Assignment 03

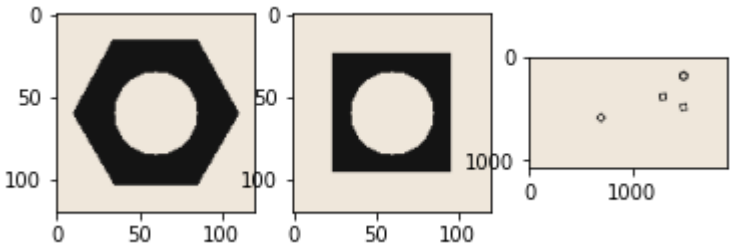
Index : 190356E

Name : N.D.Liyanage

Github : https://github.com/NaduniDamsariLiyanage/en_2550

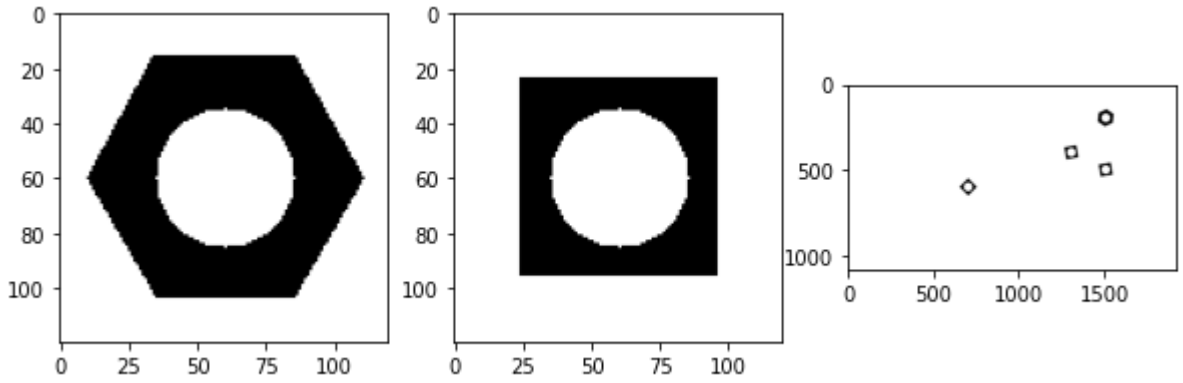
Connected Component Analysis

In [ ]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

hexnut_template = cv.imread('hexnut_template.png', cv.IMREAD_COLOR)
squarenut_template =  cv.imread('squarenut_template.png', cv.IMREAD_COLOR)
conveyor_f100 =  cv.imread('conveyor_f100.png', cv.IMREAD_COLOR)

fig, ax = plt. subplots(1,3)
ax[0].imshow(cv.cvtColor(hexnut_template, cv.COLOR_RGB2BGR))
ax[1].imshow(cv.cvtColor(squarenut_template, cv.COLOR_RGB2BGR))
ax[2].imshow(cv.cvtColor(conveyor_f100, cv.COLOR_RGB2BGR))
plt.show()
```

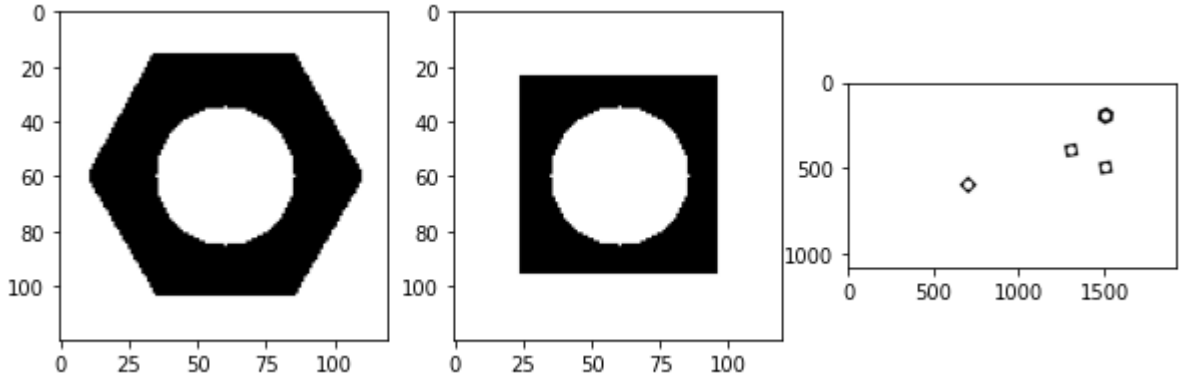


In [ ]:
```
hexnut_gray=cv.cvtColor(hexnut_template, cv.COLOR_BGR2GRAY)
squarenut_gray=cv.cvtColor(squarenut_template, cv.COLOR_BGR2GRAY)
conveyor_gray=cv.cvtColor(conveyor_f100,cv.COLOR_BGR2GRAY)

ret1,hexnut_bin = cv.threshold(hexnut_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
ret2,squarenut_bin = cv.threshold(squarenut_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
ret3,conveyor_bin = cv.threshold(conveyor_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)

print("Threshold for hexnut template = "+str(ret1))
print("Threshold for squarenut template = "+str(ret2))
print("Threshold for conveyor = "+str(ret3))

fig, ax = plt. subplots(1,3, figsize = (10,10))
ax[0].imshow(hexnut_bin,'gray', vmin=0, vmax=255)
ax[1].imshow(squarenut_bin,'gray', vmin=0, vmax=255)
ax[2].imshow(conveyor_bin,'gray', vmin=0, vmax=255)
plt.show()
```

```
Threshold for hexnut template = 20.0
Threshold for squarenut template = 20.0
Threshold for conveyor = 20.0
```



In [ ]:
```
kernel = np.ones((3,3),np.uint8)

List = [hexnut_bin, squarenut_bin, conveyor_bin]
closed_list = []
fig, ax = plt. subplots(1,3, figsize = (10,10))
for i in range(3):
    closed_list.append(cv.morphologyEx((cv.cvtColor(List[i], cv.COLOR_GRAY2RGB)), cv.MO
    ax[i].imshow(closed_list[i])
```



Connected components analysis for Hexnut template

In [ ]:
```
connectivity = 4
hex_num_labels,hex_labels,hex_stat,hex_cent = cv.connectedComponentsWithStats(hexnut_bi
print('There are {} connected components in Hexnut template.'.format(hex_num_labels))
print('Statistics:')
print(hex_stat)
print()
print('Centroids:')
print(hex_cent)
```

```
There are 3 connected components in Hexnut template.
Statistics:
[[  10   16  101   88 4724]
 [   0    0  120  120 7715]
 [  35   35   51   51 1961]]

Centroids:
[[59.83361558 59.22290432]
 [59.16863253 59.54257939]
 [60.         60.        ]]
```

Connected components analysis for Squarenut template

In [ ]:
```
connectivity = 4
```

```
sqr_num_labels,sqr_labels,sqr_stat,sqr_cent = cv.connectedComponentsWithStats(squarenut
print('There are {} connected components in Squarenut template.'.format(sqr_num_labels)
print('Statistics:')
print(sqr_stat)
print()
print('Centroids:')
print(sqr_cent)
```

```
There are 3 connected components in Squarenut template.
Statistics:
[[  24   24   72   72 3223]
 [   0    0  120  120 9216]
 [  35   35   51   51 1961]]

Centroids:
[[59.19578033 59.19578033]
 [59.5        59.5       ]
 [60.         60.        ]]
```

Connected components analysis for Conveyor belt

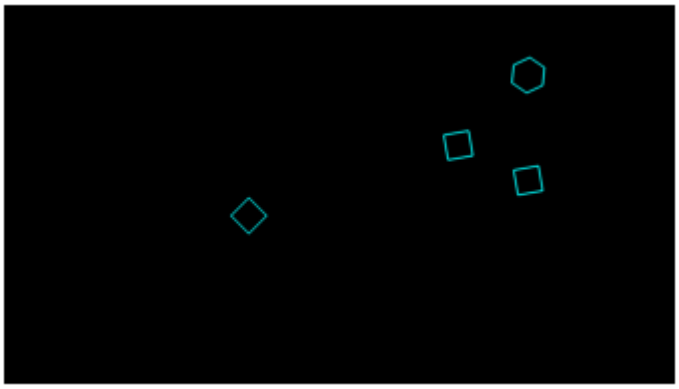
In [ ]:
```
connectivity = 8
belt_num_labels,belt_labels,belt_stat,belt_cent = cv.connectedComponentsWithStats(conve
print('There are {} connected components in Conveyor belt.'.format(belt_num_labels))
print('Statistics:')
print(belt_stat)
print()
print('Centroids:')
print(belt_cent)
```

```
There are 6 connected components in Conveyor belt.
Statistics:
[[    650     150     896     501   13938]
 [      0       0    1920    1080 2051818]
 [   1475     175      51      51    1961]
 [   1275     375      51      51    1961]
 [   1475     475      51      51    1961]
 [    675     575      51      51    1961]]

Centroids:
[[1274.92050509  400.1106328 ]
 [ 956.24678115  540.8845999 ]
 [1500.          200.        ]
 [1300.          400.        ]
 [1500.          500.        ]
 [ 700.          600.        ]]
```

Contour analysis

In [ ]:
```
black_img=np.zeros(conveyor_f100.shape)
contours, hierarchy = cv.findContours(conveyor_bin, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPL
cnt = [contours[i] for i in range(1,9,2)]
cv.drawContours(black_img, cnt, -1, (0,255,150), 3)
plt.imshow(black_img)
plt.axis('off')
plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```



Detecting Objects on a Synthetic Conveyor

In [ ]:
```
cv.namedWindow('Conveyor', cv.WINDOW_NORMAL)
cap = cv.VideoCapture('conveyor.mp4')
f = 0
frames=[]
frame = []
while cap.isOpened():
    ret, frame = cap.read()
    frames.append(frame)
    if not ret:
        print("Can't receive frame (stream end?). Exiting.")
        break

    f += 1
    text = 'Frame:' + str(f)
    cv.putText(frame,text , (100, 100), cv.FONT_HERSHEY_COMPLEX, 1, (0,250,0), 1, cv.LI
    cv.imshow('Conveyor', frame)

    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

```
Can't receive frame (stream end?). Exiting.
```

Count the number of matching hexagonal nuts

In [ ]:
```
total_matches=0
for i in range(len(contours)):
    ret = cv.matchShapes(contours[i],contours[7],1,0.0)
    if ret==0.0:
        total_matches+=1
print('Number of matching hexagonal nuts = ',total_matches)
```

```
Number of matching hexagonal nuts =  1
```

Detecting Objects on a Synthetic Conveyor and counting hexagonal nuts

In [ ]:
```
nuts_total=0
frame_count=0
detected=[]
for frm in frames[:-1]:
    frame_total=0
    image_gray = cv.cvtColor(frm,cv.COLOR_BGR2GRAY)
    ret_f,thresh = cv.threshold(image_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
```

```
    contours_frame,hierarchy_frame = cv.findContours(thresh,cv.RETR_TREE, cv.CHAIN_APPR

    for cnt in contours_frame:
        ret_count = cv.matchShapes(cnt,contours[7],1,0.0)
        if ret_count<=0.006 and abs(cv.contourArea(cnt)-cv.contourArea(contours[7]))<=8
            temp=0
            for c in detected:
                if np.sum(c)-np.sum(cnt)<75000:
                    temp=1
            if temp==0:
                detected.append(cnt)
                nuts_total+=1
            frame_total+=1


    frame_count+=1
    in_text= 'In frame : '+str(frame_total)
    upto_text='Upto frame : '+str(nuts_total)
    cv.putText(img=frm, text=in_text, org=(50, 70), fontFace=cv.FONT_HERSHEY_TRIPLEX, f
    cv.putText(img=frm, text=upto_text, org=(50, 150), fontFace=cv.FONT_HERSHEY_TRIPLEX
```

In [ ]:

```
# Writing the video

frame_array = frames[:-1]
shape = (1080, 1920, 3)

out = cv.VideoWriter('./conveyor_result_190356E.mp4',cv.VideoWriter_fourcc(*'h264'), 30

for i in range(len(frame_array)):
    cv.imshow('Frame', frame_array[i])
    if cv.waitKey(1) == ord('q'):
        break
    out.write(frame_array[i])

out.release()
cv.destroyAllWindows()
```