import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import numpy as np

import cv2

# Read in and grayscale the image

image = mpimg.imread('exit-ramp.jpg')

gray = cv2.cvtColor(image,cv2.COLOR\_RGB2GRAY)

# Define a kernel size and apply Gaussian smoothing

kernel\_size = 5

blur\_gray = cv2.GaussianBlur(gray,(kernel\_size, kernel\_size),0)

# Define our parameters for Canny and apply

low\_threshold = 50

high\_threshold = 150

edges = cv2.Canny(blur\_gray, low\_threshold, high\_threshold)

# Next we'll create a masked edges image using cv2.fillPoly()

mask = np.zeros\_like(edges)

ignore\_mask\_color = 255

# This time we are defining a four sided polygon to mask

imshape = image.shape

vertices = np.array([[(0+20,imshape[0]),((imshape[1]/2)-20, imshape[0]/2), ((imshape[1]/2)+20, imshape[0]/2), (imshape[1]-20,imshape[0])]], dtype=np.int32)

cv2.fillPoly(mask, vertices, ignore\_mask\_color)

masked\_edges = cv2.bitwise\_and(edges, mask)

# Define the Hough transform parameters

# Make a blank the same size as our image to draw on

rho = 1 # distance resolution in pixels of the Hough grid

theta = np.pi/180 # angular resolution in radians of the Hough grid

threshold = 1 # minimum number of votes (intersections in Hough grid cell)

min\_line\_length = 10 #minimum number of pixels making up a line

max\_line\_gap = 5 # maximum gap in pixels between connectable line segments

line\_image = np.copy(image)\*0 # creating a blank to draw lines on

# Run Hough on edge detected image

# Output "lines" is an array containing endpoints of detected line segments

lines = cv2.HoughLinesP(masked\_edges, rho, theta, threshold, np.array([]),

min\_line\_length, max\_line\_gap)

# Iterate over the output "lines" and draw lines on a blank image

for line in lines:

for x1,y1,x2,y2 in line:

cv2.line(line\_image,(x1,y1),(x2,y2),(255,0,0),10)

# Create a "color" binary image to combine with line image

color\_edges = np.dstack((edges, edges, edges))

# Draw the lines on the edge image

lines\_edges = cv2.addWeighted(color\_edges, 0.8, line\_image, 1, 0)

plt.imshow(lines\_edges)