Finding Lane Lines on the Road

Vincent Young - 20559325

The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report

Reflection

1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

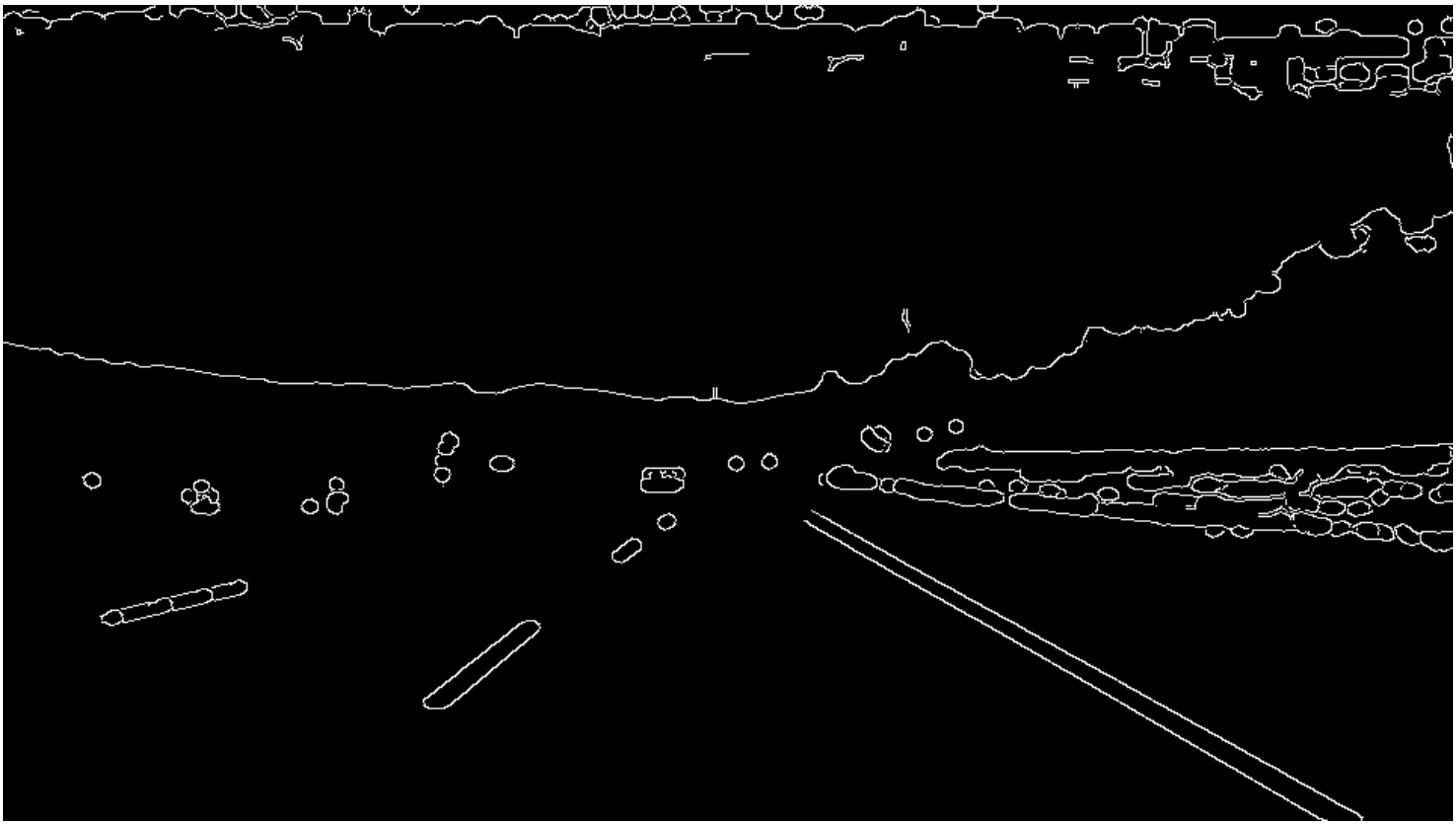
My pipeline consisted of 7 steps. In order to demonstrate the steps taken in my pipeline, pictures will be shown at each stage. First, the test image is included before any modifications.

The first step taken in the pipeline was to convert the image from the RGB domain to the HSV domain, then isolating the V channel. This was done because when examining the individual channels, yellow and white (the colours of the lanes) were visually the easiest to isolate using the V channel. Thus, the V channel was thresholded at 200, assuming 8-bit input images, producing the image below.

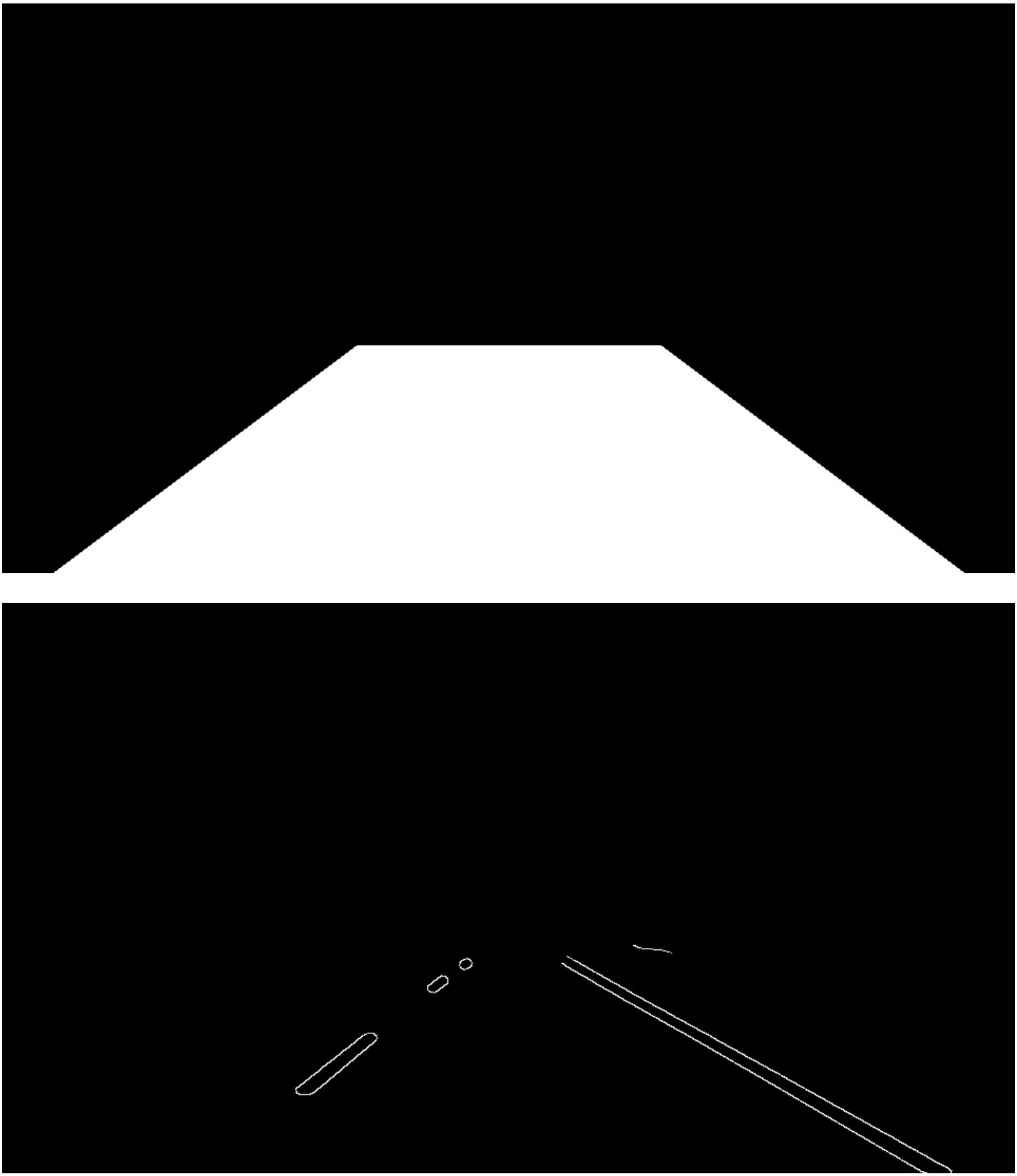
Next, a Gaussian blur filter was applied in order to reduce noise. Thus, if there were any thin, high-frequency features that made it past the thresholding step, they would be blurred out, reducing their ability to skew the results later. This produced the image below.

Next, a dilation operation was performed, in order to enhance the lane features in the image, and ensure that edge detection later would be able to detect a continuous lane, especially as the lane thins out towards the horizon.

A Canny edge detector was then used on the image to produce a binary bitmap of the edges in the image. This edge detector was chosen as it is fairly robust and able to create high quality edges, producing single-pixel thickness edge hypotheses, as well as fairly continuous edges (as opposed to being disjointed, or detecting a lane edge multiple times).

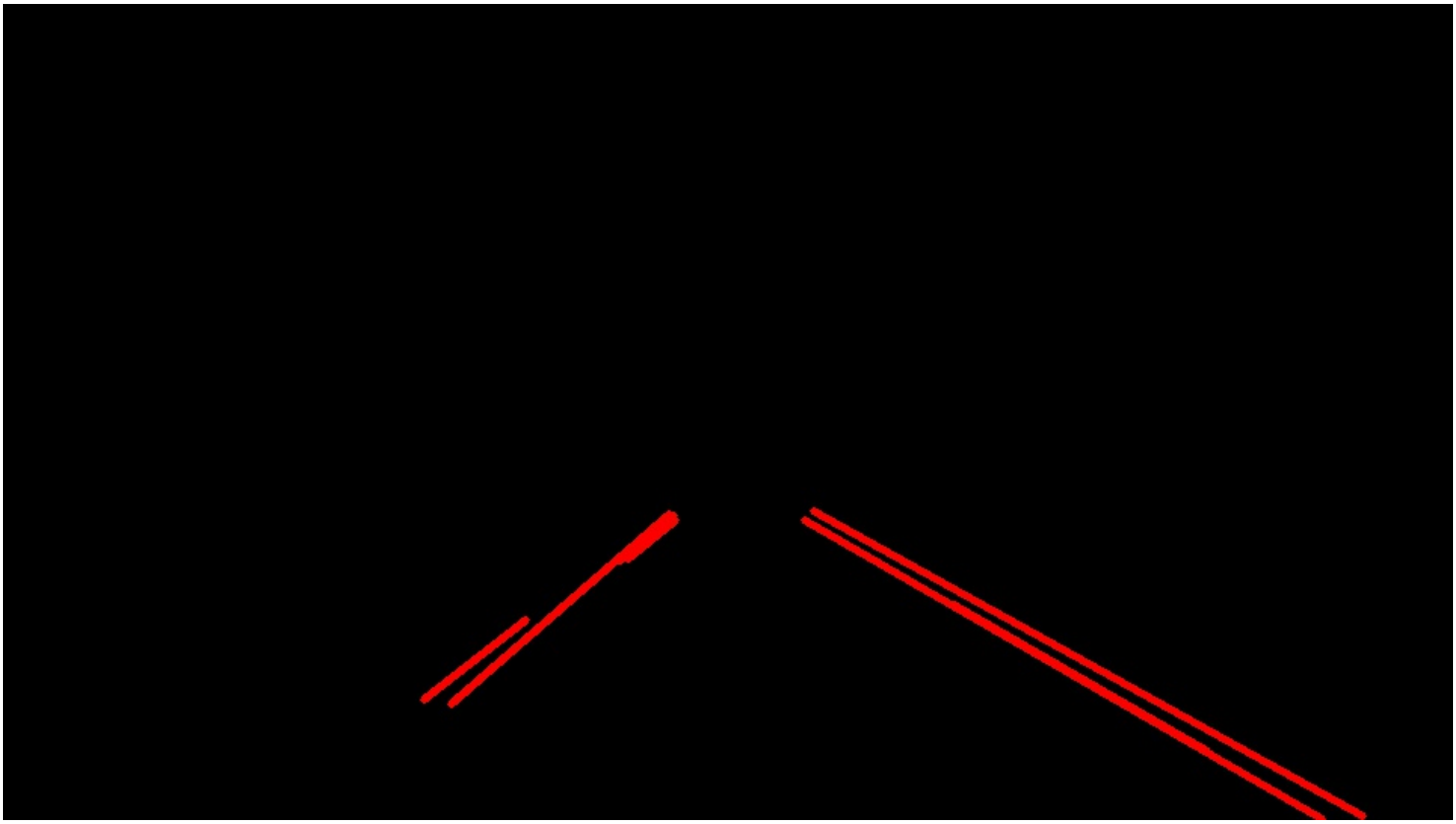


After edge detection was performed, the trapezoidal region of interest (ROI) shown below was applied to generate the image below that. This was done to remove the other edges detected in the image, and isolate only the edges corresponding to the lane markings. The ROI was chosen as this is where we expect to find the lane markings in the image.

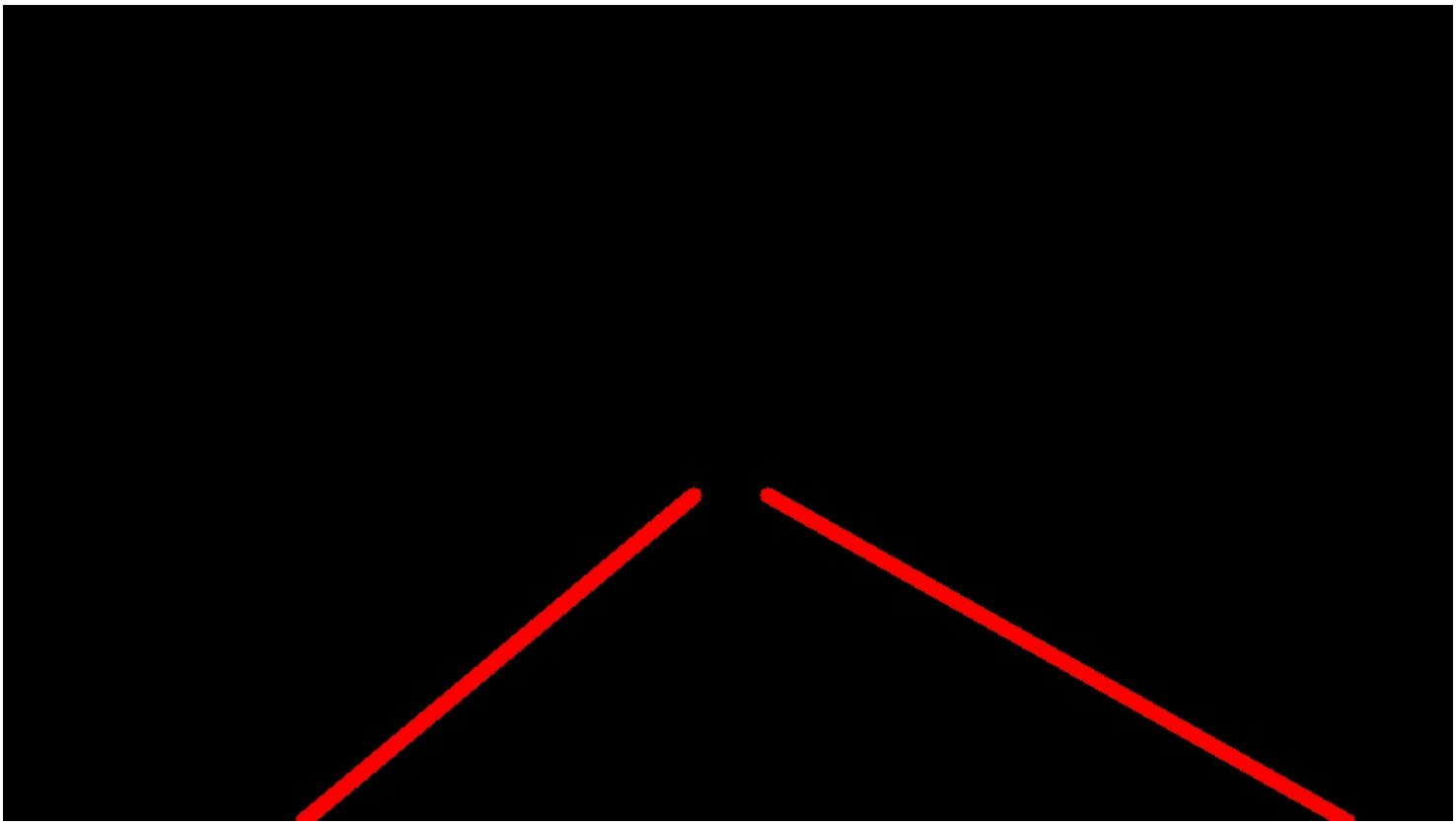


These edges were then fed into a Hough Line Transform in order to generate hypotheses for line segments that are present in the image. This is done to move from the bitmap created by the Canny detector to actual

line segments (with proposed endpoints) that can then be used to come up with exact lane markings.



The lane lines shown above were then combined in a weighted average, and run through a linear regression operation. The (x, y) coordinates of the line segment endpoints were used as the data, and the lengths of the lines were used as their weights in this operation, placed in the draw_lines() function, producing the image below.



This final line drawing was overlaid over the original image producing the combined image below.

2. Identify potential shortcomings with your current pipeline

With the current pipeline, if lighting conditions were to change drastically, the isolated V channel in the HSV domain image may become insufficient for detecting the lanes. The thresholding on this works for the test images and videos provided, but the hard-coded threshold is not robust.

Another potential shortcoming with the pipeline is that none of the test data provided has a car appear in the ROI, and thus the pipeline doesn't account for rejecting any lines that may result from a car appearing in frame.

The pipeline also assumes that lane lines will be fairly straight when shown in frame. This is reflected by the choice of using a Hough Line Transform to detect line segments in the frames. If the lanes had significant curvature (as in the case of a sharper turn), the detection would not be robust.

3. Suggest possible improvements to your pipeline

In order to improve the pipeline, the step in which thresholding is performed on the isolated V channel in the HSV image could be replaced by more robust colour picking using the colour carrying channels, hue and saturation (H and S). Instead of thresholding by high intensity, a colour picking algorithm could account for a wider variation of yellow and white objects in the frame. This could be done through experimentation (looking at a wide variety of lane marking images) or through research to determine the appropriate ranges on these values to apply.

Another potential improvement is to improve the rejection of irrelevant lines detected in the Hough transform, as would happen if another car entered the center of the frame. This could be done by taking any detected lines and extrapolating them to the bottom of the frame, and ensuring they fall in the range that the lanes are expected (left-most quarter and right-most quarter potentially).

Another improvement that could improve the consistency of the lane detection in the video cases is to use inter-frame data. Even assuming the video is shot at a slower 30fps, there is only 1/30 of a second between frames, meaning there will not be too much variation with where the lanes are between frames. Thus, the location of the lanes detected in the previous frame can be weighed into the detection of the lane in the current frame, or replicated in the current frame in the case that a hypothesis for a lane can't be generated.

Finally, in order to account for sharper turns, a generalized Hough Transform or other algorithm can be used to fit the lanes, allowing for detection of non-linear lane shapes in the frame.