ANIKET CHANDRAKANT DEVLE (1220351418)
Perception in Robotics - Project_3b (Pursuit Evasion Game)

Introduction: This project dealt with creation of a map using SLAM, Autonomous navigation using turtle bot and its visualization using Rviz for GUI, taking input from the camera on waffle pi turtle bot and detecting the person who is trying to evade, following the evading man. If used correctly we can use this object detection for mundane activities to even in space research.

TASK 1: SLAM with GMapping in ROS

SLAM stands for simultaneous localization and mapping. SLAM is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of a bot location within it. Initially the bot doesn't know anything about its surroundings but it keeps updating the map of the surrounding obstacles after receiving the input from the camera. We use the Gmapping package of the SLAM for making the map of the surrounding. I used turtlebot3_teleop to move the robot in the map, and hence created the map in the process. Following are images of the maps obtained.

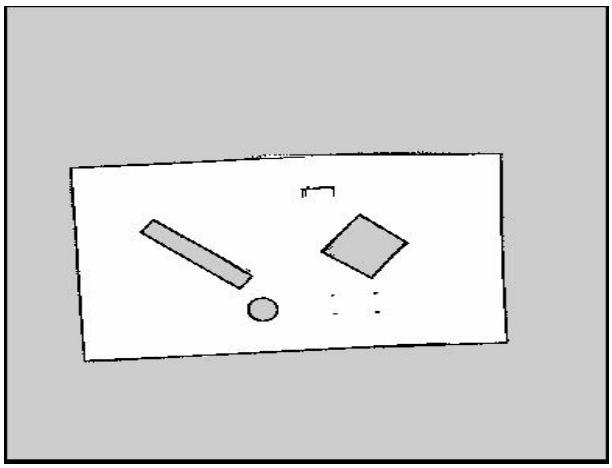

Fig : Map world 1

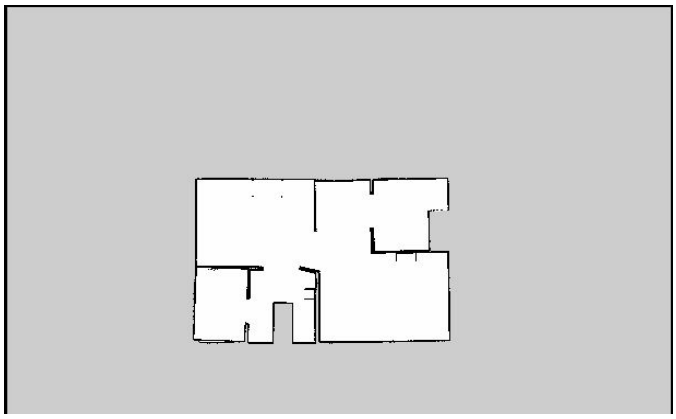

Fig Map world 2

Link for Video: https://drive.google.com/file/d/12xE9iiVbUdlV2L6Cu77NPo-wo-axmsrR/view?usp=sharing

TASK 2:  Autonomous Navigation

Autonomous navigation refers to a vehicle's ability to plan and execute its route without the need for human interference. In some cases, remote navigation aids are used in the planning phase, while in others, the only data available to compute a direction comes from sensors on board the vehicle. This is very useful when we are dealing in outer space. Sending the exact path is very difficult when the communication with the bot itself takes minutes. In that case the bot figures out the path and follow it once it has the map that it produced with the Gmapping stage.

Turtlebot3_navigation:

This package provides the roslaunch scripts for starting the navigation. It has 5 different dependencies viz amcl, catkin , map_server , move_base , turtlebot3_bringup. Amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive Monte Carlo localization approach , which uses a particle filter to track the pose of a robot against a known map. Catkin, Low-level build system macros and infrastructure for ROS. map_server provides the map_server ROS Node, which offers map data as a ROS service. It also provides the map_saver command-line utility, which allows dynamically generated maps to be saved to file. Move_base : The move_base package provides an implementation of an action  that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. turtlebot3_bringup : roslaunch scripts for starting the TurtleBot3

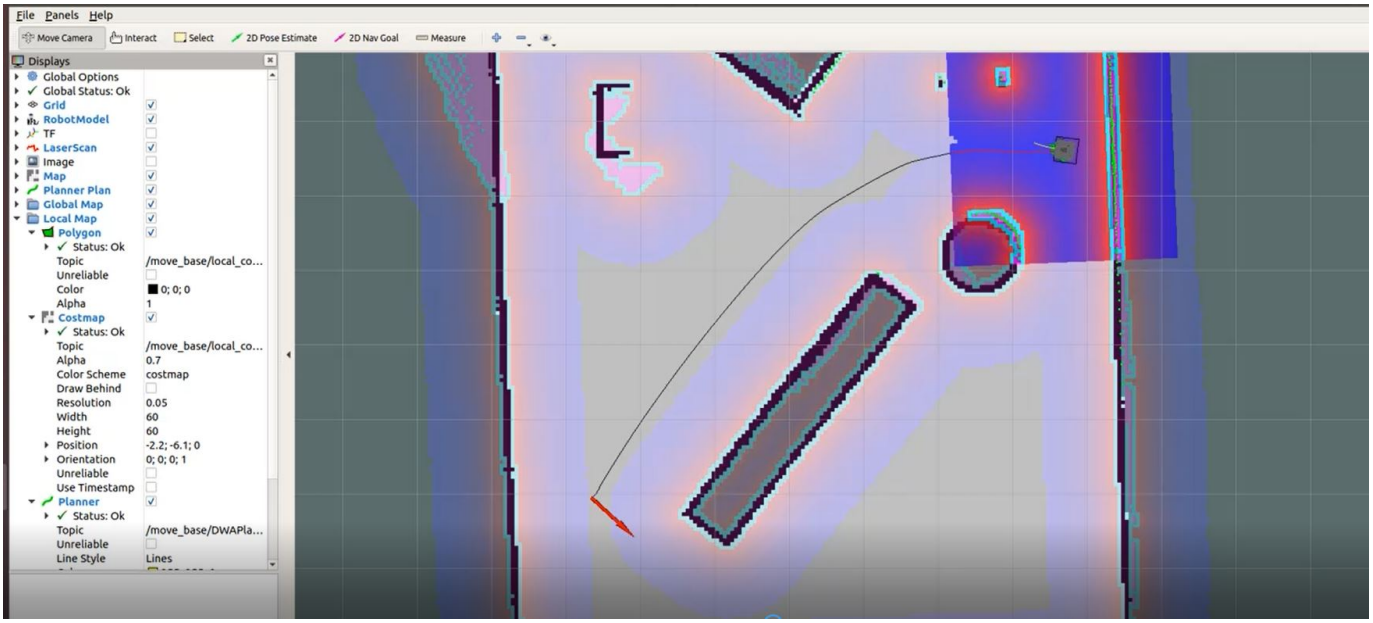


Fig: Task 2

Link to task 2 video : https://drive.google.com/file/d/1ZF6an1EB3VIj4ow_aKTzfdS2aDqY-mvN/view?usp=sharing

Task 3 and 4:

I have used the Yolov3 package for object detection. Yolo is an acronym for 'You Only Look Once'. Yolo looks at the whole image as the test time so its predictions are informed by global context in the image. It also makes predictions with a single network unlike systems like R-CNN which require thousands for a single image.

We pass the image from the turtlebot3 camera as input for the object detection algorithm. But the image from the camera is not in the format acceptable by CV2 so we use cv_bridge class to convert the image from turtlebot . The class id for humans in the YoloV3 model is 0. There is a chance that multiple boxes with class id 0 are returned so we only consider the one which has the maximum confidence.

Then I measured the angle of the person by taking the center of the detected bounding box and using the given FOV. On every detection, I produced a target of a fixed distance towards the person, which I published to the goal subject.

Link to final video: https://drive.google.com/file/d/116N5KJFx11dNRcCosvPr-0Kpm-GDrNza/view?usp=sharing

Running the code:

1. After extracting add both the folder to the catkin_ws
2. Change the path of the maps in yaml file to the appropriate path in that particular system.
3. Change the path of MODEL_PATH_WEIGHTS and MODEL_PATH_CFG in the Project_3b.py file to the path of the file in that particular system.
4. I have also made some changes in the launch file so you will have to use different command for directly launching a particular map( I have made two different launch files for the same)
5. For map1 : roslaunch pursuit_evasion robot_amcl_map1.launch world_index:=0
6. For map2: roslaunch pursuit_evasion robot_amcl_map2.launch world_index:=1
7. For moving the evader run: roslaunch pursuit_evasion move_evader.launch world_index:= [index] -> index for map1 and map2 are 0 and 1 respectively.
8. The run the python file by: rosrun pursuit_evasion Project_3b.py