
Assignment 3: Object Counting on a Conveyor Belt

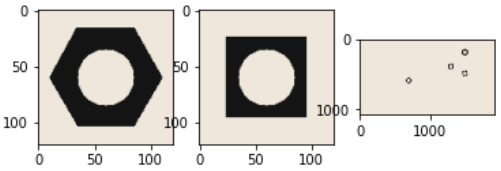
Name: Bandara D.R.K.W.M.S.D.

Index Number: 190071B
GitHub Profile: https://github.com/Saliya-99/In19-S4-EN2550---Fundamentals-of-Image-Processing-and-Machine-Vision/blob/master/Assignments/Assignment3/190071B.ipynb

---

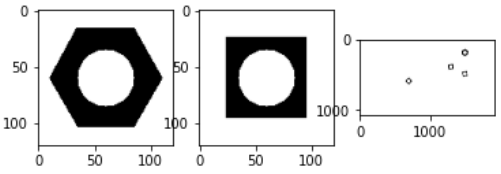
Opening of the images

In [1]:
```
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()
```



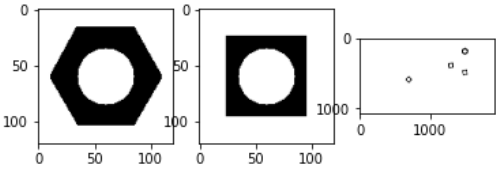
Thresholding

In [2]:
```
hexnut_template_gray = cv.cvtColor(hexnut_template,cv.COLOR_BGR2GRAY)
squarenut_template_gray = cv.cvtColor(squarenut_template,cv.COLOR_BGR2GRAY)
conveyor_f100_gray = cv.cvtColor(conveyor_f100,cv.COLOR_BGR2GRAY)
ret1,th1 = cv.threshold(hexnut_template_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
ret2,th2 = cv.threshold(squarenut_template_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
ret3,th3 = cv.threshold(conveyor_f100_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
fig, ax = plt. subplots(1,3)
ax[0].imshow(cv.cvtColor(th1, cv.COLOR_RGB2BGR))
ax[1].imshow(cv.cvtColor(th2, cv.COLOR_RGB2BGR))
ax[2].imshow(cv.cvtColor(th3, cv.COLOR_RGB2BGR))
plt.show()
```



Morphological closing

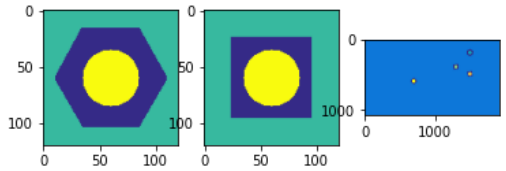
In [3]:
```
kernel = np.ones((3,3),np.uint8)
closing_1 = cv.morphologyEx(th1, cv.MORPH_CLOSE, kernel)
closing_2 = cv.morphologyEx(th2, cv.MORPH_CLOSE, kernel)
closing_3 = cv.morphologyEx(th3, cv.MORPH_CLOSE, kernel)
fig, ax = plt. subplots(1,3)
ax[0].imshow(cv.cvtColor(closing_1, cv.COLOR_RGB2BGR))
ax[1].imshow(cv.cvtColor(closing_2, cv.COLOR_RGB2BGR))
ax[2].imshow(cv.cvtColor(closing_3, cv.COLOR_RGB2BGR))
plt.show()
```



Conncted Component analysis with the statistics

In [4]:
```
connectivity = 4

retval_1, labels_1, stats_1, centroids_1 = cv.connectedComponentsWithStats(closing_1, connectivity, cv.CV_32S)
retval_2, labels_2, stats_2, centroids_2 = cv.connectedComponentsWithStats(closing_2, connectivity, cv.CV_32S)
retval_3, labels_3, stats_3, centroids_3 = cv.connectedComponentsWithStats(closing_3, connectivity, cv.CV_32S)
colormapped_1 = cv.applyColorMap((labels_1/np.amax(labels_1)*255).astype('uint8'), cv.COLORMAP_PARULA)
```

```
colormapped_2 = cv.applyColorMap((labels_2/np.amax(labels_2)*255).astype('uint8'), cv.COLORMAP_PARULA)
colormapped_3 = cv.applyColorMap((labels_3/np.amax(labels_3)*255).astype('uint8'), cv.COLORMAP_PARULA)
fig, ax = plt. subplots(1,3)
ax[0].imshow(cv.cvtColor(colormapped_1, cv.COLOR_RGB2BGR))
ax[1].imshow(cv.cvtColor(colormapped_2, cv.COLOR_RGB2BGR))
ax[2].imshow(cv.cvtColor(colormapped_3, cv.COLOR_RGB2BGR))
plt.show()
print("Detected Connected Components in hexnut_template Image: ", len(stats_1))
print("Detected Connected Components in squarenut_template Image: ", len(stats_2))
print("Detected Connected Components in conveyor_f100 Image: ", len(stats_3))
print("-----------------------------------------------Data of the first object of hex nut image---------------------------------")
print(" The leftmost (x) coordinate which is the inclusive start of the bounding box in the horizontal direction: ", stats_1[0,0])
print(" The topmost (y) coordinate which is the inclusive start of the bounding box in the vertical direction: ", stats_1[0,1])
print(" The horizontal size of the bounding box: ", stats_1[0,2])
print(" The vertical size of the bounding box: ", stats_1[0,3])
print(" The total area (in pixels) of the connected component: ", stats_1[0,4],'\n')
```




```
Detected Connected Components in hexnut_template Image:  3
Detected Connected Components in squarenut_template Image:  3
Detected Connected Components in conveyor_f100 Image:  6
-----------------------------------------------Data of the first object of hex nut image---------------------------------------
---------------------------------
 The leftmost (x) coordinate which is the inclusive start of the bounding box in the horizontal direction:  11
 The topmost (y) coordinate which is the inclusive start of the bounding box in the vertical direction:  16
 The horizontal size of the bounding box:  99
 The vertical size of the bounding box:  88
 The total area (in pixels) of the connected component:  4722
```

Stats is a matrix contains the

1. The leftmost (x) coordinate which is the inclusive start of the bounding box in the horizontal direction
2. The topmost (y) coordinate which is the inclusive start of the bounding box in the vertical direction
3. The horizontal size of the bounding box
4. The vertical size of the bounding box
5. The total area (in pixels) of the connected component

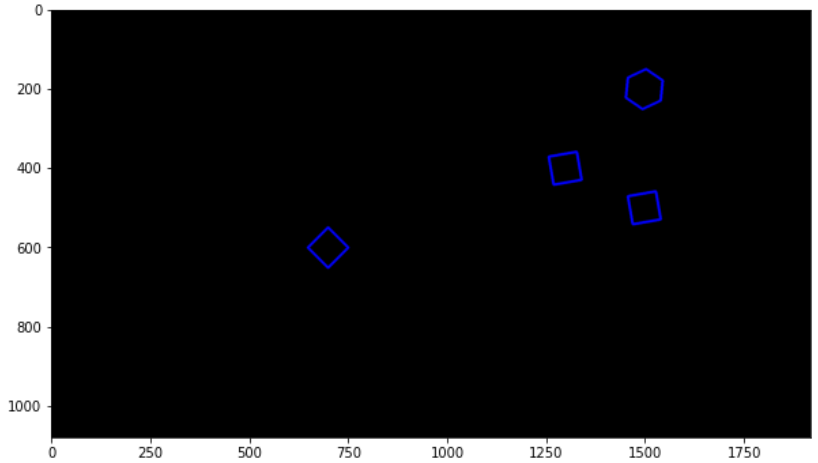
   in each column of the matrix and one row for one object

Plotting extreme Contours

In here, Used areas of contours to filter the outrmost contour instead of using cv..RETR_EXTERNAL.

In [29]:
```
img = np.zeros((conveyor_f100.shape[0],conveyor_f100.shape[1],conveyor_f100.shape[2]))
contours_1, hierarchy_1 = cv.findContours(th1, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
contours_2, hierarchy_2 = cv.findContours(th2, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
contours_3, hierarchy_2 = cv.findContours(th3, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
areas = []
for i in contours_3[1:]: # do not consider the background as a connected region
    areas.append(cv.contourArea(i))

for j,k in enumerate(areas):
    if abs(k - max(areas))<2000:
        cv.drawContours(img, [contours_3[j+1]], 0, (0,0,255), 5)

fig, ax = plt. subplots(1,1,figsize = (10,10))
ax.imshow(img)
plt.show()
```

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



Above picture shows the extreme countours of the detected objects.

Opening Video

In [27]:
```
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.LINE_AA)
    cv.imshow('Conveyor', frame)

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

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

Counting the number of specific object in an image

In here, number of hexagonal nuts in the image is counting using matchShapes function.

In [30]:
```
count = 0
for i in contours_3[1:]:
    ret = cv.matchShapes(contours_1[1],i,1,0.0)
    if ret < 10**(-3):
        count += 1
print("Count of hexaganal nuts in conveyor_f100: ", count)
```

```
Count of hexaganal nuts in conveyor_f100:  1
```

Object counting, Tracking, and Displaying on video frame

In here, using centroids of objects to track the object througout the video in order to calculate total number of nuts up to a frame.

In [10]:
```
# Yor code here.
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

cv.namedWindow('Conveyor', cv.WINDOW_NORMAL)
cap = cv.VideoCapture('conveyor.mp4')
f = 0
frame = []
frame_array = []
shape = (1080, 1920, 3)
allhex = 0
allsqr = 0
allcenhex = [[]]
allcensqr = [[]]
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting.")
        break

    f += 1
    frame_gray = cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
    ret3,th = cv.threshold(frame_gray,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
    contours, hierarchy = cv.findContours(th, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    cv.drawContours(frame, contours, -1, (0,255,0), 3)
    count_hex = 0
    count_sqr = 0
    areas = []
    hexs = []
    sqrs = []
    simcountHex = 0
    simcountsqr = 0
    for g,i in enumerate(contours[1:]):
        res_1 = cv.matchShapes(contours_1[1],i,1,0.0)
        res_2 = cv.matchShapes(contours_2[1],i,1,0.0)
        if res_1 < 10**(-4):
            count_hex += 1
            M = cv.moments(i)
            cx, cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])
            prev_hex = allcenhex[-1]

            for cen1 in prev_hex:
                if abs(cen1[0]-cx)<=15 and abs(cen1[1]-cy) <= 15:
                    simcountHex += 1
                    break
            hexs.append((cx,cy))


        if res_2 < 10**(-3):
            count_sqr += 1
```

```
            M = cv.moments(i)
            cx, cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])
            prev_sqr = allcensqr[-1]

            for cen1 in prev_sqr:
                if abs(cen1[0]-cx)<=15 and abs(cen1[1]-cy) <= 15:
                    simcountsqr += 1
                    break
            sqrs.append((cx,cy))
    allcensqr.append(sqrs)
    allcenhex.append(hexs)
    allhex += (count_hex - simcountHex)
    allsqr += (count_sqr - simcountsqr)

    # hex_all = hex_all+count_hex-

    text = 'Frame:' + str(f)
    text2 = "Num of hexagonal nuts in current frame: " + str(count_hex)
    text3 = "Num of square nuts in current frame: " + str(count_sqr)
    text4 = "Total hex nuts: " + str(allhex)
    text5 = "Total square nuts: " + str(allsqr)
    cv.putText(frame,text , (100, 60), cv.FONT_HERSHEY_COMPLEX,  1, (0,0,255), 1, cv.LINE_AA)
    cv.putText(frame,text2 , (100, 80), cv.FONT_HERSHEY_COMPLEX, 0.7, (0,0,1), 1, cv.LINE_AA)
    cv.putText(frame,text3 , (100, 100), cv.FONT_HERSHEY_COMPLEX, 0.7, (0,0,1), 1, cv.LINE_AA)
    cv.putText(frame,text4 , (100, 120), cv.FONT_HERSHEY_COMPLEX, 0.7, (255,0,0), 1, cv.LINE_AA)
    cv.putText(frame,text5 , (100, 140), cv.FONT_HERSHEY_COMPLEX, 0.7, (255,0,0), 1, cv.LINE_AA)
    cv.putText(frame,'Index No: 190071B' , (100, 1000), cv.FONT_HERSHEY_COMPLEX, 0.7, (255,0,0), 1, cv.LINE_AA)
    frame_array.append(frame)
    cv.imshow('Conveyor', frame)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()

#Video write
out = cv.VideoWriter('D:/Study Materials/Sem 4/EN2550 - Fundamentals of Image Processing and Machine Vision/conveyor_result_190071

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()
```

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