Visual Recognition Assignment

Naga Sri Vaishnavi Dhulipalla
IMT2017514

Purpose of this Assignment :

1. To detect road lane markers and the road region.
2. To classify images into three main categories of Night, Portrait (face), Landscape using nearest Neighbors classifier.

1. Road Lane Marker Detection :

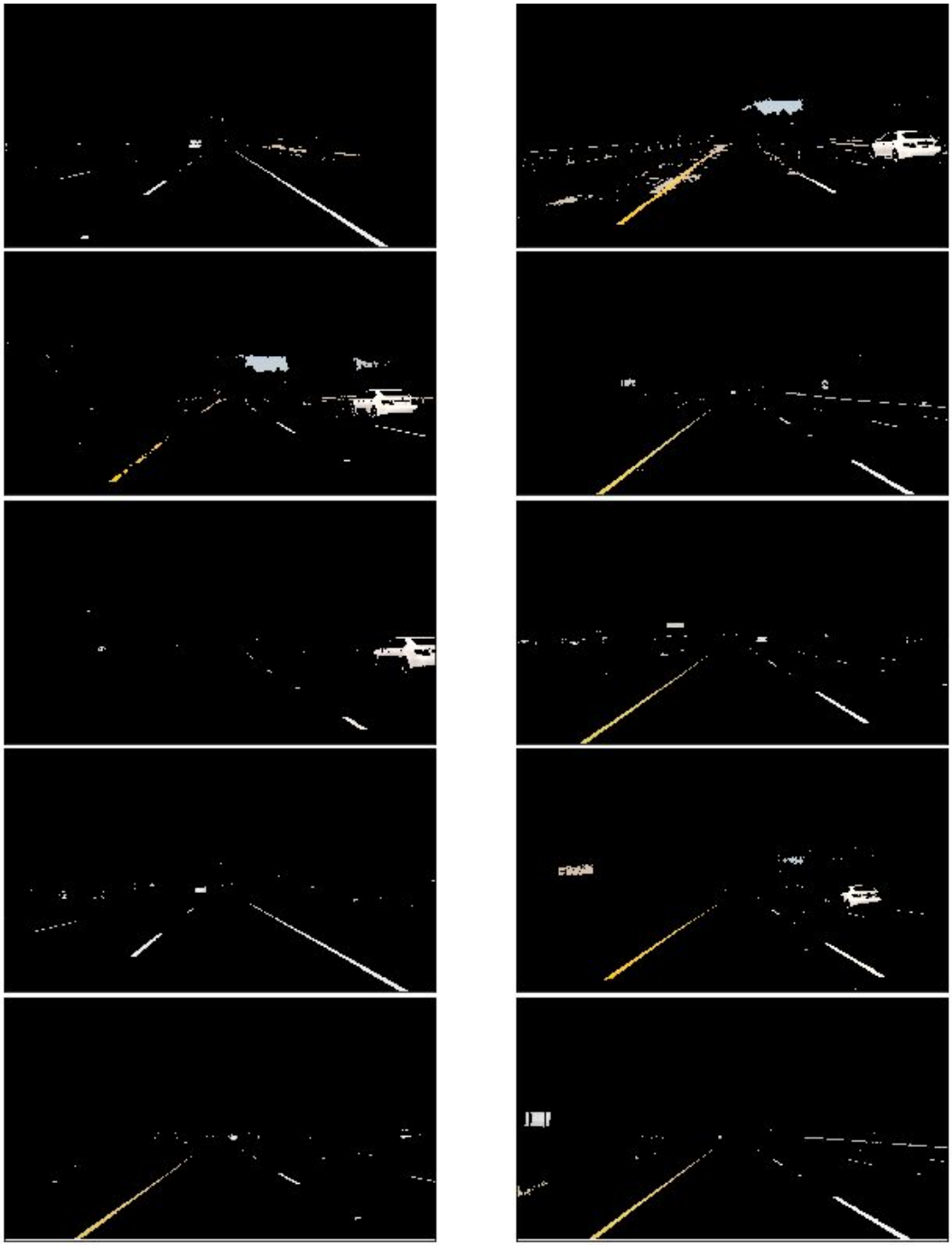
To achieve this objective, I have used Python and OpenCV. The following techniques are used :

a. Colour Selection
b. Canny Edge Detection
c. Region of Interest Selection
d. Hough transform Line Detection

I have written a function *show_images()* to display and save the displayed images into a seperate folder. When the input images are displayed, we can notice that the lines are either in yellow or white.

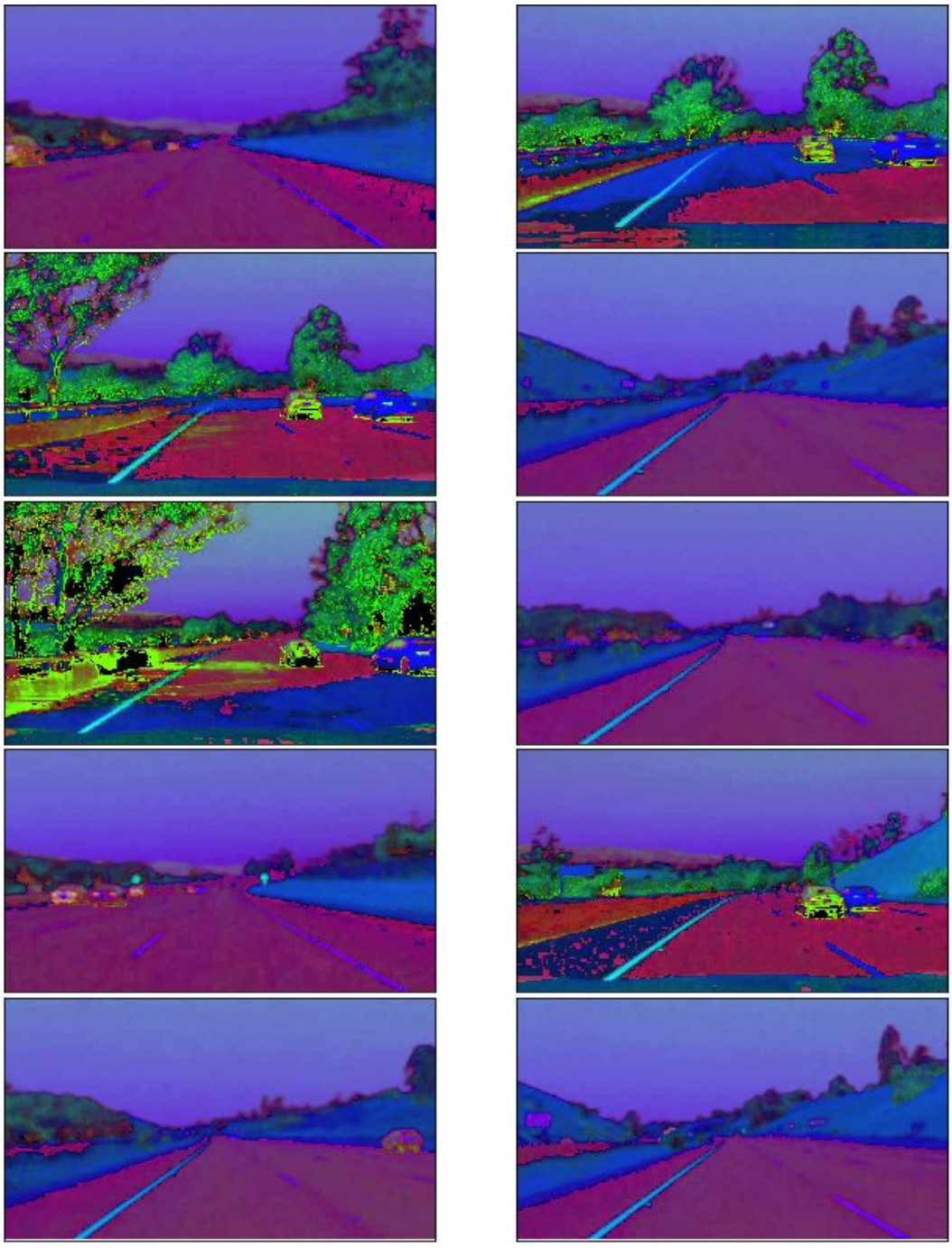
a. Colour Detection :

The images are loaded in the RGB colour space. I have tried selecting only yellow and white colours in the images using RGB channels.

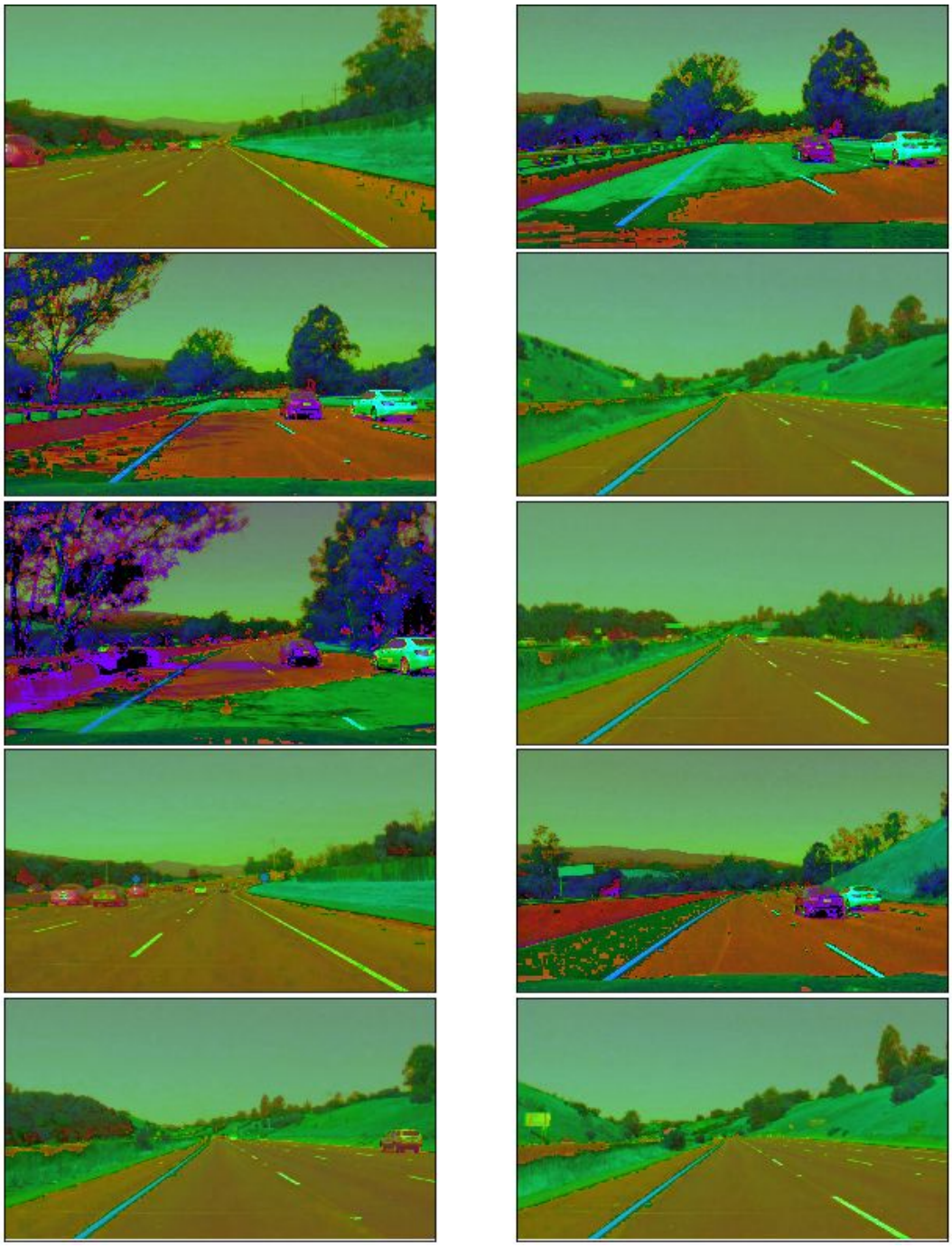


We can observe that lines are detected good except for the images where the lines were not clear due to dark shade from the trees.

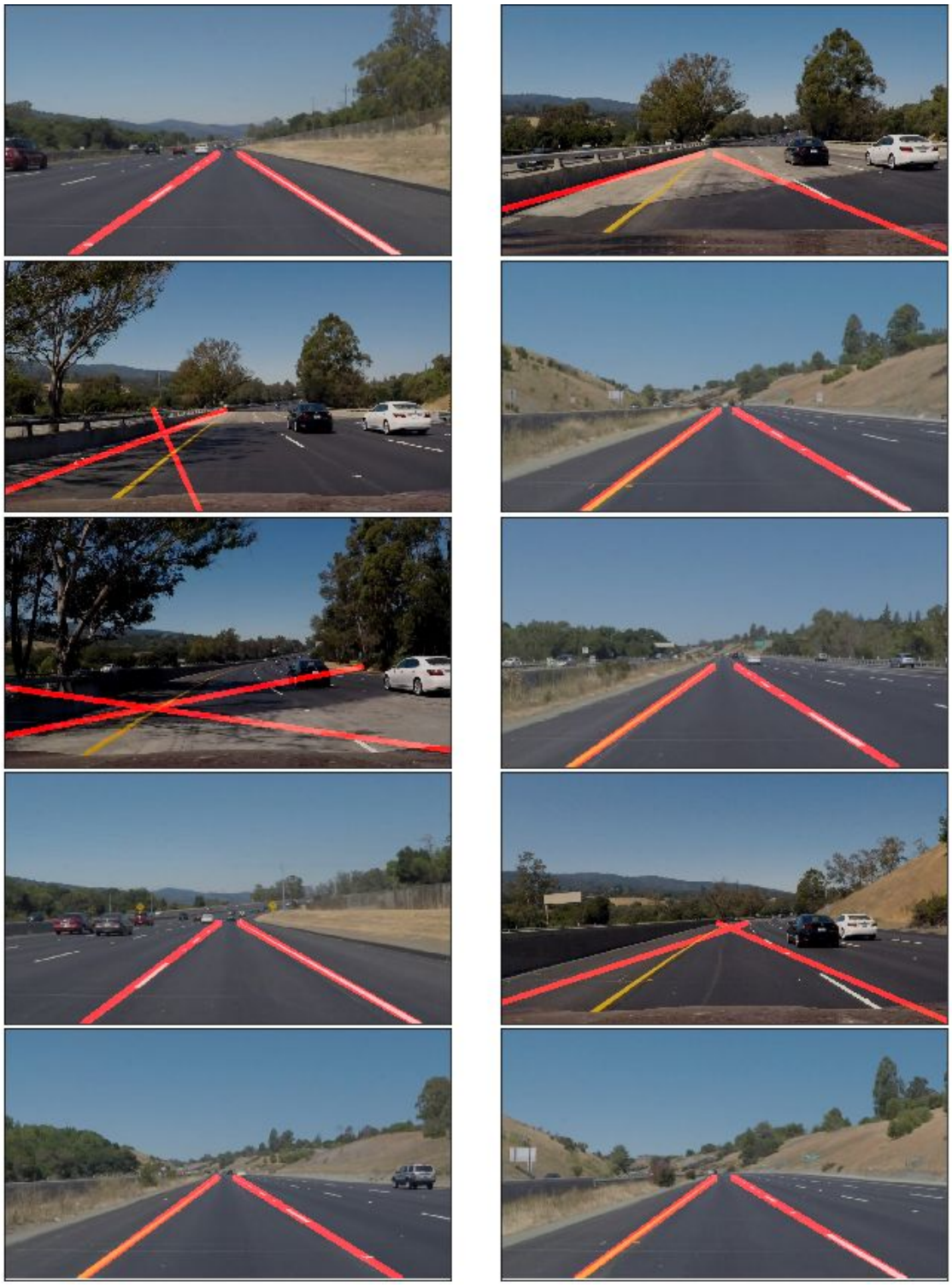
So, I have tried converting RGB images to HSV colour space. Now, the yellow lines are pretty clear but the white ones are less clear.



So, now I have converted the images to HSL colour space. Both the white and yellow lines are clearly recognizable. Also the lines under the shade are clearly visible.



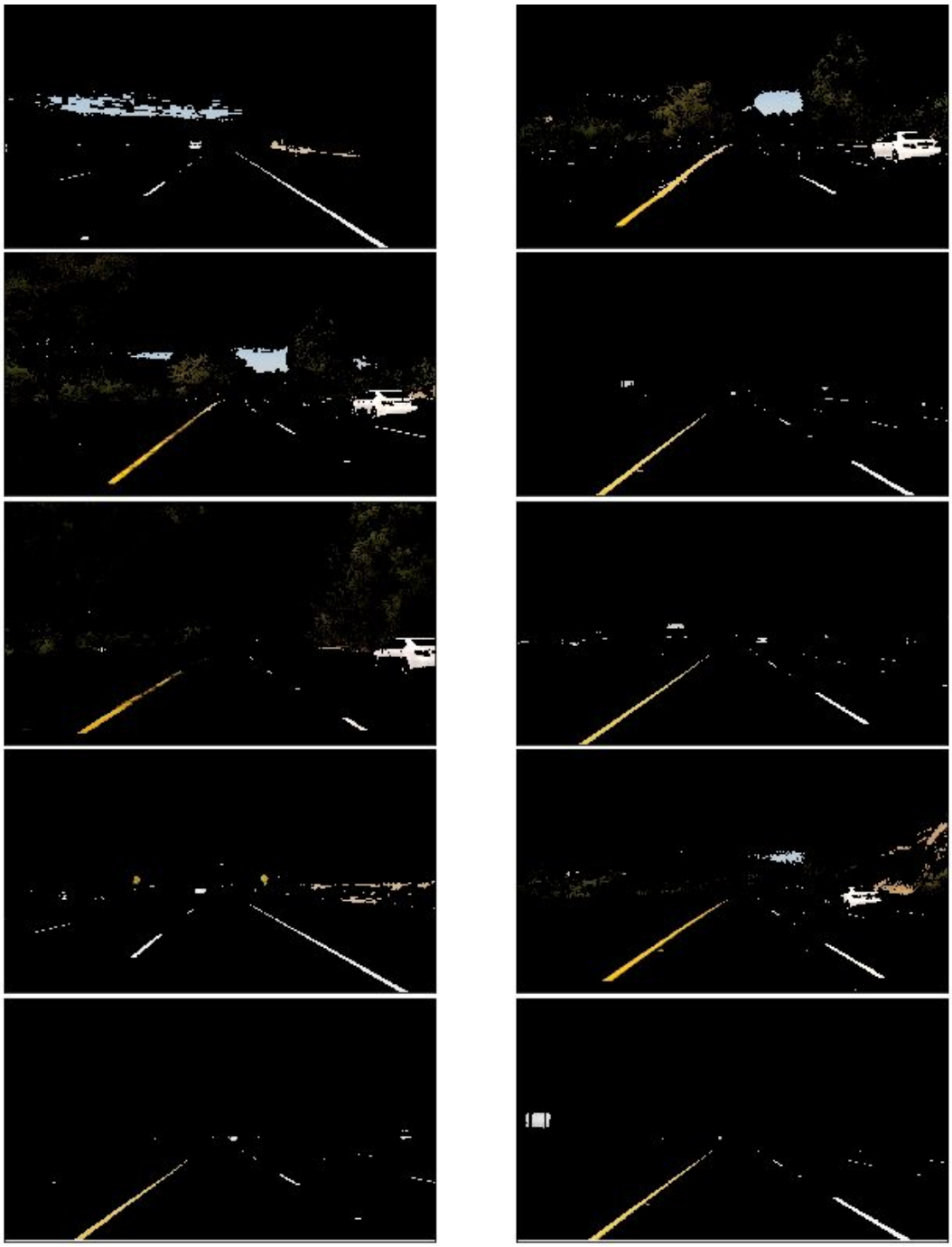
The reason I have chosen to do this extra step is to detect the lane markers which are unclear or under shade. Applying Gaussian blurring, Canny edge detection and selecting the region of interest gave us these results. Adding this extra step of choosing the right colour space gave us more accurate results.



After choosing a colour space, I have built a filter to select the white and yellow lines. I have used *cv2.inRange* to filter white colour and the yellow colour separately; *cv2.bitwisw_or* to combine the two binary masks(white and yellow)

and *cv2.bitwise_and* to apply the combined mask onto the original RGB image.

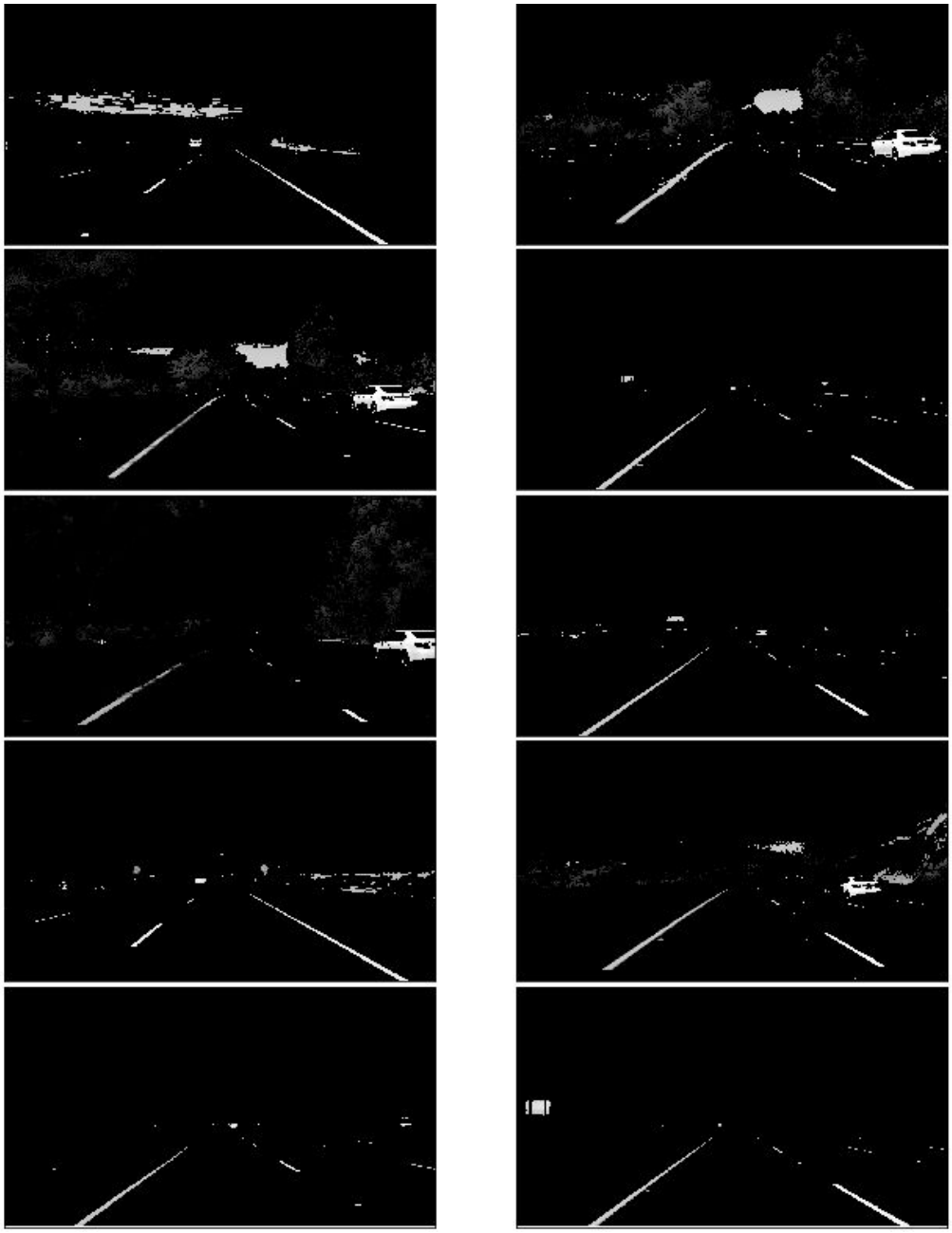
For white colour, I have chosen high **L** value and did not filter **H,L** values. For yellow colour **H** was chosen to be around 30 and relatively high **S** to exclude the environment around. These combined masks filtered yellow and white lines very clearly.



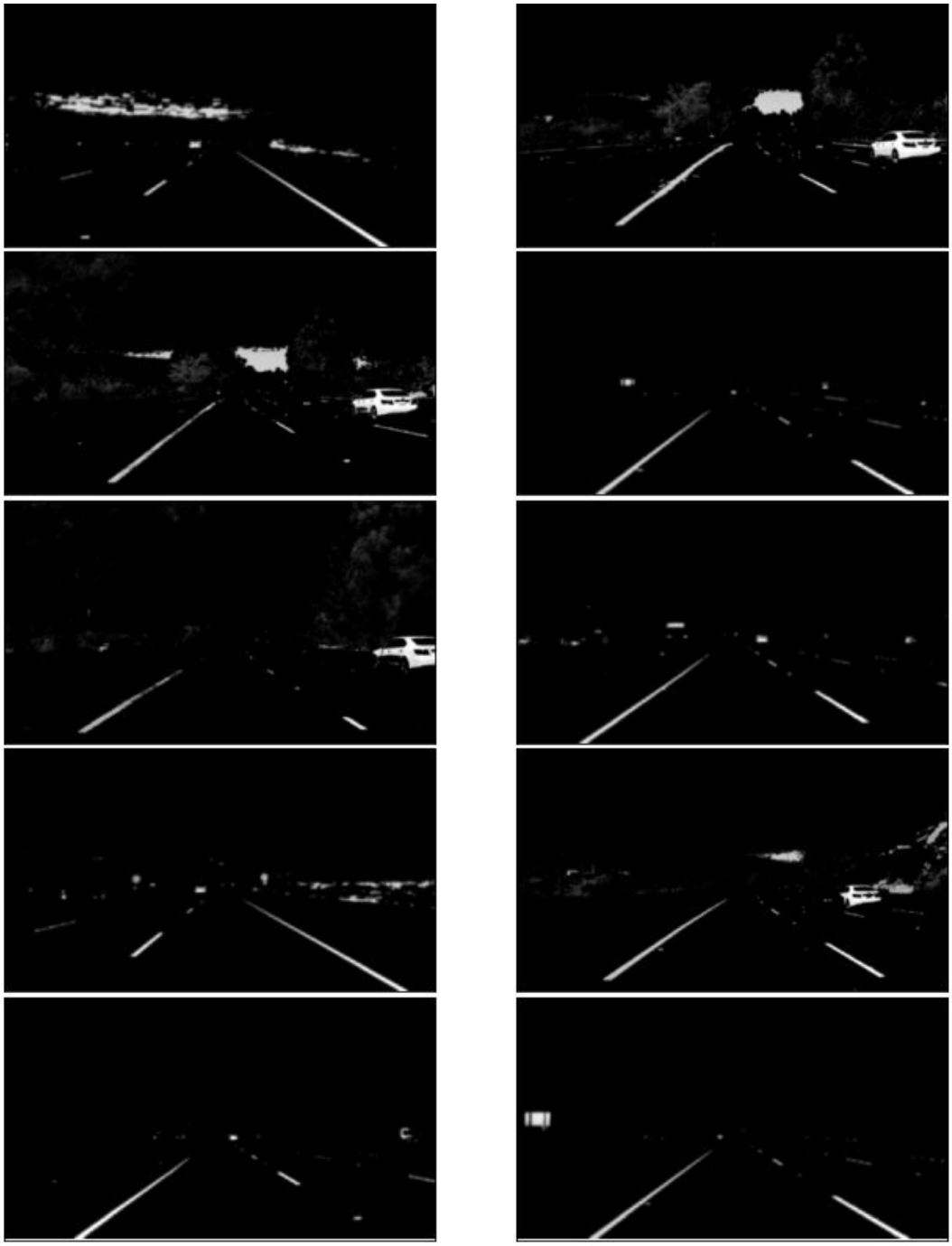
b. Canny Edge Detection :

We want to detect edges in order to find the lane lines accurately. I have used *cv2.cvtColor* to convert images to grayscale; *cv2.GaussianBlur* to smooth out the rough edges and *cv2.Canny* to find edges.

The imaged should be converted to gray scaled images in order to detect edges as canny edge detection measures the magnitude of pixel intensity changes or gradients.



When there is an edge, the pixel intensity changes rapidly which we want to detect. But before that we need to smoothen out the edges to avoid noisy edges being detected. I have used *cv2.GaussianBlur* (Gaussian Smoothing) for the same.



Gaussian Blur takes a *kernel_size* which must be positive and odd. I have tried using 3,9,11,15 and 17 and realized that the bigger the value is the more blurry the image becomes.

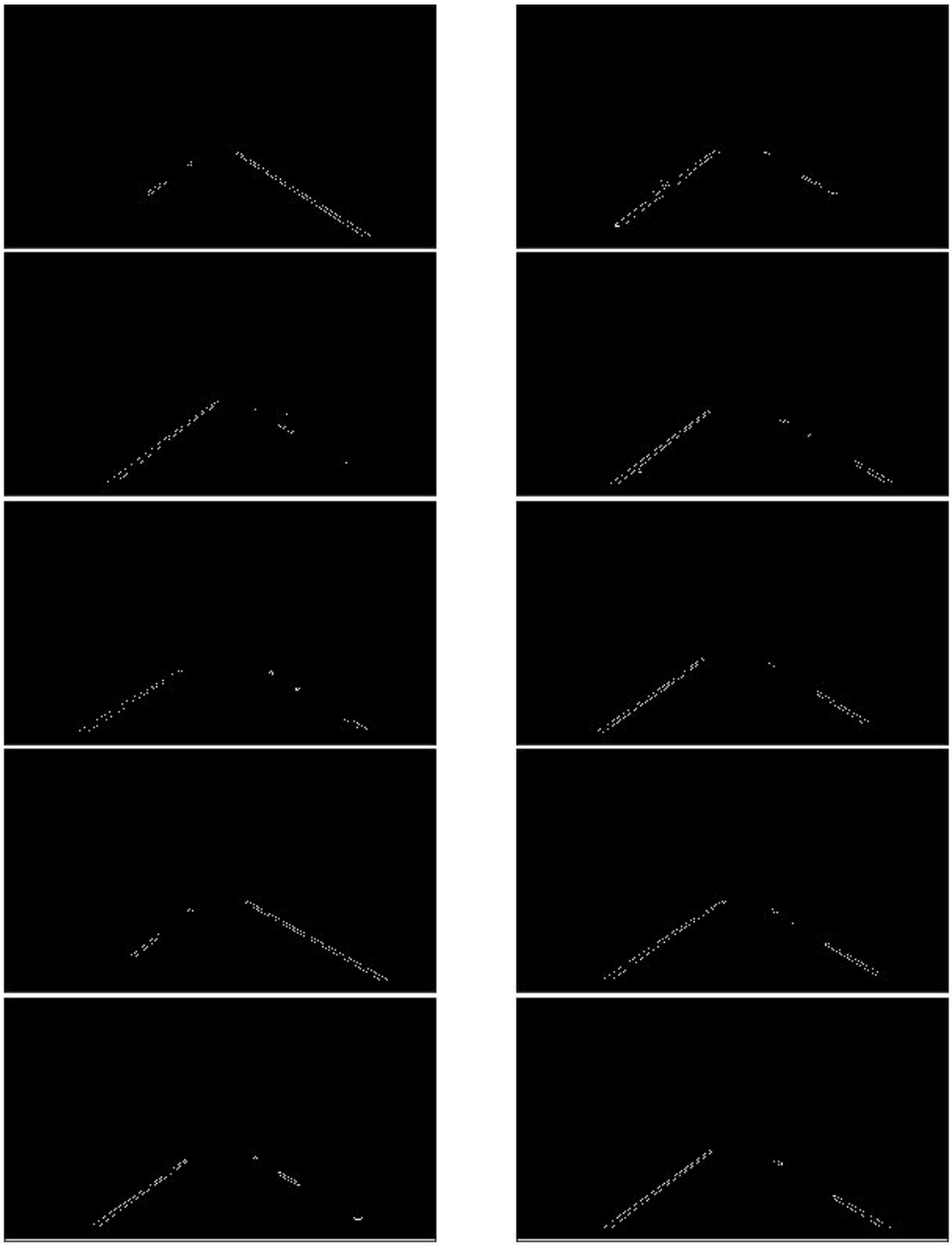
Now, I have done canny edge detection and chosen the thresholds as 50 and 150. Double thresholds are used as follows :

a. If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge.
b. If a pixel gradient value is below the lower threshold, then it is rejected.
c. If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.
d. Canny recommended an upper:lower ratio between 2:1 and 3:1.
e. If the higher threshold is too high, we find no edges and if it's too low we find too many edges. So, first set lower value to zero and adjust the upper one. Then adjust the lower one to discard the weak/noisy edges.

c. Region of Interest Selection :

When finding lane lines, we don't need to check the sky, hills and tress in the background. So, we exclude outside the region of interest by applying a mask.

After applying the mask, I got lanelines but we need to recognize them as lines. Especially, two lines: left lane and right lane.



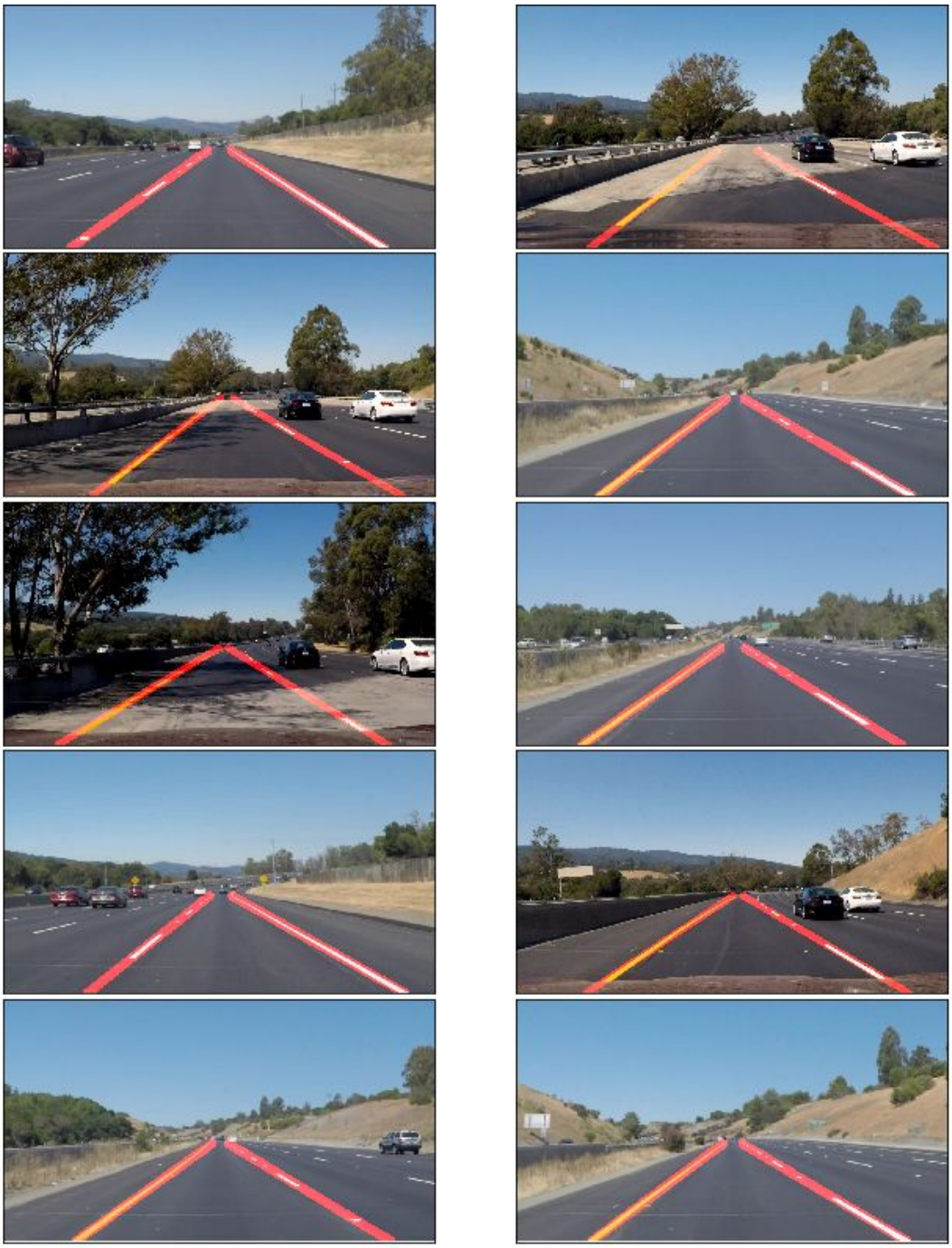
d. Hough Transform Line Detection :

To detect lines in the edge images, I have used *cv2.HoughLinesP* . There are several parameters you'll need to tweak and tune:

- rho – Distance resolution of the accumulator in pixels.

- theta – Angle resolution of the accumulator in radians.
- threshold – Accumulator threshold parameter. Only those lines are returned that get enough votes (> threshold).
- minLineLength – Minimum line length. Line segments shorter than that are rejected.
- maxLineGap – Maximum allowed gap between points on the same line to link them.

There were multiple lines detected for a lane line. So, I have averaged and extrapolated the lines to get one marking for one lane. To draw the lines, I have converted the slope and intercept into pixel points.



This is the final output of the lane line marker detection.