CS 3630 Project 5

Team 1

Name: Fei Ding
GT username: fding33
GTID: 903444656
Team #: 19

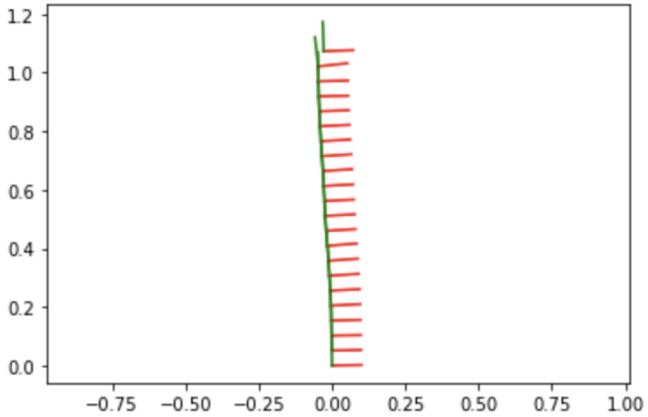
Name: Zhen Jiang
GT username: zjiang330
GTID: 903402987
Team #: 19

Team 2

Name: Francis Kim
GT username: fkim30
GTID: 903281773
Team #: 76

Name: Eric Yan
GT username: eyan30
GTID: 903195858
Team #: 76

1. Provide the screenshot of a plot showing the robot's orientation over time. Why might the final pose look slightly offset/out of place compared to the calculated trajectory (x, y, theta)?



The final pose look slight offset compared to the calculated trajectory because there are many more potential factors to consider, such as friction, the robot's quality, and so on. In order to make our calculation work, we just assume these factor are not significant to make our life easier. However, in the real-world scenario, these factors might have influence on the robot's orientation over time.

2. Explain the steps you used for detecting lane boundaries along the scan lines under “**find_lane_boundaries_on_scan_line**” function

In order to implement this function to detect lane boundaries along the scan lines, we first applied the gaussian filter. The gaussian filter enables us to blur the image slightly, resulting in improved classification performance by smoothing away random noise in the image. Then, we generated a new image by utilizing a max filter. A max filter allows us to extract the maximum pixel value from the corresponding pixel and its neighborhood in the original image. In our scenario, the lane pixels is white, and therefore the pixel has a high intensity value. Applying the max filter let us find the lane pixel much easier. After that, we used low thresholding to find the two peaks on the scan line. Using a high threshold may result in the potential loss of detected lane boundaries. Thus, using a low threshold is a safe solution so we would not miss any of the lane boundary. In the end, we did some manipulation to find the centers of the two lane boundaries.

3. Can your current lane boundary detection method also detect curved lane boundaries? Can you suggest some changes/methods for detecting curved lanes?

Our current lane boundary detection method can detect curved lane boundaries because we use a soft threshold, which basically enables us to detect any potential edge. However, this method has a disadvantage. There might be so many detected edges on the scanline. In our case, this will not be such a big problem since the image taken by the robot only contains the two lanes and the track, which are easy to classify based on the pixel intensity. However, in the real world, the image taken would include so many detected edges, and it is difficult to classify them. One suggested method to detect curved lanes is to use hysteresis thresholding. For hysteresis thresholding, we use a high threshold to start edge curves and a low threshold to continue them. This approach allows us to follow a faint section of an edge we have previously seen, without meaning that every noisy pixel in the image is marked down as an edge.

4. Include a screenshot of your Unit Test results below:

```
......
----------------------------------------------------------------------
Ran 6 tests in 0.015s

OK
```

5. What might be the reasons for error in final orientation of the robot when compared with the ground truth orientation angle that you recorded in Lab part 1?

There are several factors that might result in this deviation. The first one is the friction between the robot's wheels and the ground. When we did our project to calculate the final orientation of the robot, we did not take consideration of friction into our calculation. We just assumed the friction is not significant in order to simplify the calculation. However, in the real world, the friction might play a great role in this type of scenario, and hence it may affect our result to some extent. The second factor is the inner quality of the robot, which may vary greatly even between different robots. Our project is all based on the simple assumption that our robot performs perfectly. However, in the real-world scenario, this is not correct. There might be some issues within the robot's components (such as possibly with the motors), which affect the overall performance of the robot and can lead to error in the final orientation.

6. Specify the challenges faced during the lab

The challenge that we faced during the lab is to implement the function find_lane_boundaries_on_scan_line. The other coding portions were intuitive, because we have pinhole camera equation and a lot of images as guidance. However, we had difficulties when we tried to implement this function, because the implementation was somewhat open-ended. There were some hints given to us, such as utilizing a convolutional filter, using a threshold, and so on, but it didn't specify a certain track to lead us to the goal. Hence, we had to experiment a bit to find the final answer. Additionally, we had to code filters and thresholds, the concepts that we talked about abstractly during lecture. It therefore made our implementation much more difficult, but it was also a great learning experience. We did a lot of work and finally decided to use the gaussian filter, the max filter, and the soft threshold in order to get lane boundaries.