**NAME - AYUSH VASHISTHA PERFORMED AT-KING MONGKUT UNIVERSITY**

**REG NO - 169108037 OF TECHNOLOGY BANGKOK**

**INTERNSHIP TUTOR -MAHASAK KETCHAM**

**UNIVERSITY TUTOR - DR. AKHILESH KUMAR SHARMA INTERNSHIP DATE-01/06/2019 -03/07/2019**

horizontal line

**ACKNOWLEDGEMENT**

First of all I would like to thank Mrs. YOON Airi, a PhD student, for her constant guidance, advice, encouragement and every possible help in the overall preparation of this report.

I would like to express my gratitude and respect to my internship supervisor, Prof. KETCHAM Mahasak for having me for this project.

I also want to thank my tutor, DR AKHILESH KUMAR SHARMA, for keeping informed on my situation.

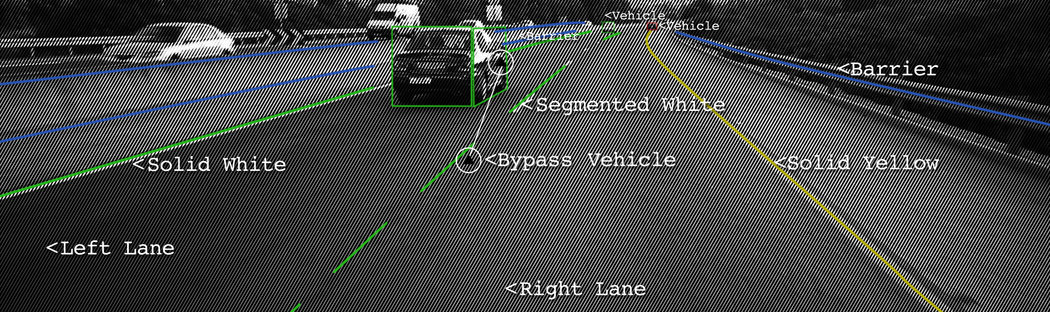
This internship would not have been possible without DR SUNANTHA SODSEE who not only welcomed me in Thailand but guided me in every step of my internship.

Last but not the least, I want to express my deep gratitude to my parents MR AVINASH CHANDRA MISHRA AND MRS NIDHI MISHRA and also my elder sister MS ANUKRITI VASHISTHA for their continuous supports in my studies.

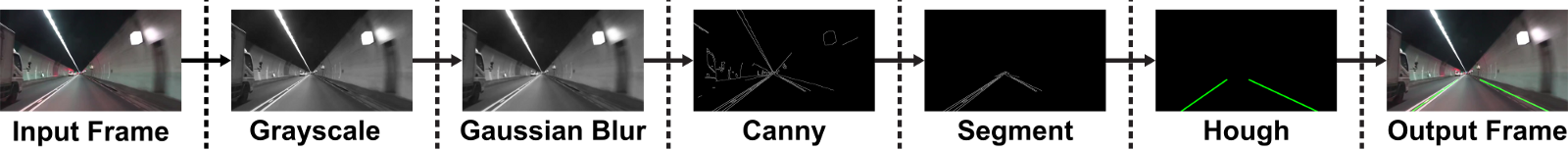
# Introduction

We have heard a lot about self driving cars , but what most people do not know is their ability to detect the lanes on their own.so how do they do it? Let us find out.

The lines drawn on roads indicate to human drivers where the lanes are and act as a guiding reference to which direction to steer the vehicle accordingly and convention to how vehicle agents interact harmoniously on the road. Likewise, the ability to identify and track lanes is cardinal for developing algorithms for driverless vehicles.



zz



### 

**SETUP**

The various requirements are -

1. OPENCV- if you do not have opencv install then do as following ,open terminal and write - pip install opencv-python
2. Next, open detector.py with your text editor. We will be writing all of the code of this section in this Python file.

**PROCEDURE**

**1. Converting images to video**

I was provided with a set of 3000 images and so it would be a hectic task to apply lane detection on each and every image so the best way was to write a code in python to combine all to images(frames) and make it in a video and then apply lane

detection to it. BELOW you will find a code to convert images to a video.

And the output video is -<http://localhost:8981/files/video.avi> (the link)

As the images have now been turned into a video , now the next thing would be to apply lane detection on this video.

#### **2. Processing a video**

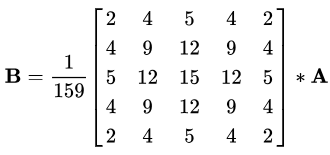
We will use our video for lane detection as a series of continuous frames (images) by intervals of 10 milliseconds. We can also quit the program anytime by pressing the ‘q’ key.

#### **3. Applying Canny Detector**

The Canny Detector is a multi-stage algorithm optimized for fast real-time edge detection. The fundamental goal of the algorithm is to detect sharp changes in luminosity (large gradients), such as a shift from white to black, and defines them as edges, given a set of thresholds. The Canny algorithm has four main stages:

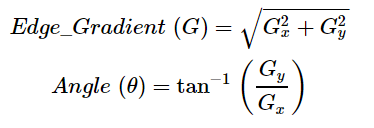
**A. Noise reduction**

As with all edge detection algorithms, noise is a crucial issue that often leads to false detection. A 5x5 Gaussian filter is applied to convolve (smooth) the image to lower the detector’s sensitivity to noise. This is done by using a kernel (in this case, a 5x5 kernel) of normally distributed numbers to run across the entire image, setting each pixel value equal to the weighted average of its neighboring pixels.

5x5 Gaussian kernel. Asterisk denotes convolution operation.

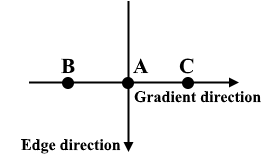
**B. Intensity gradient**

The smoothened image is then applied with a Sobel, Roberts, or Prewitt kernel (Sobel is used in OpenCV) along the x-axis and y-axis to detect whether the edges are horizontal, vertical, or diagonal.

Sobel kernel for calculation of the first derivative of horizontal and vertical directions

**C. Non-maximum suppression**

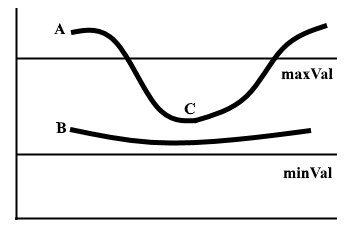
Non-maximum suppression is applied to “thin” and effectively sharpen the edges. For each pixel, the value is checked if it is a local maximum in the direction of the gradient calculated previously.

Non-maximum suppression on three points

A is on the edge with a vertical direction. As gradient is normal to the edge direction, pixel values of B and C are compared with pixel values of A to determine if A is a local maximum. If A is local maximum, non-maximum suppression is tested for the next point. Otherwise, the pixel value of A is set to zero and A is suppressed.

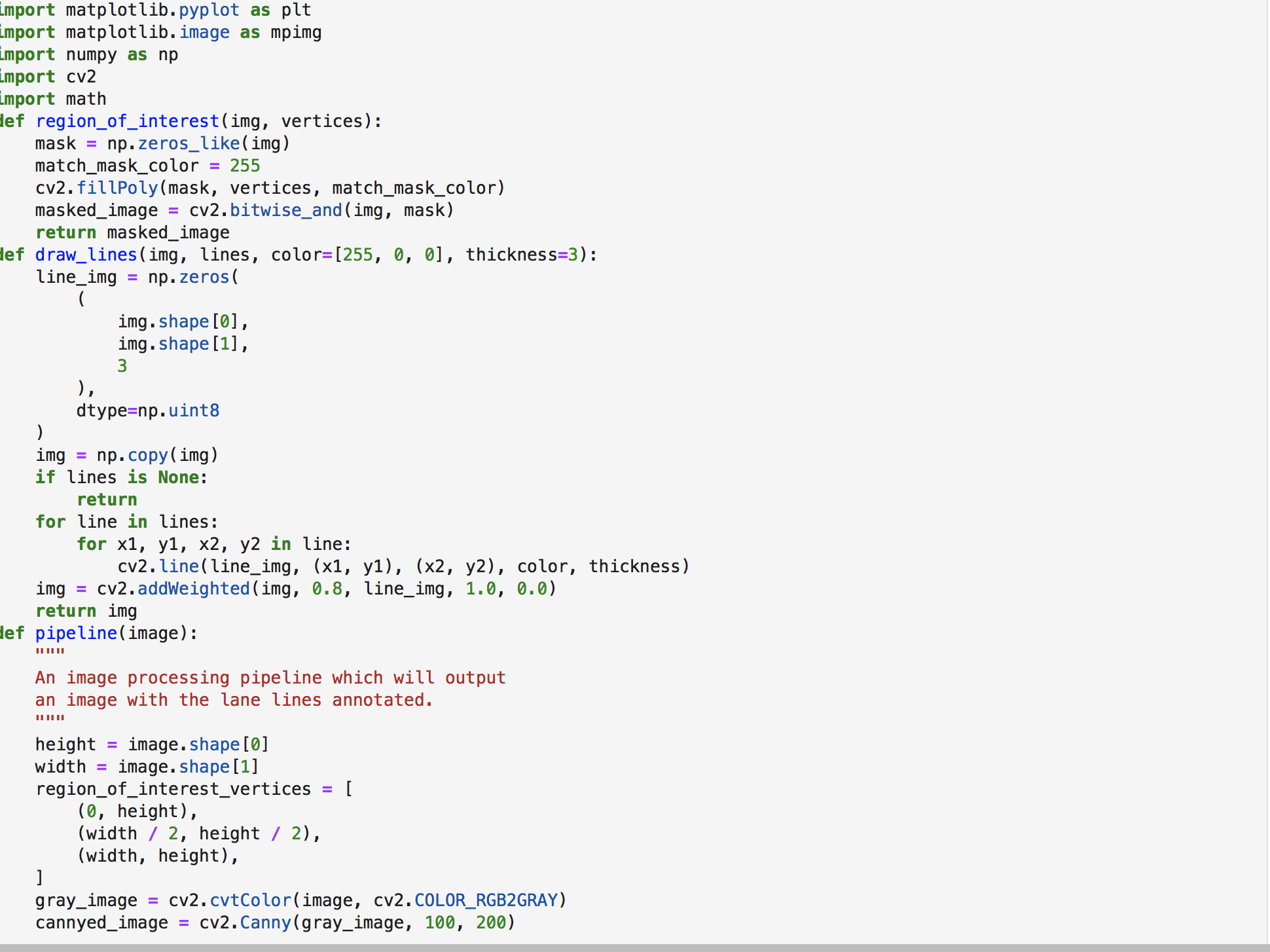
**D. Hysteresis thresholding**

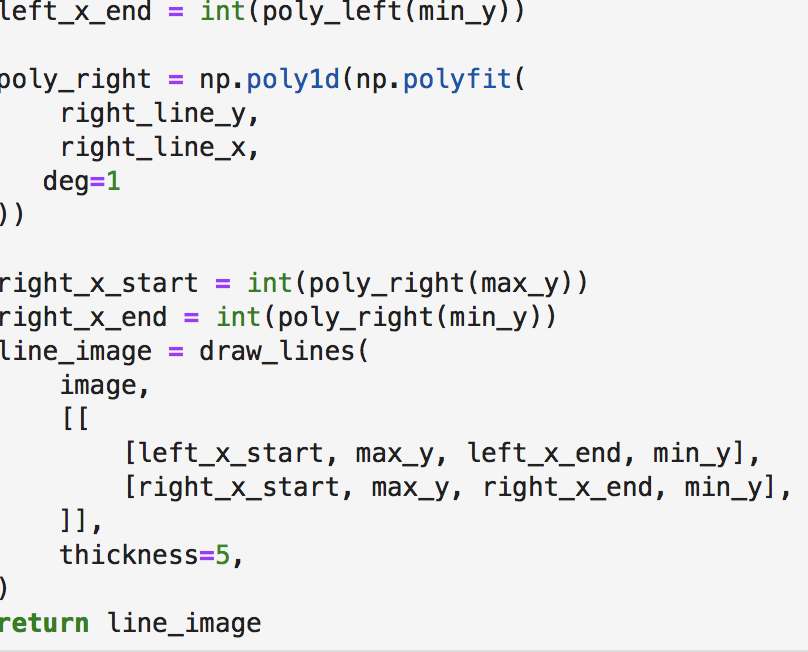
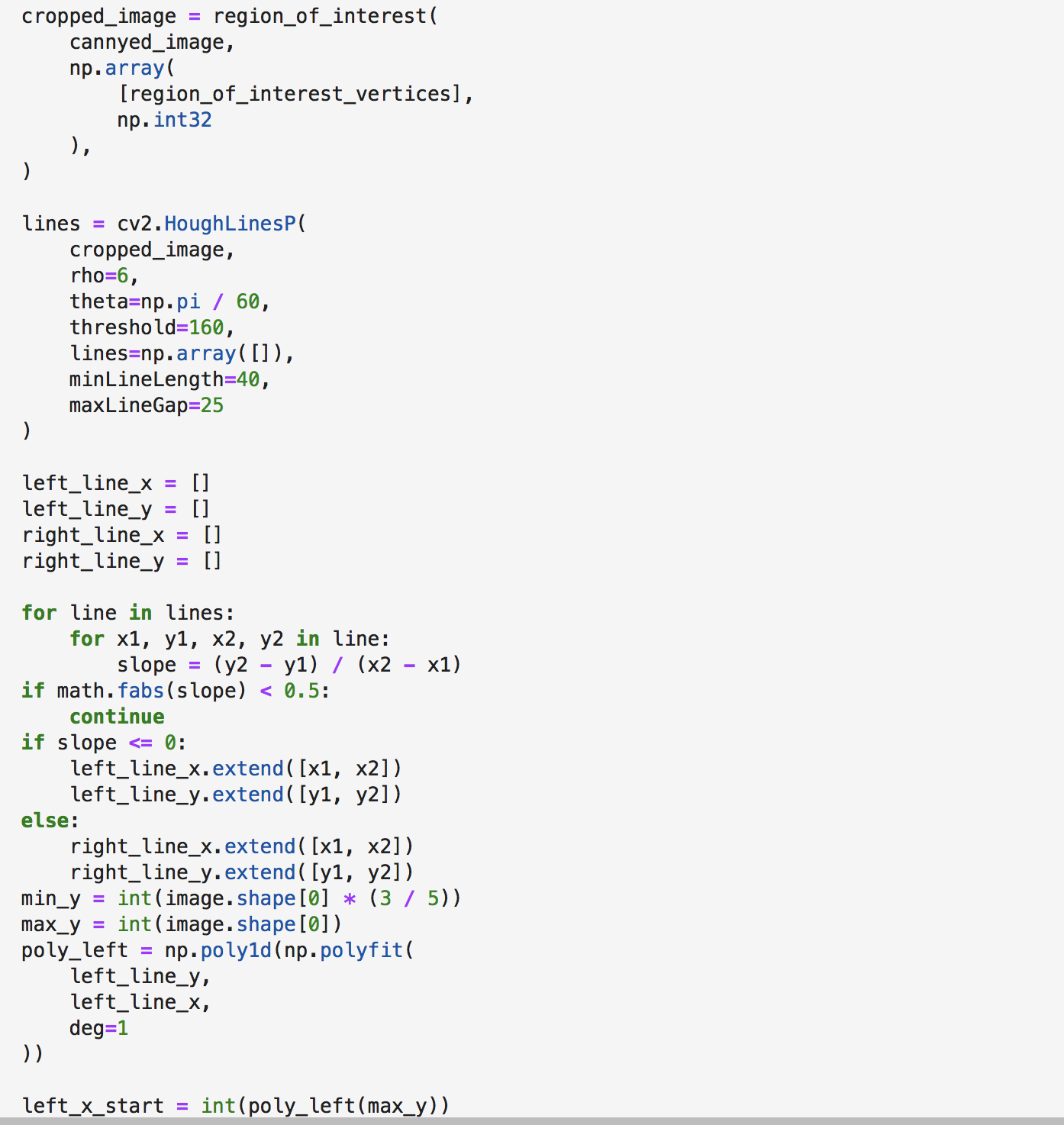
After non-maximum suppression, strong pixels are confirmed to be in the final map of edges. However, weak pixels should be further analyzed to determine whether it constitutes as edge or noise. Applying two pre-defined minVal and maxVal threshold values, we set that any pixel with intensity gradient higher than maxVal are edges and any pixel with intensity gradient lower than minVal are not edges and discarded. Pixels with intensity gradient in between minVal and maxVal are only considered edges if they are connected to a pixel with intensity gradient above maxVal.

Hysteresis thresholding example on two lines

Edge A is above maxVal so is considered an edge. Edge B is in between maxVal and minVal but is not connected to any edge above maxVal so is discarded. Edge C is in between maxVal and minVal and is connected to edge A, an edge above maxVal, so is considered an edge.

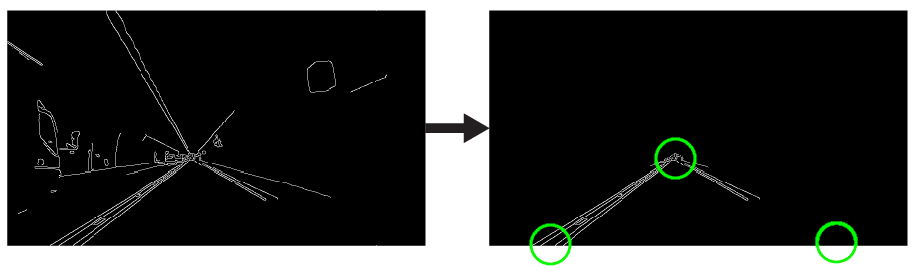
For our pipeline, our frame is first grayscaled because we only need the luminance channel for detecting edges and a 5 by 5 gaussian blur is applied to decrease noise to reduce false edges.





#### **4. Segmenting lane area**

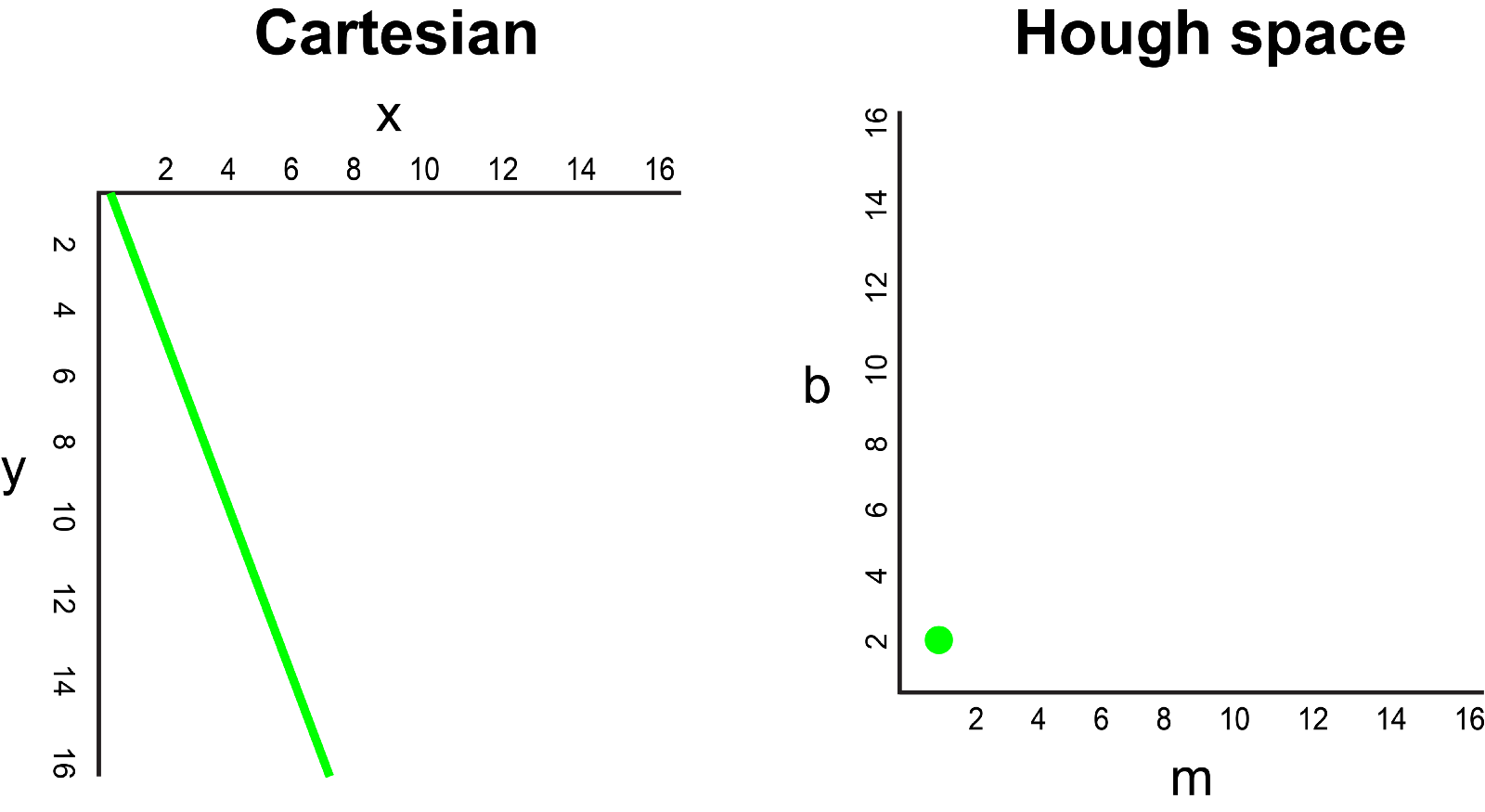
We will handcraft a triangular mask to segment the lane area and discard the irrelevant areas in the frame to increase the effectiveness of our later stages.



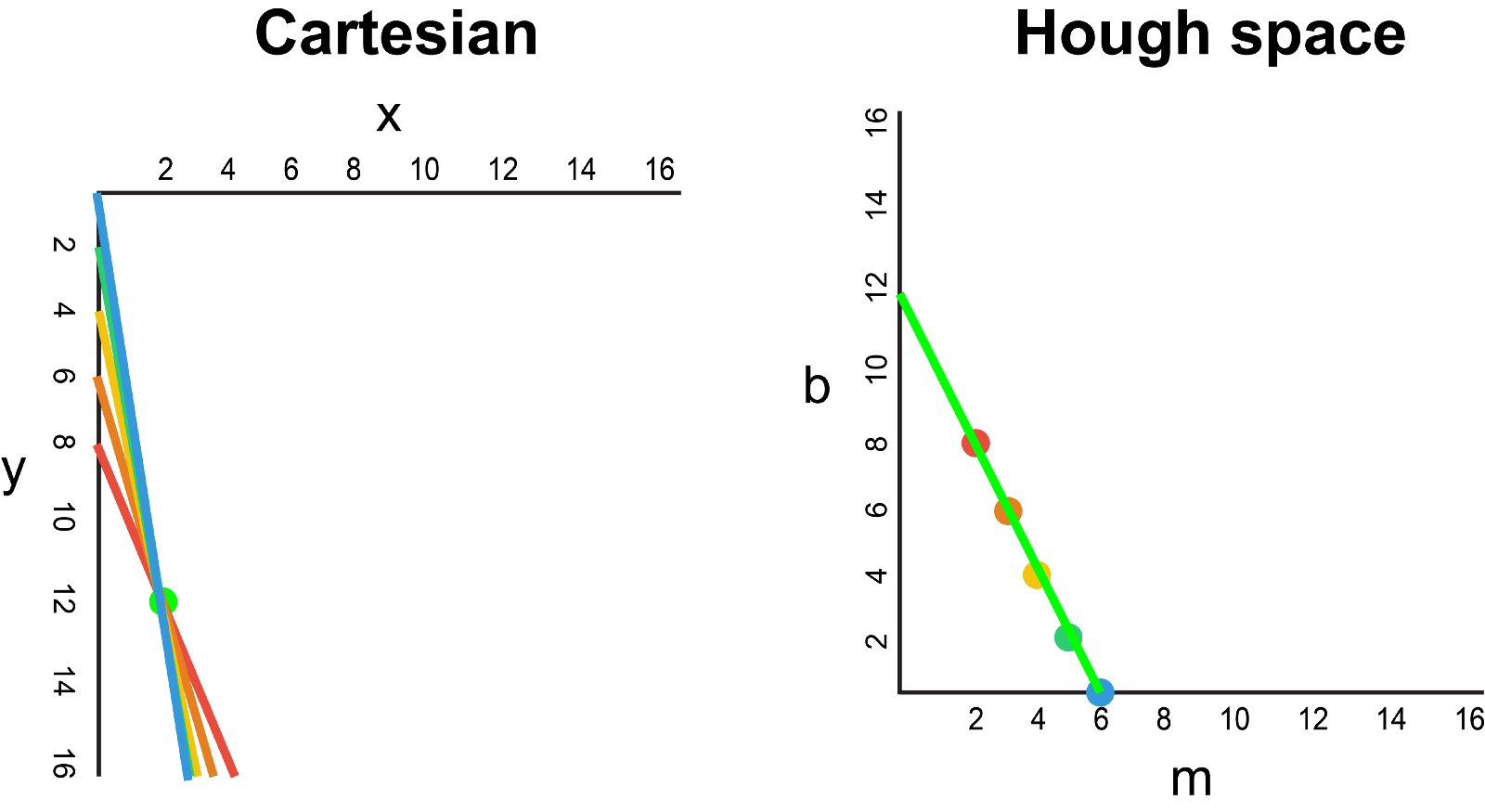


#### **5. Hough transform**

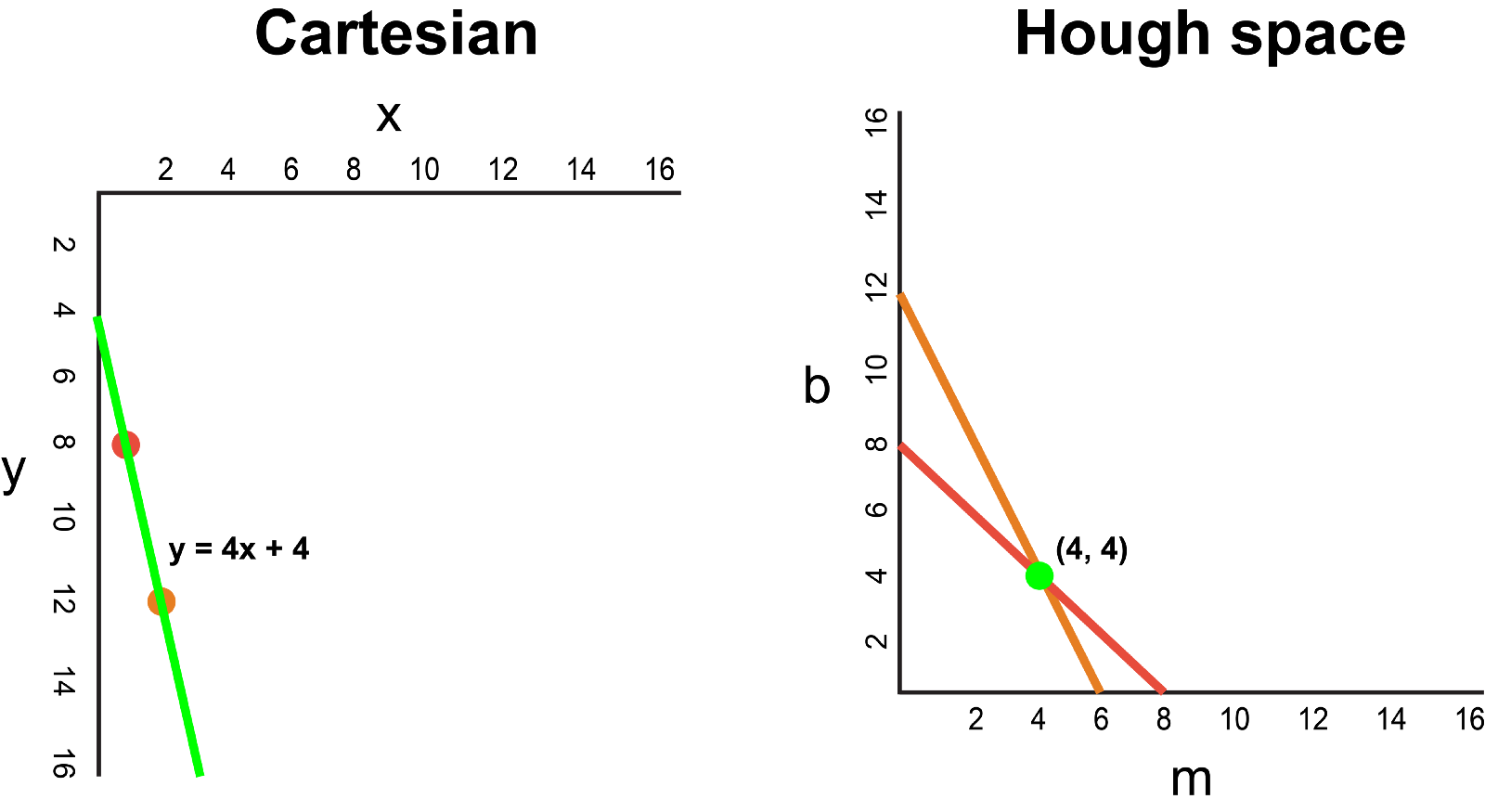
In the Cartesian coordinate system, we can represent a straight line as y = mx + b by plotting y against x. However, we can also represent this line as a single point in Hough space by plotting b against m. For example, a line with the equation y = 2x + 1 may be represented as (2, 1) in Hough space.



Now, what if instead of a line, we had to plot a point in the Cartesian coordinate system. There are many possible lines which can pass through this point, each line with different values for parameters m and b. For example, a point at (2, 12) can be passed by y = 2x + 8, y = 3x + 6, y = 4x + 4, y = 5x + 2, y = 6x, and so on. These possible lines can be plotted in Hough space as (2, 8), (3, 6), (4, 4), (5, 2), (6, 0). Notice that this produces a line of m against b coordinates in Hough space.

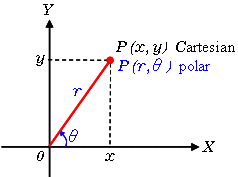


Whenever we see a series of points in a Cartesian coordinate system and know that these points are connected by some line, we can find the equation of that line by first plotting each point in the Cartesian coordinate system to the corresponding line in Hough space, then finding the point of intersection in Hough space. The point of intersection in Hough space represents the m and b values that pass consistently through all of the points in the series.

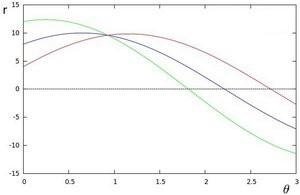


Since our frame passed through the Canny Detector may be interpreted simply as a series of white points representing the edges in our image space, we can apply the same technique to identify which of these points are connected to the same line, and if they are connected, what its equation is so that we can plot this line on our frame.

For the simplicity of explanation, we used Cartesian coordinates to correspond to Hough space. However, there is one mathematical flaw with this approach: When the line is vertical, the gradient is infinity and cannot be represented in Hough space. To solve this problem, we will use Polar coordinates instead. The process is still the same just that other than plotting m against b in Hough space, we will be plotting r against θ.



For example, for the points on the Polar coordinate system with x = 8 and y = 6, x = 4 and y = 9, x = 12 and y = 3, we can plot the corresponding Hough space.



We see that the lines in Hough space intersect at θ = 0.925 and r = 9.6. Since a line in the Polar coordinate system is given by r = xcosθ + ysinθ, we can induce that a single line crossing through all these points is defined as 9.6 = xcos0.925 + ysin0.925.

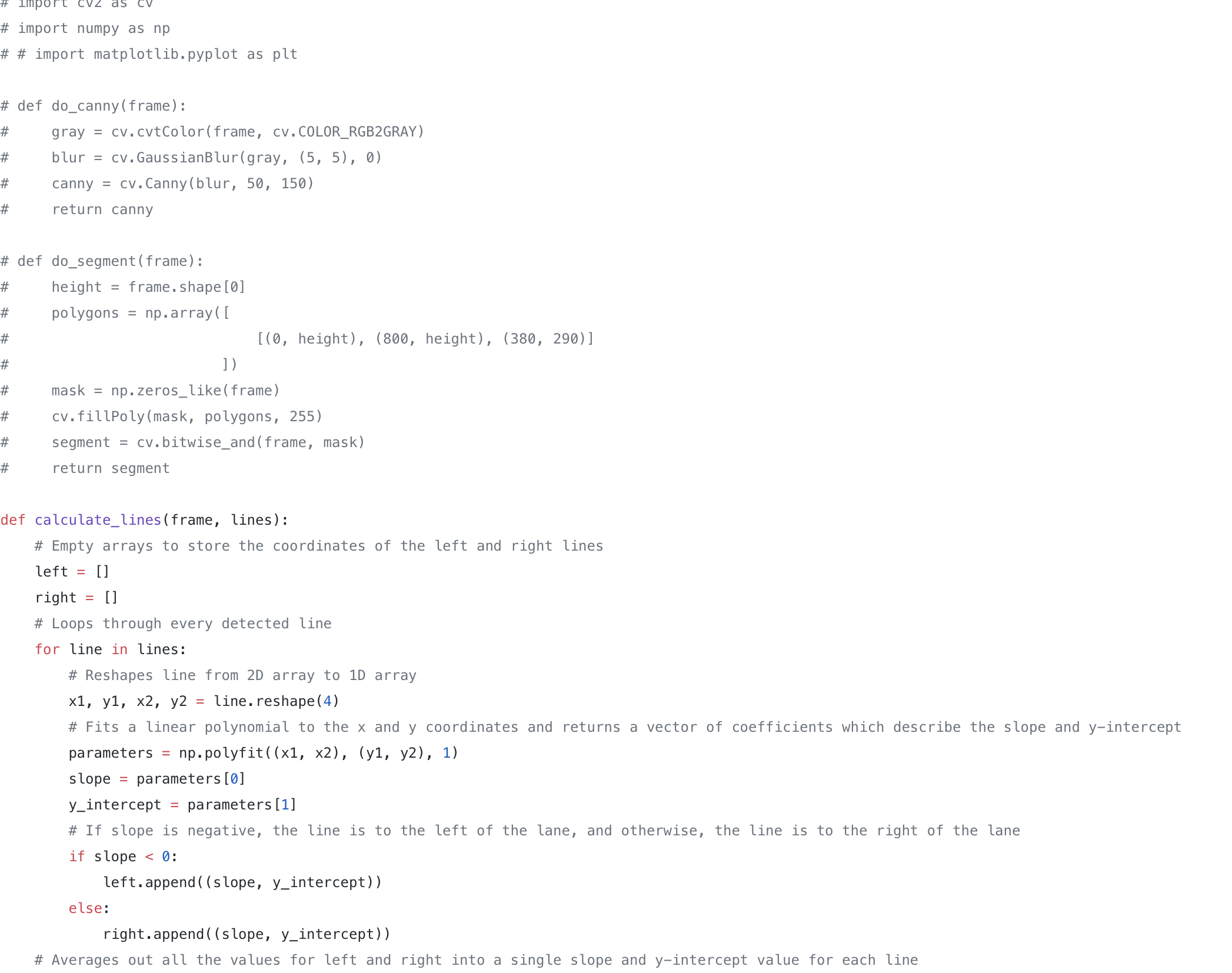
Generally, the more curves intersecting in Hough space means that the line represented by that intersection corresponds to more points. For our implementation, we will define a minimum threshold number of intersections in Hough space to detect a line. Therefore, Hough transform basically keeps track of the Hough space intersections of every point in the frame. If the number of intersections exceeds a defined threshold, we identify a line with the corresponding θ and r parameters.

We apply Hough Transform to identify two straight lines — which will be our left and right lane boundaries



#### **6. Visualization**

The lane is visualized as two light green, linearly fitted polynomials which will be overlayed on our input frame.



Now, we open Terminal and run python detector.py to test our simple lane detector which looks to be working.

**RESULTS**

In order to choose which method to implement, I performed several trials. I tested 2 approaches HOUGH TRANSFORM and the SPATIAL CNN and is it turned out the HOUGH TRANSFORM was the ideal one for the video that we created using the images and it worked completely fine.

**FURTHER OBJECTIVES**

As i was not aware of the concept of self driving RC car before but having learnt a lot about it during my 1 month internship my next step would be to build a self driving RC car and to implement Lane Detection on it.

**CONCLUSION**

To sum it up this internship helped me a lot in learning new things and concepts.

Before coming to this university applying Lane detection to a video would just have been a distant reality.I learned a lot about self driving Rc car as well.

And i would like to thank each and everyone who helped me not only in my internship but in other aspects as well.