Chapter 4

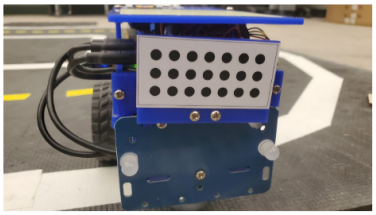


Image 1 - Grid sticker for detection

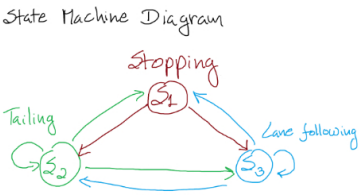


Image 2 - Simple state machine diagram of our approach

Duckling following mama duck

In this lab, we learn how to interact in the world based on the presence of another physical robot. The task at hand requires us to tail another duckiebot by taking advantage of a sticker in the back of the leader duckiebot to executes a detection similar to what we have for the intrinsic camera calibration, to then execute a command based on the detection itself.

The implementation that we went for was to use the supplied detection and distance nodes for that particular circle grid in our robot, such that the detection nodes give us the coordinates within the video frame that we made the detection on, then this information is supplied to the distance node such that it provides distance in meters of our robot to that particular grid tag we are detecting. Based on the detection of that grid tag, we then switch between three states' tail or lane following.

Tailing is when we detect the sticker grid in front of us and execute our motion based on where we see the grid in our camera frame. If it's fairly in the middle, we just follow it straight; if the leader either turn left or right, the follower tries to center with the leader robot such that the dot grid is situated in the middle of the follower's camera frame therefore making it move left or right, this, unfortunately, can be done incrementally based on how well our detections are. Lane following only activates based on whether we cannot detect the previously mentioned grid sticker after so many readings; in our case, it was 10. This state then changes its detection based on the yellow lane grid. Our strategy also uses the z coordinate of our detection to modulate the distance between the leader and follower robots. Finally, the stopping state has precedent over the other states and simply uses the detection of the red stop line to execute the action of stopping on it.

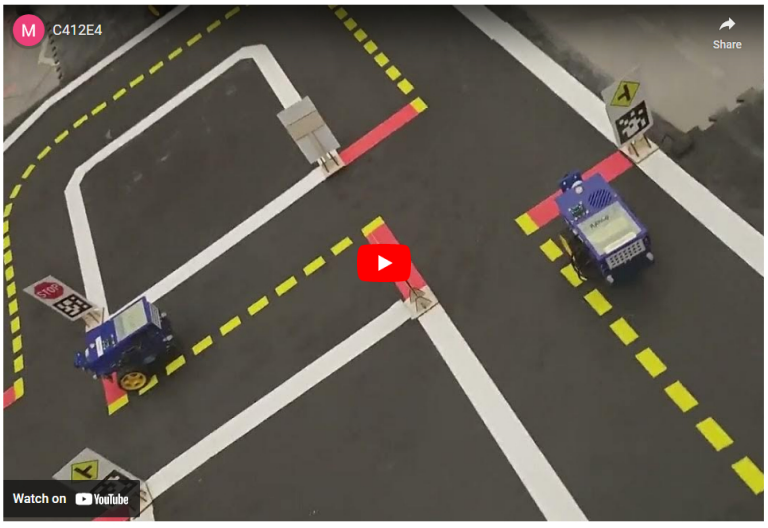
Despite our efforts, we couldn't execute the lab as neatly due to our strategy for turning at the intersections. We initially solely based it on detecting the grid and then adjusting our follower robot such that the gird stays in the center of the frame, but it wasn't very efficient; the other strategy was to move based on a selected number of last detections where we average each image to determine where we should turn after we stopped. Neither approach worked well, and these are the potential reasons for it.

1. There are Apriltags at the corners obstructing the view so that the following bot could not detect the dot pattern.

2. Both bots are moving at the same time, it is possible that the motion blur is making the dot pattern detection more unreliable.

3. Observing how differential drive robots turn, we can see that the dot pattern shifts to an extreme angle fairly quickly. Also, the dot pattern detection fails a lot, even with fairly moderate angles. With only a few frames of valid detections, it is very hard to determine which direction the leading bot has turned into.

4. Our bot needs to stop at the intersection, but by the time it starts moving, the leading bot could already have cleared the intersection.



Video 1 - Our Attempt at the video submission

One of the challenges we faced for this lab was incorporating my partner Marcus' code with mine. Since our methods of implementation is a bit different, it can sometimes be hard to establish a bridge between the other person's code. I hope this is a skill that is acquired over time. Another challenge I found was to test the capabilities of this exercise since it's hard with the keyboard control to achieve a consistent test drive of our leader such that our follower bot can execute its commands smoothly. Finally, another improvement we should do is to try to seek solutions outside the "box" since we were so eager to use the code provided to us as a resource for our work that we didn't search for other options sooner (using masking of the sticker grid as detection. etc.).

Questions for this exercise:

1. How well did your implemented strategy work?

This implementation theoretically can work well for the stopping and the tailing process on straights, and even turning on the edges (occasionally relying on lane following) also keeps a very reasonable distance to the leader robot and even on the intersections, the only downside is that our implementation of the strategy did not perform well on intersections.

2. Was it reliable?

   It wasn't very consistent, so it wasn't that reliable in high sight because we tried to do multiple detections (lane, stop line, robot) and organized several states (tailing, lane following, stopping, etc.). This can lead to occasional misdetection if the processes or states are not sorted properly.

3. In what situations did it perform poorly?

   A situation that occurred was that at the intersections, our follower robot would lose detection rapidly and would have trouble finding the leader afterwards. Another pointer of our approach was that the bot would struggle with some of the turns; when the bot follows the leader into a corner and then loses detection, it would also not be able to detect the yellow lane markers.

References :

- OpenCV:
  1. https://docs.opencv.org/3.3.0/dc/dbb/tutorial_py_calibration.html
  2. https://www.tutorialspoint.com/detection-of-a-specific-color-blue-here-using-opencv-with-python
  3. https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html (Others from the OpenCV documentation as well)
- P.I.D controllers:
  1. https://www.youtube.com/watch?v=y3K6FUgrgXw
  2. https://github.com/vazgriz/PID_Controller/blob/master/Assets/Scripts/PID_Controller.cs

Received help from: Xiao (TA), Justin (TA), Marcus (Partner & student)