180717E
WIJITHARATHNA KMR

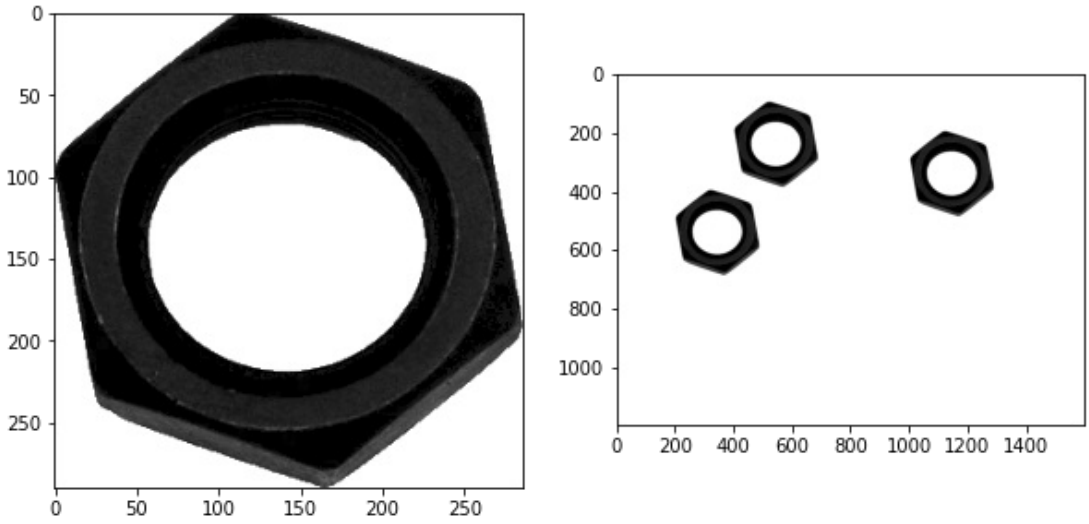
Part I

In [69]:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
%matplotlib inline
```

In [70]:

```
template_im = cv.imread('template.png', cv.IMREAD_GRAYSCALE)
belt_im = cv.imread('belt.png', cv.IMREAD_GRAYSCALE)
fig, ax = plt. subplots(1,2,figsize=(10,10))
ax[0].imshow(template_im, cmap='gray')
ax[1].imshow(belt_im, cmap='gray')
plt.show()
```



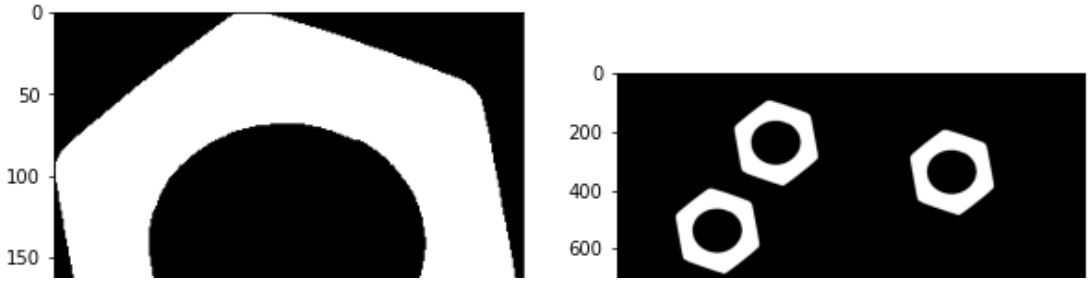
Otsu's thresholding

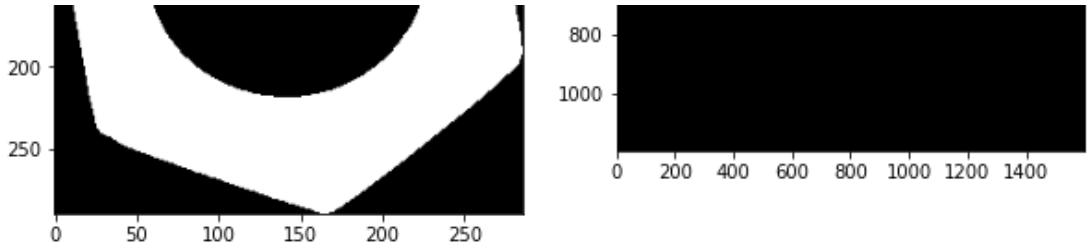
In [71]:

```
th_t, img_t = cv.threshold(template_im,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU) #first variable gets assigned the threshold value
th_b, img_b = cv.threshold(belt_im,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
```

In [72]:

```
fig, ax = plt. subplots(1,2,figsize=(10,10))
ax[0].imshow(img_t, cmap='gray')
ax[1].imshow(img_b, cmap='gray')
plt.show()
```





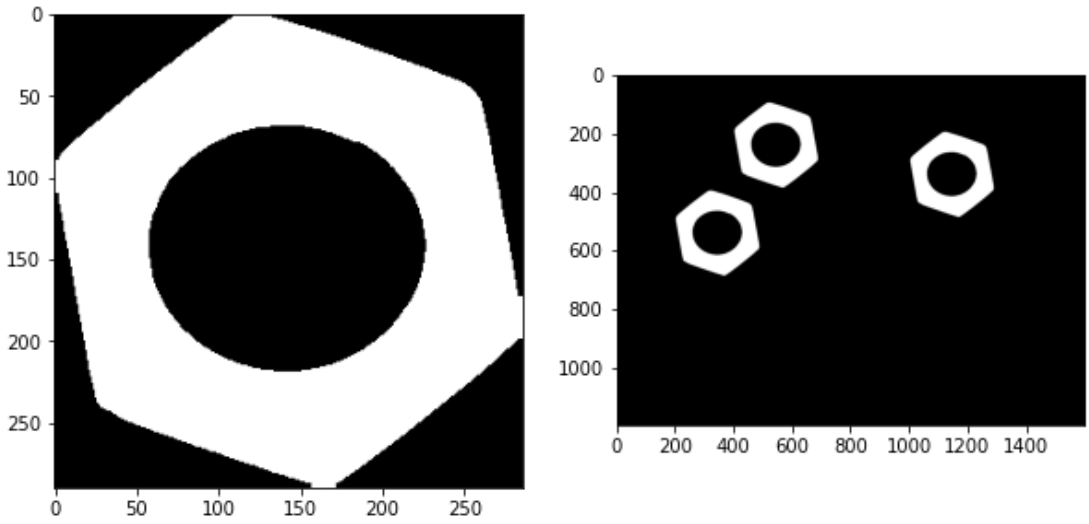
Morphological closing

In [73]:

```
kernel = np.ones((5,5),np.uint8)
closing_t = cv.morphologyEx(img_t, cv.MORPH_CLOSE, kernel)
closing_b = cv.morphologyEx(img_b, cv.MORPH_CLOSE, kernel)
```

In [74]:

```
fig, ax = plt. subplots(1,2,figsize=(10,10))
ax[0].imshow(closing_t, cmap='gray')
ax[1].imshow(closing_b, cmap='gray')
plt.show()
```



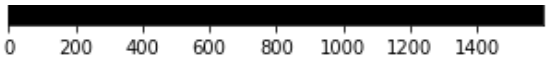
Connected component analysis

In [75]:

```
retval_t, labels_t, stats_t, centroids_t = cv.connectedComponentsWithStats(closing_t)
#numofLabels , labels, stats, centroids
retval_b, labels_b, stats_b, centroids_b = cv.connectedComponentsWithStats(closing_b)
```

In [76]:

```
plt.imshow(labels_b,cmap="gray")
plt.show()
```



# of connected components
In image template, 2 connected components including the background , in image belt, 4 connected components also including the background as a connect compenent.

In [142]:

```
print("template.png\t",retval_t)
print("belt.png\t",retval_b)
```

```
template.png	 2
belt.png	 4
```

Stats
Stats includes the bounding boxes and the area of each of the connected components.
[starting coordinates of the bounding box(top left corner), width and height of the bounding box, the area]

In [138]:

```
print("First component is the background")
print("[top left x , top left y, b box width, b box height, b box area ]\n")
print("template.png")
for stat in stats_t:
  print(stat)
print("\n")

print("belt.png")
for stat in stats_b:
  print(stat)
```

```
First component is the background
[top left x , top left y, b box width, b box height, b box area ]

template.png
[    0     0   286   290 42242]
[    0     0   286   290 40698]


belt.png
[      0       0    1600    1200 1798161]
[  400   100   286   290 40613]
[ 1000   200   286   290 40613]
[  200   400   286   290 40613]
```

Centroids
Centroids give the center (x,y) coordinates of a connected component.

In [140]:

```
print("template.png")
for centroid in centroids_t:
  print(centroid)
print("\n")
print("belt.png")
for centroid in centroids_b:
  print(centroid)
```

```
template.png
[142.17589129 145.20387292]
[142.83640474 143.76942356]


belt.png
[807.85728475 614.56805258]
[542.82567158 243.78479797]
[1142.82567158  343.78479797]
[342.82567158 543.78479797]
```

Contour analysis

In [79]:

```
contours_t, hierarchy_t = cv.findContours(closing_t, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPL
E)  #https://docs.opencv.org/4.5.2/d9/d8b/tutorial_py_contours_hierarchy.html
contours_b, hierarchy_b = cv.findContours(closing_b, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPL
E)

#Contours is a Python list of all the contours in the image. Each individual contour is a
Numpy array of (x,y) coordinates of boundary points of the object.
```

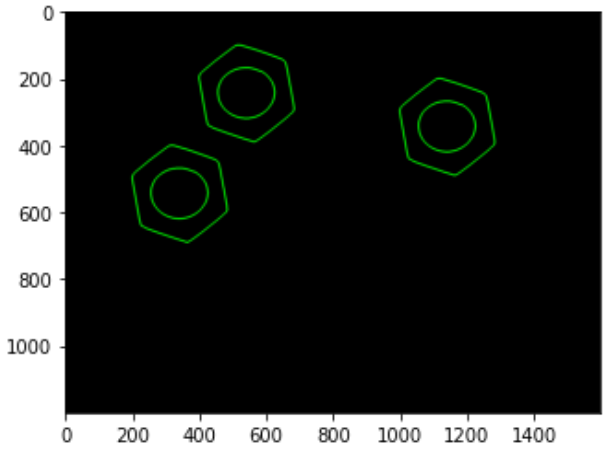
In [80]:

```
print(hierarchy_b)
```

```
[[[ 2 -1  1 -1]
  [-1 -1 -1  0]
  [ 4  0  3 -1]
  [-1 -1 -1  2]
  [-1  2  5 -1]
  [-1 -1 -1  4]]]
```

In [81]:

```
im_contours_belt = np.zeros((belt_im.shape[0],belt_im.shape[1],3), np.uint8) #empty imag
e to draw the contours
conts = cv.drawContours(im_contours_belt, contours_b, -1, (0,255),3).astype('uint8') # s
ource image, contours as a list, color range, no of channels
plt.imshow(conts)
plt.show()
```



Count the number of matching hexagonal nuts in belt.png.

In a perfect match the returned value of the function cv.matchShapes() is zero. Here we have gotten a very small value, which means a good similarity.

In [82]:

```
label = 1 # remember that the label of the background is 0
belt = ((labels_b >= label)*255).astype('uint8') #labels is an image containing the conn
ected components, what is this??
belt_cont, template_hierarchy = cv.findContours(belt, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_
SIMPLE)
for j,c in enumerate(belt_cont):
  print(cv.matchShapes(contours_t[0], c, cv.CONTOURS_MATCH_I1, 0.0))
```

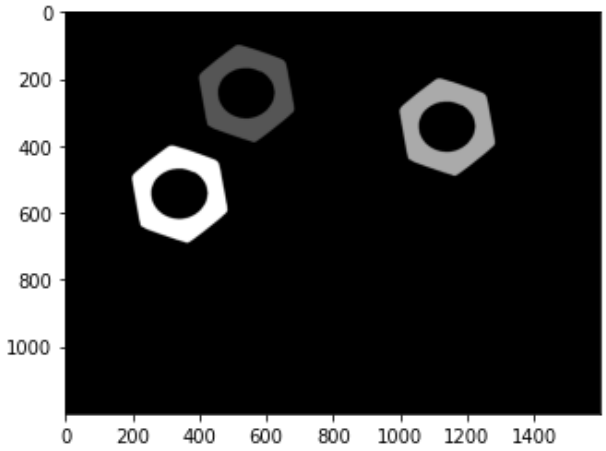
```
0.0002170059775208344
0.0002170059775752860595
0.0002170059775752416506
```

In [136]:

```
plt.imshow(labels_b,cmap='gray')
```

Out[136]:

```
<matplotlib.image.AxesImage at 0x7f271b84a350>
```



Part II

Frame tracking through image moments

In [84]:

```
ca = cv.contourArea(contours_b[1])
M = cv.moments(contours_b[1])
cx,cy = int(M['m10']/M['m00']),int(M['m01']/M['m00'])
count = 1
object_prev_frame = np.array([cx,cy,ca,count])  # count acts as an identifier to object
delta_x = 15
```

Part III

Implement the function get_indexed_image, which takes an image as theinput, performs thresholding, closing, and connected component analysis and return retval, labels, stats, centroids

In [85]:

```
def get_indexed_image(im):

  thresh, img_tresholded = cv.threshold(im,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
  kernel = np.ones((5,5),np.uint8)
  closed_img = cv.morphologyEx(img_tresholded, cv.MORPH_CLOSE, kernel)
  retval, labels, stats, centroids = cv.connectedComponentsWithStats(closed_img)

  return retval, labels, stats, centroids, closed_img
```

Implement the function is_new, which checks the dissimilarity between 2 vectors

In [86]:

```
'''def is_new(a,b,delta,i):
  for row in range(len(a)):
    flag = True
```

```
    for col in i:
      print(abs(a[row][col]-b[col]))
      print(delta[col])
      if abs(a[row][col]-b[col]) <= delta[col]:
        flag = False
        break
    if flag:
      return flag
  return False'''


def is_new(a, b, delta, i):
  b_np = np.array(b)
  b_np = b_np[i]
  delta_np = np.array(delta)
  delta_np = delta_np[i]
  for row in np.array(a):
    L1 = abs(row[i]-b_np)
    if np.less_equal(L1,delta_np).all():
      return False
  return True
```

In [87]:

```
a = np.array([[1.36100e+03, 5.53000e+02, 5.99245e+04, 2.00000e+00],
[7.61000e+02, 4.53000e+02, 5.99385e+04, 1.00000e+00],
[1.55200e+03, 2.43000e+02, 6.00585e+04, 3.00000e+00]])
b = np.array([7.51000e+02, 4.53000e+02, 5.99385e+04, 3.00000e+00])
delta = np.array([delta_x,delta_x])
i = np.array([0,1])
is_new(a, b, delta, i)
```

Out[87]:

```
False
```

In [88]:

```
delta_x
```

Out[88]:

```
15
```

If the array a is in the shape of (number of nuts , len (object_prev_frame)) (i.e. array a is made by stacking all the object_prev_frame for each frame.) If b is in the form of [cx, cy, ca, count], write the function prev_index to find the index of a particular nut in the previous frame

In [89]:

```
def prev_index(a, b, delta, i):
  """ Returns Previous Index
  Returns the index of the apppearance of the object in the previous frame.
  (See thee example in the next cell)
  """
  index = -1

  for index in range(len(a)):
    flag = False     #set flag to false each time we start
    for col in i:
      if abs(a[index][col]-b[col]) > delta[col]:
        flag =True    #set flag to notify that the break has occured
        break
    if flag == False:    #break has occured, so we don't return index yet, there is some
thing causing diff larger than the delta in that row
      return index
  #return index
```

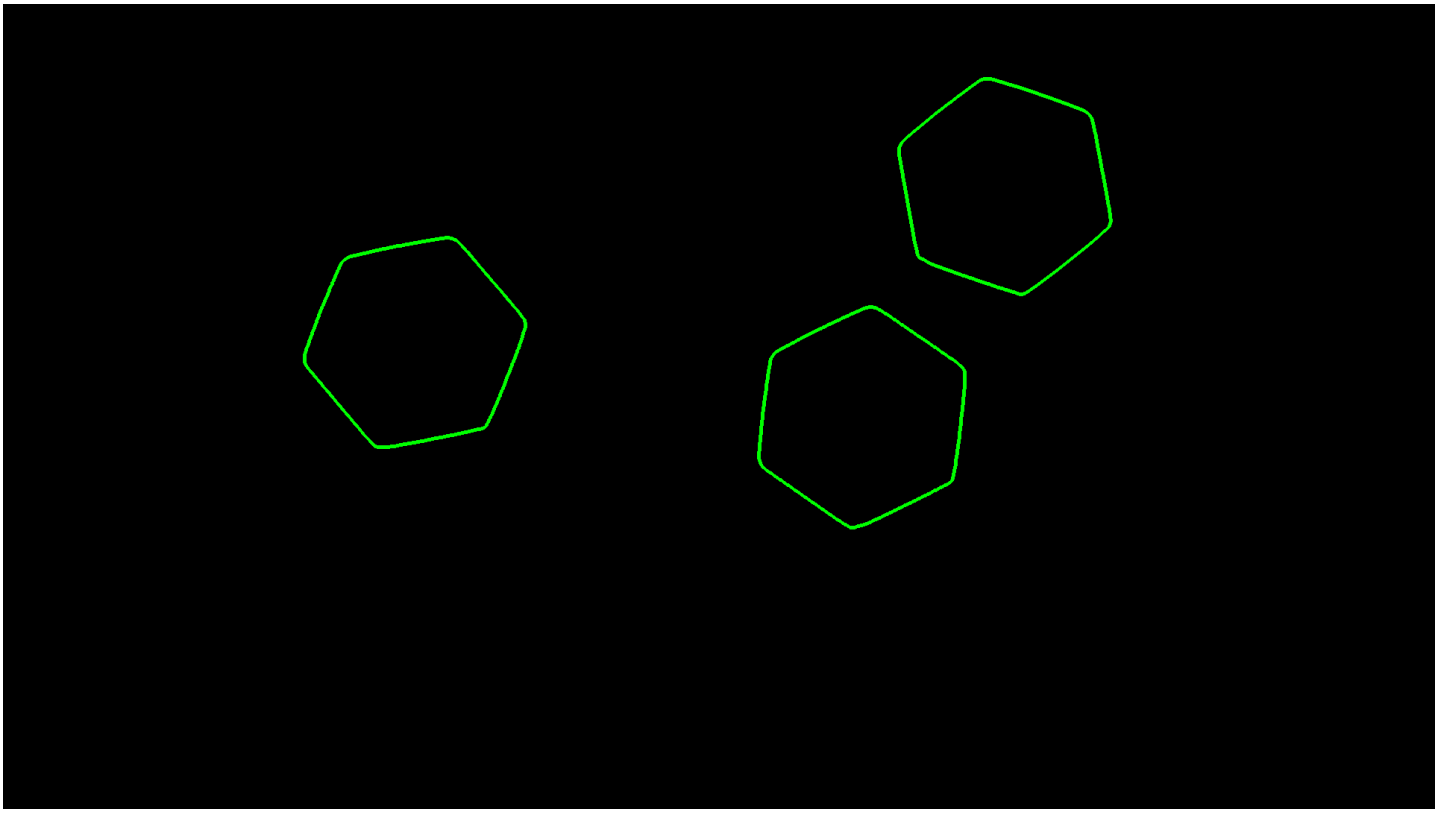
In [90]:

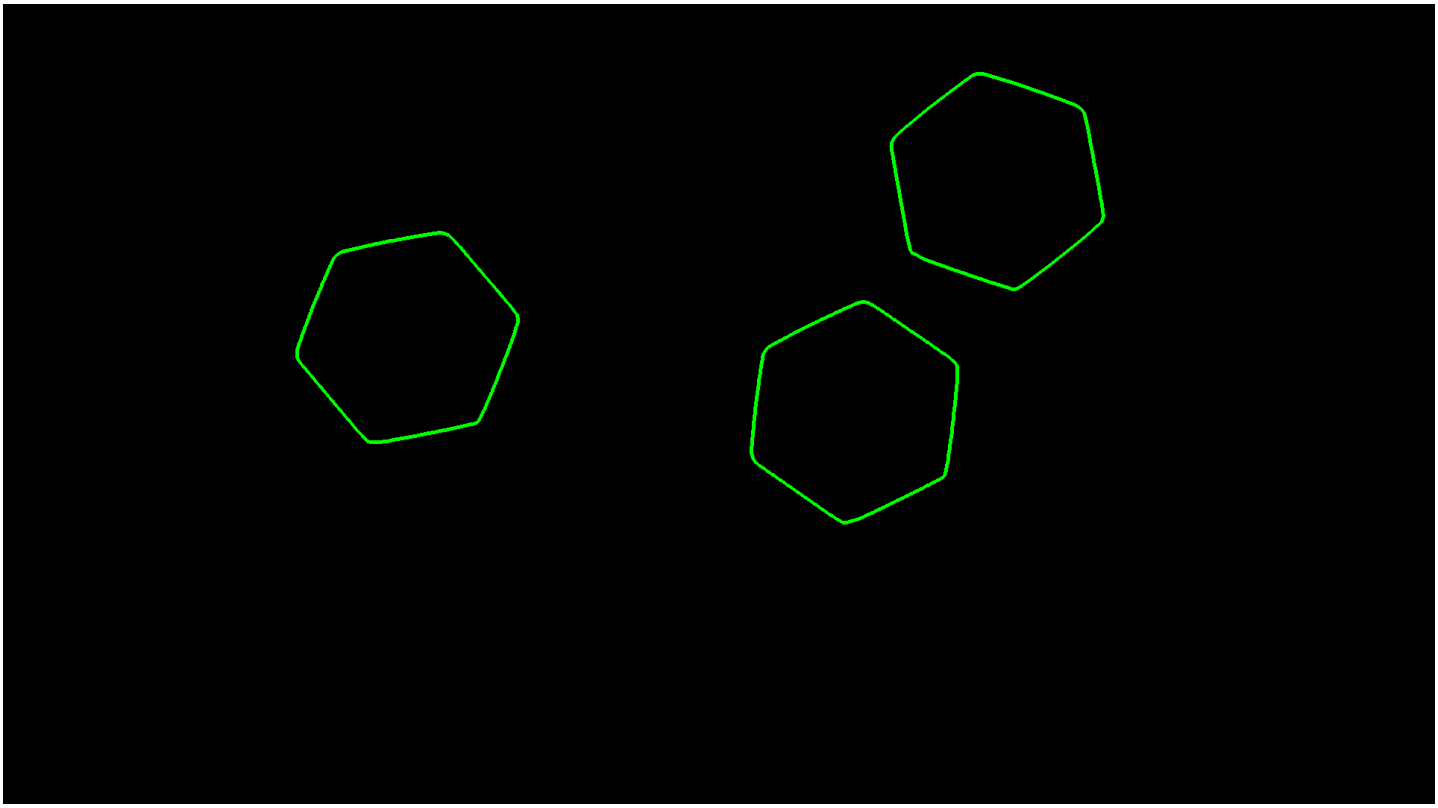
```
# check prev_index expected answer 1
a = np.array([[1.36100e+03, 5.53000e+02, 5.99245e+04, 2.00000e+00],
[7.61000e+02, 4.53000e+02, 5.99385e+04, 1.00000e+00],
[1.55200e+03, 2.43000e+02, 6.00585e+04, 3.00000e+00]])
b = np.array([7.51000e+02, 4.53000e+02, 5.99385e+04, 3.00000e+00])
delta = np.array([delta_x])
i = np.array([0])
assert prev_index(a,b,delta,i) == 1, " Check the function "
```

Implement a code to detect hexagonal nuts in a moving convey belt.

In [91]:

```
cap = cv.VideoCapture('conveyor_two_frame.mp4') # give the correct path here
while cap.isOpened():
  ret, frame = cap.read()
  if not ret:
    print("Can't receive frame (stream end?). Exiting ...")
    break
  grey = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
  retval, labels, stats, centroids,closed_img = get_indexed_image(grey)
  contours, hierarchy = cv.findContours(closed_img, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE
)
  im_contours_belt = np.zeros((closed_img.shape[0],closed_img.shape[1],3), np.uint8) #em
pty image to draw the contours

  #checking for the hexagon contours
  selected_conts = []
  for j,c in enumerate(contours):  #in enumerate j correspnds to the index and c correspo
nds to the item in contours
    if (cv.matchShapes(contours_t[0], c, cv.CONTOURS_MATCH_I1, 0.0)< 0.00025): #contours
_t[0] template's contour of hexagon, 0.00025 was selected by looking at the previous valu
es
      selected_conts.append(c)
  cv.drawContours(im_contours_belt, selected_conts, -1, (0,255,0), 3)
  cv2_imshow(im_contours_belt)
  print("\n")
  if cv.waitKey(1) == ord('q'):
    break
cap.release()
cv.destroyAllWindows()
```





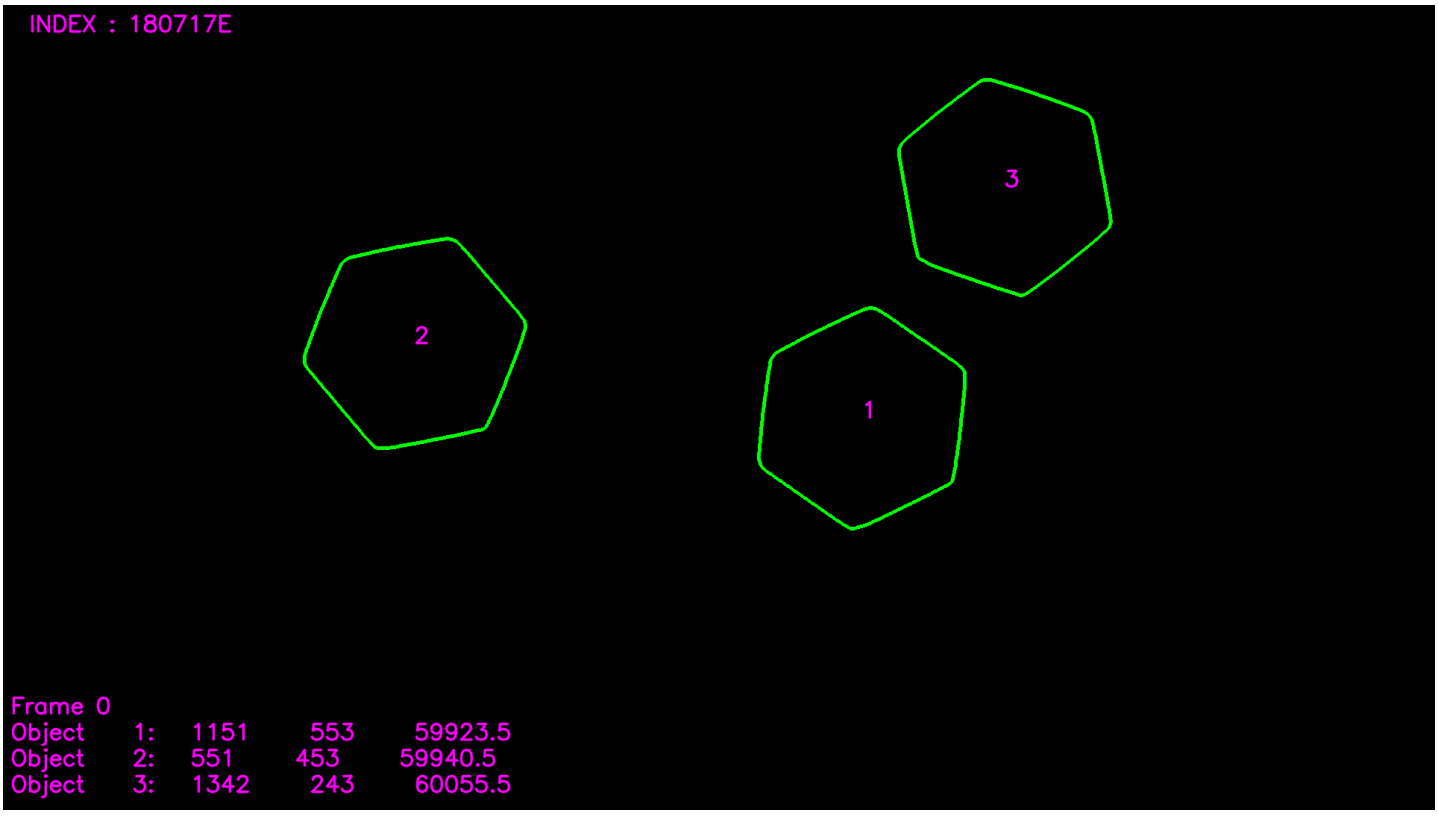
```
Can't receive frame (stream end?). Exiting ...
```

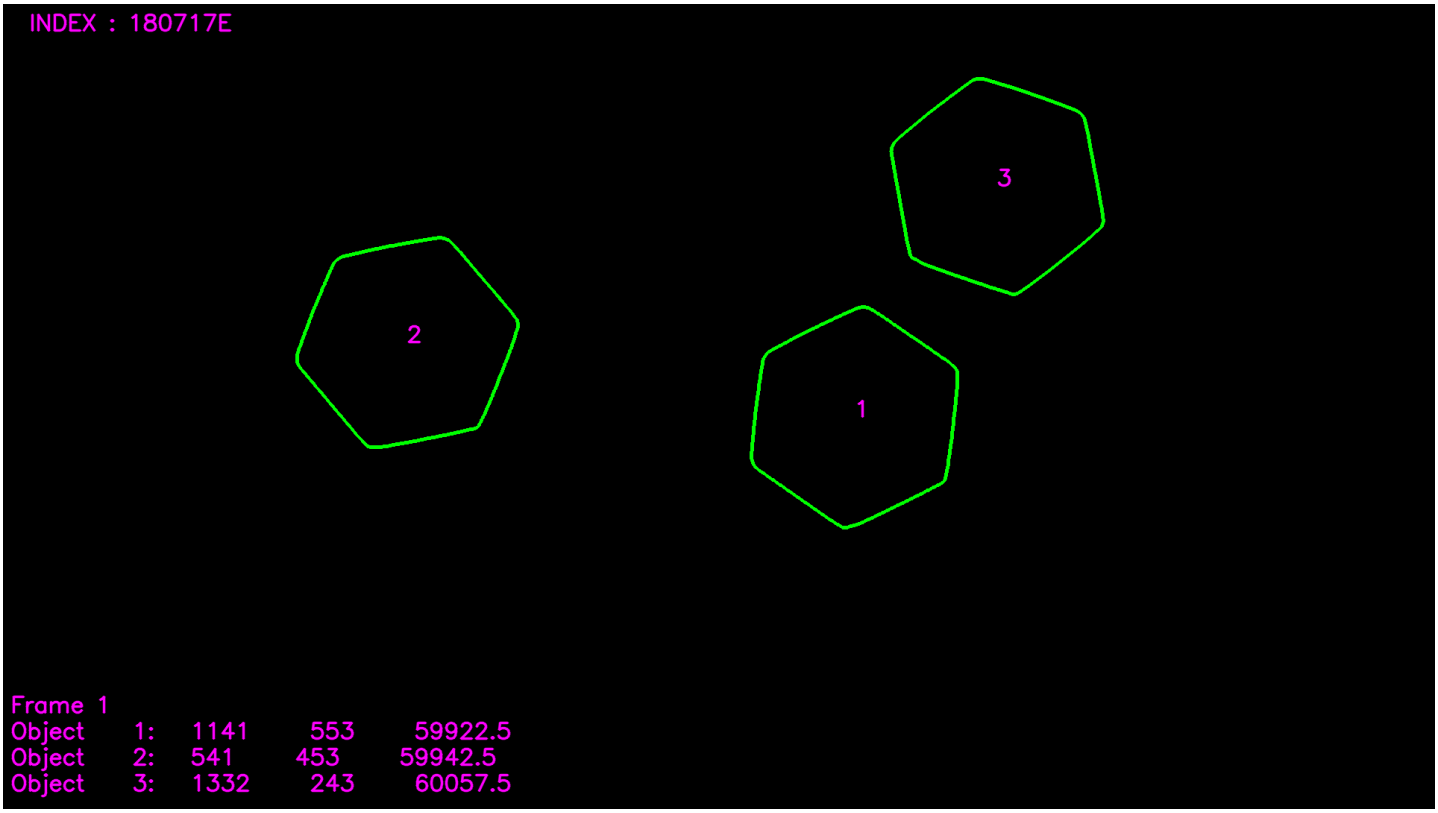
Object detection and tracking

In [124]:

```
cap = cv.VideoCapture('conveyor_two_frame.mp4') # give the correct path here
frm = 0
while cap.isOpened():
  ret, frame = cap.read()
  if not ret:
    print("Can't receive frame (stream end?). Exiting ...")
    break
  grey = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
  retval, labels, stats, centroids,closed_img = get_indexed_image(grey)
  contours, hierarchy = cv.findContours(closed_img, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE
)
  im_contours_belt = np.zeros((closed_img.shape[0],closed_img.shape[1],3), np.uint8) #em
pty image to draw the contours

  #checking for the hexagon contours
  selected_conts = []
  for j,c in enumerate(contours):  #in enumerate j correspnds to the index and c correspo
nds to the item in contours
    if (cv.matchShapes(contours_t[0], c, cv.CONTOURS_MATCH_I1, 0.0)< 0.00025): #contours
_t[0] template's contour of hexagon, 0.00025 was selected by looking at the previous valu
es
      selected_conts.append(c)
  cv.drawContours(im_contours_belt, selected_conts, -1, (0,255,0), 3)

  curr_frame =[]
  if frm ==0:     #if this is the first frame no need to check if the objects are new or
not
    for i in range(len(selected_conts)):
      M = cv.moments(selected_conts[i])
      cx, cy = int(M['m10']/M['m00']),int(M['m01']/M['m00'])
      ca = cv.contourArea(selected_conts[i])
      obj_curr_frame = [cx,cy,ca,i+1]
      curr_frame.append(obj_curr_frame)
      cv.putText(im_contours_belt,str(obj_curr_frame[3]),(obj_curr_frame[0],obj_curr_fra
me[1]), cv.FONT_HERSHEY_SIMPLEX, 1,(255,0,255),2,cv.LINE_AA)
    prev_count = i+1
```

```
  else:
    i=[0,1,2]
    delta_x, delta_y, delta_ca = 15,15,3000
    delta = [delta_x, delta_y, delta_ca]
    for k in range(len(selected_conts)):
      M = cv.moments(selected_conts[k])
      cx, cy = int(M['m10']/M['m00']),int(M['m01']/M['m00'])
      ca = cv.contourArea(selected_conts[k])
      obj_curr_frame = [cx,cy,ca]        # we do not know if this is a new object(positi
on) or not, hence we do not add count

      if is_new(prev_frame, obj_curr_frame, delta, i):
        prev_count += 1
        obj_curr_frame.append(prev_count)
        print("new object")
      else:
        prev_ind = prev_index(prev_frame,obj_curr_frame,delta,i)
        obj_curr_frame.append(prev_frame[prev_ind][3])
      curr_frame.append(obj_curr_frame)
      cv.putText(im_contours_belt,str(obj_curr_frame[3]),(obj_curr_frame[0],obj_curr_fra
me[1]), cv.FONT_HERSHEY_SIMPLEX, 1,(255,0,255),2,cv.LINE_AA)
  prev_frame = curr_frame
  prev_count = max(row[3] for row in curr_frame)

  #adding index
  cv.putText(im_contours_belt,"INDEX : 180717E",(35,35), cv.FONT_HERSHEY_SIMPLEX, 1,(255
,0,255),2,cv.LINE_AA)

  #addding bottom text
  bottom_text = 'Frame {}'.format(frm)
  cv.putText(im_contours_belt,bottom_text,(10,950), cv.FONT_HERSHEY_SIMPLEX, 1,(255,0,25
5),2,cv.LINE_AA)
  j =1
  for obj in curr_frame: #putText doesnt support newline nor tab calls
    bottom_text = 'Object    {}:   {}     {}     {}'.format(obj[3],obj[0],obj[1],obj[2])
    cv.putText(im_contours_belt,bottom_text,(10,950+35*j), cv.FONT_HERSHEY_SIMPLEX, 1,(2
55,0,255),2,cv.LINE_AA)
    j+= 1
  cv2_imshow(im_contours_belt)
  frm +=1
  print("\n")
  if cv.waitKey(1) == ord('q'):
    break
cap.release()
cv.destroyAllWindows()
```






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

Video 2

In [127]:

```
cap = cv.VideoCapture('conveyor_with_rotation.mp4') # give the correct path here
width=1920
height=1080
fps = 30        # obtained by checking the video properties in windows

file_obj = cv.VideoWriter('180717e_en2550_a05.mp4', cv.VideoWriter_fourcc(*'mp4v'), fps,
(width, height), True)
frm = 0
while cap.isOpened():
  ret, frame = cap.read()
  if not ret:
    print("Can't receive frame (stream end?). Exiting ...")
    break
  grey = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
  retval, labels, stats, centroids,closed_img = get_indexed_image(grey)
  contours, hierarchy = cv.findContours(closed_img, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE
)
  im_contours_belt = np.zeros((closed_img.shape[0],closed_img.shape[1],3), np.uint8) #em
pty image to draw the contours

  #checking for the hexagon contours
  selected_conts = []
  for j,c in enumerate(contours):  #in enumerate j correspnds to the index and c correspo
nds to the item in contours
    if (cv.matchShapes(contours_t[0], c, cv.CONTOURS_MATCH_I1, 0.0)< 0.00025): #contours
_t[0] template's contour of hexagon, 0.00025 was selected by looking at the previous valu
es
      selected_conts.append(c)
  cv.drawContours(im_contours_belt, selected_conts, -1, (0,255,0), 3)

  curr_frame =[]
  if frm ==0:     #if this is the first frame no need to check if the objects are new or
not
    for i in range(len(selected_conts)):
```

```
      M = cv.moments(selected_conts[i])
      cx, cy = int(M['m10']/M['m00']),int(M['m01']/M['m00'])
      ca = cv.contourArea(selected_conts[i])
      obj_curr_frame = [cx,cy,ca,i+1]
      curr_frame.append(obj_curr_frame)
      cv.putText(im_contours_belt,str(obj_curr_frame[3]),(obj_curr_frame[0],obj_curr_fra
me[1]), cv.FONT_HERSHEY_SIMPLEX, 1,(255,0,255),2,cv.LINE_AA)
    prev_count = i+1
  else:
    i=[0,1,2]
    delta_x, delta_y, delta_ca = 15,15,2000
    delta = [delta_x, delta_y, delta_ca]
    for k in range(len(selected_conts)):
      M = cv.moments(selected_conts[k])
      cx, cy = int(M['m10']/M['m00']),int(M['m01']/M['m00'])
      ca = cv.contourArea(selected_conts[k])
      obj_curr_frame = [cx,cy,ca]        # we do not know if this is a new object(positi
on) or not, hence we do not add count

      if is_new(prev_frame, obj_curr_frame, delta, i):
        prev_count += 1
        obj_curr_frame.append(int(prev_count))
        #print("new object {}".format(prev_count))
      else:
        prev_ind = prev_index(prev_frame,obj_curr_frame,delta,i)
        obj_curr_frame.append(prev_frame[prev_ind][3])
      curr_frame.append(obj_curr_frame)
      cv.putText(im_contours_belt,str(int(obj_curr_frame[3])),(obj_curr_frame[0],obj_cur
r_frame[1]), cv.FONT_HERSHEY_SIMPLEX, 1,(255,0,255),2,cv.LINE_AA)

  #cv2_imshow(im_contours_belt)
  prev_frame = np.concatenate((curr_frame,prev_frame))
  prev_count = int(max(row[3] for row in curr_frame))

  #adding index no
  cv.putText(im_contours_belt,"INDEX : 180717E",(35,35), cv.FONT_HERSHEY_SIMPLEX, 1,(255
,0,255),2,cv.LINE_AA)

  #adding bottom text
  bottom_text = 'Frame {}'.format(frm)
  cv.putText(im_contours_belt,bottom_text,(10,880), cv.FONT_HERSHEY_SIMPLEX, 1,(255,0,25
5),2,cv.LINE_AA)
  j =1
  for obj in curr_frame: #putText doesnt support newline nor tab calls
    bottom_text = 'Object    {}:   {}     {}     {}'.format(obj[3],obj[0],obj[1],obj[2])
    cv.putText(im_contours_belt,bottom_text,(10,880+35*j), cv.FONT_HERSHEY_SIMPLEX, 1,(2
55,0,255),2,cv.LINE_AA)
    j+= 1


  file_obj.write(im_contours_belt)
  frm +=1
  #print("max count {} \n".format(prev_count))
  if cv.waitKey(1) == ord('q'):
    break
cap.release()
file_obj.release()
cv.destroyAllWindows()
```

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