MEEM 4707: Autonomous system

Spring, 2024

Lab - 6

By

Colton Kriecher & Amanda West

Example)

Name	Logic Build	Coding	Report Writing	Total
Colton	50%	50%	50%	150%
Amanda	50%	50%	50%	150%

Problem 1

Write a piece of code in the Python node perception_camera.py. Your code should determine the command velocities and publish it on the topic "/cmd_vel." Find the RED cylindrical wall using camera images and follow the wall.

1) Demonstrate the robot to identify and follow the red cylinder wall. **Capture the trajectory in the Gazebo and real-world (using dead-reckoning) and include them in your lab report.**

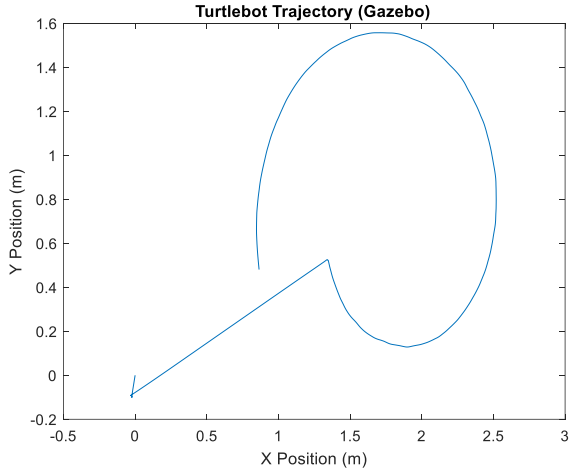

Figure 1: Simulation TurtleBot Trajectory

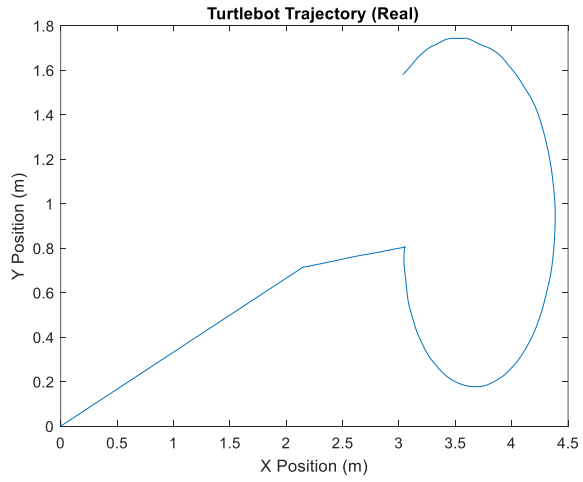

Figure 2: Real World TurtleBot Trajectory

Problem 2

1) Discuss how the robot finds the RED cylinder.

Each color has a HSV (hue/saturation/brightness) range. The camera can be tuned to identify specific HSV ranges for color. By tuning it for red, the camera can mask the field and determine where it sees the color red. The net result is a black/white image, which represents the pixels that fail/pass the given condition (in our case, being within a specific range of HSV values). We can then code the robot to determine the horizontal center line of those red pixels by taking the average X-value of the pixels that "pass", and then using the distance (in pixels) between that line and the center of the image to estimate angular error.

2) What can be done to improve the performance further? Explain your idea briefly.

To improve the performance, we could integrate additional cameras to cover the full 360^{o} field-of-view that the LIDAR covers. This would allow us to find and align to our target even when it is not directly in front of the robot. Alternatively, using SLAM to record the estimated positions of previously-seen cylinders could provide that same context that's only present outside the camera's FOV. If this were implemented, then when we are circling the object using LIDAR, we could specifically ignore other cylinders and only accept the angle of closest distance if it's near the remembered position of the red cylinder.

Objective

The objective of this assignment is for the TurtleBot to identify obstacles by color. The TurtleBot must identify the red cylinder and move towards it, the traverse around its wall.

Approach to achieve the Objective

To separate out the different movements, we created a state machine, like in our last few labs. The state machine includes scanning, approach, aligning, and following. We changed our code in the scanning state to identify the cylinder by color. Using the given code, we adjusted the range of the lower and upper HSV values to fall in the range of the color red. The TurtleBot then masked the area with the color sensor to find which obstacle falls within the HSV range we set.

The code identifies the dimensions of the mask in pixels. It computes the horizontal centerline by dividing the width in two. We then calculate the average passing X value by determining the X centroid, or the center of the light pixels on the mask. We set a variable called 'px_error' which computes the distance between the centerline and the centroid. We then used an if else statement to tell the TurtleBot to move either clockwise or counterclockwise based on the value of the 'px_error' and if it falls within a set tolerance. Because we implemented a PID controller into the code, we set a kP value for the pixel-estimated angular error. This is used when determining how fast the angular velocity will be.

Challenges faced and countermeasures taken

What problems did you face?

While we were able to get a perfectly-clean mask in the simulation, where there is no scattering or ambient light to distract the robot, we initially had issues with masking the real robot's camera. After printing a list of all unique HSV color tuples in the frame, we found that this was because the actual

material of the “red” cylinder was perceived as orange by the real camera. To fix the mask, we changed our hue range from 0-10 (covering the transition from red to blue) to 220-255 (covering the transition from orange to red). This gave us a much cleaner mask that was sufficient for our algorithm to work on the real bot.

The largest issue we had, however, was in getting the hardware to connect properly. There were a few different issues which either caused the camera to not be detected, or for the data type stored in the RGB image matrices to be of the wrong type (uint16_t instead of uint8_t). We were able to fix this by specifying the data type for opencv to use.

The difference in strategy: Pre-lab vs. Lab strategy

Explain the modifications in your original plan.

In the pre-lab the goal was to determine the HSV range for the red color and test it by masking the field. We were also required to determine how to align the center of the red cylinder with the center of the masking field. The original plan for this was to determine the midpoint of the masking field, then find the center of the red cylinder by calculating the average center of the bright zone pixels. This would give a distance between those two points, and the TurtleBot could decrease the distance by turning either clockwise or counterclockwise.

The modifications we made in the code during the lab were adding tolerance for the pixels so we could let the distance between the two centerlines fall within a range. This is because robot’s drivetrain is not incredibly responsive, so precise hit-or-miss conditions would result in either a lot of oscillations or a lot of go-arounds.

Observations and Learnings

- You can detect color using an HSV range.
- By masking a field, you can use those pixel points to identify the center of a colored object.
- You can use a PID controller to fine tune your path by finding the error between your actual path and desired position and decreasing it.
- Using both the lidar and the camera color sensor, we were able to identify an object in two different ways and maneuver around it.