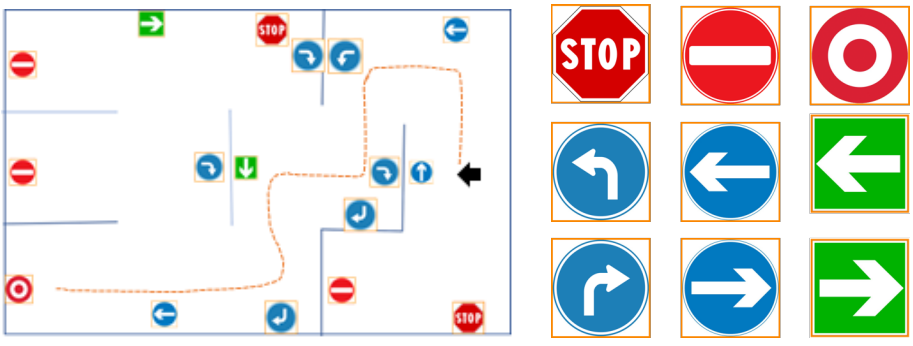
CS/ECE/ME/BME/AE 7785
Lab 6/Final Project

Checkpoint: November 22^{nd}, 2021.
Final Due Date: Friday, December 6^{th}, 2021.

1 Overview

The objective of the final project is to integrate multi-modal sensing and navigation into a single reasoning system. Our goal will be for the robot to complete a maze by following signs. As always you can solve this problem however you would like. Note: Since this is your final project, you may use any ROS configuration for computation that you would like. Everything can be run on the robot or you can have specific nodes processing data and sending commands from your computer. However, if you run nodes on your computer (e.g. passing images to do images processing on your machine) you must accept the risk or have a backup strategy if the GT network is slow that day.

The robot will be placed in a world such as the one pictured below. You may map the environment a priori and you will be able to localize your robot before starting. Your objective is for the robot to find the goal (designated by the red target sign) in the shortest path possible dictated by the signs in the maze. The location of the goal nor your starting position will not be known ahead of time, but signs throughout the world will always direct you from any initial starting position to the goal position.



(a) (b)

Figure 1

In the example world above (Figure 1), if the robot were to start on the black arrow, it would be able to follow the signs on the wall in order to reach the goal in the bottom left.

Nine different signs will be present in the world, organized into four categories: wrong way (stop and do-no-enter signs) indicating the robot should turn around, goal (red target sign), turn 90 degrees to the left (three left arrow signs), and turn 90 degrees to the right (three right arrow signs). The signs are printed in color on white paper and are taped to the walls of the robot space. There is a small orange border around each sign, as shown above.

The project is split into two parts, a **Vision Checkpoint**, and the **Final Demo**. As always, we strongly encourage you to use all available resources to complete this assignment. This includes looking at the sample code provided with the robot, borrowing pieces of code from online tutorials, and talking to classmates. You may discuss solutions and problem solve with others in the class, but this remains a team assignment and each team must submit their own solution. Multiple teams should not jointly write the same program and each submit a copy, this will not be considered a valid submission.

2 Vision Checkpoint (Due: November 22^{nd})

The first step of the project is to create an algorithm to classify the signs, without the robot running. For this checkpoint, we are only testing image classification performance.

You may design any solution of your choice to identify the signs. If you want to hand-code a solution, that is acceptable. The best results, however, will likely be achieved with a classification algorithm, and we are providing two suggested solutions:

Use KNN and OpenCV: You are already somewhat familiar with OpenCV, so one solution is to use the tools provided within. We are providing example code for the use of KNN, although you will need to improve it to get good performance. The current code reads the images in grayscale, and represents them simply as an array of raw pixel values. The resulting model achieves 26% accuracy. Suggested improvements to the model (in order of importance):

- Crop the image to focus classification only on the sign portion instead of the entire image.
- Incorporate color
- Incorporate other high-level features

Use scikit-learn toolkit (Python only): scikit-learn provides a wide range of machine learning tools, including classification algorithms. For those who dont mind adding another tool to their software arsenal, scikit will likely provide the greatest benefit and best overall performance. To make integration easier, we are providing a ROS node that shows how to process an image using scikit-learn, available at: https://github.com/GT-RAIL/image_classifier

You may also use a different third-party library or tool if you wish. Also feel free to collect more training data to train your model. Remember, for your final, this classification algorithm must be able to run in real time on your computer (passing images from the Turtlebot to your PC) or onboard the Turtlebot entirely.

2.1 Submission

Submit a single zip file called `LastName1_LastName2_Lab6.zip` containing your code and a `readme.txt` file describing how to run it on Canvas under Assignments–Lab 6. Please have your file output a confusion matrix as well as an accuracy score. We will test the code against a withheld dataset of images similar to the ones provided with the assignment.

2.2 Grading

Grade will be equivalent to the classification accuracy of your algorithm on the withheld test set.

E.g. Accuracy = 93% → Grade = 93%

2.3 Files Provided

- `2021imgs/` directory containing example images of signs taken from the robot.
- `2021imgs/train_images/train.txt` example files listing images to be included in the training set and their corresponding class label.
  - **labels:** 0: empty wall, 1: left, 2: right, 3: do not enter, 4: stop, 5: goal.
- `2021imgs/test_images/test.txt` example files listing images to be included in the testing set and their corresponding class label.
  - **labels:** 0: empty wall, 1: left, 2: right, 3: do not enter, 4: stop, 5: goal.
- `knn_example.py` example use of KNN with OpenCV in Python.

3 Final Demo (Due December 6^{th})

For the final demo, your robot should navigate the maze set up in the lab. To aid in developing code for this component, the Gazebo environment provided in lab 5 may be used for for testing. The Gazebo environment is identical to the one in the lab, but instead provides an idealized environment for testing. We highly recommend debugging in Gazebo before starting on the robot.

The figure on the right (Figure 2) shows a map of the world that has been created in the lab. For testing, the robot should start in the bottom middle (R), facing south. The goal is in the top middle, facing east. A significant number of signs such as the ones shown in the map will be present to guide the robot to the goal. There will not be a sign on every wall or surface, but all key intersections will have a sign on every adjacent wall.

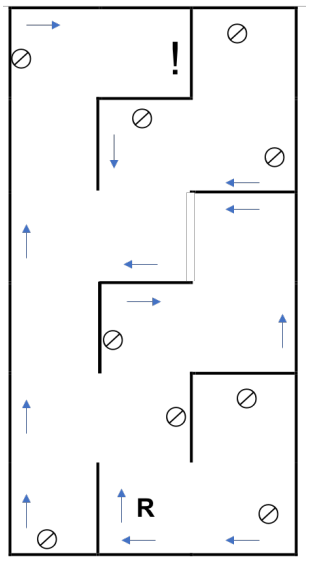


Figure 2

If your robot is lost, consider having it turn in place to check other orientations. Note that we recommend that the robot check for signs only when stationary and facing a wall directly in front (which can be verified using the laser data). All example images provided were taken with the robot approximately a foot from the wall. This ensures that only a single sign is in view.

The world will contain only right angles. As a result, you can assume that a right/left turn arrow indicates that the robot should perform a 90° turn and drive to the next wall from there. The stop and do-not-enter signs signal that the robot should turn around and drive to the next wall.

Some parts of the world near the wall can be a bit dark due to the lighting in the room. If needed, you can use a cell phone or flashlight to help light up the area when your robot is running.

3.1 Grading

Consider the map of the world as a large 3x6 occupancy grid. Project grade will be determined based on the following formulas:

If the goal is reached:

$$grade = 1 - (|r - n| * 0.01)$$

If the goal is not reached:

$$grade = (1 - \frac{d}{n}) - (|r - m| * 0.01)$$

where:

- n is the length of the optimal path to the goal when following the signs.
- r is the length of the path taken by the robot.
- d is the distance remaining from the robot's furthest progress point to the goal.
- m is the length of the optimal path to the robot's furthest progress point.

All above distances are Manhattan distances. The robots furthest progress point is defined as the point along its path that is closest to the goal. Note that the above equations are basically identical if you consider that $d = 0$ and $m = n$ in the case that the goal is reached.

Example grading scenarios given the map on the previous page (incorrect robot actions are underlined for clarity):

- The robot correctly turns right two times, drives north, turns right, drives to the east wall, turns left, drives north, turns left, drives to the west wall, turns right, drives north, turns right, drives east, and finds the goal.

$$grade = 1 - (|9 - 9| * 0.01) = 1(100\%)$$

- The robot correctly turns right two times, drives north, turns right, drives to the east wall, turns left, drives north, turns left, drives to the west wall, turns left, drives south, turns around, drives north, turns right, drives east, and finds the goal.

$$grade = 1 - (|15 - 9| * 0.01) = 0.94(94\%)$$

- The robot correctly turns two times, drives north one block and cannot see a sign so begins to spin, sees the do not enter sign on the right, turns around, drives stright, turns right, drives straight, turns right, and finds the goal .

$$grade = 1 - (|7 - 9| * 0.01) = 0.98(98\%)$$

- The robot correctly turns right two times, drives north, turns right, drives to the east wall, turns left, drives north, turns left, drives to the west wall, turns left, drives south, then stops and makes no further progress.

$$grade = (1 - \frac{3}{9}) - (|9 - 6| * 0.01) = 0.67(67\%)$$

- The robot doesn't move.

$$grade = (1 - 9/9) - (|0 - 0| * 0.01) = 0(0\%)$$

3.2 Demo and Grading Process

The world will remain unchanged for the grading demo, so you can continue using the same map you created for testing. However, we will change the robots start and goal locations on the day of the demo, as well as the locations of the signs as appropriate. You will be able to localize your robot at the beginning of your run.

Demos will be conducted during **pre-assigned 20 minute time slots** throughout the day on December 3^{rd}. Each person will sign up for a time using the spreadsheet below and will be graded during this time. Only one team may demo at a time and only those who are demoing will be allowed into Klaus 1210. You may set up a robot in Klaus 1210 away from those demoing if you are signed up for one of the next two slots (i.e. you can start setting up 40 minutes before your session). Please work quietly and respect the work environment of the students demoing (they have priority).

This demo process means you should not plan on any last-minute hacking, since neither robots nor the maze world will be available for testing.

Each team will have one demo session in which to test. A second session will only be made available later in the day if there is a major hardware failure during the first run (i.e. lidar stops working). **NO LATE DAYS MAY BE USED ON THE FINAL PROJECT!**

Sign up sheet: here

`https://docs.google.com/spreadsheets/d/1qK45rPcTwMU7s_dA-Z_DplNNcYqHs3nrqLVNpmukNAo/edit?usp=sharing`

3.3 Submission

You have two required submissions for the final.

1 Your ROS package in a single zip file called `LastName1_LastName2_Final.zip` uploaded on Canvas under Assignments–Final Project.

2 A live demonstration of your code to one of the instructors during your specified timeslot.