

SPRINT - 4

Team ID	PNT2022TMID29193
Project name	Smart Solutions for Railways

Python Code:

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\SOOSAI\Downloads\crack2.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
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()
```