Gregory Cooke

Partner: Connor Napolitano

Learning Module 8: Autonomous Vehicles/Navigation

Module 1: Wandering

Week 4: Vision-Based Wandering

Homework Question: You should roughly reiterate the activity, discuss how it was accomplished, demonstrate through images (if possible) the functionality, then briefly note any observations (including cases where it may not work so faithfully, or what is needed to work very well):

In this activity we were to create a “wandering zombie” robot. It was supposed to split the laserscan data from the Kinect into 5 sectors, average the values in those sectors, then choose to move in the direction that appeared the most open. We took the laserscan data gathering from the example code and integrated that into our state machine. The reason for this was because the way the example code was structured was completely different than the averaging method, so after discussing it with Professor Vela, we decided it would be easier to integrate the laser scan capabilities into our old state machine structure rather than integrate the state machine into the laser scan capabilities.

This appeared to be relatively simple, taking advantage of python’s max() and index() functions for lists. However, the appearance of NaNs when an object was either too close or too far for the sensor to detect caused a lot of issues. At first, we just threw these NaNs away from the average, but after discussing it with Professor Vela we decided that it was better to assume the NaNs were far away and to let the bumpers handle when it gets too close to something.

We ran into an issue where the robot would “wiggle” to the left and right. That was because, when it was facing down a long hallway, it saw all NaNs. Then, because the left sector was the first index, it would turn left. It would then appear, as it turned left and began to see the wall of the hallway, that the right was the most open and would turn back towards the center. Then, the all NaNs issue would happen again and it would turn left, and it would continue doing this. We fixed this by restructuring the list that we fed into the max() and index() function. When multiple values were tied for the max, it would end up returning that we should towards the sector that was the close to the start of the list. We changed the list order to have the center as the further left index of the list. This fixed the issue we were having, because then when the robot looked down the hallways and the laserscanner returned all NaNs, the robot would go straight instead of veering left.

After doing the fix above, the wandering worked quite well. It’s biggest issue comes from the fact that we treat all NaNs as the furthest possible case. This causes the robot to run straight into something that gets very close to the Kinect. However, as discussed above, there is no good way to avoid this via the approach we are taking, so we just let the bumper logic handle those cases. The bumper logic reorients the robot in such a way that is should continue moving in a good direction instead of straight into a wall again.

Turtlebot Adventures Answers:

1. What is the coding error in the stock code?
    a. The stock code splits the image into 3 sectors, but has the first sector as the first third of the laser scan, the second sector as 1/3 – ½ of the laser scan, and the third sector as the last half of the image. That doesn't make sense in terms of deciding left, straight, or right, so the fix was to make each sector 1/3 of the image.