Project Story 2

SeungU Lyu, HK Rho

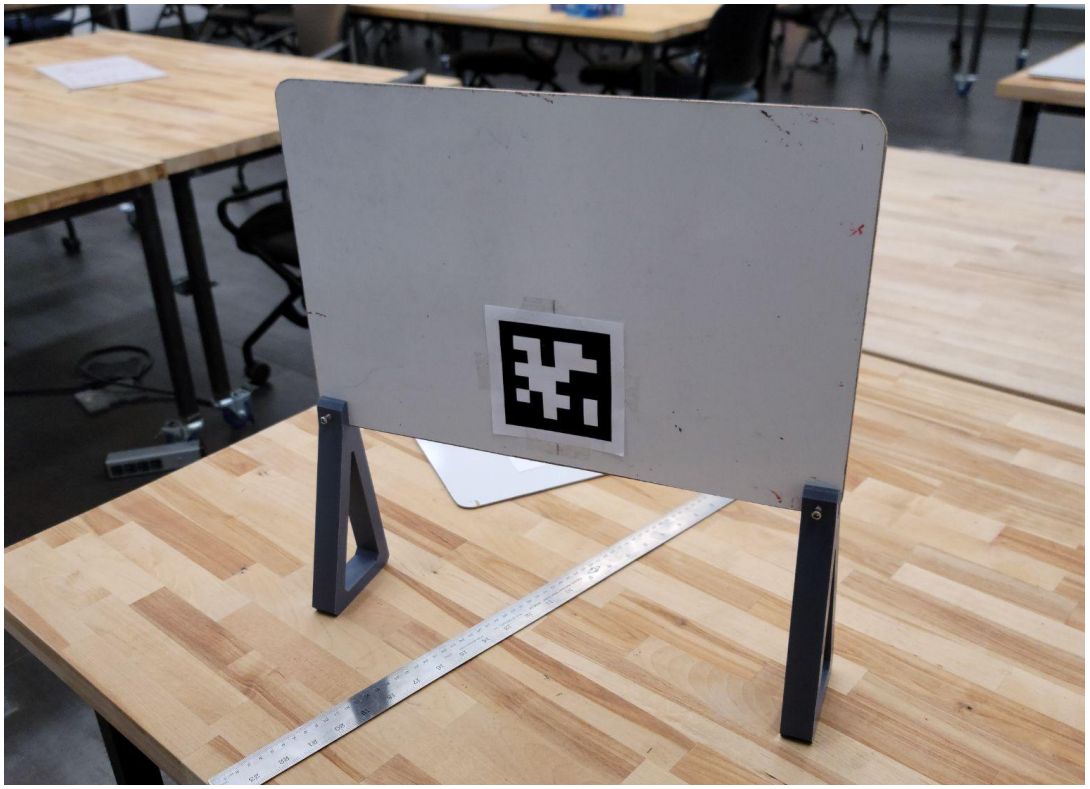
Fix Critical Angle Bug

1. A critical bug related to the April Tag angle happened while converting Quaternion to Euler angles. This was fixed by extracting the necessary angle from the rotation matrix, instead of going through the conversion of rotation matrix -> quaternion -> Euler angle.
2. After fixing this bug, recording and loading the map works as intended.

Test in a Real Environment

1. The simulated environment had issues with the map and the real world not matching each other due to the wrong camera parameters. Tested in the real environment, it had good accuracy simulating the map but had an issue with the Odom frame not being correct due to how Neato estimated its own position (ex. moving on a carpet will have less movement in the real world than Odom).
2. To solve this issue, we decided that it will be best to have more checkpoints so that we can get the map updated before the Odom and the real world gets too different.

3D Print Checkpoints



1. To have more checkpoints, we decided to 3D print the parts so that we can turn some common objects into checkpoints.

2. We have printed 5 pairs, which will be enough to create a map that goes around the classroom.

Multiple Neato Control

1. Using Paul's multiple neato code, we successfully connected to two different neatos with different topic names.
2. All the codes were modified so that they have a parameter for the robot name, and all the topics published/subscribed are changed based on the parameter.
3. To be able to show the image and control both Neatos at the same time, a node for Pygame was created to get processed images from both Neatos and send Twist messages to them.

Pygame Integration

1. We have created a structure using ros topics to efficiently send information to each node.
2. drive_neato class will detect the April tag and calculate the map's position on the Odom frame. After the image is processed, the image and Neato's position on Map/Odom/base will be all sent to different nodes.
3. Pygame will receive the processed images from both Neatos and show them at the same time. Users will be able to use different keys on the keyboard to control each Neato in real-time.

Minimap

1. From the map data and the Neato's position on the map, we created a Minimap that shows the position of the player's Neato in the map frame.
2. This is done through Pygame's Draw function - connecting each point in the map as a line.

Track Generation

1. From the map data, the Neato's position on the map, and the camera intrinsics, we draw the tracks in OpenCV. The tracks are visualized accordingly to the Neato's location and orientation.

Gazebo Demo

1. https://youtu.be/muAUlZxc-00

2.