Finding Lane Lines on the Road

The pipeline steps **1**

Example of the pipeline implementation **2**

Improvement **2**

Introduction

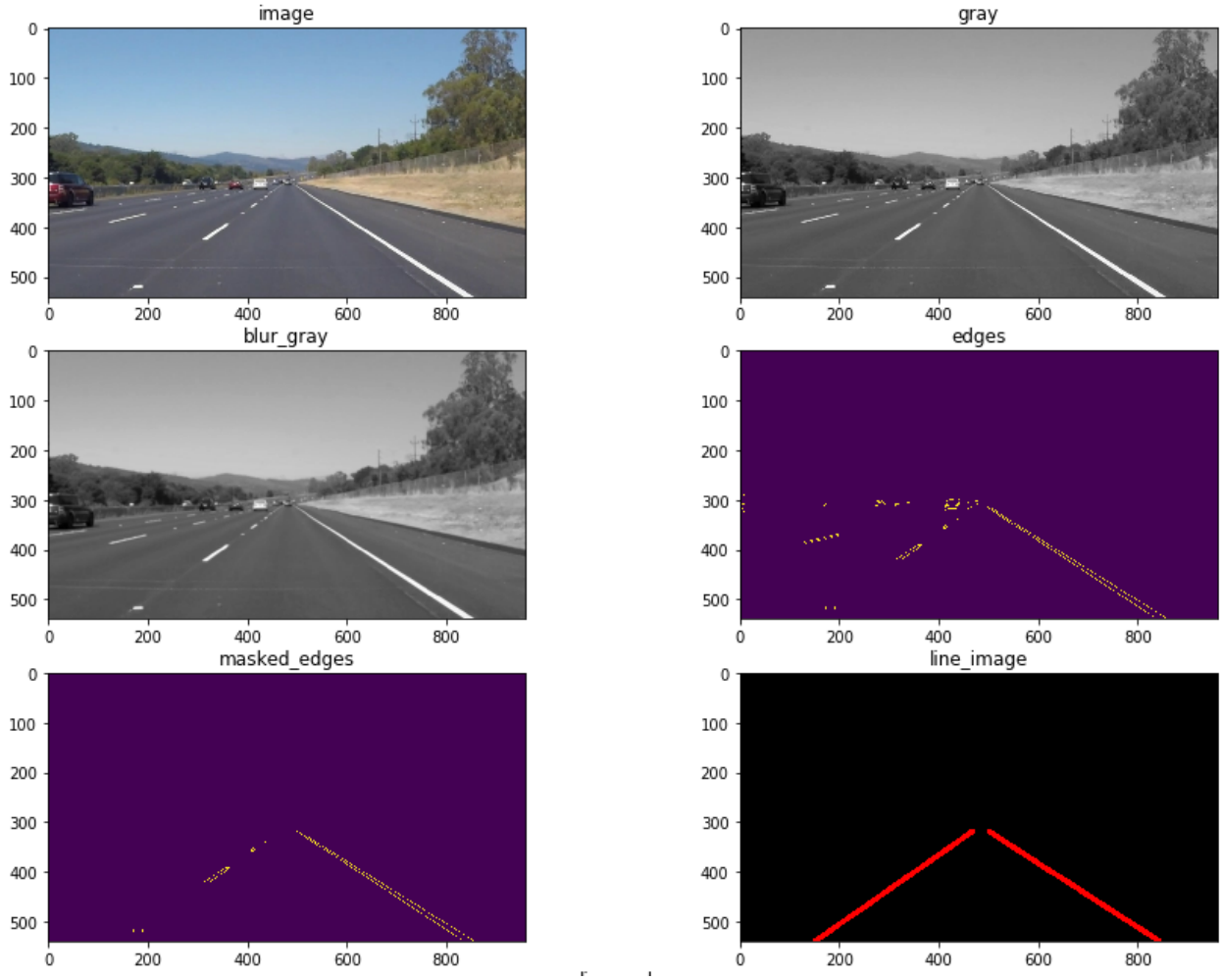
The main target for this project is to be familiar with the main concepts of computer vision. So we are implementing a Lane detection algorithm. The concept covered in the project are:

1. Be familiar with OpenCV library in python
2. The concept Hough transform and how it is used in the computer vision
3. What is the Canny edge detection and how we are using it
4. Main concept of detecting objects based on color, shape, orientation, position in the picture.

The pipeline steps

1. Convert the image to grayscale.
    a. We used the function “cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)”
    b. This is done because the colors are not important in Lane detection because the model will be learning from the geometry present in the image. The image-binarization will help in sharpening the image by identifying the light and dark areas.
2. We applied a Gaussian filter to the image.
    a. We used the function “cv2.GaussianBlur(img, (kernel_size, kernel_size), 0)”
    b. Image blurring is achieved by convolving the image with a low-pass filter kernel. It actually removes high frequency content (e.g: noise, edges) from the image resulting in edges being blurred when this filter is applied. We should specify the width and height of the kernel which should be positive and odd.
3. We applied canny edge detection.
    a. We used the function “cv2.Canny(img, low_threshold, high_threshold)”
    b. Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stage.
4. We defined the region of interest
    a. We used the functions “cv2.fillPoly(mask, vertices, ignore_mask_color)”
5. We applies Hough line transform
    a. We used the function “cv2.HoughLinesP(img, rho, theta, threshold, np.array([]), minLineLength=min_line_len, maxLineGap=max_line_gap)”
    b. After detecting the two points that consist of the lines, we needed to average the lines and extrapolate them so that we have a nice look for the lane line.
    c. The steps for getting a smooth line are:
        i. Divide the lines into “left” and “right” based on the slope
        ii. Getting the slope and y intersection for each line
        iii. Averaging the slope and intersection for the “left” and “right” category.
        iv. Construction of new lines that extend the overall field of interest in the image.

Example of the pipeline implementation



Improvement

The algorithm is working fine for straight lines Lanes, so the next improvement that can be done is to update the algorithm to accommodate the curved roads. Another potential improvement could be taking into account the vehicle is not moving within the lane line for example crossing the lanes. The algorithm is assuming that the lanes are in the middle of the picture so we need to create an algorithm that track the change in the car and reflect it to the region of interest in the algorithm