1. Pipeline Description

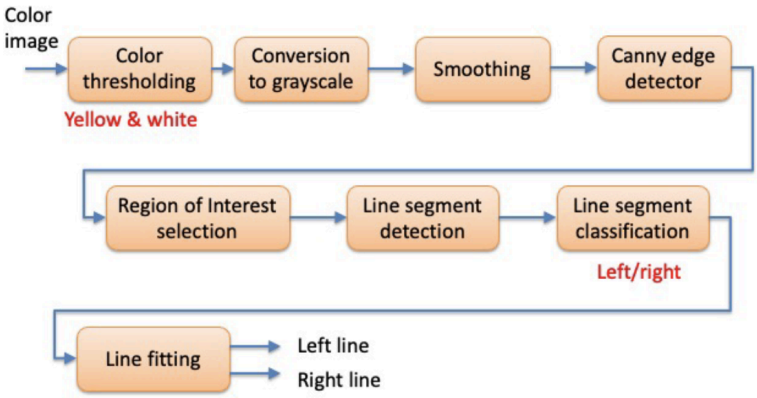


Figure 1. Suggested Pipeline.

Figure 2. Base Test Image.

My pipeline consists of the same eight steps that were provided in the assignment description as shown in Figure 1. The eight steps include and their subcomponents are:

---

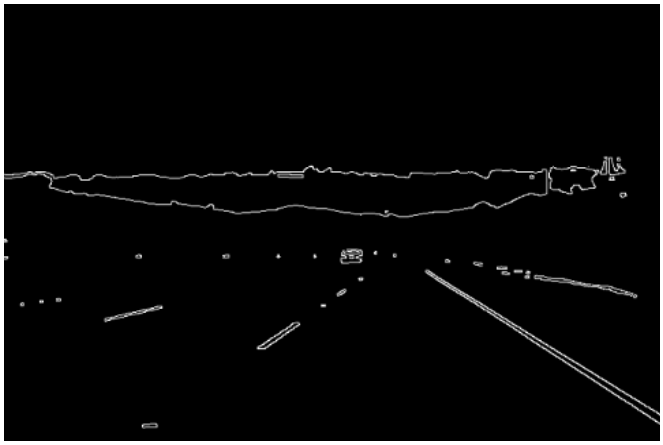
1. Color thresholding
   - convert the image to HSV format to leverage the hue plane which helps to identify specific colours better as compared to RGB format
   - apply a mask to keep the white and yellow areas of the image which are possible lanes

2. Conversion to grayscale
   - simplify the image from three intensity values (RGB) to a single intensity channel(0 to 255) to improve the efficiency of next steps

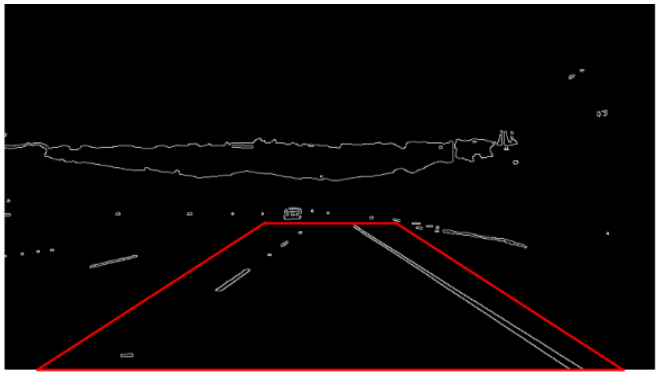
3. Smoothing
   - apply a 2D Gaussian filter to remove both horizontal and vertical noise to prepare for the edge detection
   - kernel spread determines the amount of smoothing

- high kernel performs more smoothing but can blur small details
- low kernel performs less smoothing, retaining fine details but noise can persist
- selected a kernel size of 5x5 for this application because as mentioned in the lecture slides, it “effectively captures 98.76% of the area under the 1D Gaussian” and the 2D Gaussian filter used is a product of two 1D filters.

4. Canny edge detector
   - the Canny filter provides single pixel edges with good continuity
   - high threshold determines where to begin tracing the edge
   - low threshold determines where to stop tracing the edge
   - According to the OpenCV documentation, “Canny recommended an upper:lower ratio between 2:1 and 3:1.”

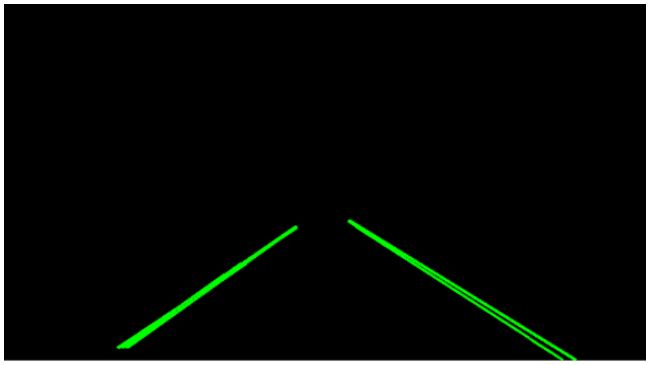


5. Region of interest selection
   - mask area of road where lanes are expected using trapezoid
   - all edges outside of trapezoid are removed
   - bottom left and bottom right are 5% off the sides
   - top left and top right are at 40% and 60% width respectively and at 60% height



6. Line segment detection
   - using the Hough line detector to find straight line segments from edges

Parameter	Value	Description
rho	1	distance resolution in pixels
theta	π/180	angular resolution in radians
threshold	10	minimum number of votes or intersections



min_line_length	20	minimum length of a line segment in pixels
max_line_gap	200	maximum gap in pixels between connectable line segments

7. Line segment classification
   - classify left and right line segments to differentiate between the two lanes
   - one approach is to split the lines by the center of the trapezoid
   - combined with step 8

8. Line fitting
   - overlay the lines on the original image
   - use linear regression for fitting
   - use RANSAC algorithm to remove outliers

2. Potential Shortcomings

- computer vision takes a lot of compute
- curved roads are harder to implement
- The implementation relies heavily on clearly marked lanes. However, due to various reasons, such as snow, mud, off path roads, or night conditions, the lanes may not be visible.
- In parking lots, there are no lanes.

3. Possible Improvements to the Pipeline

- Use deep learning and neural networks to create real-time lane detection and improve efficiency
- Implement fallback methods to accommodate for situations when the lane is not visible. For example, we can estimate a lane based on the path of the cars in front, or if the user is consistently travelling in a straight line, then we can estimate their lane.
- Lane changes
- Adapt kernel size based on image resolution and noise level. For example, larger images would require a larger kernel size and therefore, we can make kernel size a factor of the image size. Likewise, adjust thresholds depending on brightness.
- Detect splines instead of lines