CSE 598 Perception in Robotics Project 3b: Pursuit Evasion Game

Rohan Sharma

April 28, 2021

1 Task 1: SLAM with GMapping in ROS

After downloading the pursuit evasion module,I started with creation of map of the environment. Firstly entered the following command

```
roslaunch pursuit_evasion robot_mapping.launch world_index:=[index]
gui:=[true/false]
```

where **index =0/1** which means 0 will launch the first world while 1 will launch the 2 second world. The **gui =True/False** signifies weather gazebo would be switched on or not. I used the approach of gmapping as my slam algorithm for mapping the whole environment. After opening the desired world, **RVIZ** also pops up for verification and mapping. Next command using teleoperations was set to move the robot for creation of the map.

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

The gmapping slam creates a binary map in which the white area is the non obstacle zone and the robot and move around this area. The black map is the boundary of the obstacle. The gray area signifies that the area is still uncovered and robot is unaware of the area. The Turtlebot waffle was moved around the whole world. After creation of the map, the map is saved using the following command

```
rosrun map_server map_saver -f  /map
```

The following ,aps were obtained with their **.pgm and .yaml** extensions

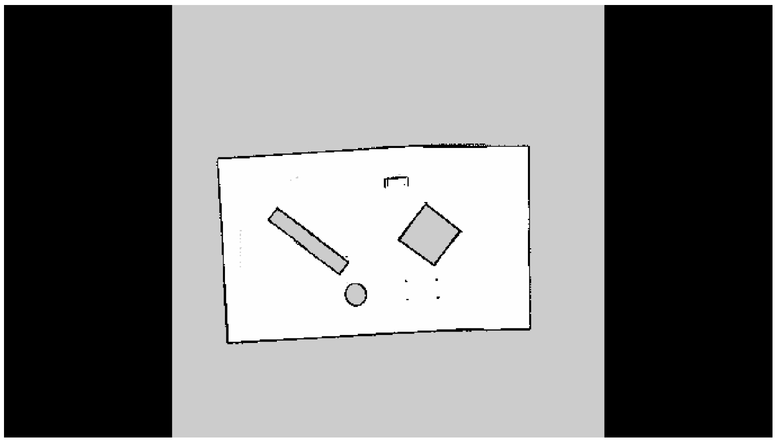


Figure 1: Map of world 0

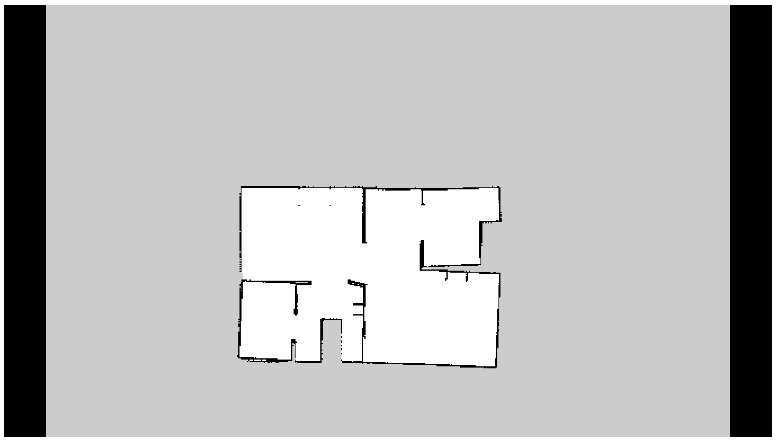


Figure 2: Map of world 1

2 Task 2: Autonomous Navigation

After creation of the map using the above task, moving the waffle with the RVIZ gui was simple. For autonomous navigation both the meathod of move and turtlebot navigation was used. Move base automatically moves the robot. In the other part, I used the in build turtlebot navigation stack for moving the robot.

```
roslaunch pursuit_evasion sim.launch world_index:=[index]
gui:=[true/false]
```

for move_base navigation a launch file was already made which is launched using the roslaunch command and furthur giving the Goto location. I also used the turtlebot navigation which gave the following result.

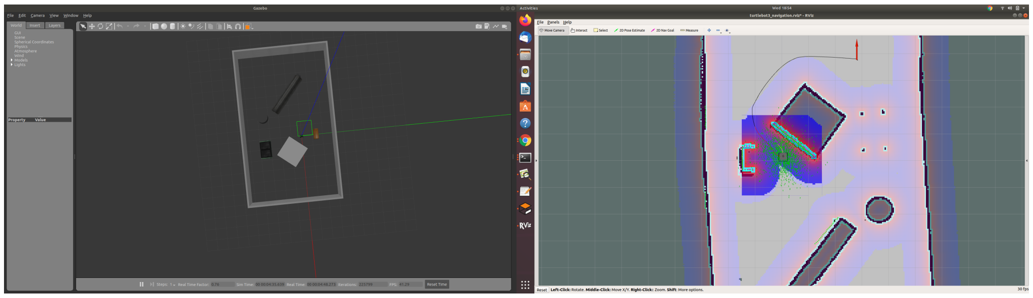


Figure 3: Using 2D Nav Goal

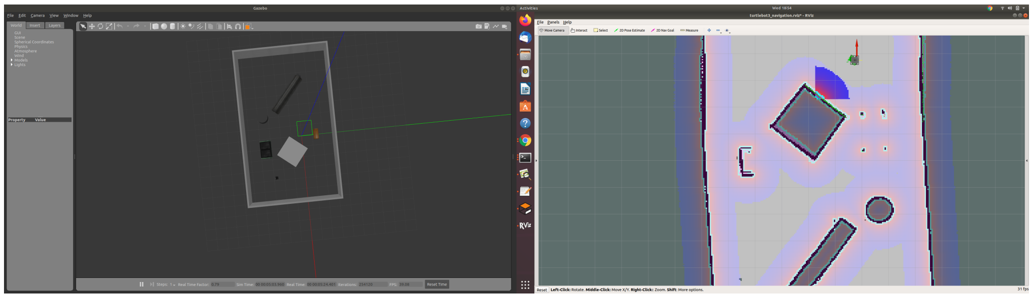


Figure 4: Reaching goal location

3 Task 3: Tracking Node

After running the below command, the acml package pops up. With that a separate node was written that subscribes to the topic **/tb3_1/camera/rgb/image_raw**. This gives the live feed of what the turtle bot is seeing.For tracking and detection hog detector was used, which helps in detecting persons. An example is shown below where turtle bot identifies the legs of a person and makes a bounding box around it. The code (Hyperlinked) was used for coding the HOG detector.

```
roslaunch pursuit_evasion robot_amcl.launch map_file:=\home \
rohan18 \ map.yaml world_index := 0
```

Cv_bridge was used for converting the video feed into the image and cv2 readable formats

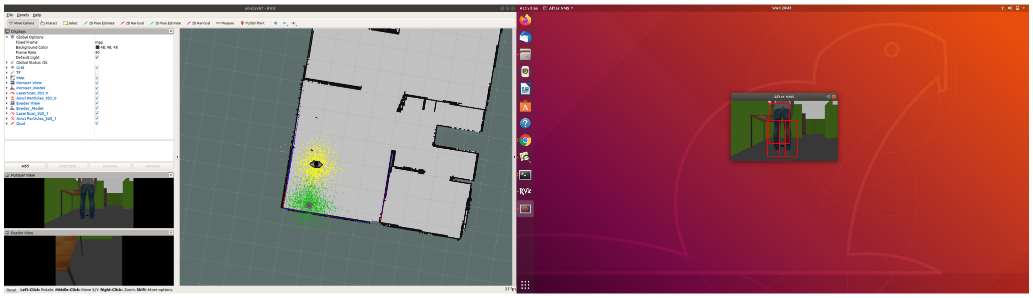
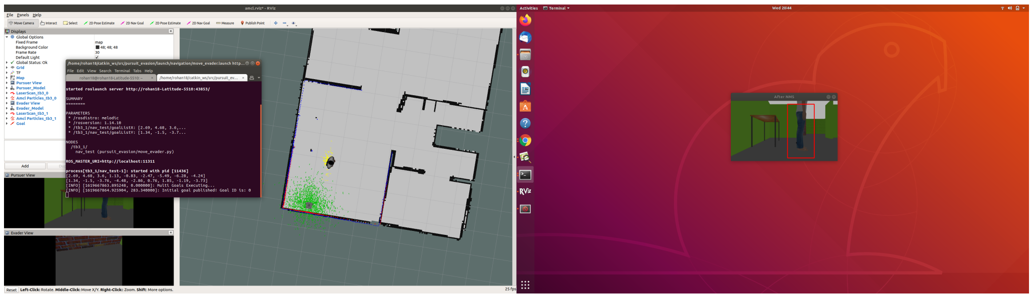
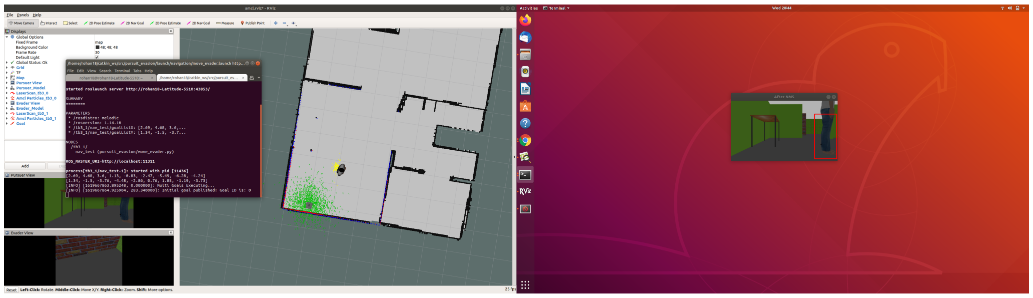


Figure 5: Hog Image detector without Supression



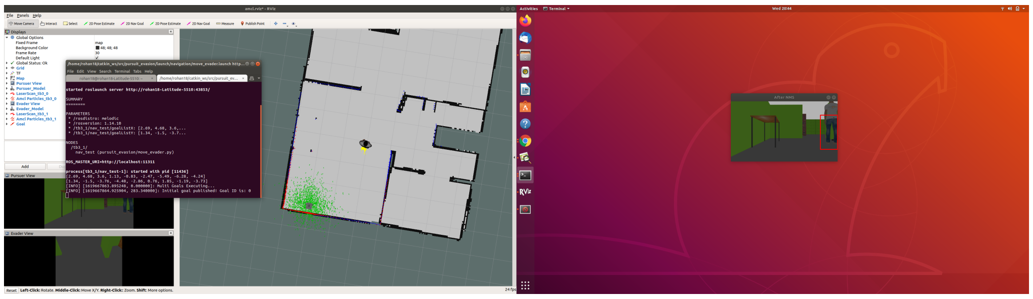
```
started roslaunch server http://rohan18-Latitude-5510:43853/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.10
 * /tb3_1/nav_test/goalListX: [2.69, 4.68, 3.6,...
 * /tb3_1/nav_test/goalListY: [1.34, -1.5, -3.7...

NODES
  /tb3_1/
    nav_test (pursuit_evasion/move_evader.py)

ROS_MASTER_URI=http://localhost:11311

process[tb3_1/nav_test-1]: started with pid [11436]
[2.69, 4.68, 3.6, 1.13, -0.83, -2.47, -5.49, -6.28, -4.24]
[1.34, -1.5, -3.76, -4.48, -2.86, 0.76, 1.85, -1.19, -3.73]
[INFO] [1619667863.895248, 0.000000]: Multi Goals Executing...
[INFO] [1619667864.925904, 283.340000]: Initial goal published! Goal ID is: 0
```

Figure 6: General motion 1



```
started roslaunch server http://rohan18-Latitude-5510:43853/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.10
 * /tb3_1/nav_test/goalListX: [2.69, 4.68, 3.6,...
 * /tb3_1/nav_test/goalListY: [1.34, -1.5, -3.7...

NODES
  /tb3_1/
    nav_test (pursuit_evasion/move_evader.py)

ROS_MASTER_URI=http://localhost:11311

process[tb3_1/nav_test-1]: started with pid [11436]
[2.69, 4.68, 3.6, 1.13, -0.83, -2.47, -5.49, -6.28, -4.24]
[1.34, -1.5, -3.76, -4.48, -2.86, 0.76, 1.85, -1.19, -3.73]
[INFO] [1619667863.895248, 0.000000]: Multi Goals Executing...
[INFO] [1619667864.925904, 283.340000]: Initial goal published! Goal ID is: 0
```

Figure 7: General motion 2



```
started roslaunch server http://rohan18-Latitude-5510:43853/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.10
 * /tb3_1/nav_test/goalListX: [2.69, 4.68, 3.6,...
 * /tb3_1/nav_test/goalListY: [1.34, -1.5, -3.7...

NODES
  /tb3_1/
    nav_test (pursuit_evasion/move_evader.py)

ROS_MASTER_URI=http://localhost:11311

process[tb3_1/nav_test-1]: started with pid [11436]
[2.69, 4.68, 3.6, 1.13, -0.83, -2.47, -5.49, -6.28, -4.24]
[1.34, -1.5, -3.76, -4.48, -2.86, 0.76, 1.85, -1.19, -3.73]
[INFO] [1619667863.895248, 0.000000]: Multi Goals Executing...
[INFO] [1619667864.925904, 283.340000]: Initial goal published! Goal ID is: 0
```

Figure 8: General motion 3

4 Task 4: Control Node for Pursuit

In this task we were asked to move the turtlebot using according to the evader. I used the technique where the robot moves towards when it sees an evader. Eventually rotates in anticlockwise direction when no evader is found. Since Hog detector has trouble detecting the real person, there were some faulty detection which moved the evader. I tried using the pose calculation for estimating the pose but was unable to. The evader was moved using the following command

```
roslaunch pursuit_evasion move_evader.launch
```

After that the tracking algorithm was run and the track was partially successful for both the maps.