Region Prediction Lane Detection Algorithm

Yunfei Guo
Xingyu Lu

1 Abstract

This paper describes the Region Prediction Lane Detection Algorithm. The algorithm is a vision based lane detection algorithm implemented using C++ with OpenCV API. This algorithm can be used for autonomous driving vehicle. The idea of this algorithm is using the lane detected in previous frame to highlight a region of interest, and using the region of interest to predict where the lane might be in future frame. Compare with detecting lane in entire image, using region of interest will dramatically decrease noise and increase processing speed.

2 Introduction

The two biggest challenge for lane detections are processing speed and noise. To solve these problems, area prediction method is introduced.

The prediction method is to use the lane detected in a previous frame to highlight a region of interest. The region of interest is the prediction of next frame, new Lane will be detected within region of interest. The region of interest is relatively small area in the original image, to process a smaller area will reduce noise and increase processing speed. The overview region of interest is shown in figure 1-1.

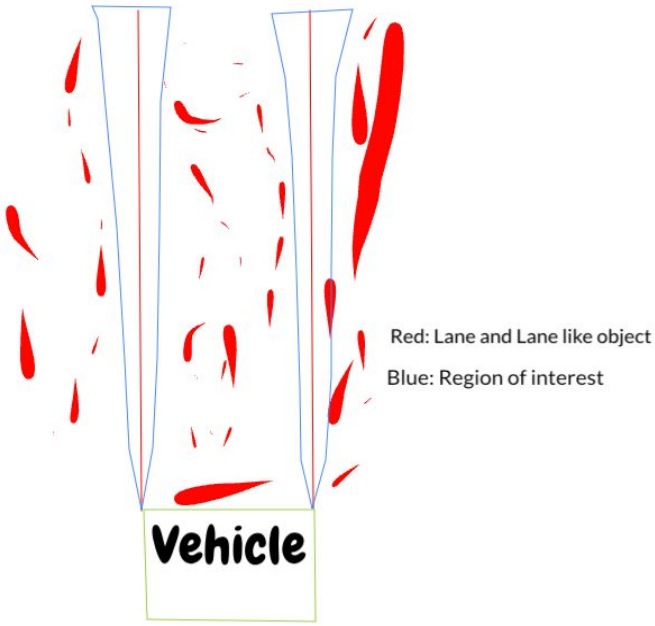


Figure 2-1 region of interest demo

3 Program Design

3.1 High Level Design

The program take a video as input. For the first frame, the program use a very strict filter to filter out any lane-like object, fit a lane and calculate the region of interest. For the second frame and all consecutive frames, the program use a very loose filter to filter out any lane-like object, fit a lane within the region of interest and calculate the region of interest. The flowchart of the algorithm is shown in figure 2-1.

This program will display area of interest on runtime for debug. It does not have output unless record mode is used. When record mode is used, the program will pack the runtime display image into a video.

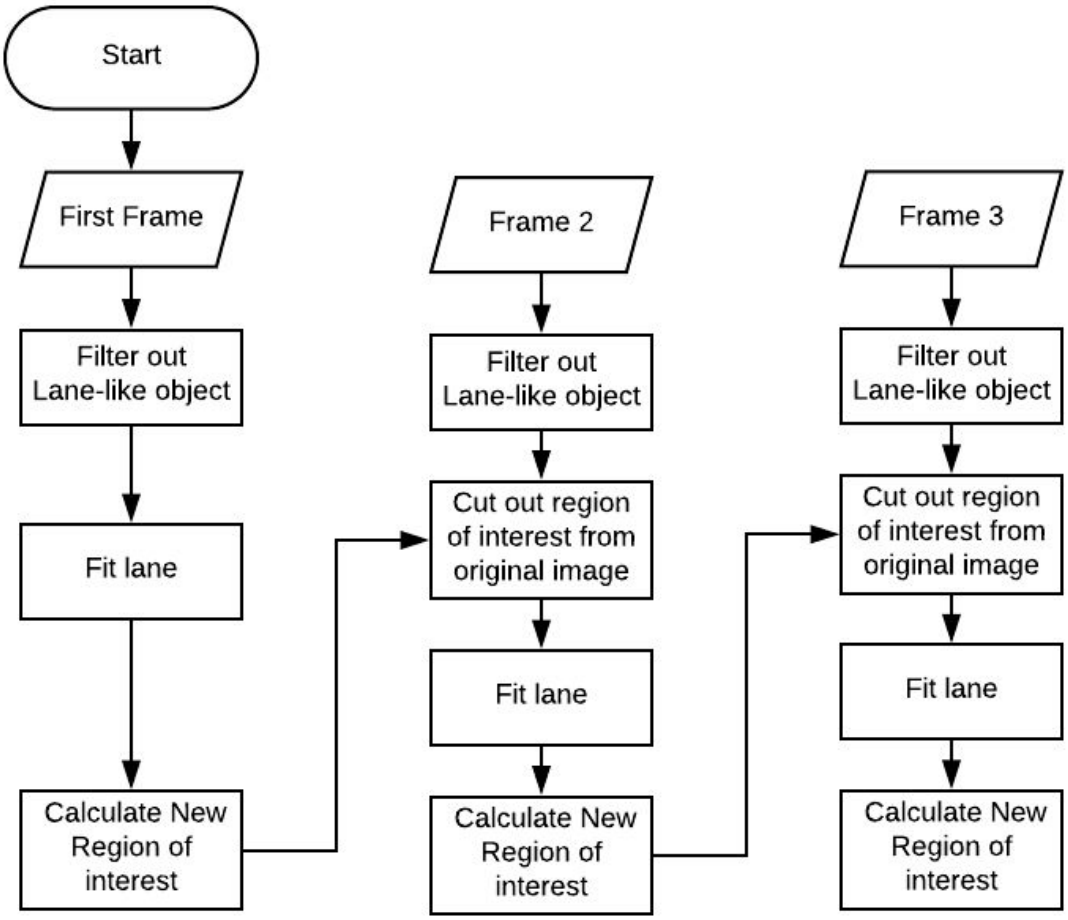


Figure 3-1 program flowchart

3.2 Operation modes and tools

There are 2 operation modes, real road mode and robot test mode. To switch between these different modes, uncomment corresponding defined macro in “detect.hpp”.

When test the program in different environment, the height of the camera various and create a huge difference in the result. Real road mode is used and adjusted for real vehicle running on real road. Robot test mode is used and adjusted for smaller robot, where camera are mount relatively close to ground.

For real road mode and robot test mode, there are slightly difference in filter design and the angle adjustment for IPM(inverse perspective mapping) view.

There is also a record tool used to record the result as a video. To use record mode, uncomment corresponding defined macro in “detect.hpp” and change the output file name and path.

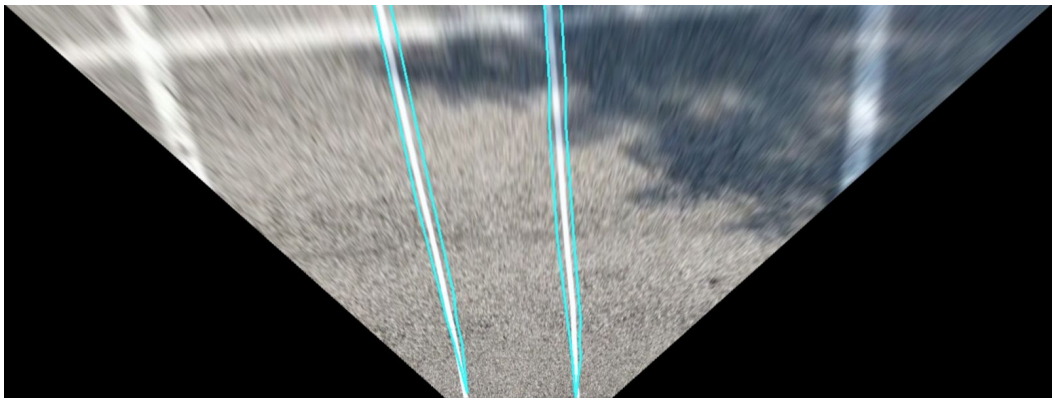


Figure 3-2 Display result in robot test mode

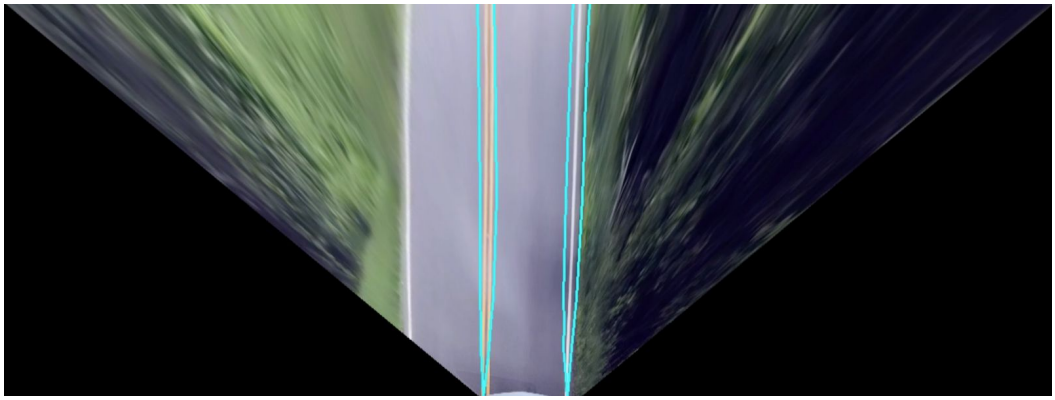


Figure 3-3 Display result in real road mode

3.3 Filter design

There are two filter designs. Filter A is used for robot platform and filter B is used for real vehicle. Filter B is a simplified version of filter A. The flowchart of both filter are shown below.

For filter A, the original image go through Gaussian filter. It will remove texture that is not necessary, for example, the texture of ground. There will be 2 different operations to filter out

lane. The first operation is color filter. The blurred image will transform to HSV and compare with a white color threshold and turned into a binary image. The binary image will go through open morphology and close morphology, where open morphology remove small holes in the foreground and close morphology remove small holes in the background. The second operation is edge filter. The blurred image will transform to HSV and then equalized HSV image to eliminate the influence of shade. Equalized HSV image will be split into 3 channels and Canny edge detector will be performed for each filter and the edge will be highlighted and convert into binary image. Perform a bitwise OR on the binary image from each channel will give the result. Perform a bitwise AND on the binary image from 2 operations above and get the final result of filter A. The final result of filter A will be transformed into Inverse Perspective Mapping(IPM) view for easy lane fit and curvature calculation.

For filter B, it is a simplified version of filter A. It removed color filter and other remain the same.

The flowchart of filter A and filter B are shown in figure 3-4 and figure 3-5.

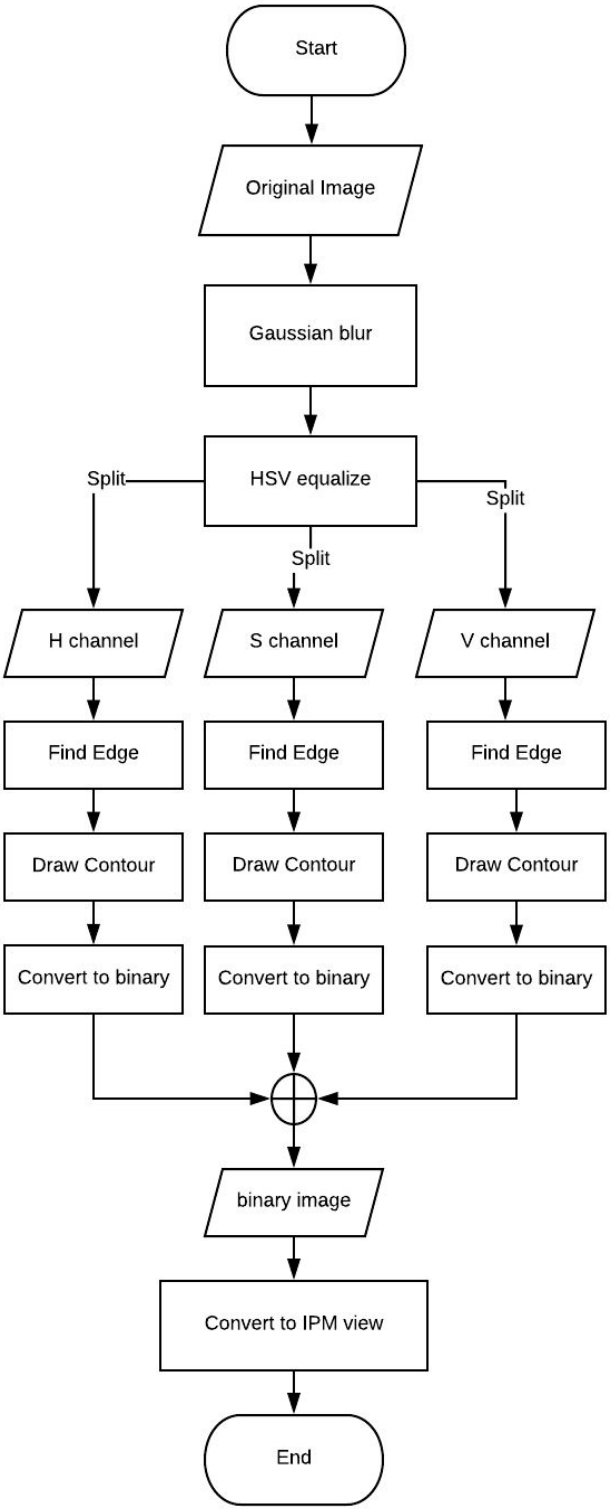


Figure 3-4 Filter B design flowchart

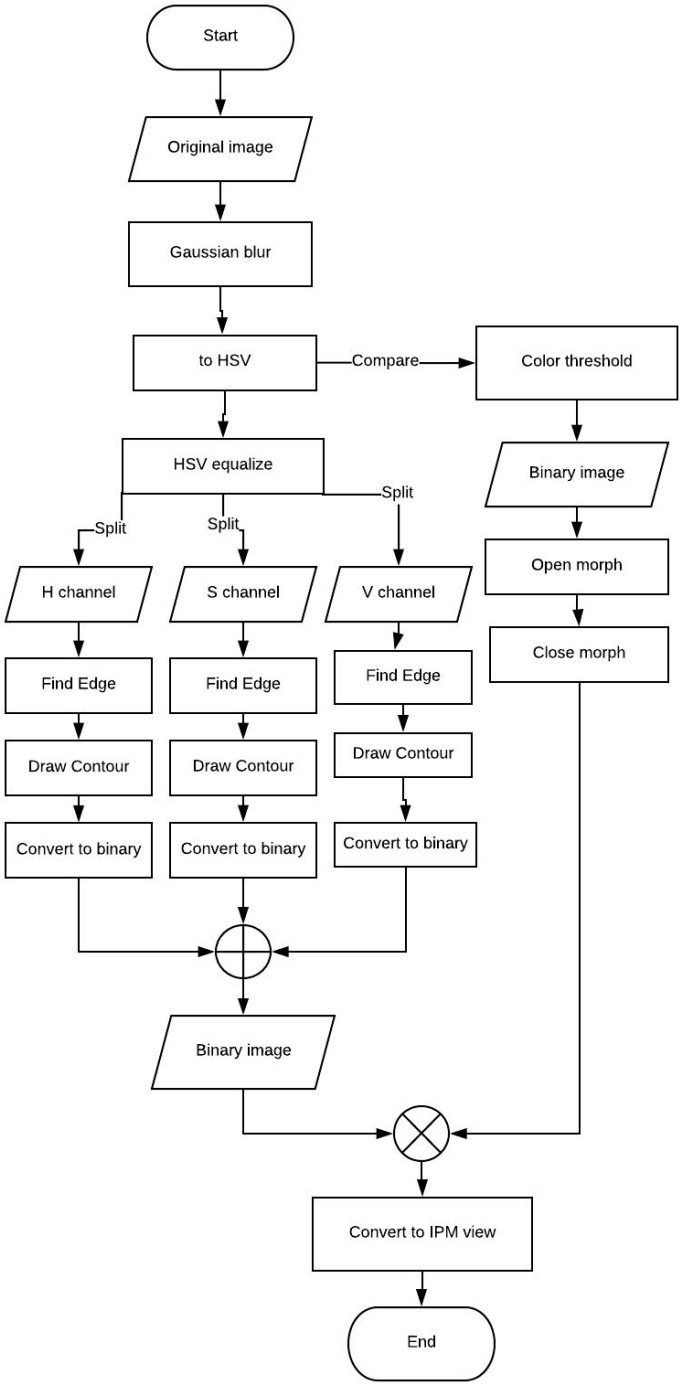


Figure 3-5 Filter A design flowchart

3.4 Lane fitting

The binary image produced from filters are used to fit a lane. The image are similar to figure 3-6. The region enclosed by red line are region of interest, we only fit a lane within the region. The black spots are all valid lane-like pixels. These lane-like pixels are result from the filter discussed in previous section. To fit a lane, the medium point of lane-like pixels are highlight in light blue in each row. The region of interest are split into 8 sub-region by 9 green lines. Distance between these lines are proportional to distance from the vehicle to the lane. At further distance, there will be less valuable information, so the density of the line will decrease. Connect the intersection of the light blue spots and the green lines using dark blue line. The dark blue lane will be the result of lane fitting.

To prevent losing lanes, there are 2 special operations. First, if there is no lane-like pixel at top of the region of interest, the region of interest for next frame will be extend to the top of image for 50 pixels follow the direction of detected lane. Second, if there is no lane-like pixel at bottom of region of interest, the region of interest for next frame will be extend to the bottom of image follow the direction of detected lane.

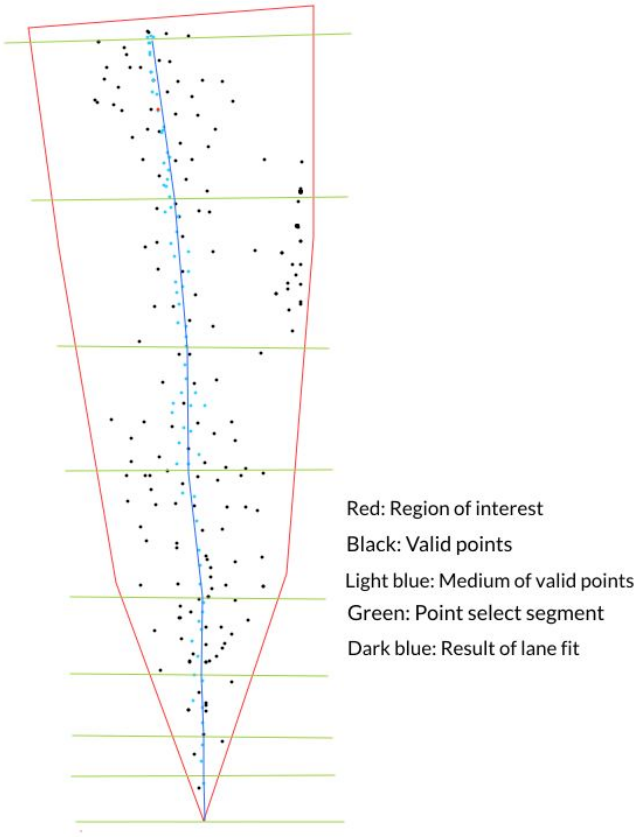


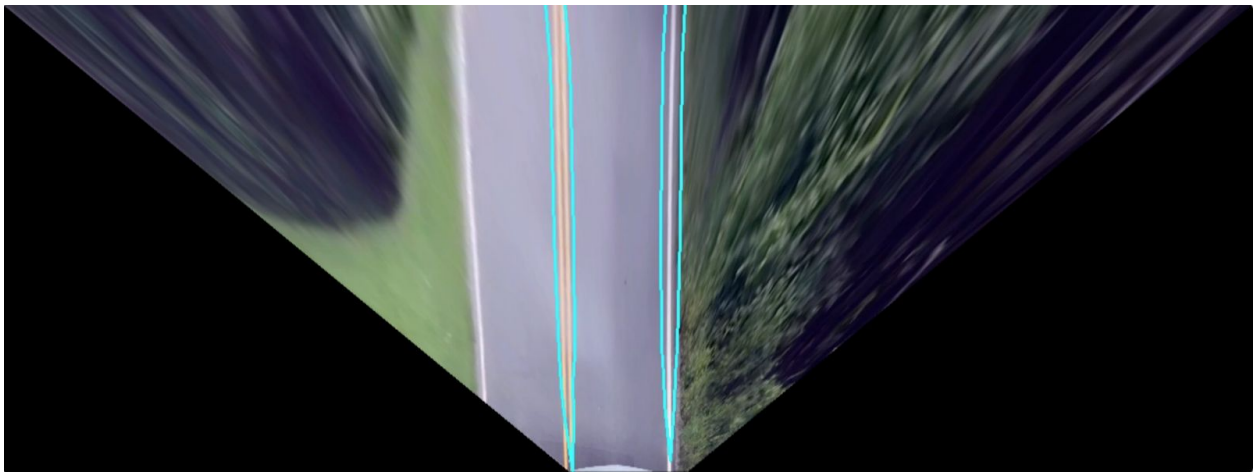
Figure 3-6 Demonstration of lane fitting

4 Testing environment and result

The algorithm are tested using videos having resolution at 1280x720. There are 2 different test environments. The first environment is real road environment. A camera is mount on a vehicle running on a real road with traffic coming on both sides. Multiple lighting condition are included in the test. The second environment is robotic platform environment. A camera is mounted on a tank drive robot and the robot are running in a parking lot. Multiple lighting condition are included in the test.

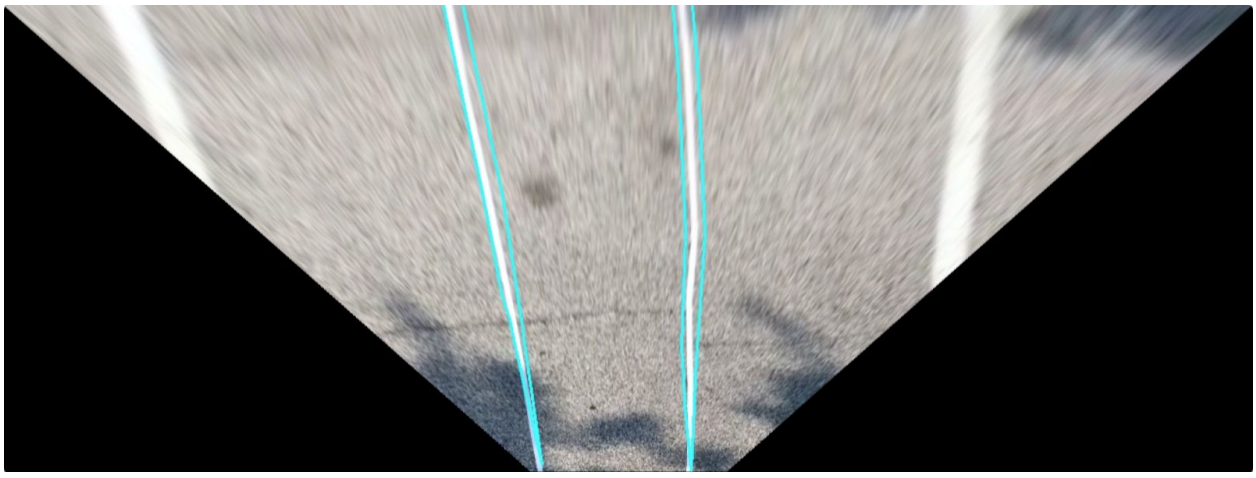
4.1 Real Road

This video is taken on a sunny day at 3:00 pm. This test use video from a camera mount on a vehicle running on road around a mountain. Different lighting condition are engaged. The vehicle start in shade of tree, after a while, the road is half cover by the shade. At the end of the video, there is no shade on the road. The algorithm works well under all conditions.



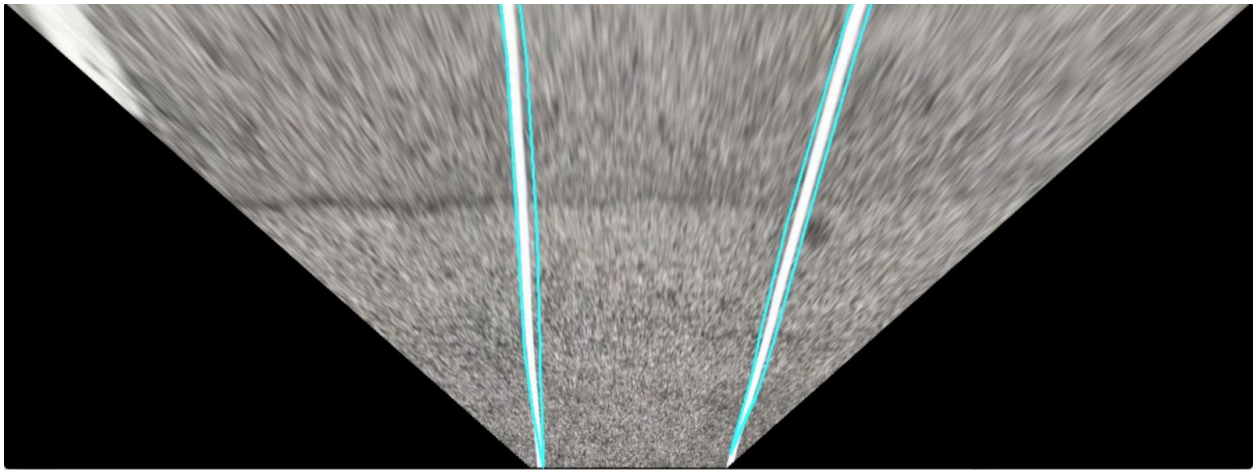
4.2 Sunny

This video is taken on a sunny day at 12:30pm. This test use video taken by a camera mount on a robot to verify the performance under intense lighting condition.



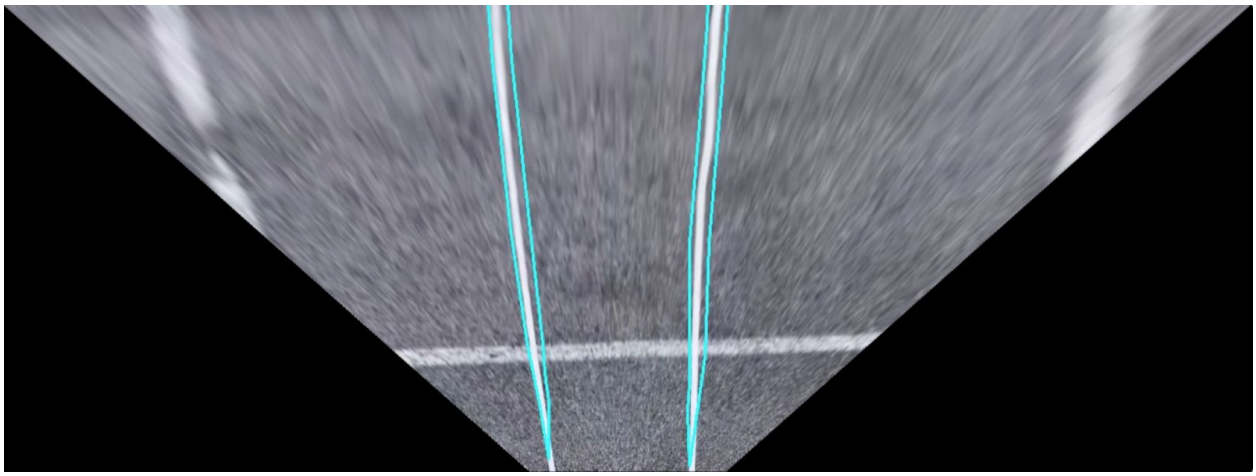
4.3 Cloudy

This video is taken at on cloudy day at 9 am. This test use video taken by a camera mount on a robot to verify the performance under midium lighting condition.



4.4 Low Light

This video is taken on a cloudy day at 8:30pm. This test use video taken by a camera mount on a robot to verify the performance under low lighting condition.



5 Limitations

There are certain limitations for this algorithm.

The first frame detection is not perfect, especially under intense lighting condition. Some of the test videos need to be processed in order to detect lane on first frame.

When test in robotic platform, the shake of the robot will blur the video and affect recognition.

When the robot/vehicle climbing up or down hills, the angle of view changed. The camera will facing up to the sky or down to the ground. When there is no lane in the view of camera, the algorithm will not work.

7 Conclusions

The algorithm works well when the region of interest is successfully selected. It also showed the ability to auto correct some wrong detection. It can handle various lighting conditions. The filter has a flexibility to add or remove any layer to improve performance.

8 Appendix

8.1 Memory usage:

Processing a video has resolution 1280x720 will use around 133MB memory.

8.2 Test videos and code:

Real Road:
Input: https://youtu.be/2CIxM7x-Clc

Output: https://youtu.be/IQAtYWv1a-Q

Robot platform:

Sunny: https://youtu.be/fpBS96Azh_g
Sunny Output: https://youtu.be/mqIsPW7D2J0

Cloudy: https://youtu.be/PG_dfm0TQxk
Cloudy Output: https://youtu.be/SljiRq4iwAk

Low light: https://youtu.be/ta3Hf2GNLsg
Low light Output: https://youtu.be/1yiZ1r7yWdg

Source Code:

https://github.com/yunfei96/Lane_Detect_Research