University of Moratuwa

EN2550 - Fundamentals of Image Processing and Machine Vision

Assignment III

Name - Weerasinghe K.N.

Index No. - 190672T

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

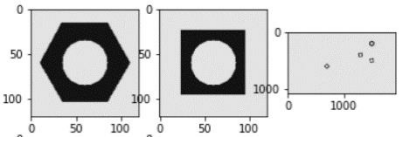
The Tasks given in the assignment leads to object detection in an array of frames. To do the job this has divided into two parts,

(a) Connected Component Analysis – (Analysis of objects using contours in a single frame) and

(b) 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…

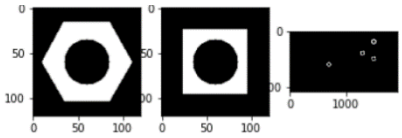
Original Images…



1. Convert the images into binary state and apply threshold.

```
hexnut_template = cv.cvtColor(hexnut_template,cv.COLOR_RGB2GRAY)
squarenut_template =  cv.cvtColor(squarenut_template,cv.COLOR_RGB2GRAY)
conveyor_f100 =  cv.cvtColor(conveyor_f100,cv.COLOR_RGB2GRAY)

ret1,th1 = cv.threshold(hexnut_template,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
ret2,th2 = cv.threshold(squarenut_template,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
ret3,th3 = cv.threshold(conveyor_f100,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
```



In the given code Otsu's thresholding[1] has been applied. Also, the values are inverted so that the background becomes black. The applied threshold value is 20.0.

2. Covering the small holes inside the foreground.

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

In the given code `cv.morphologyEx` has been used which uses Dilation[2] and Erosion[3] in the mentioned order. This is used to remove small black spots on the foreground and white spots in the background.

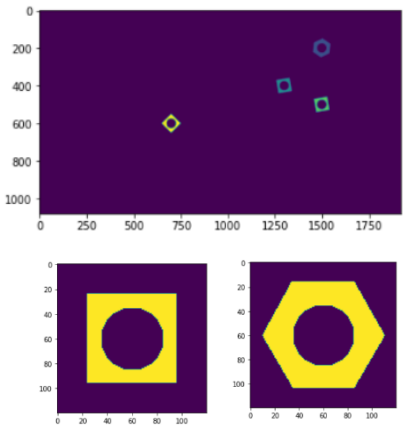
3. Connected Component Analysis

```
class Output:
    def __init__(self,num_lables,lables,stats,centroids):
        self.num_lables=num_lables
        self.lables=lables
        self.stats=stats
        self.centroids=centroids

    def Print(abc,i):
        print("Calculations of image -",i)
        print("     Number of connected components =",abc.num_lables)
        fig, ax = plt.subplots()
        Image=abc.lables
        ax.imshow(Image)
        print("     Statistics... ")
        for j in range(1,abc.num_lables):
            print("         1. The leftmost (x) coordinate =",abc.stats[j,cv.CC_STAT_LEFT])
            print("         2. The topmost (y) coordinate =",abc.stats[j,cv.CC_STAT_TOP])
            print("         3. The horizontal size of the bounding box =",abc.stats[j,cv.CC_STAT_WIDTH])
            print("         4. The vertical size of the bounding box =",abc.stats[j,cv.CC_STAT_HEIGHT])
            print("         5. The total area (in pixels) =",abc.stats[j,cv.CC_STAT_AREA])
        print("     Centroids =",abc.centroids)

Connectivity=4
O1 = cv.connectedComponentsWithStats(closing_1, connectivity, cv.CV_32S)
O2 = cv.connectedComponentsWithStats(closing_2, connectivity, cv.CV_32S)
O3 = cv.connectedComponentsWithStats(closing_3, connectivity, cv.CV_32S)
Os=[O1,O2,O3]

for j,i in enumerate(Os):
    Os[j]=Output(i[0],i[1],i[2],i[3])
    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_lables`: 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 = 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 = 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

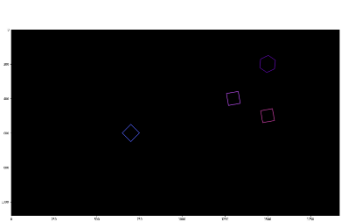
1. The leftmost (x) coordinate = 650
2. The topmost (y) coordinate = 550
3. The horizontal size of the bounding box = 101
4. The vertical size of the bounding box = 101
5. The total area (in pixels) = 3144

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))
```



The function cv.findContours finds and returns the respective contours with the hierarchy of contours. The retrieval mode used is, RETR_EXTERNAL and the approximation method is CHAIN_APPROX_NONE.

RETR_EXTERNAL: retrieves only the extreme outer contours. i.e., All the child contours and the parent contours of the outermost contours are set to -1.

CHAIN_APPROX_NONE: All the points belonging to the contour are stored.

Then the task is Detecting Objects on a Synthetic Conveyor.

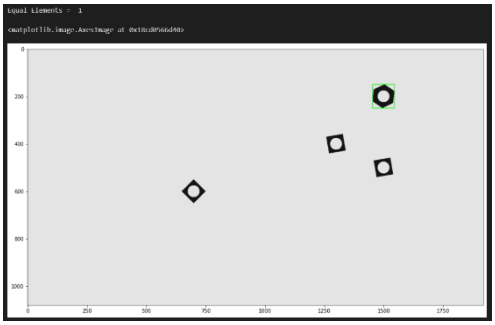
Explanation: The procedure to the second part of the problem is as follows…

1. Test using a single frame

```python
Base_Ctr=Contours[0][0]
conveyor_f100 =  cv.cvtColor(conveyor_f100,cv.COLOR_GRAY2BGR)

Count=0
for i in Contours[2]:
    ret = cv.matchShapes(Base_Ctr,i,1,0.0)
    if ret<0.001:
        Count+=1
        x,y,w,h = cv.boundingRect(i)
        cv.rectangle(conveyor_f100,(x,y),(x+w,y+h),(0,255,0),2)
print("Equal Elements = ",Count)

fig, ax = plt. subplots(figsize = (20,20))
ax.imshow(cv.cvtColor(conveyor_f100, cv.COLOR_BGR2RGB))
```



The base contour is the contour used to compare with the extracted contours in conveyor_100. This is done using cv.matchshapes function which returns similarity between the two contours calculated using hu moments. For the same sample returned value is 0. Then a threshold is applied to the ret value and measured if the respective contour is the same as the hexnut template. If so a bounding box is drawn on conveyor_100 and Count is increased by one which is ultimately printed as the result.

By changing Base_Ctr to Contours[1][0] the whole code can be changed to check with squarenut_template.

2. Implement for the video.

This function returns the number of matching contours in a given frame and draws a bounding box on the frame around the detected objects. The procedure is same as before…

- Convert the images into binary state and apply threshold
- Covering the small holes inside the foreground
- Contour Analysis
- Apply a threshold and selecting the matching labels.

```python
def Count_Ctr(frame):
    global Base_Ctr
    frame_gr =  cv.cvtColor(frame,cv.COLOR_RGB2GRAY)
    ret,th = cv.threshold(frame_gr,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
    kernel=np.ones((3,3),np.uint8)
    closing = cv.morphologyEx(th, cv.MORPH_CLOSE, kernel)
    contours, hierarchy = cv.findContours(closing, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
    Count=0
    for i in contours:
        ret = cv.matchShapes(Base_Ctr,i,1,0.0)
        if ret<0.001:
            Count+=1
            x,y,w,h = cv.boundingRect(i)
            cv.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
    return (Count,frame)
```

```python
f = 0
frame_array = []
Running_Sum={"Prev":0,"Total":0}
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting.")
        break

    f += 1
    text = 'Frame:' + str(f)
    Count,frame=Count_Ctr(frame)
    cv.putText(frame,text , (100, 100), cv.FONT_HERSHEY_COMPLEX, 1, (0,250,0), 1, cv.LINE_AA)
    text = 'Current Contours:' + str(Count)
    cv.putText(frame,text , (100, 130), cv.FONT_HERSHEY_COMPLEX, 1, (250,0,0), 1, cv.LINE_AA)
    if Running_Sum["Prev"]==Count:
        Total=Running_Sum["Total"]
    else:
        added_val=max(Count-Running_Sum["Prev"],0)
        Running_Sum["Total"]+=added_val
    Total=Running_Sum["Total"]
    Running_Sum["Prev"]=Count
    text = 'Total Contours:' + str(Total)
    cv.putText(frame,text , (100, 160), cv.FONT_HERSHEY_COMPLEX, 1, (0,0,250), 1, cv.LINE_AA)
    cv.imshow('Conveyor', frame)
    if cv.waitKey(1) == ord('q'):
        break
    frame_array.append(frame)
```

This process is looped for every frame using the code to the left. Total values are calculated through a simple dictionary containing the previous number of frames and the total up to previous frame.

The resultant video can be found here.
https://github.com/KavinduWeerasinghe/Assignment-3/blob/main/Result%20-%20Low%20Size.m4v