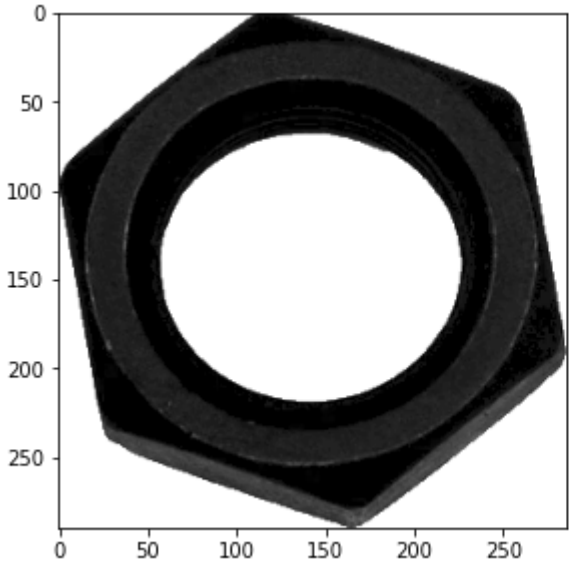
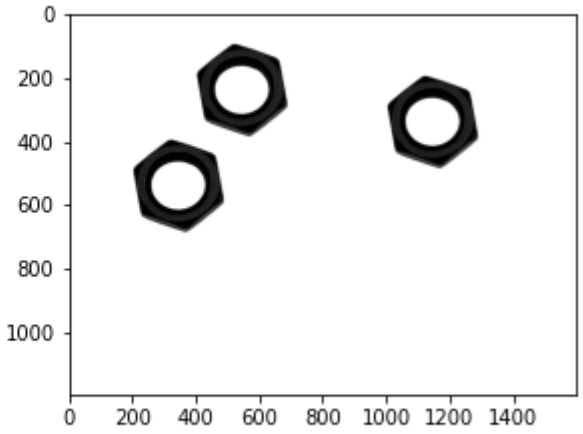
Object Counting on a Convey Belt

In [1]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
```

Load and visualize the template image and the convey belt snapshot

In [2]:
```
template_im = cv.imread(r'template.png', cv.IMREAD_GRAYSCALE)
belt_im = cv.imread(r'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()
```

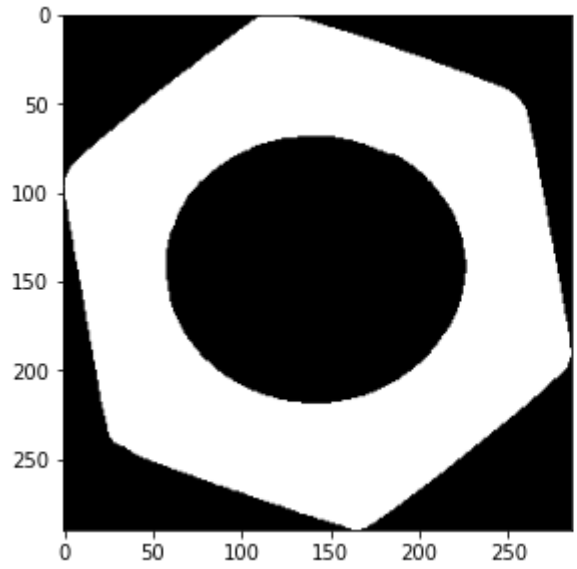


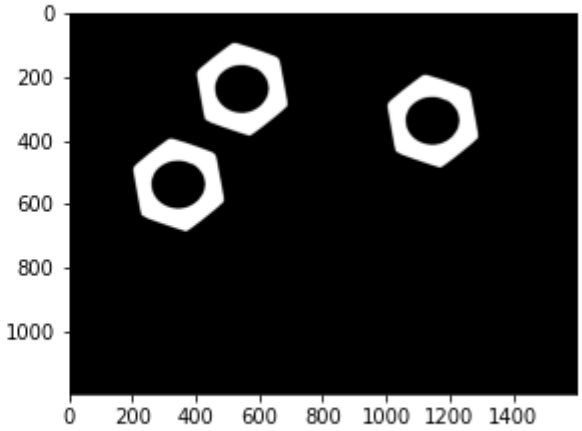


Part 1

Otsu's thresholding

In [3]:
```
th_t, img_t=cv.threshold(template_im,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
th_b, img_b=cv.threshold(belt_im,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
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()
```

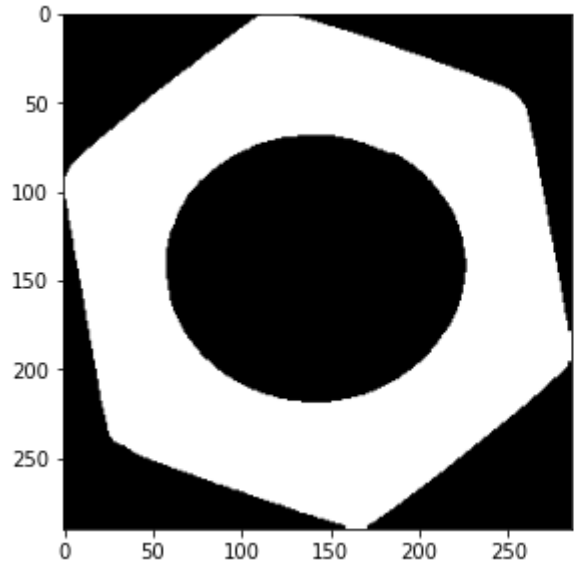


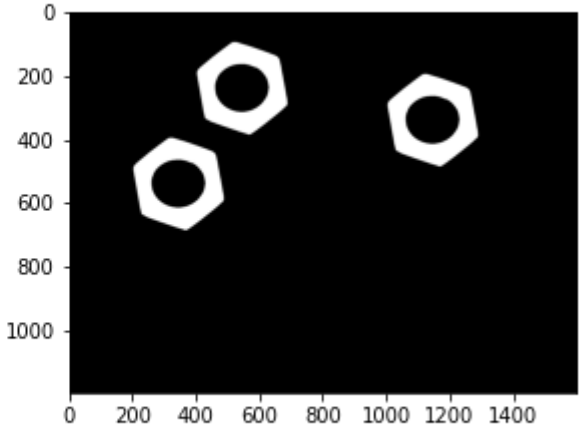


Morphological closing

In [4]:

```
kernel = np.ones((3,3),dtype=np.uint8)
closing_t = cv.morphologyEx(img_t, cv.MORPH_CLOSE, kernel)
closing_b = cv.morphologyEx(img_b, cv.MORPH_CLOSE, kernel)
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 [5]:

```
retval_t, labels_t, stats_t, centroids_t = cv.connectedComponentsWithStats(closing_t)
retval_b, labels_b, stats_b, centroids_b = cv.connectedComponentsWithStats(closing_b)
print("Number of connected components in template:",retval_t)
print(stats_t)
print("Centroids of the template: ",centroids_t)
print("Number of connected components in belt:",retval_b)
print(stats_b)
print("Centroids of the belt: ",centroids_b)
```

```
Number of connected components in template: 2
[[    0     0   286   290 42290]
 [    0     0   286   290 40650]]
Centroids of the template:  [[142.18770395 145.19172381]
 [142.82489545 143.780369  ]]
Number of connected components in belt: 4
[[      0       0    1600    1200 1798161]
 [    400     100     286     290   40613]
 [   1000     200     286     290   40613]
 [    200     400     286     290   40613]]
Centroids of the belt:  [[ 807.85728475  614.56805258]
 [ 542.82567158  243.78479797]
 [1142.82567158  343.78479797]
 [ 342.82567158  543.78479797]]
```

The statistics given in the 5 element array are:

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.

Contour analysis

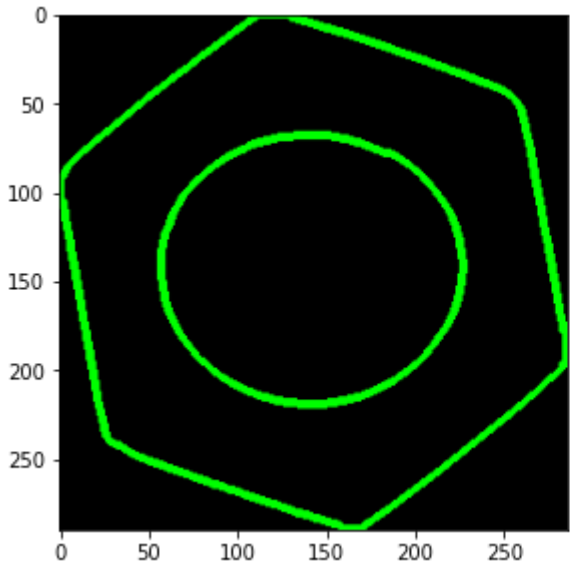
In [6]:

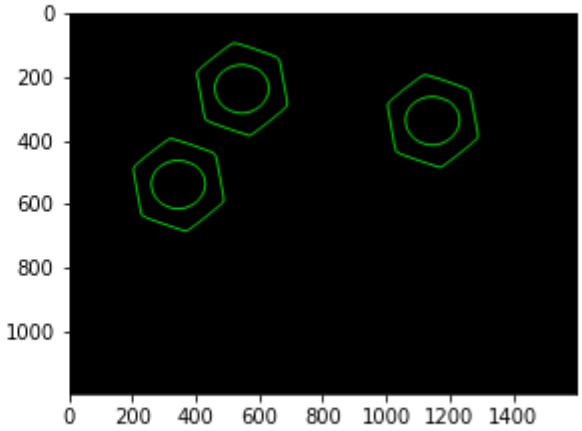
```
contours_t, hierarchy_t = cv.findContours(closing_t, cv.RETR_TREE, cv.CHAIN_APPROX_SIMP
contours_b, hierarchy_b = cv.findContours(closing_b, cv.RETR_TREE, cv.CHAIN_APPROX_SIMP
print(len(contours_b))
print(len(contours_b[0]))
print(len(contours_b[1]))
```

```
6
402
235
```

In [7]:

```
im_contours_template = np.zeros((template_im.shape[0],template_im.shape[1],3), np.uint8
conts_t = cv.drawContours(im_contours_template, contours_t, -1, (0,255,0), 3).astype('u
im_contours_belt = np.zeros((belt_im.shape[0],belt_im.shape[1],3), np.uint8)
conts_b = cv.drawContours(im_contours_belt, contours_b, -1, (0,255,0), 3).astype('uint8
fig,ax=plt. subplots(1,2,figsize=(10,10))
ax[0].imshow(conts_t, cmap='gray')
ax[1].imshow(conts_b, cmap='gray')
plt.show()
```





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

In [8]:
```
label = 1 # remember that the label of the background is 0
belt = ((labels_b >= label)*255).astype('uint8')
belt_cont, template_hierarchy = cv.findContours(belt, cv.RETR_EXTERNAL, cv.CHAIN_APPROX
for j,c in enumerate(belt_cont):
    print(cv.matchShapes(contours_t[0], c, cv.CONTOURS_MATCH_I1, 0.0))
```

```
0.00010071698397173812
0.00010071698397950968
0.00010071698397506879
```

Part 2

Frame tracking through image moments

In [9]:
```
ca = cv.contourArea(contours_b[1])
print("Area:",ca)
```

```
Area: 20080.0
```

In [10]:
```
M = cv.moments(contours_b[1])
cx,cy =int(M['m10']/M['m00']),int(M['m01']/M['m00'])
print("X coordinate:",cx)
print("Y coordinate:",cy)
```

```
X coordinate: 341
Y coordinate: 542
```

In [11]:
```
count = 1
object_prev_frame =  [cx, cy, ca, count]
print(object_prev_frame)
```

```
[341, 542, 20080.0, 1]
```

In [12]:
```
delta_x=15
```

Part 3

In [13]:
```
def get_indexed_image(im):
    threshold, th_image=cv.threshold(im,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
    kernel = np.ones((3,3),dtype=np.uint8)
    closing_image = cv.morphologyEx(th_image, cv.MORPH_CLOSE, kernel)
    retval, labels, stats, centroids = cv.connectedComponentsWithStats(closing_image)
    return retval, labels, stats, centroids
```

In [14]:
```
def is_new(a, b, delta, i):
    for k in range(len(a)):
        if abs(a[k][i]-b[i])<delta:
            return False
            break
    else:
        return True
```

In [15]:
```
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 is_new(a, b, delta, i) == False
```

In [16]:
```
def prev_index(a, b, delta, i):
    index = -1
    for k in range(len(a)):
        if abs(a[k][i]-b[i])<delta:
            return k
    else:
        return index
```

In [17]:
```
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
```

In [18]:
```
cap = cv.VideoCapture('conveyor_with_rotation.mp4')
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    cv.imshow('frame',frame)
    if cv.waitKey(1) == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

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

In [19]:

```python
# convert the frame into grayscale

gray_frame=[]
cap = cv.VideoCapture('conveyor_with_rotation.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
    gray= cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
    gray_frame.append(gray)
    cv.imshow("frame",gray)
    if cv.waitKey(1) == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
print(len(gray_frame))
```

```
Can't receive frame (stream end?). Exiting ...
280
```

In [20]:

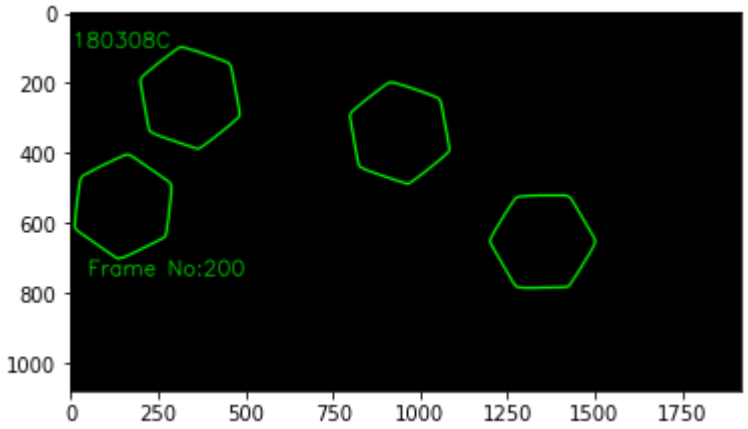
```python
# Draw contours in each frame
contours_plots=[]
contours_list = []
video = []
for i,image in enumerate (gray_frame):
    retval, labels, stats, centroids = get_indexed_image(image)
    belt = ((labels >= 1)*255).astype('uint8')
    contours,hierarchy = cv.findContours(belt, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE
    count = 0 # nut count in each frame
    frame = []
    for contour in contours:
        metric = cv.matchShapes(contours_t[0], contour, cv.CONTOURS_MATCH_I1, 0.0)
        if metric <= 0.5:
            count +=1
            M  = cv.moments(contour)
            ca = M['m00']
            cx, cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])
            frame.append(np.array([cx, cy, ca, count])) #center cordinates,area and ind
    video.append(frame)
    contours_list.append(contours)
    im_contours_belt = np.zeros((belt.shape[0],belt.shape[1],3), np.uint8)
    cont = cv.drawContours(im_contours_belt, contours, -1, (0,255,0), 5).astype('uint8'
    contours_plots.append(cont)

    cv.putText(cont,'180308C',(5,100),cv.FONT_HERSHEY_SIMPLEX, 2,(0,255,0),2,cv.LINE_AA
    cv.putText(cont,'Frame No:%i'%(i),(50,750),cv.FONT_HERSHEY_SIMPLEX, 2,(0,255,0),2,c
    cv.imshow('contours',cont)
    if cv.waitKey(1) == ord('q'):
        break

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

In [21]:

```python
print("Total no. of frames: ",len(video))
```

```
frame_no=200
print("Number of nuts in the given frame: ",int(video[frame_no][-1][-1]))
print(video[frame_no])
fig,ax=plt.subplots()
ax.imshow(contours_plots[frame_no])
plt.show()
```

```
Total no. of frames:  280
Number of nuts in the given frame:  4
[array([1.35200e+03, 6.53000e+02, 5.99555e+04, 1.00000e+00]), array([1.51000e+02, 5.5300
0e+02, 5.99225e+04, 2.00000e+00]), array([9.42000e+02, 3.43000e+02, 6.00565e+04, 3.00000
e+00]), array([3.42000e+02, 2.43000e+02, 6.00585e+04, 4.00000e+00])]
```



Object detection and tracking

In [22]:

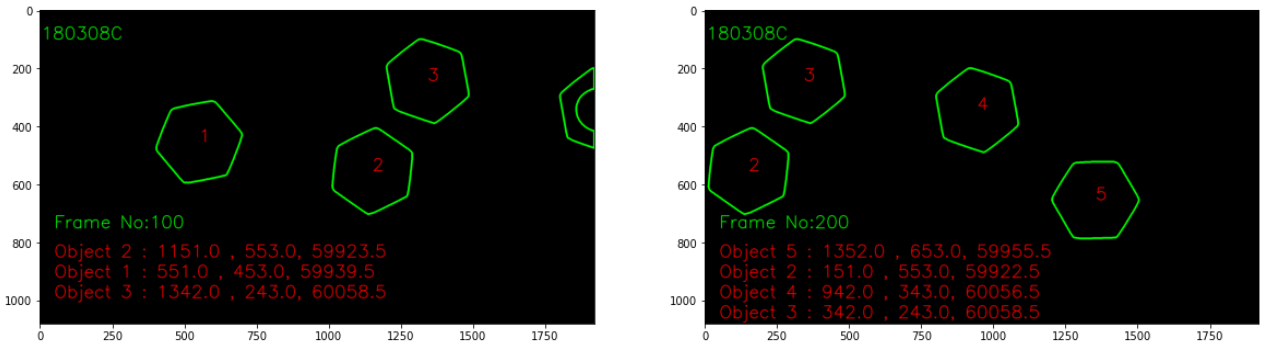
```
#tracking hexagonal nuts in each frame
total_nuts = int(video[0][-1][-1]) # initial number of nuts.
delta_x = np.array([15])
i = np.array([0])
previous_frame = video[0]
for j,frame in enumerate(video):
    current_frame=video[j]
    for nut in frame:
        if is_new(previous_frame, nut, delta_x, i):
            total_nuts +=1
            nut[-1]=total_nuts #give a new index if the nut is new
        else:
            index_in_previous_frame = prev_index(previous_frame, nut, delta_x, i)
            nut_index = previous_frame[int(index_in_previous_frame)][-1]
            nut[-1] = nut_index # if the nut is not a new one nut index doesn't change
    previous_frame=current_frame
print(total_nuts)
```

```
5
```

In [23]:

```
#put details of the nuts to video
for frame,cont in zip(video,contours_plots):
    y=0
    for nut in (frame):
        cv.putText(cont,(str(int(nut[-1]))),(int(nut[0]),int(nut[1])),cv.FONT_HERSHEY_S
        cv.putText(cont,'Object {no} : {cx} , {cy}, {area}'.format(no=int(nut[-1]),cx=n
        y=y+1
```

In [24]:
```
fig,ax=plt.subplots(1,2,figsize=(20,10))
ax[0].imshow(cv.cvtColor(contours_plots[100],cv.COLOR_BGR2RGB))#showing random two fram
ax[1].imshow(cv.cvtColor(contours_plots[200],cv.COLOR_BGR2RGB))
plt.show()
```



In [25]:
```
#create and save the video
time = 9 #time of the output video
fps = int(len(contours_plots)/time)# No of frames per second
height, width, channels= contours_plots[0].shape
frame_size = (width, height)
out = cv.VideoWriter('180308C_en2550_a05.mp4', cv.VideoWriter_fourcc(*'MP4V'), fps, fra
for frame in contours_plots:
    out.write(frame)

out.release()
```