Code:

# %%

#EDGE DETECTION USING BINARY AND SOBEL

import cv2

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

from matplotlib import pyplot as plt

from matplotlib.gridspec import GridSpec

from unittest import result

fpath = 'image1.jpg'

img = cv2.imread(fpath)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

val, bin\_img = cv2.threshold(img,100,150, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

img\_canny = cv2.Canny(bin\_img,200,400)

img\_sobel\_x = cv2.Sobel(bin\_img, cv2.CV\_64F, 1,0)

img\_sobel\_y = cv2.Sobel(bin\_img, cv2.CV\_64F, 0,1)

grad = np.sqrt(img\_sobel\_x\*\*2+ img\_sobel\_y\*\*2)

img\_sobel\_xy = (grad\*255 / grad.max()).astype(np.uint8)

fig = plt.figure(figsize=(15,15))

gs = GridSpec(1,3)

fig.add\_subplot(gs[0,0])

plt.title('Binary Image')

plt.imshow(bin\_img, cmap='gray')

fig.add\_subplot(gs[0,1])

plt.title('Canny Edge')

plt.imshow(img\_canny, cmap='gray')

fig.add\_subplot(gs[0,2])

plt.title('Sobel Edge')

plt.imshow(img\_sobel\_xy, cmap='gray')

plt.savefig('1.jpg')

# %%

#TEMPLATE MATCHING

filepath1 = 'template1.jpg'

filepath2 = 'target1.jpg'

img\_template = cv2.imread(filepath1)

img\_target = cv2.imread(filepath2)

h,w,c = img\_template.shape

img\_template = cv2.cvtColor(img\_template, cv2.COLOR\_BGR2GRAY)

img\_target = cv2.cvtColor(img\_target, cv2.COLOR\_BGR2GRAY)

img\_template = cv2.resize(img\_template,[133,128])

h2,w2 = img\_template.shape

print(img\_template.shape)

print(img\_target.shape)

img\_target = img\_target.copy()

img\_target\_2 = img\_target.copy()

img\_target\_3 = img\_target.copy()

img\_target\_4 = img\_target.copy()

img\_target\_5 = img\_target.copy()

img\_target\_6 = img\_target.copy()

result1 = cv2.matchTemplate(img\_target, img\_template, cv2.TM\_CCOEFF)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(result1)

top\_left = max\_loc

top\_left\_norm = min\_loc

bottom\_right = [top\_left[0]+w2, top\_left[1]+h2]

cv2.rectangle(img\_target,top\_left,bottom\_right, 255,5)

result2 = cv2.matchTemplate(img\_target\_2, img\_template, cv2.TM\_CCORR)

min\_val\_2, max\_val\_2, min\_loc\_2, max\_loc\_2 = cv2.minMaxLoc(result2)

top\_left\_2 = max\_loc\_2

top\_left\_norm\_2 = min\_loc\_2

bottom\_right\_2 = [top\_left\_2[0]+w2, top\_left\_2[1]+h2]

cv2.rectangle(img\_target\_2,top\_left\_2,bottom\_right\_2, 255,5)

result3 = cv2.matchTemplate(img\_target\_3, img\_template, cv2.TM\_SQDIFF)

min\_val\_3, max\_val\_3, min\_loc\_3, max\_loc\_3 = cv2.minMaxLoc(result3)

top\_left\_3 = max\_loc\_3

top\_left\_norm\_3 = min\_loc\_3

bottom\_right\_3 = [top\_left\_3[0]+w2, top\_left\_3[1]+h2]

cv2.rectangle(img\_target\_3,top\_left\_3,bottom\_right\_3, 255,5)

result4 = cv2.matchTemplate(img\_target\_4, img\_template, cv2.TM\_CCOEFF\_NORMED)

min\_val\_4, max\_val\_4, min\_loc\_4, max\_loc\_4 = cv2.minMaxLoc(result4)

top\_left\_norm\_4 = min\_loc\_4

bottom\_right\_4 = [top\_left\_norm\_4[0]+w2, top\_left\_norm\_4[1]+h2]

cv2.rectangle(img\_target\_4,top\_left\_norm\_4,bottom\_right\_4, 255,5)

result5 = cv2.matchTemplate(img\_target\_5, img\_template, cv2.TM\_CCORR\_NORMED)

min\_val\_5, max\_val\_5, min\_loc\_5, max\_loc\_5 = cv2.minMaxLoc(result5)

top\_left\_norm\_5 = min\_loc\_5

bottom\_right\_5 = [top\_left\_norm\_5[0]+w2, top\_left\_norm\_5[1]+h2]

cv2.rectangle(img\_target\_5,top\_left\_norm\_5,bottom\_right\_5, 255,5)

result6 = cv2.matchTemplate(img\_target\_6, img\_template, cv2.TM\_SQDIFF\_NORMED)

min\_val\_6, max\_val\_6, min\_loc\_6, max\_loc\_6 = cv2.minMaxLoc(result6)

top\_left\_norm\_6 = min\_loc\_6

bottom\_right\_6 = [top\_left\_norm\_6[0]+w2, top\_left\_norm\_6[1]+h2]

cv2.rectangle(img\_target\_6,top\_left\_norm\_6,bottom\_right\_6, 255,5)

fig = plt.figure(figsize=(25,25))

gs = GridSpec(1,3)

'''

fig.add\_subplot(gs[0,0])

plt.title('Template image')

plt.imshow(img\_template, cmap='gray')

'''

fig.add\_subplot(gs[0,0])

plt.title('TM\_CCOEFF')

plt.imshow(img\_target, cmap='gray')

fig.add\_subplot(gs[0,1])

plt.title('TM\_CCORR')

plt.imshow(img\_target\_2, cmap='gray')

fig.add\_subplot(gs[0,2])

plt.title('TM\_SQDIFF')

plt.imshow(img\_target\_3, cmap='gray')

plt.savefig('2.jpg')

# %%

fig = plt.figure(figsize=(25,25))

gs = GridSpec(2,3)

fig.add\_subplot(gs[1,0])

plt.title('TM\_CCOEFF\_NORMED')

plt.imshow(img\_target\_4, cmap='gray')

fig.add\_subplot(gs[1,1])

plt.title('TM\_CCORR\_NORMED')

plt.imshow(img\_target\_5, cmap='gray')

fig.add\_subplot(gs[1,2])

plt.title('TM\_SQDIFF\_NORMED')

plt.imshow(img\_target\_6, cmap='gray')

plt.savefig('3.jpg')

Output:





