Average iteration is 1661.28

1st trial	2nd trial	3rd trail
51.35	133.77	123.17
51.83	161.95	132.56
52.60	133.83	132.86
51.81	152.11	142.75
57.60	154.65	143.00

Trial runs of environment without obstacles

1st trail	2nd trail	3rd trail
19.13	19.79	17.34
17.02	18.54	14.68
14.66	16.99	13.11
19.99	17.97	14.06
17.86	25.29	12.70

Trial runs of environment with obstacles

FUTURE WORK

- We plan to implement more optimized path planning algorithms in different environments to find the shortest path between the goal and the target including obstacles.
- We can tune various parameters of ROS Navigation Stack which works differently for different environments to get the optimal path planning performance.
- A more efficient algorithm is RRT* which we plan to implement in the future.

THANK YOU