EN2550: Assignment 03 on Object Counting on a Conveyor Belt

Name - R.M.K.L.Rathnayake
Index - 190520D
Github - https://github.com/rmklakshan/Assignment-03.git

Connected Component Analysis

In this part, we will generate an indexed image representing connected components in conveyor_f101.png image. Notice that, as there are three square nuts and one hexagonal nut in the image, there will be five connected components (backgound will be assigned the label 0).

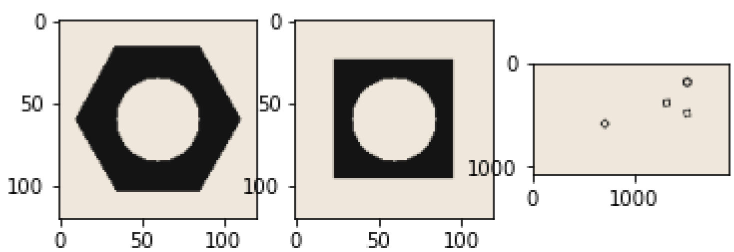
1.Open the hexnut_template.png, squarenut_template.png and conveyor_f100.png and display. This is done for you.

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()
```



Convert the images to grayscale and apply Otsu's thresholding to obtain the binarized image. Do this for both the templates and belt images. See https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html for a guide. State the threshold value (automatically) selected in the operation. Display the output images.

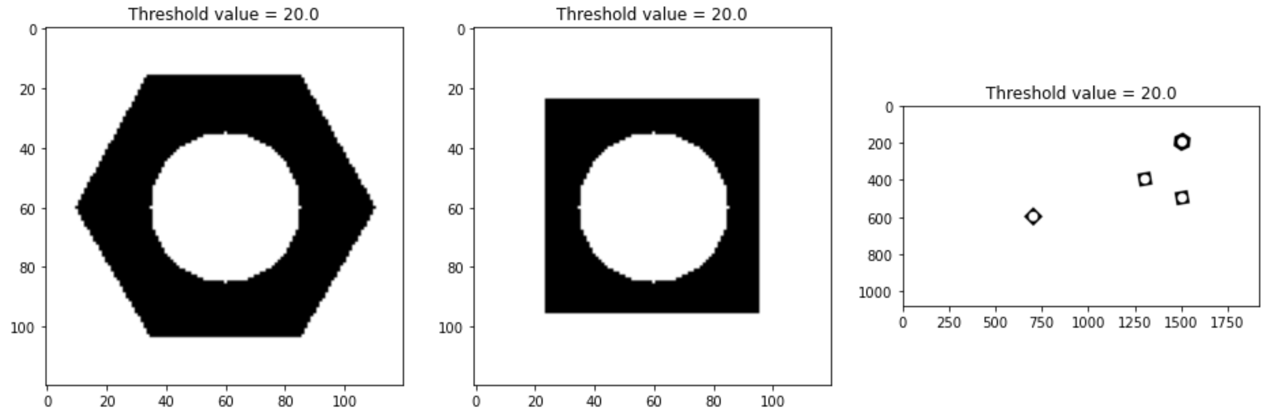
In [ ]:

```
#Convert the images to grayscale
gray_hexnut_template = cv.cvtColor(hexnut_template, cv.COLOR_BGR2GRAY)
gray_squarenut_template =  cv.cvtColor(squarenut_template, cv.COLOR_BGR2GRAY)
gray_conveyor_f100 =  cv.cvtColor(conveyor_f100, cv.COLOR_BGR2GRAY)

#Apply Otsu's thresholding to obtain the binarized image
th_hexnut, bi_hexnut = cv.threshold(gray_hexnut_template,0,255,cv.THRESH_BINARY+cv.THRE
```

```
th_squarenut, bi_squarenut = cv.threshold(gray_squarenut_template,0,255,cv.THRESH_BINAR
th_conveyor, bi_conveyor = cv.threshold(gray_conveyor_f100,0,255,cv.THRESH_BINARY+cv.TH

fig, ax = plt. subplots(1,3, figsize=(16,8))
ax[0].imshow(bi_hexnut,cmap='gray')
ax[0].set_title("Threshold value = "+str(th_hexnut))
ax[1].imshow( bi_squarenut ,cmap='gray')
ax[1].set_title("Threshold value = "+str(th_squarenut))
ax[2].imshow( bi_conveyor ,cmap='gray')
ax[2].set_title("Threshold value = "+str(th_conveyor))
plt.show()
print(" Thresold of hexnut = ",th_hexnut,"\n","Thresold of squarenut = ",th_squarenut,"
```



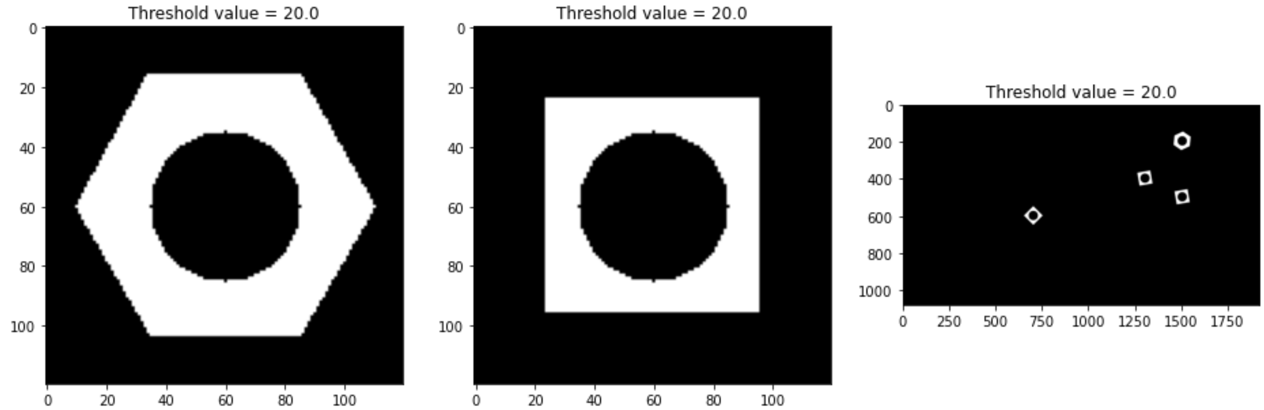

```
Thresold of hexnut =  20.0
Thresold of squarenut =  20.0
Thresold of conveyor =  20.0
```

Using cv.THRESH_BINARY_INV we can obtion the white forground and black background template to apply morphological closing in next part.

In [ ]:

```
#Apply Otsu's thresholding to obtain the binarized image
th_hexnut, bi_hexnut = cv.threshold(gray_hexnut_template,0,255,cv.THRESH_BINARY_INV+cv.
th_squarenut, bi_squarenut = cv.threshold(gray_squarenut_template,0,255,cv.THRESH_BINAR
th_conveyor, bi_conveyor = cv.threshold(gray_conveyor_f100,0,255,cv.THRESH_BINARY_INV+c

fig, ax = plt. subplots(1,3, figsize=(16,8))
ax[0].imshow(bi_hexnut,cmap='gray')
ax[0].set_title("Threshold value = "+str(th_hexnut))
ax[1].imshow( bi_squarenut ,cmap='gray')
ax[1].set_title("Threshold value = "+str(th_squarenut))
ax[2].imshow( bi_conveyor ,cmap='gray')
ax[2].set_title("Threshold value = "+str(th_conveyor))
plt.show()
print(" Thresold of hexnut = ",th_hexnut,"\n","Thresold of squarenut = ",th_squarenut,"
```



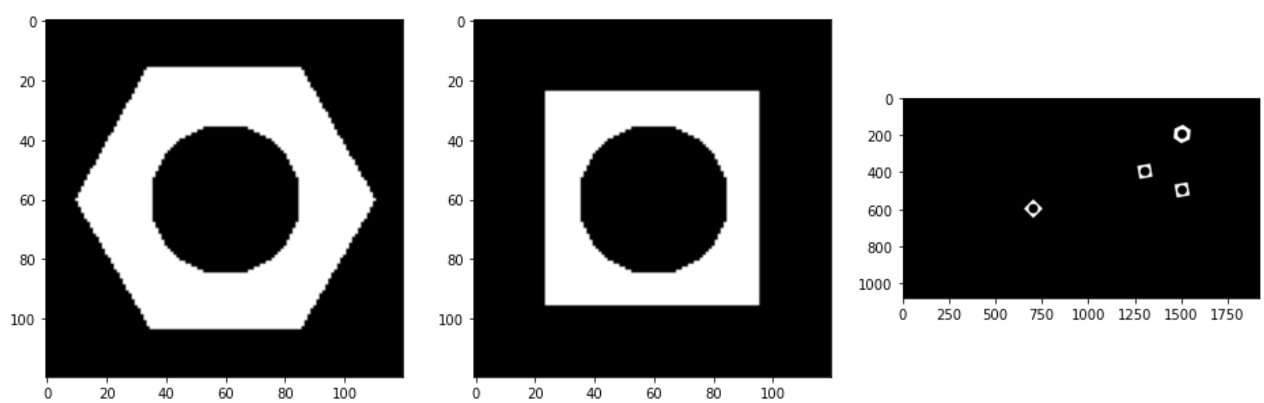

```
Thresold of hexnut =  20.0
Thresold of squarenut =  20.0
Thresold of conveyor =  20.0
```

Carry out morphological closing to remove small holes inside the foreground. Use a 3 × 3 kernel. See https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html for a guide.

In [ ]:

```
kernel = np.ones((3,3),np.uint8)
closing_hexnut = cv.morphologyEx(bi_hexnut, cv.MORPH_CLOSE, kernel)
closing_squarenut = cv.morphologyEx(bi_squarenut, cv.MORPH_CLOSE, kernel)
closing_conveyor = cv.morphologyEx(bi_conveyor, cv.MORPH_CLOSE, kernel)

fig, ax = plt. subplots(1,3, figsize=(16,8))
ax[0].imshow(closing_hexnut,cmap='gray')
ax[1].imshow( closing_squarenut ,cmap='gray')
ax[2].imshow( closing_conveyor ,cmap='gray')
plt.show()
```



Connected components analysis: apply the connectedComponentsWithStats function (see https://docs.opencv.org/4.5.5/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e and display the outputs as colormapped images. Answer the following questions

- How many connected components are detected in each image?
- What are the statistics? Interpret these statistics.
- What are the centroids?

For the hexnut template, you should get the object area in pixel as approximately 4728.

In [ ]:

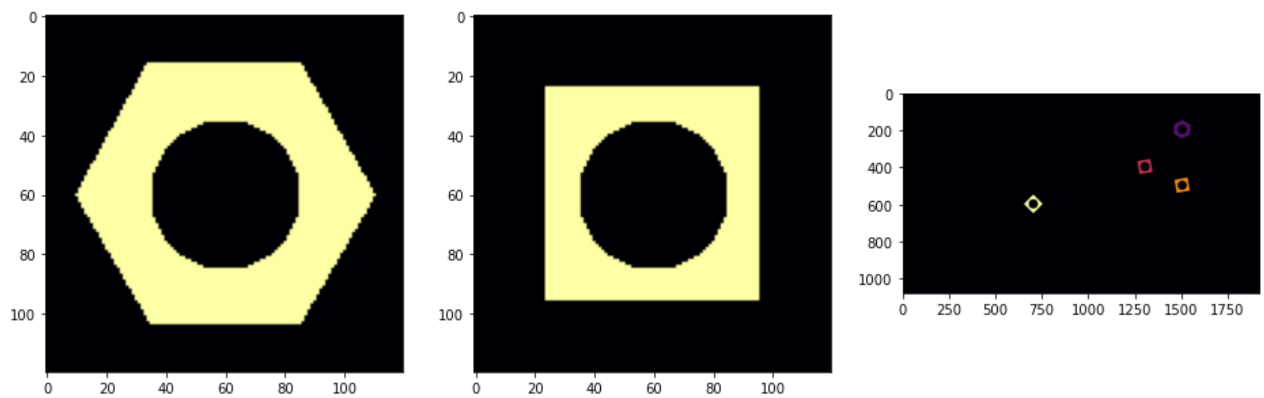
```
nb_hexnut,lb_hexnut,st_hexnut,cntd_hexnut = cv.connectedComponentsWithStats(closing_hex
```

```
nb_squarenut,lb_squarenut,st_squarenut,cntd_squarenut = cv.connectedComponentsWithStats
nb_conveyor,lb_conveyor,st_conveyor,cntd_conveyor = cv.connectedComponentsWithStats(clo

lb_hexnut = np.uint8(cv.normalize(lb_hexnut,None,0,255,cv.NORM_MINMAX))
lb_squarenut = np.uint8(cv.normalize(lb_squarenut,None,0,255,cv.NORM_MINMAX))
lb_conveyor = np.uint8(cv.normalize(lb_conveyor,None,0,255,cv.NORM_MINMAX))

clr_hexnut = cv.applyColorMap(lb_hexnut, cv.COLORMAP_INFERNO )
clr_squarenut = cv.applyColorMap(lb_squarenut, cv.COLORMAP_INFERNO )
clr_conveyor = cv.applyColorMap(lb_conveyor, cv.COLORMAP_INFERNO )

fig, ax = plt. subplots(1,3, figsize=(16,8))
ax[0].imshow(cv.cvtColor(clr_hexnut,cv.COLOR_BGR2RGB))
ax[1].imshow(cv.cvtColor(clr_squarenut,cv.COLOR_BGR2RGB) )
ax[2].imshow( cv.cvtColor(clr_conveyor,cv.COLOR_BGR2RGB))
plt.show()
```



- How many connected components are detected in each image?

In [ ]:

```
print ("Number of connected components of hexnut_template = ",nb_hexnut)
print ("Number of connected components of hexnut_template = ",nb_squarenut)
print ("Number of connected components of hexnut_template = ",nb_conveyor)
```

```
Number of connected components of hexnut_template =  2
Number of connected components of hexnut_template =  2
Number of connected components of hexnut_template =  5
```

- What are the statistics? Interpret these statistics.
- What are the centroids?

1. hexnut_template

In [ ]:

```
for i in range(nb_hexnut):
    print('Component',i)
    print('The leftmost (x) coordinate','\t\t\t:',st_hexnut[i][cv.CC_STAT_LEFT])
    print('The topmost (y) coordinate','\t\t\t:',st_hexnut[i][cv.CC_STAT_TOP])
    print('The horizontal size of the bounding box','\t:',st_hexnut[i][cv.CC_STAT_WIDTH
    print('The vertical size of the bounding box','\t\t:',st_hexnut[i][cv.CC_STAT_HEIGH
    print('The total area  of the connected component','\t:',st_hexnut[i][cv.CC_STAT_AR

    print('\nCentroid',':',str(cntd_hexnut[i]))

    print("\n")
```

```
Component 0
The leftmost (x) coordinate                    : 0
The topmost (y) coordinate                     : 0
The horizontal size of the bounding box        : 120
The vertical size of the bounding box          : 120
The total area  of the connected component     : 9672

Centroid : [59.33684864 59.63513234]


Component 1
The leftmost (x) coordinate                    : 10
The topmost (y) coordinate                     : 16
The horizontal size of the bounding box        : 101
The vertical size of the bounding box          : 88
The total area  of the connected component     : 4728

Centroid : [59.83375635 59.22356176]
```

1. squarenut_template

In [ ]:

```
for i in range(nb_squarenut):
    print('Component',i)
    print('The leftmost (x) coordinate','\t\t\t:',st_squarenut[i][cv.CC_STAT_LEFT])
    print('The topmost (y) coordinate','\t\t\t:',st_squarenut[i][cv.CC_STAT_TOP])
    print('The horizontal size of the bounding box','\t:',st_squarenut[i][cv.CC_STAT_WI
    print('The vertical size of the bounding box','\t\t:',st_squarenut[i][cv.CC_STAT_HE
    print('The total area  of the connected component','\t:',st_squarenut[i][cv.CC_STAT

    print('\nCentroid',':',str(cntd_squarenut[i]))

    print("\n")
```

```
Component 0
The leftmost (x) coordinate                    : 0
The topmost (y) coordinate                     : 0
The horizontal size of the bounding box        : 120
The vertical size of the bounding box          : 120
The total area  of the connected component     : 11173

Centroid : [59.5875772 59.5875772]


Component 1
The leftmost (x) coordinate                    : 24
The topmost (y) coordinate                     : 24
The horizontal size of the bounding box        : 72
The vertical size of the bounding box          : 72
The total area  of the connected component     : 3227

Centroid : [59.19677719 59.19677719]
```

1. conveyor_f100

In [ ]:

```
for i in range(nb_conveyor):
```

```
    print('Component',i)
    print('The leftmost (x) coordinate','\t\t\t:',st_conveyor[i][cv.CC_STAT_LEFT])
    print('The topmost (y) coordinate','\t\t\t:',st_conveyor[i][cv.CC_STAT_TOP])
    print('The horizontal size of the bounding box','\t:',st_conveyor[i][cv.CC_STAT_WID
    print('The vertical size of the bounding box','\t\t:',st_conveyor[i][cv.CC_STAT_HEI
    print('The total area  of the connected component','\t:',st_conveyor[i][cv.CC_STAT_

    print('\nCentroid',':',str(cntd_conveyor[i]))

    print("\n")
```

```
Component 0
The leftmost (x) coordinate                      : 0
The topmost (y) coordinate                       : 0
The horizontal size of the bounding box          : 1920
The vertical size of the bounding box            : 1080
The total area  of the connected component       : 2059646

Centroid : [957.36323524 540.44416273]


Component 1
The leftmost (x) coordinate                      : 1454
The topmost (y) coordinate                       : 150
The horizontal size of the bounding box          : 92
The vertical size of the bounding box            : 100
The total area  of the connected component       : 4636

Centroid : [1499.24201898  199.28515962]


Component 2
The leftmost (x) coordinate                      : 1259
The topmost (y) coordinate                       : 359
The horizontal size of the bounding box          : 82
The vertical size of the bounding box            : 82
The total area  of the connected component       : 3087

Centroid : [1299.18302559  399.18302559]


Component 3
The leftmost (x) coordinate                      : 1459
The topmost (y) coordinate                       : 459
The horizontal size of the bounding box          : 82
The vertical size of the bounding box            : 82
The total area  of the connected component       : 3087

Centroid : [1499.18302559  499.18302559]


Component 4
The leftmost (x) coordinate                      : 650
The topmost (y) coordinate                       : 550
The horizontal size of the bounding box          : 101
The vertical size of the bounding box            : 101
The total area  of the connected component       : 3144

Centroid : [700. 600.]
```

Contour analysis: Use findContours function to retrieve the extreme outer contours. (see

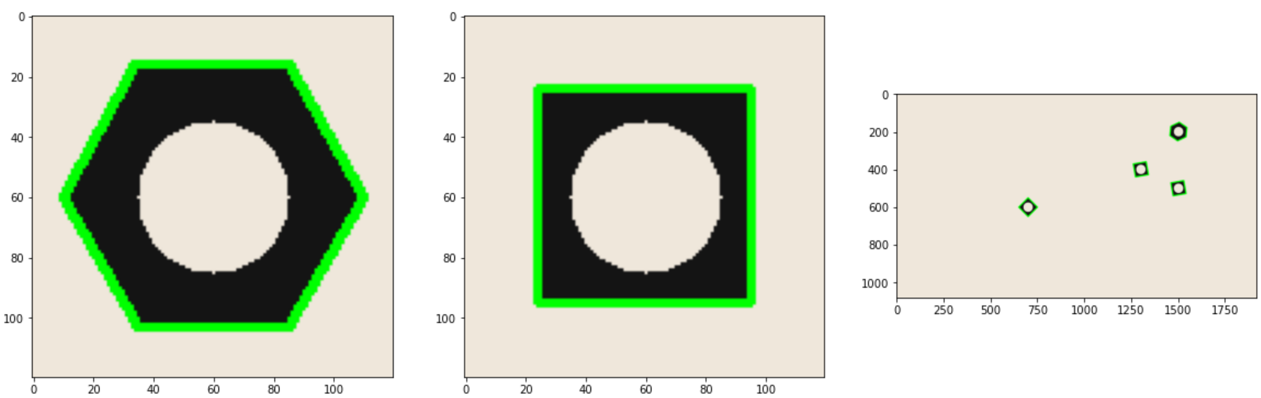
https://docs.opencv.org/4.5.2/d4/d73/tutorial_py_contours_begin.html for help and https://docs.opencv.org/4.5.2/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f2(
for information.

In [ ]:

```
hex_contours, hierarchy_h = cv.findContours(closing_hexnut, cv.RETR_EXTERNAL, cv.CHAIN_
square_contours, hierarchy_s = cv.findContours(closing_squarenut, cv.RETR_EXTERNAL, cv.
conveyor_contours, hierarchy_c = cv.findContours(closing_conveyor, cv.RETR_EXTERNAL, cv

cv.drawContours(hexnut_template, hex_contours, -1, (0,255,0), 2)
cv.drawContours(squarenut_template, square_contours, -1, (0,255,0), 2)
cv.drawContours(conveyor_f100, conveyor_contours, -1, (0,255,0), 5)

fig, ax = plt. subplots(1,3, figsize=(20,10))
ax[0].imshow(cv.cvtColor(hexnut_template,cv.COLOR_BGR2RGB))
ax[1].imshow(cv.cvtColor(squarenut_template,cv.COLOR_BGR2RGB) )
ax[2].imshow( cv.cvtColor(conveyor_f100,cv.COLOR_BGR2RGB))
plt.show()
```



Detecting Objects on a Synthetic Conveyor

In this section, we will use the synthetic conveyor.mp4 sequence to count the two types of nuts.

1.Open the sequence and play it using the code below.

In [ ]:

```
cv.namedWindow('Conveyor', cv.WINDOW_NORMAL)
cap = cv.VideoCapture('conveyor.mp4')
f = 0
frame = []
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)
    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.
```

1.Count the number of matching hexagonal nuts in conveyor_f100.png. You can use matchCountours function as shown in https://docs.opencv.org/4.5.2/d5/d45/tutorial_py_contours_more_functions.html to match contours in each frame with that in th template.

In [ ]:

```
hexnut_count = 0
hex_contours, hierarchy_h = cv.findContours(closing_hexnut, cv.RETR_EXTERNAL, cv.CHAIN_
conveyor_contours, hierarchy_c = cv.findContours(closing_conveyor, cv.RETR_EXTERNAL, cv

hexnut_shape = hex_contours[0]

for c in conveyor_contours:
    val = cv.matchShapes(c,hexnut_shape,1,0.0)
    if val < 0.001:
        hexnut_count+=1
print("Number of matching hexagonal nuts in conveyor = ",hexnut_count)
```

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

2.Count the number of objects that were conveyed along the conveyor belt: Display the count in the current frame and total count upto the current frame in the output video. Please compress your video (using Handbreak or otherwise) before uploading. It would be good to experiment first with the two adjacent frames conveyor_f100.png and conveyor_f101.png. In order to disregard partially appearing nuts, consider comparing the contour area in addition to using the matchCountours function.

In [ ]:

```
# My code
cv.namedWindow('Conveyor',cv.WINDOW_NORMAL)
cap = cv.VideoCapture('conveyor.mp4')
f = 0
kernel = np.ones((3,3),np.uint8)
frame_array = []
shape = (1080, 1920, 3)
count_hex_total=0
count_sqr_total=0
left_ref=0

color_hex=(0,0,255)
color_sqr=(255,0,0)
color_txt=(168, 50, 121)

hex_contours, hierarchy_h = cv.findContours(closing_hexnut, cv.RETR_EXTERNAL, cv.CHAIN_
square_contours, hierarchy_s = cv.findContours(closing_squarenut, cv.RETR_EXTERNAL, cv.

# Writing the video
frame_array = []
shape = (1080, 1920, 3)

# My code

while cap.isOpened():
```

```
    ret, frame_bgr = cap.read()
    if not ret:  break

    frame=cv.cvtColor(frame_bgr,cv.COLOR_BGR2GRAY)
    th_value,frame = cv.threshold(frame,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
    frame = cv.morphologyEx(frame, cv.MORPH_CLOSE, kernel)

    count_hex_frame=0
    count_sqr_frame=0

    conts, hi = cv.findContours(frame, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

    left_max=0

    for cont in conts:
        if cv.matchShapes(cont,square_contours[0],1,0)<0.0015:
            count_sqr_frame+=1
            left=np.min(cont[:,:,0])
            if left>left_ref: count_sqr_total+=1
            if left>left_max: left_max=left
            frame_bgr= cv.drawContours(frame_bgr,[cont],0,color_sqr,5)

        elif cv.matchShapes(cont,hex_contours[0],1,0)<0.0015:
            count_hex_frame+=1
            left=np.min(cont[:,:,0])
            if left>left_ref: count_hex_total+=1
            if left>left_max: left_max=left
            frame_bgr= cv.drawContours(frame_bgr,[cont],0,color_hex,5)

    left_ref=left_max

    f += 1
    text1 = 'Frame No:   {}'.format(f)
    text2 = '          Current   Total'
    text3 = 'Hexanut     {}        {}'.format(count_hex_frame,count_hex_total)
    text4 = 'Squarenut   {}        {}'.format(count_sqr_frame,count_sqr_total)
    text5 = 'Total         {}        {}'.format(count_hex_frame+count_sqr_frame,count_h

    cv.putText(frame_bgr,text1 , (100, 90), cv.FONT_HERSHEY_COMPLEX, 1, color_txt, 1, c
    cv.putText(frame_bgr,text2 , (100, 150), cv.FONT_HERSHEY_COMPLEX, 1, color_txt, 1,
    cv.putText(frame_bgr,text3 , (100, 200), cv.FONT_HERSHEY_COMPLEX, 1, color_hex, 1,
    cv.putText(frame_bgr,text4 , (100, 250), cv.FONT_HERSHEY_COMPLEX, 1, color_sqr, 1,
    cv.putText(frame_bgr,text5 , (100, 310), cv.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 1,

    cv.imshow('Conveyor', frame_bgr)
    frame_array.append(frame_bgr)

    if cv.waitKey(2) == ord('q'):
        break

cap.release()

out = cv.VideoWriter('./conveyor_result_190520D.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()
```