Atom Open CV codes:

1)

import cv2

import numpy as np

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

gray = cv2.cvtColor(lane\_image, cv2.COLOR\_RGB2GRAY)

cv2.imshow('result', gray)

cv2.waitKey(0)

2)

import cv2

import numpy as py

image = cv2.imread('test\_image.jpg')

cv2.imshow('result', image)

cv2.waitKey(0)

3)

import cv2

import numpy as np

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

gray = cv2.cvtColor(lane\_image, cv2.COLOR\_RGB2GRAY)

blur = cv2.GaussianBlur(gray, (5,5), 0)

cv2.imshow('result', blur)

cv2.waitKey(0)

4)

import cv2

import numpy as np

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

gray = cv2.cvtColor(lane\_image, cv2.COLOR\_RGB2GRAY)

blur = cv2.GaussianBlur(gray, (5,5), 0)

canny = cv2.Canny(blur,50,150)

cv2.imshow('result', canny)

cv2.waitKey(0)

5)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny = canny(lane\_image)

plt.imshow(canny)

plt.show()

5)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

return mask

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny = canny(lane\_image)

cv2.imshow('result', region\_of\_interest(canny))

cv2.waitKey(0)

6)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny = canny(lane\_image)

cropped\_image = region\_of\_interest(canny)

cv2.imshow('result', cropped\_image)

cv2.waitKey(0)

7)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def display\_lines(image, lines):

line\_image = np.zeros\_like(image)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

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

return line\_image

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny = canny(lane\_image)

cropped\_image = region\_of\_interest(canny)

lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

line\_image = display\_lines(lane\_image, lines)

combo\_image = cv2.addWeighted(lane\_image, 0.8, line\_image, 1, 1)

cv2.imshow('result', combo\_image)

cv2.waitKey(0)

8)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def average\_slope\_intercept(image, lines):

left\_fit =[]

right\_fit = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

print(parameters)

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def display\_lines(image, lines):

line\_image = np.zeros\_like(image)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

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

return line\_image

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny\_image = canny(lane\_image)

cropped\_image = region\_of\_interest(canny\_image)

lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

average\_lines = average\_slope\_intercept(lane\_image, lines)

line\_image = display\_lines(lane\_image, lines)

combo\_image = cv2.addWeighted(lane\_image, 0.8, line\_image, 1, 1)

cv2.imshow('result', combo\_image)

cv2.waitKey(0)

9)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def average\_slope\_intercept(image, lines):

left\_fit =[]

right\_fit = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

intercept = parameters[1]

if slope < 0:

left\_fit.append((slope, intercept))

else:

right\_fit.append((slope, intercept))

print(left\_fit)

print(right\_fit)

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def display\_lines(image, lines):

line\_image = np.zeros\_like(image)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

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

return line\_image

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny\_image = canny(lane\_image)

cropped\_image = region\_of\_interest(canny\_image)

lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

average\_lines = average\_slope\_intercept(lane\_image, lines)

line\_image = display\_lines(lane\_image, lines)

combo\_image = cv2.addWeighted(lane\_image, 0.8, line\_image, 1, 1)

cv2.imshow('result', combo\_image)

cv2.waitKey(0)

10)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def average\_slope\_intercept(image, lines):

left\_fit =[]

right\_fit = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

intercept = parameters[1]

if slope < 0:

left\_fit.append((slope, intercept))

else:

right\_fit.append((slope, intercept))

left\_fit\_average = np.average(left\_fit, axis = 0)

right\_fit\_average = np.average(right\_fit, axis = 0)

print(left\_fit\_average, 'left')

print(right\_fit\_average, 'right')

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def display\_lines(image, lines):

line\_image = np.zeros\_like(image)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

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

return line\_image

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny\_image = canny(lane\_image)

cropped\_image = region\_of\_interest(canny\_image)

lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

average\_lines = average\_slope\_intercept(lane\_image, lines)

line\_image = display\_lines(lane\_image, lines)

combo\_image = cv2.addWeighted(lane\_image, 0.8, line\_image, 1, 1)

cv2.imshow('result', combo\_image)

cv2.waitKey(0)

11)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def make\_coordinates(image, line\_parameters):

slope, intercept = line\_parameters

y1 = image.shape[0]

y2 = int(y1\*(3/5))

x1 = int((y1-intercept)/slope)

x2 = int((y2-intercept)/slope)

return np.array([x1,y1,x2,y2])

def average\_slope\_intercept(image, lines):

left\_fit =[]

right\_fit = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

intercept = parameters[1]

if slope < 0:

left\_fit.append((slope, intercept))

else:

right\_fit.append((slope, intercept))

left\_fit\_average = np.average(left\_fit, axis = 0)

right\_fit\_average = np.average(right\_fit, axis = 0)

left\_line = make\_coordinates(image, left\_fit\_average)

right\_line = make\_coordinates(image, right\_fit\_average)

return np.array([left\_line, right\_line])

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def display\_lines(image, lines):

line\_image = np.zeros\_like(image)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

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

return line\_image

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

image = cv2.imread('test\_image.jpg')

lane\_image = np.copy(image)

canny\_image = canny(lane\_image)

cropped\_image = region\_of\_interest(canny\_image)

lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

average\_lines = average\_slope\_intercept(lane\_image, lines)

line\_image = display\_lines(lane\_image, average\_lines)

combo\_image = cv2.addWeighted(lane\_image, 0.8, line\_image, 1, 1)

cv2.imshow('result', combo\_image)

cv2.waitKey(0)

12)

import cv2

import numpy as np

import matplotlib.pyplot as plt

def make\_coordinates(image, line\_parameters):

slope, intercept = line\_parameters

y1 = image.shape[0]

y2 = int(y1\*(3/5))

x1 = int((y1-intercept)/slope)

x2 = int((y2-intercept)/slope)

return np.array([x1,y1,x2,y2])

def average\_slope\_intercept(image, lines):

left\_fit =[]

right\_fit = []

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

intercept = parameters[1]

if slope < 0:

left\_fit.append((slope, intercept))

else:

right\_fit.append((slope, intercept))

left\_fit\_average = np.average(left\_fit, axis = 0)

right\_fit\_average = np.average(right\_fit, axis = 0)

left\_line = make\_coordinates(image, left\_fit\_average)

right\_line = make\_coordinates(image, right\_fit\_average)

return np.array([left\_line, right\_line])

def canny(image):

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

blur = cv2.GaussianBlur(gray,(5,5),0)

canny = cv2.Canny(blur, 50, 100)

return canny

def display\_lines(image, lines):

line\_image = np.zeros\_like(image)

if lines is not None:

for line in lines:

x1, y1, x2, y2 = line.reshape(4)

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

return line\_image

def region\_of\_interest(image):

height = image.shape[0]

polygons = np.array([[(200, height), (1100, height), (550,250)]])

mask = np.zeros\_like(image)

cv2.fillPoly(mask, polygons, 255)

masked\_image = cv2.bitwise\_and(image, mask)

return masked\_image

#image = cv2.imread('test\_image.jpg')

#lane\_image = np.copy(image)

#canny\_image = canny(lane\_image)

#cropped\_image = region\_of\_interest(canny\_image)

#lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

#average\_lines = average\_slope\_intercept(lane\_image, lines)

#line\_image = display\_lines(lane\_image, average\_lines)

#combo\_image = cv2.addWeighted(lane\_image, 0.8, line\_image, 1, 1)

#cv2.imshow('result', combo\_image)

#cv2.waitKey(0)

cap = cv2.VideoCapture("test2.mp4")

while(cap.isOpened()):

\_, frame = cap.read()

canny\_image = canny(frame)

cropped\_image = region\_of\_interest(canny\_image)

lines = cv2.HoughLinesP(cropped\_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)

average\_lines = average\_slope\_intercept(frame, lines)

line\_image = display\_lines(frame, average\_lines)

combo\_image = cv2.addWeighted(frame, 0.8, line\_image, 1, 1)

cv2.imshow('result', combo\_image)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()