E28 Project 3 Write-Up - “Robot Slalom Course”

Team lchien1-skalidi1-tle1

Who did what?

Suraj helped create the drive_towards() function in the program which the robot uses to navigate towards the destination point.

Lonnie wrote the code for task 1 that solves the location of the destination point for the pure pursuit implementation and debugged the code by physically testing the robot in the lab. Lonnie also used the code from task 1 to create slalom.py which added the functionality of the robot to read a course file, and execute a list of actions including switching from the pre-action state to the driving state (described below) and adapted code from project 1, following the tape.

Toby helped Lonnie test his code on the robot to make finishing touches once the code was working.

A description of the functionality of the system, including diagrams (state diagrams, gate geometry, etc.) where helpful.

When the program reads the course file it creates a list of rows from the text, such as “left yellow yellow”. I created a global variable to keep track of which row I am on, which starts from zero and increments every time a gate is passed. There is also a “pre-action” boolean variable which is true when the robot should be turning left, right, or following the tape and false when the robot should be driving to a gate. This is useful because the program needs to know when to

read the first word from the course row, or when to read the cone colors. When pre-action is true, it reads the first word which should be left, right, or tape.

The program is set so the robot is always in either a “pre-action” state or a “driving” state. If the line from the course reads “left yellow yellow”, the robot should turn left until two yellow cones are in view. While in the “pre-action” state, the program is constantly checking whether the gates in its field of vision match the correct color description. As soon as a correct gate comes into view, pre_act switches to false and the robot begins to drive to the gate.

When driving, the robot navigates towards the gate using pure-pursuit. The program establishes the location of the robot in the gate frame, and projects that location to the gate x-axis by just taking the x-coordinate of the robot. It then creates a destination point that is the same as the projection, but a distance of alpha closer to the gate. I chose an alpha of 0.3. Then the program converts the destination point to robot coordinates.

As the robot moves, the destination point is always being updated–even before the robot reaches the point. So as the robot moves, its destination also moves closer to the gate. Once the robot is sufficiently close to the gate, the “pre-action” variable switches back to true and the robot will follow the instruction of the first word in the next line.

If that first word is “tape” then the robot follows a special set of instructions to follow the tape. The linear forward velocity is set to a constant value and the robot steers towards a “blob” that it recognizes as a red tape using a pd controller. The tape it steers towards is always the biggest red blob in its field of vision, so this helps to prevent the robot from drifting off or cutting corners when the path bends along the tape. If the largest blob is far to the left of the center of the robot’s vision, then the robot will steer more aggressively to the left.

Just like when the word is “left” or “right”, the robot will continue to follow the tape until the correct colored gate comes into view. As soon as that happens, pre_act switches to false and the robot drives to the gate.

What problems you ran into, and how you solved them.

Initially, the program was not recognizing some of the member variables in the code that I wrote, which is why I used a global variable to keep track of which row in the course the robot should be following. Professor Phillips helped me solve the problem which turned out to be issues when using tabs versus spaces while coding in vim.

Another challenge was when testing the code, the robot would turn until it saw the gate, drive forward for a little bit, then keep turning. This was because the program was searching for a new gate once the gate went out of vision. To solve this problem, I needed to associate the gate’s location with the pre-action state. Once pre_act switched to false, the program needed to have a memory of where the gate was and then no longer look for new gates.

We ran into more problems when testing the robot on the course, and the robot was failing to find the gate after following the line, so it would just try to follow tape indefinitely. The solution to this was simply physically pushing the cones closer together or up and down until they were in a position that the robot could see before it reached the end of the tape.

Commentary on the videos – how does each clip relate to the two items above?

In the video named “slalom.MOV” on Google Drive, the robot follows the course by starting from the tape labeled “start” and turning left until it sees the first gate. The program was meant to stop the robot’s linear motion every time it stopped the gate, but because we used velocity filtering, the robot would start turning and looking for the next gate before it reached a full stop. This turned out to work well, as the robot could still look for the new gate and move more efficiently without fully stopping.

A link to a Google drive folder shared with me and containing your video

https://drive.google.com/file/d/1eKulb4VPo4EO8W1yivpjpHuiHwxEQ8x_/view?usp=sharing

This link should work for anyone logged in with a .swarthmore email.