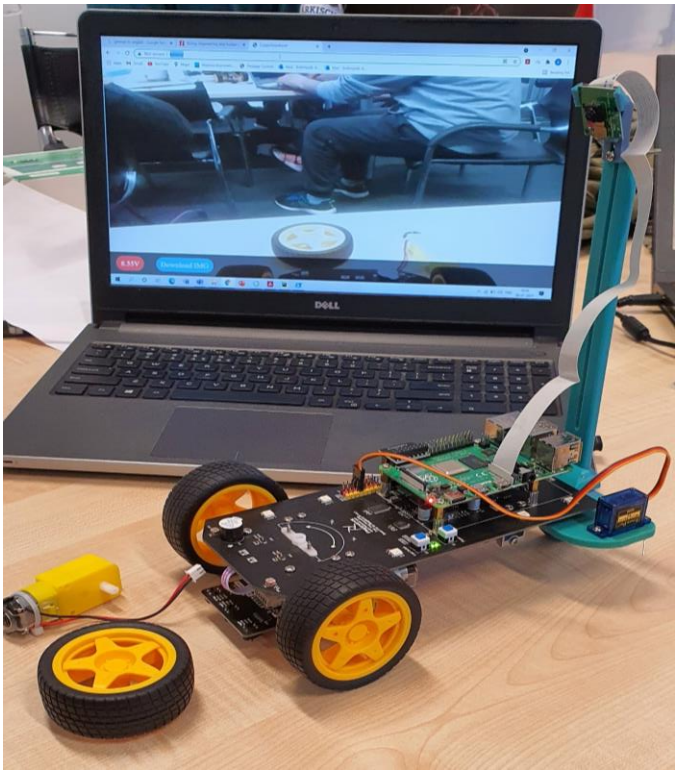
SRH Berlin University of Applied Sciences – SoSe21



Lane Line Keep Assistance Self-Driving Car

Group: Ivision

Under the guidance of,

Professor Klaus Schwarz

By,

Anil Bollimpalli – 3105796

Kumar Rajendrababu – 3105771

Content

Pg. no.

1. Introduction:

Today Tesla, Google, Uber, and GM are all trying to create their own self-driving cars that can run on real-world roads. Many analysts predict that within the next 5 years, we will start to have fully autonomous cars running in our cities, and within 30 years, nearly all the cars will be fully autonomous. Wouldn't it be cool to build your very own self-driving car using some of the same techniques the big guys use? In this paper, we would like to tell our experience about making a self-driving car especially about Lane Line Detection in detail under the guidance of Professor, Klaus Schwarz. As we already know about the hardware part and its installation which we learnt in the class, we shall focus on Lane Line Detection technique. Our objective is to use python and OpenCV to teach our Ivision car to navigate autonomously on a winding single lane road by detecting lane lines and steer accordingly.

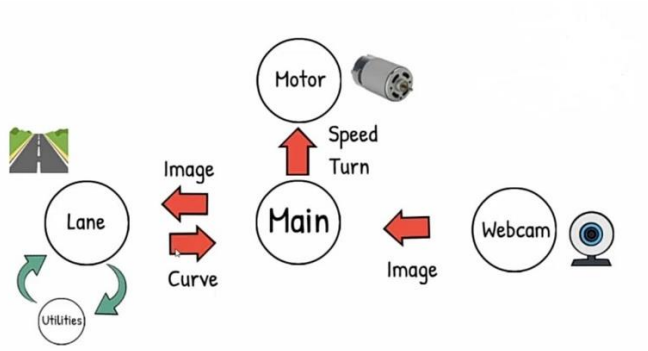


Figure 1: Components of self-driving car

2. Motivation

Once when we were on a road trip with our friends driving an Audi A6 car of a Car-sharing company from Berlin to Hannover during our 2nd semester, the Audi A6, which has both ACC and LKAS did an excellent job on the autobahns, our round trip was of 650kms and 90% of the Autobahn distances was self-driven by the car. The person driving just had to do was to put his hands on the steering wheel (but did not have to steer) and just stare at the road ahead. It was an amazing experience to have the car steering, breaking, and accelerating all by itself when the road curved, or when the vehicle in front of us slowed down or stopped. That experience made us curious

to know how it works. It was a common thought in all our mind's that, wouldn't it be nice if we could replicate this ourselves on a smaller scale? At the same time, we got an opportunity via New Mobility and Automotive Technologies subject from our university to work on self-driving car. In this course, we built Lane Keep Assist System LKAS in our car named *'Ivision'*. Let us discuss about the algorithms behind Lane Line Detection technique.

3. Autonomous Lane line navigation

Currently, there are a few cars from 2018–2019 in the market that have these two features onboard, namely, Adaptive Cruise Control (ACC) and some forms of Lane Keep Assist System (LKAS). Adaptive cruise control uses radar to detect and keep a safe distance with the car in front of it. This feature has been around since around 2012–2013. Lane Keep Assist System is a relatively new feature, which uses a windshield mount camera to detect lane lines, and steers so that the car is in the middle of the lane. This is an extremely useful feature when you are driving on both highways and in bumper-to-bumper traffic.

We have used OpenCV to detect color, edges, and line segments. Then compute steering angles so that Ivision car can navigate itself within a lane.

3.1 Perception: Lane detection

A lane keep assist system has two components, namely, perception (lane detection) and Path/Motion Planning (steering). Lane detection's job is to turn a video of the road into the coordinates of the detected lane lines. One way to achieve this is via the computer vision package and OpenCV. But before we can detect lane lines in a video, we must be able to detect lane lines in a single image. Once we can do that, detecting lane lines in a video is simply repeating the same steps for all frames in a video. There are many steps, so let's get started!

3.2 Isolate the color of the lane

When we set up lane lines for my Ivision car in the living room, we used the blue color tape to make the lanes as we had to build our car to detect blue lines. The first thing to do is to isolate all the blue areas on the image. To do this, we first need to

turn the color space used by the image, which is RGB (Red/Green/Blue) into the HSV (Hue/Saturation/Value) color space. The main idea behind this is that in an RGB image, different parts of the blue tape may be lit with different light, resulting them appears as darker blue or lighter blue. However, in HSV color space, the Hue component will render the entire blue tape as one color regardless of its shading.

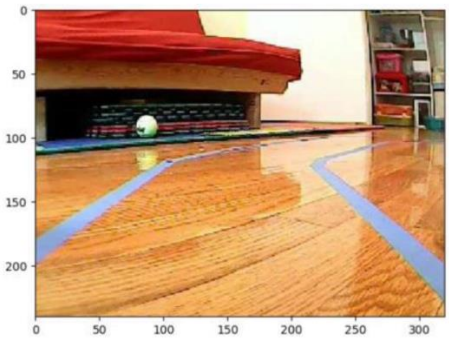


Figure 2: Blue Lane lines

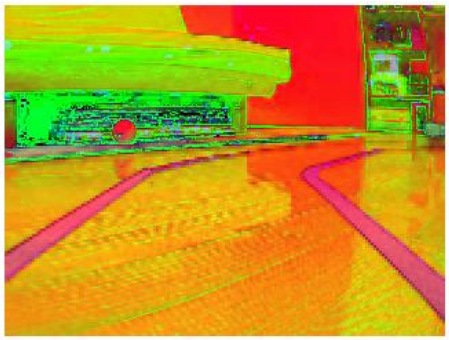


Figure 3: Image in HSV color space

Once the image is in HSV, we can "lift" all the blueish colors from the image. This is by specifying a range of blue. In Hue color space, the blue color is in about 120–300 degrees range, on a 0–360 degrees scale. You can specify a tighter range for blue, say 180–300 degrees, but it doesn't matter too much.

Figure 4: Blue masked image

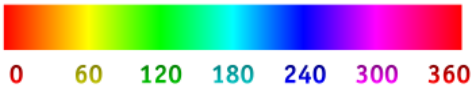


Figure 5: Hue in 0-360 range

3.3 Detecting edges of lane lines

Next, we need to detect edges in the blue mask so that we can have a few distinct lines that represent the blue lane lines. The Canny Edge Detection Function is a powerful command that detects edges in an image.

3.4 Isolate region of interest

From the image below, we see that we detected quite a few blue areas that are NOT our lane lines. A closer look reveals that they are all at the top half of the screen. Indeed, when doing lane navigation, we only care about detecting lane lines that are closer to the car, where the bottom of the screen. So, we will simply crop out the top half. Two clearly marked lane lines as seen on the image on the right.

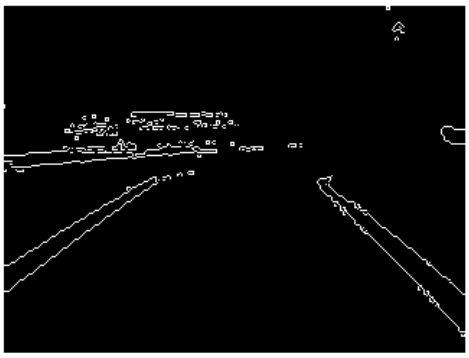


Figure 6: Edges of blue area

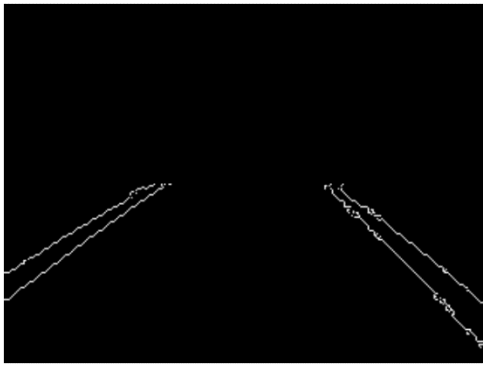


Figure 7: Cropped image

3.5 Detect line segments

In the cropped edges image above, to us humans, it is obvious that we found four lines, which represent two lane lines. However, to a computer, they are just a bunch of white pixels on a black background. Somehow, we need to extract the coordinates of these lane lines from these white pixels. Luckily, OpenCV contains a magical function, called Hough Transform, which does exactly this. Hough Transform is a technique used in image processing to extract features like lines, circles, and ellipses. We will use it to find straight lines from a bunch of pixels that seem to form a line. The function HoughLinesP essentially tries to fit many lines through all the white pixels and return the most likely set of lines, subject to certain minimum threshold constraints.

3.6 Combine line segments into two lane lines

Now that we have many small line segments with their endpoint coordinates (x1, y1) and (x2, y2), how do we combine them into just the two lines that we really care about, namely the left and right lane lines? One way is to classify these line segments

by their slopes. We can see from the picture above that all line segments belonging to the left lane line should be upward sloping and on the left side of the screen, whereas all line segments belonging to the right lane line should be downward sloping and be on the right side of the screen. Once the line segments are classified into two groups, we just take the average of the slopes and intercepts of the line segments to get the slopes and intercepts of left and right lane lines.

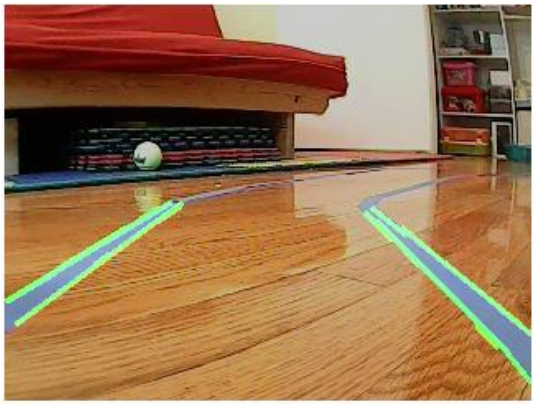


Figure 8: Lane line slopes

3.7 Motion planning: Steering

Now that we have the coordinates of the lane lines, we need to steer the car so that it will stay within the lane lines, even better, we should try to keep it in the middle of the lane. Basically, we need to compute the steering angle of the car, given the detected lane lines.

3.8 Two detected lane lines

This is the easy scenario, as we can compute the heading direction by simply averaging the far endpoints of both lane lines. The red line shown below is the heading direction.

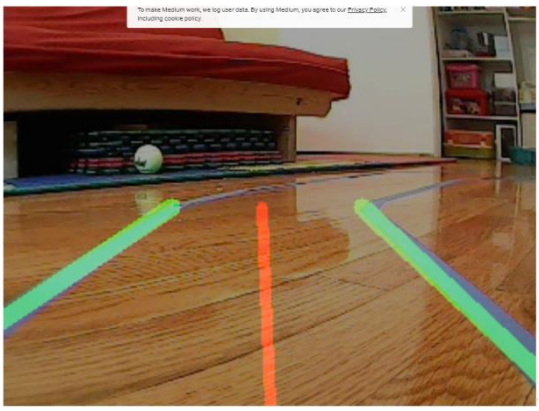


Figure 9: Mean line

3.9 One detected lane line

If we only detected one lane line, this would be a bit tricky, as we can't do an average of two endpoints anymore. But observe that when we see only one lane line, say only the left or right lane, this means that we need to steer hard towards the right or left, so we can continue to follow the lane. One solution is to set the heading line to be the same slope as the only lane line to be followed.

4. Steering angle

Now that we know where we are headed, we need to convert that into the steering angle, so that we tell the car to turn. Remember that for this Ivision car, the steering angle of 90 degrees is heading straight, 45–89 degrees is turning left, and 91–135 degrees is turning right.

4.1 Stabilization of the steering angle

Initially, when we computed the steering angle from each video frame, we simply told the Ivision car to steer at this angle. However, during actual road testing, we have found that the car sometimes bounces left and right between the lane lines like a drunk driver. We then found out that it is caused by the steering angles, computed from one video frame to the next frame are not very stable. You should run your car in the lane without stabilization logic to see what we mean. Sometimes, the steering angle may be around 90 degrees (heading straight) for a while, but, for whatever reason, the computed steering angle could suddenly jump wildly, to say 120 (sharp right) or 70

degrees (sharp left). As a result, the car would jerk left and right within the lane. Clearly, this is not desirable. We need to stabilize steering. Indeed, in real life, we have a steering wheel, so that if we want to steer right, we turn the steering wheel in a smooth motion, andf the steering angle is sent as a continuous value to the car, namely, 90, 91, 92, …. 132, 133, 134, 135 degrees, not 90 degrees in one millisecond, and 135 degrees in next millisecond.

5. Conclusion

Now, we are combining all the individual files namely: Motor.py, Servo.py and Lane_Line_Detection.py together into one main file called Main.py which will drive the Ivision car autonomously by detecting the lanes.

6. Links for

6.1 Lane detection code

https://github.com/kumarnaidu1996/Lane-Line-Detection-for-Self-Driving-Car/blob/main/hand_coded_lane_follower.py

Source: https://github.com/OanaGaskey/Lane-Lines-Detection

6.2 Main file for running the car

https://github.com/kumarnaidu1996/Lane-Line-Detection-for-Self-Driving-Car/blob/ed634850c37e826335ed413d7b5870ed939a76de/main.py

Source:
main.py,Kumar_Rajendrababu&Anil_Bollimpalli,kumarnaidur17@gmail.com

7. Summary of the course

The course “New Mobility and Automotive Technologies incl. Assistance and Autonomous Systems Group” was a very interesting topic for most of us as most of us are from Mechanical background. With the introduction of this topic, we were very much excited to start our work as we had an expectation to learn about new technologies in automotive sector with incorporating autonomous and software solutions into the mechanical world and making todays automotive sector smarter.

Emerging to the end of course we as mechanical engineering students felt we have got a lot of insight about the current technologies and updates in automotive sector which were very much necessary for today's world. We felt the new technologies we have learnt with respect to mobility, autonomous systems and self-driving cars gave us a deeper insight about the developing smarter solutions for vehicles.

We feel that this course was structured in a decent manner, where most of us have gained some knowledge with proper understanding of concepts.

We would like to talk about the professor Klaus Schwarz who had been a great support and motivate since the beginning of the subject. He used to explain everything so well about the content he wanted to deliver to the students. Professor used stay even after the class to resolve everyone's issues and to discuss more about the concept with the interested students. He is very user friendly and very supportive. We were able to schedule meetings with him personally to discuss about the errors we faced while coding for Lane Line Detection. He used to resolve the errors on time and guide me in a right path. It was wonderful and great to work with Professor Klaus Schwarz.

8. Further Expectations

We would expect more about real time object detection using computer vision. Also, as there is more research in the improvements of self-driving cars, we would expect the course contains some challenging research topics which will uplift the autonomous system for the cars. As we worked on Lane Line Keep Assistance technology in the New Mobility and Automotive Technologies course now, we would expect "Vehicle Over Taking on Highway" which includes the lane line detection, object detection and real time surrounding environment observation. We know, this concept will be beyond challenging for students like Swimming in frozen ocean. However, this is going to be a fun ride for the ones who really wants to dive into autonomous vehicle concepts. To support this task, we would expect an advanced and better version of car, than what we already worked in this course. As this technology is not in advanced stage, students along with the Professor can work together to generate the ideas or road path to design algorithm, so that the students can develop and execute the code.