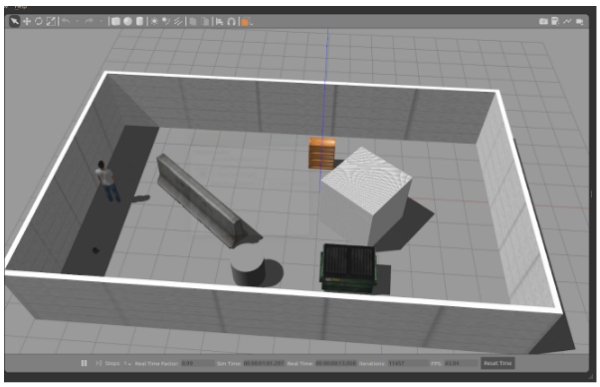
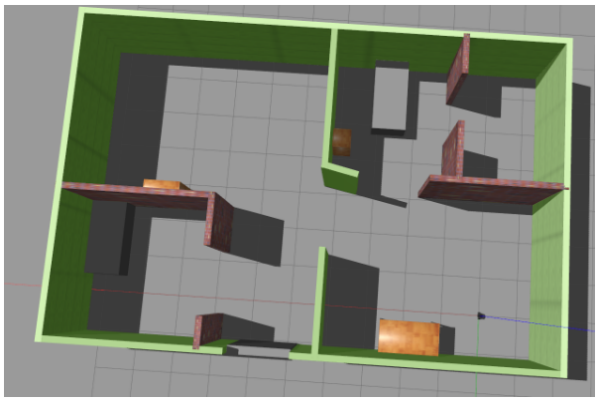
CSE 598 Perception in Robotics Project 3b: Pursuit Evasion Game

Varun Jammula <vjammula@asu.edu>, Duo Lu <duolu@asu.edu>,
Yezhou Yang <yz.yang@asu.edu>

This project is due on April 23rd, 2020. It is the second part of project 3. In this part, students will learn to use SLAM using the gmapping package in ROS to create maps of the simulated world and use these maps for navigation tasks as well as to avoid obstacles during navigation. These methods can also be used to map real-world environments. After creating a map that can assist navigation for both pursuer and evader, we move to the actual task of pursuit-evasion. Turtlebot3 acts as a Pursuer and the human model as an Evader. Teams will create a node for tracking the evader and then use the bounding box, localization information to send goals to the pursuer to track the evader.



(a) Environment-1 (world_index=0)



(b) Environment-2(world_index=1)

Note that this is more like an open end project. It is designed to allow students to gain hands-on experience of a whole robotic system containing perception, decision making, and action. There are no strict rules on deliverables, though a rough set of deliverables are specified at the end of this document. Instead, you are supposed to build the system and demo that it is working. The grading is effort based and emphasizes on the demo.

Resources:

Please see the project resources in the following link.

- https://drive.google.com/open?id=1lna2TeNoV35sM6bL-mQVRg6Mp9maJvBz
- https://piazza.com/class/kjw0dktqq7c6vj?cid=69

1. pursuit_evasion
   This ROS package contains launch files that you can use for this project.

2. Human_description
   This ROS package contains required files for the evader. This evader model is equipped with a depth camera, LIDAR and a planar plugin to move in the world.

Pre-requisites:

- Install turtlebot3 packages
- sudo apt install ros-melodic-slam-gmapping (required for mapping)

Task 1: SLAM with GMapping in ROS

Please see the instructions in the following links

1. http://wiki.ros.org/gmapping
2. https://emanual.robotis.com/docs/en/platform/turtlebot3/slam_simulation/

Although "gmapping" is recommended here, you may use any SLAM package readily available for turtlebot3.

You can use the tutorial provided in (2) to create a map of the world. Note that we use a different world than the one provided in the tutorial. You can either choose to use 1 or more turtlebots to create the map but make sure you follow the appropriate steps. If you choose to use multiple turtlebots, you need to use map merging to combine the individual maps from the robots. After the mapping is done, be sure to save the created map on your machine. For your convenience, our TA has provided a launch file that loads the environment and the gmapping module.

Step (1): Launch gazebo simulator along with the world file provided in the “worlds” folder in the "pursuit_evasion" package.

Step (2): Use the "turtlebot3_slam" package to launch the gmapping node to start the process of creating a map.

Step (3): Next, use the "turtlebot3_teleop" package to run the teleop node to move the robot around the map to create a map.

Step (4): After visualizing in "rviz" that mapping is complete, use the "map_server" package to save the created map to the maps folder in pursuit_evasion package.

A launch file has been provided in the "pursuit_evasion" package that combines Step (1) and Step (2) if you wish to use it. You can also set the world_index to either 0 or 1 to select the environment you want to map. You can also set the argument for GUI to enable to disable it using the following parameters at launch. If you wish to enable, Gazebo GUI will be enabled and you can see the simulator.

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

The meaning of the parameters are as follows.

- world_index : 0 - using environment 1; 1 - using environment 2
- gui: true - enables gazebo view; false - disables gazebo view

For example,

```
roslaunch pursuit_evasion robot_mapping.launch world_index:=0 gui:=true
```

Rough Deliverables: For this task, add the maps you created to the maps folder provided in the pursuit_evasion package when submitting.

Task 2: Autonomous Navigation

Please see the instructions in the following link.

http://wiki.ros.org/move_base

Once you have a map of the environment, it is easy to navigate in the known environment using the "move_base" package, which is a part of the ROS navigation stack. For more information, please refer to the ROS package link above and the turtlebot manual.

https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/

https://emanual.robotis.com/docs/en/platform/turtlebot3/slam_simulation/

The "turtlebot3_navigation" package provides an abstract way to load the map you created above and to use it with the "move_base" package to send goal locations to turtlebot3 using

RViz. This step will help you understand how the navigation stack works and how to autonomously move in a known map while avoiding obstacles.

Step (1): Launch the gazebo environment with a world of your choice. A launch file for this step is in the provided resources and you can use the following command to launch the packages.

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

Step (2): Using the turtlebot3_navigation package to load the map you created earlier.

Step (3): The above step will launch rviz, to visualize the created map and provide goal locations using the RVIZ-GUI.

For step (1), do not use the launch file that was provided for Task 1 because it launches the gmapping package which is not needed for this purpose.

Please go through the "turtlebot3_navigation" package and document how the map is loaded, what other packages are being used internally and attach a screenshot of the RVIZ with the world in the report.

Rough Deliverables: For this task, add a description in the report on how "turtlebot3_navigation.launch" file works. For example, what package is being used to load the map that you created, what other nodes are used for localization, moving the turtlebot autonomously without collision(hint: navigation stack) etc.

In the first two tasks, you learnt how to map a given environment and to navigate autonomously by avoiding obstacles using GUI in RVIZ. For the next two tasks, we load the environment with both pursuer and evader models and to learn to send navigation goals for the pursuer using a ROS node.

Task 3: Tracking Node

For this task, we provide a launch file that loads the Turtlebot3 and a human model equipped with a camera and LIDAR sensor that can utilize the map that was created in Task (1). Once you start the launch file, the map is loaded. Each model uses a different namespace to avoid the ambiguity in the topics they publish and subscribe to. For the pursuing turtlebot, it's namespace will be **tb3_0** and the evading human model will have the namespace **tb3_1.** So for example, if you want to subscribe to the image topic of the evader your node will subscribe to "/tb3_1/camera/rgb/image_raw".

In this task, you can use the camera that is attached to Turtlebot in the previous phase to detect and track a person, given an image. Students can use any method they would like to perform

person detection and tracking. The major hurdle here, however, is that ROS produces messages that might not be compatible to use directly with the detection algorithms. Hence, we use the "cv_bridge" package in ROS that converts ROS "sensor_msgs/Image" to a format which can then be used in OpenCV. Please see the instructions in the following link.

http://wiki.ros.org/cv_bridge

Create a node that subscribes to the Image topic produced by the Turtlebot and use the image converted by "cv_bridge" to detect and track the person model if present in the image.

Step (1): Write a node that subscribes to the image topic of the Turtlebot camera.

Step (2): In callback, use the detection and tracking algorithm to track the evader.

roslaunch pursuit_evasion robot_amcl.launch map_file:=**[PATH_TO_MAP]** world_index:=**[index]**

Here is the meaning of the parameters.

- map_file - path to the map you created in Task 1

Task 4: Control Node for Pursuit

1. http://docs.ros.org/melodic/api/geometry_msgs/html/msg/PoseStamped.html 2. http://docs.ros.org/fuerte/api/move_base_msgs/html/msg/MoveBaseActionResult.html

To access the location of the robot model on the map you can subscribe to the "/tb3_0/amcl_pose" topic. To send dynamic goals to move_base, you would be required to publish the goals to "/tb3_0/move_base_simple/goal" which is of type "PoseStamped". To check the status of the goal without using actionlib, you can subscribe to "/tb3_0/move_base/result" of type "MoveBaseActionResult". The evader moves on a predefined path based on the selected map. The task is to track the evader for a total of 5 minutes and report the average performance of tracking for a total of 10 runs.

You can either create a new node or use the node that you created in Task 3 to send the goals to the pursuer to keep tracking the evader.

Step (1): Create a node that uses a timer to shutdown after 5 minutes.

Step (2): Create a publisher to send the PoseStamped messages to the pursuer.

Step (3): Create a subscriber to check the status of the navigation results. You can use the "actionlib_msgs/GoalStatus" status to check for the status from the "move_base". Please refer to the documentation to get an understanding of the status codes.

At the start of every trial, run the following launch file to start the evader node that moves the evader in a pre-defined path based on the selected world.

```
roslaunch pursuit_evasion move_evader.launch world_index:=[index]
```

Relevant Info

Gazebo and other nodes cause some issues when we try to exit the simulation. Some nodes might not shutdown since gazebo keeps running in the background. To avoid this issue, add the following to your ~/.bashrc and use "kill rosgazebo" in the terminal to shutdown gazebo and other ros related processes.

```
alias killrosgazebo='killall -9 rosout roslaunch rosmaster rviz gzserver nodelet
robot_state_publisher gzclient'
```

Help Resources: Please use the following link on Canvas to post your questions so that we have a common thread to answer them all.

https://canvas.asu.edu/courses/43532/discussion_topics/1172883

Deliverables:

1. Maps that were created in Task 1, code of ROS node created in Task 3 and Task 4.
2. **Two demo videos/a single combined video, one showing the SLAM, and the other showing the Turtlebot3 pursuing the evader in both environments. Please narrate in the demo videos.**
3. A report paragraph describing the node you constructed and the results of tracking for both environments.

Submission Guidelines:

1. Please submit a PDF document as your project report. The name of the document must be "xxx_project_3b.pdf". Xxx is your ASUID.

2. Please also submit a zipped file containing your code. The name of the zipped file must be "xxx_project_3b_resources.zip". Please do not use bzip, 7z, rar, or other compression methods. Just use the plain zip. Please do not put your PDF report file in this zipped file. There should be two separate files submitted, one is the PDF report, and the other is the zipped project file.
3. Your code must be able to run properly with ROS, Gazebo, and simulated Turtlebot3. Please make it clear how to run your code in the report. Please also mention any dependent libraries and environment requirements.

Grading Rubrics:

There are 10 points in total for this project. There are 2.5 points for each task.