Crack Detection Project

Basit Ali (FA19-Bse-031)

Etsham Zaid (FA19-Bse-042)

import cv2  
import math  
import numpy as np  
import scipy.ndimage  
  
def orientated\_non\_max\_suppression(mag, ang):  
 ang\_quant = np.round(ang / (np.pi/4)) % 4  
 winE = np.array([[0, 0, 0],[1, 1, 1], [0, 0, 0]])  
 winSE = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])  
 winS = np.array([[0, 1, 0], [0, 1, 0], [0, 1, 0]])  
 winSW = np.array([[0, 0, 1], [0, 1, 0], [1, 0, 0]])  
  
 magE = non\_max\_suppression(mag, winE)  
 magSE = non\_max\_suppression(mag, winSE)  
 magS = non\_max\_suppression(mag, winS)  
 magSW = non\_max\_suppression(mag, winSW)  
  
 mag[ang\_quant == 0] = magE[ang\_quant == 0]  
 mag[ang\_quant == 1] = magSE[ang\_quant == 1]  
 mag[ang\_quant == 2] = magS[ang\_quant == 2]  
 mag[ang\_quant == 3] = magSW[ang\_quant == 3]  
 return mag  
  
def non\_max\_suppression(data, win):  
 data\_max = scipy.ndimage.filters.maximum\_filter(data, footprint=win, mode='constant')  
 data\_max[data != data\_max] = 0  
 return data\_max  
  
# start calulcation  
gray\_image = cv2.imread(r'C:\Users\Asif Bashir\Downloads\road images\crack.jpg', 0)  
  
with\_nmsup = True #apply non-maximal suppression  
fudgefactor = 1.3 #with this threshold you can play a little bit  
sigma = 21 #for Gaussian Kernel  
kernel = 2\*math.ceil(2\*sigma)+1 #Kernel size  
  
gray\_image = gray\_image/255.0  
blur = cv2.GaussianBlur(gray\_image, (kernel, kernel), sigma)  
gray\_image = cv2.subtract(gray\_image, blur)  
  
# compute sobel response corner dector  
sobelx = cv2.Sobel(gray\_image, cv2.CV\_64F, 1, 0, ksize=3)  
sobely = cv2.Sobel(gray\_image, cv2.CV\_64F, 0, 1, ksize=3)  
mag = np.hypot(sobelx, sobely)  
ang = np.arctan2(sobely, sobelx)  
  
# threshold  
threshold = 4 \* fudgefactor \* np.mean(mag)  
mag[mag < threshold] = 0  
  
#either get edges directly  
if with\_nmsup is False:  
 mag = cv2.normalize(mag, 0, 255, cv2.NORM\_MINMAX)  
 kernel = np.ones((5,5),np.uint8)  
 result = cv2.morphologyEx(mag, cv2.MORPH\_CLOSE, kernel)  
 cv2.imshow('im', result)  
 cv2.waitKey()  
  
#or apply a non-maximal suppression  
else:  
  
 # non-maximal suppression  
 mag = orientated\_non\_max\_suppression(mag, ang)  
 # create mask  
 mag[mag > 0] = 255  
 mag = mag.astype(np.uint8)  
  
 kernel = np.ones((5,5),np.uint8)  
 result = cv2.morphologyEx(mag, cv2.MORPH\_CLOSE, kernel)  
  
 cv2.imshow('im', result)  
 cv2.waitKey()

ORGINAL IMAGE



RESULT

