

Assignment III

Name - Weerasinghe K.N.
Index No. - 190672T

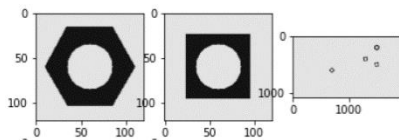
GitHub link - <https://github.com/KavinduWeerasinghe/Assignment-3>

The Tasks given in the assignment leads to object detection in an array of frames. To do the job this has divided into two parts,

- Connected Component Analysis – (Analysis of objects using contours in a single frame) and
- Detecting Objects on a Synthetic Conveyor – (Detecting objects considering a frame at a time in the given video)

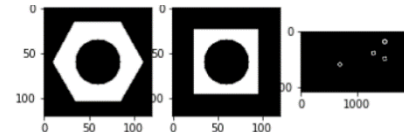
Explanation: The procedure to get the contours is as follows...

Original Images...



- Convert the images into binary state and apply threshold.

```
1 hexnut_template = cv.cvtColor(hexnut_template,cv.COLOR_RGB2GRAY)
2 squarenut_template = cv.cvtColor(squarenut_template,cv.COLOR_RGB2GRAY)
3 conveyor_f100 = cv.cvtColor(conveyor_f100,cv.COLOR_RGB2GRAY)
4
5 ret1,th1 = cv.threshold(hexnut_template,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
6 ret2,th2 = cv.threshold(squarenut_template,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
7 ret3,th3 = cv.threshold(conveyor_f100,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
```



In the given code Otsu's thresholding¹ has been applied. Also, the values are inverted so that the background becomes black. The applied threshold value is 20.0.

- Converting the small holes inside the foreground.

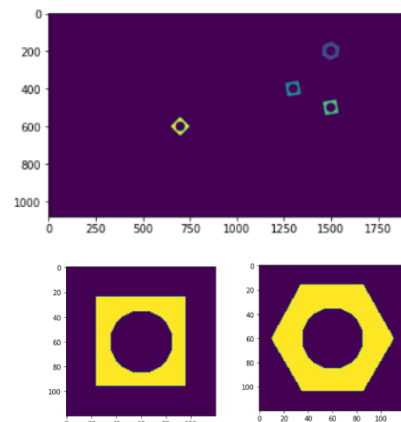
```
1 kernel=np.ones((3,3),np.uint8)
2 closing_1 = cv.morphologyEx(th1, cv.MORPH_CLOSE, kernel)
3 closing_2 = cv.morphologyEx(th2, cv.MORPH_CLOSE, kernel)
4 closing_3 = cv.morphologyEx(th3, cv.MORPH_CLOSE, kernel)
```

In the given code `cv.morphologyEx` has been used which uses Dilation² and Erosion³ in the mentioned order. This is used to remove small black spots on the foreground and white spots in the

background.

- Connected Component Analysis

```
1 class Output:
2     def __init__(self,num_labels,labels,stats,centroids):
3         self.num_labels=num_labels
4         self.labels=labels
5         self.stats=stats
6         self.centroids=centroids
7
8     def Print(abc,i):
9         print("Calculations of image -",i)
10        print("    Number of connected components =",abc.num_labels)
11        fig, ax = plt.subplots()
12        Image=abc.labels
13        ax.imshow(Image)
14        print("    Statistics... ")
15        for j in range(1,abc.num_labels):
16            print("        1. The leftmost (x) coordinate =",abc.stats[j, cv.CC_STAT_LEFT])
17            print("        2. The topmost (y) coordinate =",abc.stats[j, cv.CC_STAT_TOP])
18            print("        3. The horizontal size of the bounding box =",abc.stats[j, cv.CC_STAT_WIDTH])
19            print("        4. The vertical size of the bounding box =",abc.stats[j, cv.CC_STAT_HEIGHT])
20            print("        5. The total area (in pixels) =",abc.stats[j, cv.CC_STAT_AREA])
21        print("    centroids =",abc.centroids)
22
23    Connectivity=4
24    O1 = cv.connectedComponentsWithStats(closing_1, Connectivity, cv.CV_32S)
25    O2 = cv.connectedComponentsWithStats(closing_2, Connectivity, cv.CV_32S)
26    O3 = cv.connectedComponentsWithStats(closing_3, Connectivity, cv.CV_32S)
27    Os=[O1,O2,O3]
28
29    for j,i in enumerate(Os):
30        Os[j]=Output(i[0],i[1],i[2],i[3])
31        Os[j].Print(j+1)
```



¹ The threshold value is selected automatically by the function.

² Increases the foreground of the image.

³ Erodes away the foreground

Given the binary image and the connectivity (provided these are 4-connected images), `cv.connectedComponentsWithStats` function returns,

`num_labels`: Number of labels with 0 representing the background.
`lables`: mapping of the image's pixels according to the labels.
`stats`: An array consisting of several attributes of the given component.
`centroids`: Centroids of the selected components.

The components are color coded from the array given in `lables`.

The results interpreted from stats are as follows...

Calculations of image - 1

Number of connected components = 2
 Statistics...
 1. The leftmost (x) coordinate = 10
 2. The topmost (y) coordinate = 16
 3. The horizontal size of the bounding box = 101
 4. The vertical size of the bounding box = 88
 5. The total area (in pixels) = 4728
 Centroids = [[59.33684864 59.63513234]
 [59.83375635 59.22356176]]

Calculations of image - 2

Number of connected components = 2
 Statistics...
 1. The leftmost (x) coordinate = 24
 2. The topmost (y) coordinate = 24
 3. The horizontal size of the bounding box = 72
 4. The vertical size of the bounding box = 72
 5. The total area (in pixels) = 3227
 Centroids = [[59.5875772 59.5875772]
 [59.19677719 59.19677719]]

Calculations of image - 3

Number of connected components = 5
 Statistics...
 1. The leftmost (x) coordinate = 1454
 2. The topmost (y) coordinate = 150
 3. The horizontal size of the bounding box = 92
 4. The vertical size of the bounding box = 100
 5. The total area (in pixels) = 4636
 1. The leftmost (x) coordinate = 1459
 2. The topmost (y) coordinate = 459
 3. The horizontal size of the bounding box = 82
 4. The vertical size of the bounding box = 82
 5. The total area (in pixels) = 3087

1. The leftmost (x) coordinate = 1259
 2. The topmost (y) coordinate = 359
 3. The horizontal size of the bounding box = 82
 4. The vertical size of the bounding box = 82
 5. The total area (in pixels) = 3087
 1. The leftmost (x) coordinate = 650
 2. The topmost (y) coordinate = 550
 3. The horizontal size of the bounding box = 101
 4. The vertical size of the bounding box = 101
 5. The total area (in pixels) = 3144

Centroids = [[957.36323524 540.44416273]
 [1499.24201898 199.28515962]
 [1299.18302559 399.18302559]
 [1499.18302559 499.18302559]
 [700. 600.]]

Note that the first centroid in each data set is related to label 0 i.e., the background

4. Contour Analysis

```
Images=[closing_1,closing_2,closing_3]
Blanks=[0,0,0]
Contours=[]
for i in range(3):
    contours, hierarchy = cv.findContours(Images[i], cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
    Blanks[i]=np.zeros((Images[i].shape[0],Images[i].shape[1],3),dtype=np.uint8)
    contours.append(contours)
    for j in range(len(contours)):
        cv.drawContours(Blanks[i], contours, j,(np.random.randint(100,256),np.random.randint(256),np.random.randint(50,256)),3)
fig, ax = plt.subplots(figsize=(20,20))
ax.imshow(cv.cvtColor(Blanks[2], cv.COLOR_BGR2RGB))
```

