Grocery Shopper Final Project 2025

Yusuf Morsy, Zachary McGuire, Anna Soukhovei

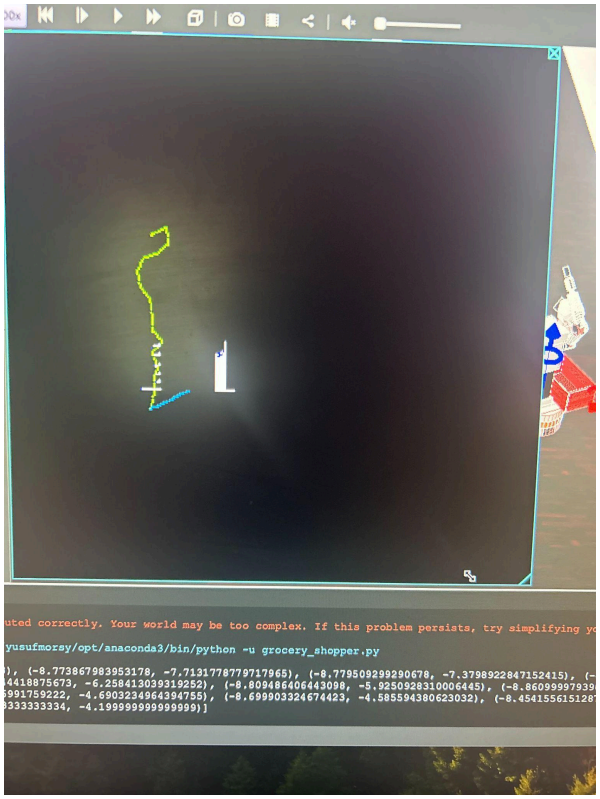
Github: https://github.com/yusufmorsy/Robotics-Final-Project
Youtube: https://youtu.be/QJKzzwn0zPI

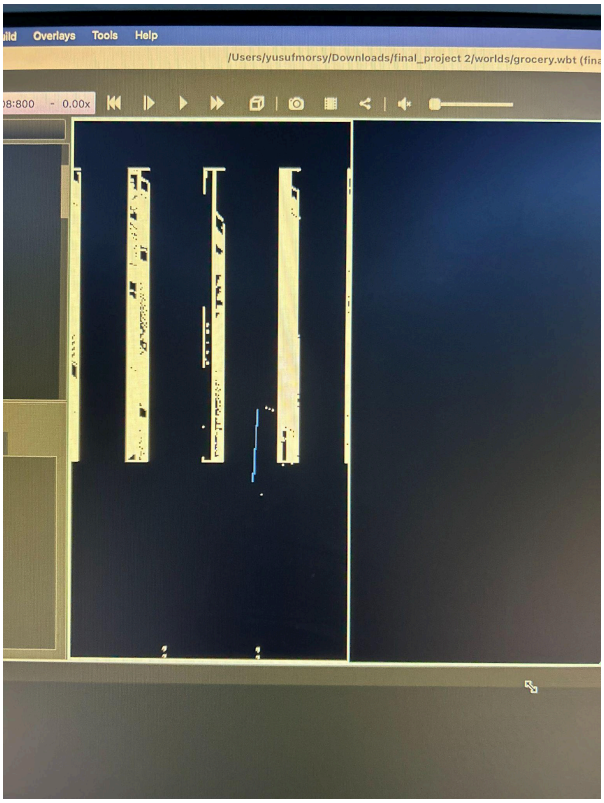
Please download the model at :
https://drive.google.com/file/d/1KVwlK7T4rta7Y7QGsymCcOVPkf4A6WsO/view?usp=sharing
It is too big to add to the zip file.

MAPPING

Lead: Yusuf

Yusuf: In this section of the project, we implemented a SLAM (manual/ autonomous mapping) system. The robot navigates and updates the map in real time using RRT* created paths. It avoids obstacles and replans its path as it populates the map. We faced many problems with making the map display properly on the display (world-to-display). We solved that by experimenting with different math formulas to understand how to completely translate the world to the map. We also had many issues updating the map as the robot moved and had to tweak values because the LIDAR's sensitivity was adding figures on our map that it thought were walls when they weren't. I got a lot of help from Zack, and we figured it out eventually.





LOCALIZATION

Lead: Yusuf

Yusuf: This was the second section we worked on for the project, and we implemented localization using GPS/Compass. We faced some issues with the angles of movement (pose_theta), but eventually figured out the direct translation of the robot to the world and made it follow the paths we wanted it to follow. We calculated many values, like the wheel speeds and error from the RRT* paths we wanted to go towards. I got help from Zack to improve the paths.

COMPUTER VISION

Lead: Anna

Anna: I trained my own machine learning color detection model with 400 images in the training set and 300 images in the validation set using yolov8 and I trained on yolov8. Originally the model was only trained using 1 epoch. When using the original model, robot wouldn't detect most cubes, and the boundary around the cubes was too big when it did detect them. Also, it had issues identifiy multiple cubes. I trained the model on Yolov5su model instead, and it improved the model's detection of yellow and green cubes. It was trained on 18 epochs and it took 1 hour and 35 minutes.After this, it was able to detect multiple boxes, and also detect a green and yellow cube.

PLANNING FOR NAVIGATION

Lead: Zack

Zack: This was the most straightforward section, although it took a long time to get it working well. I implemented RRT Star using much of the same code as from our homework, and it worked well out of the box. The hard part was the messy real-world problems. For instance, what if the robot was technically inside of the collision map? For that, I needed breadth-first search to find the nearest free map coordinate to the robot. Or what if the goal was inside of a wall? If that was the case, I chose the nearest vertex to the intended goal location. Or what if the robot was stuck against a wall? If it hadn't reached a new waypoint in too much time, it would try a different route to the same goal point.

MANIPULATION

Lead: Zack

Zack and Anna: The OpenPNP library was difficult to use. We used IK for manipulation. First, we found the blocks using our vision model, navigated towards them, and attempted to pick them up. We faced issues with getting the robot to approach the blocks it detected. We figured out that the map coordinates of the objects weren't translated correctly in our RRT* algorithm. We also faced issues positioning the robot towards the block and with the TIAGo arm. We also had issues with the inverse kinematics function from lab 5 crashing the entire program because the function thought the joints were out of bounds when they were not. So, if the robot stops adequately before a cube, the inverse kinematics will move the arms towards the cube.

Notes:

The code is run in the "autonomous" mode for the live mapping and navigation of the grocery store. The green paths are the RRT* paths the robot follows to navigate through. The code manually changes between "autonomous", "travel to cube", and "grab cube" modes depending on its position and if it finds an object. The vision model is in a Google Drive linked to the Github because it was too big to push to Github. It should be added to the root of the repository. Warning: the project may crash depending on the user's hardware in "grab cube" mode.

The best we managed to get was 3 yellow objects, as we ran out of time before we could fine-tune it enough to collect all of them.

run pip install ultranalytics, which should automatically install cv2 and the other required packages. If you already have ultranalytics installed, then you can run the yolov8 model using "yolo task=detect mode=predict model=best_18.pt source={path_directory_to_image} conf=0.8" in the command line to double check my model.