ME 4451 Lab 8 Handout

Anirban Mazumdar, Bryan Blaise, Nikhil Nandikanti

Spring 2024

1 Objectives

1. Execute a navigation task to a desired position/orientation using GPS and sensory feedback.
2. Experiment with advanced path planning algorithms for a mobile robot.

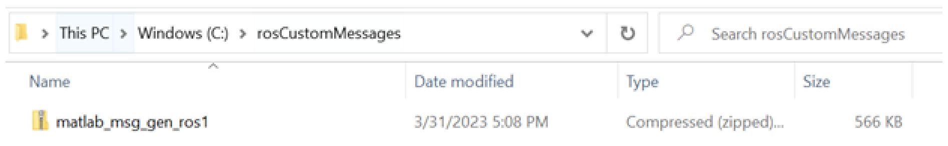
2 Prelab Assignment

- Please install MATLAB version R2022b
- Install Python 3.9 and Visual Studio Community 2022
- Follow steps found in Section 3 to set up custom ROS messages on your MATLAB
- Thoroughly review the lecture content on Mobile Robot Control
- Write the Smooth Trajectory waypoint finding function outlined in the lecture on Mobile Robot Control

3 Register ROS Custom Messages to MATLAB

Please implement the following in order to allow access to GPS functions. **These steps must be performed with the updated Matlab version.**

1. Download zip file ”matlab_msg_gen_ros1” from Lab 8 Canvas file.
2. Create a local folder, placing the zipped file inside. **DO NOT EXTRACT!**

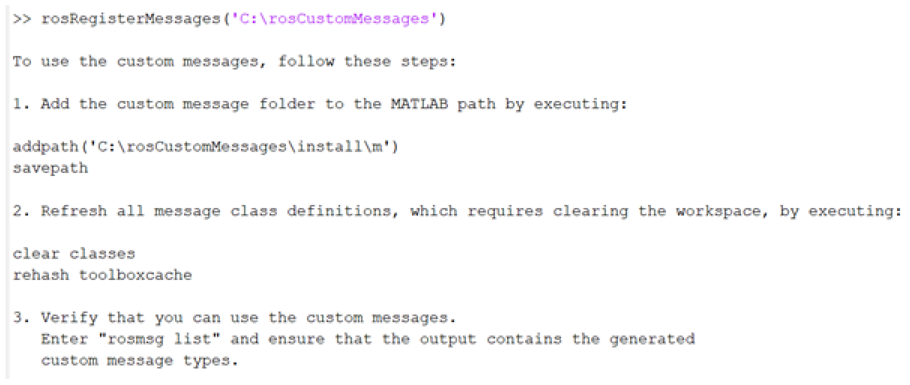


Name	Date modified	Type	Size
matlab_msg_gen_ros1	3/31/2023 5:08 PM	Compressed (zipped)...	566 KB

Figure 1: Create local file location for zipped file. NOTE THIS FILE PATH.

3. In your command window, run the command *rosRegisterMessages('YOUR_NOTED_FILE_PATH')*. For example, referencing Figure 2, my function would be called as the following: *rosRegisterMessages('C:\rosCustomMessages')*.

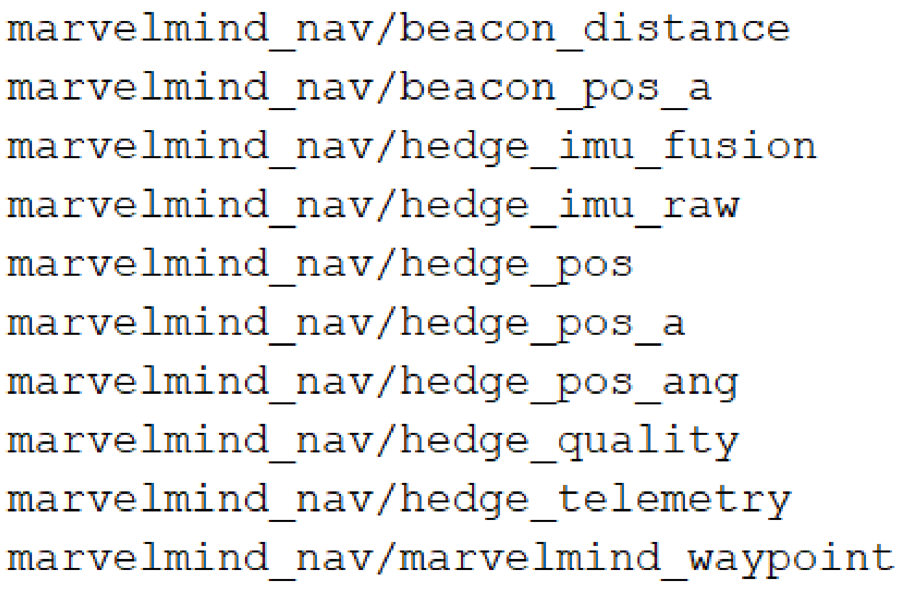
4. If step 3 is performed correctly, Matlab will prompt the following message in your command window:



```
>> rosRegisterMessages('C:\rosCustomMessages')

To use the custom messages, follow these steps:

1. Add the custom message folder to the MATLAB path by executing:

addpath('C:\rosCustomMessages\install\m')
savepath

2. Refresh all message class definitions, which requires clearing the workspace, by executing:

clear classes
rehash toolboxcache

3. Verify that you can use the custom messages.
   Enter "rosmsg list" and ensure that the output contains the generated
   custom message types.
```

Figure 2: Prompted Instructions from Step 2.

Follow these remaining steps, verifying that you have correctly installed inputting "rosmsg list" in the command window. If done correctly, the following will have output to your command window (search alphabetically).



```
marvelmind_nav/beacon_distance
marvelmind_nav/beacon_pos_a
marvelmind_nav/hedge_imu_fusion
marvelmind_nav/hedge_imu_raw
marvelmind_nav/hedge_pos
marvelmind_nav/hedge_pos_a
marvelmind_nav/hedge_pos_ang
marvelmind_nav/hedge_quality
marvelmind_nav/hedge_telemetry
marvelmind_nav/marvelmind_waypoint
```

Figure 3: Correctly installed custom ROS messages for your use.

4 Mobile Robot Control - Navigation Using GPS

Your final objective will be to generate a program that executes a navigation task on the robot. You will implement a mobile robot control algorithm that takes in desired goal poses/orientation as inputs (X_G, Y_G, γ_G), GPS and odometry data as feedback (to calculate error), command linear/rotation velocities (V_p, ω_z) for the turtlebot3, and present pose/orientation as outputs (X_P, Y_P, θ). You will also have to compute gains (K matrix) based on the error terms for this algorithm to work. The algorithm will generate a smooth trajectory while simultaneously driving the errors to zero. Refer to lecture 25 on Canvas for more details.

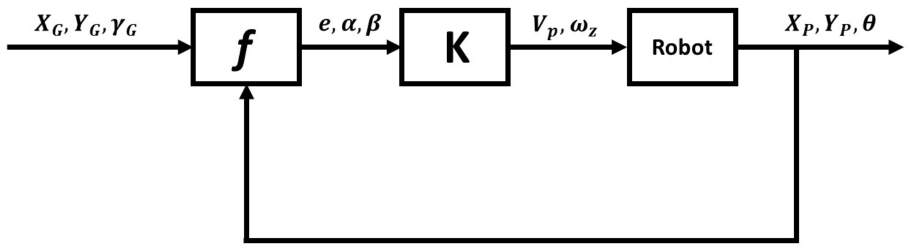


Figure 4: Diagram of the controller you will implement in this lab.

The robot will have to navigate from a starting point to an arbitrary end goal using information from GPS. The turtlebot3s have been installed with ultrasound GPS beacons and will provide you with X and Y positional data in meters. The rostopics you can use to obtain these data are: 'hedge_pos', 'hedge_pos_a', and 'hedge_pos_ang'. Create a subscriber and a rosmessage to receive the data and take a look at what is inside to get a better understanding. Keep in mind that since these beacons use ultrasound, stay out of the GPS region as much as possible to reduce noise during operation. Here is the map of the room that is covered by GPS:

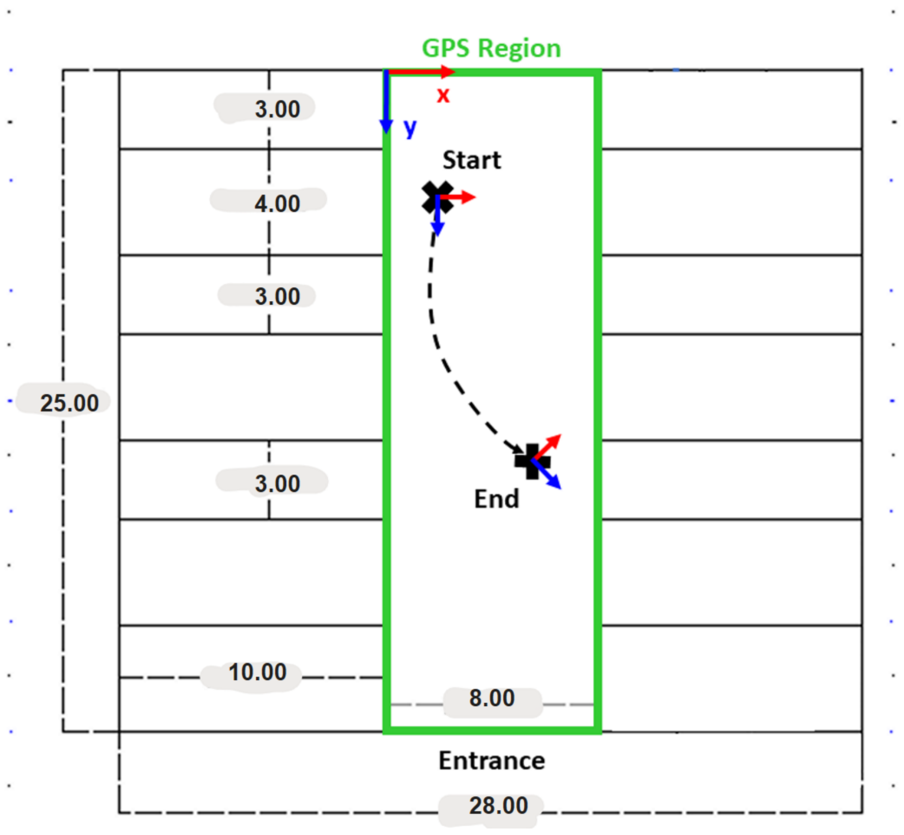


Figure 5: Map of lab. Note the starting pose includes both position and orientation.

In the Lab 8 folder under the "Files" tab, you can access this map. Dimensions are in ft. The entrance of the lab is at the bottom of the map. Note that the starting pose and ending pose include both the position and orientation of the robot. It is recommended that you align the starting orientation with the world coordinate frame as the GPS won't provide you with the orientation (you must use 'odom' to calculate orientation). The goal position/orientation can be arbitrary. Ending up anywhere within ± 2 cm and ± 5 deg is considered a success.

Recall that the Turtlebot's odometry functions consider the origin of the global frame to be the pose/orientation of the robot at the robot's startup. You can reset the odometry by using the rostopic 'reset' or by re-launching the roslaunch command on your command prompt. You will need to perform transformations to create a new global origin that matches those in the map. Once you can reliably execute the navigation task, demonstrate it to your TA for credit.

Figure 6: GPS used for this lab. These beacons mounted on your turtlebot3s send/receive data using ultrasounds.

4.1 Extra Credit: Final Orientation

The navigation task only requires your robot to end within ± 2 *cm* of your final goal position. For extra credit, program your robot to end its path facing a particular direction. Orienting within ± 5 *deg* of your goal orientation is considered a success.

4.2 Extra Credit: Obstacle Avoidance During Navigation

For additional extra credit, program your robot to perform obstacle avoidance during the navigation task. While your robot is performing its navigation task, the TA will suddenly place an object in front of your robot at their discretion. Your robot should detect the object, avoid it, and resume its path and proceed to its goal. Your robot must both avoid the obstacle and complete the path to receive credit.