Object Detection and Count using Contour

Author: Rajpal Virk | 30 May 2020

Introduction

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.
Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought.

Contours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
In this exercise, we'll use find contour feature from OpenCV to find the contours of some coins and then count the number of coins present in image. Coins image is taken using a mobile phone camera in this case.

In [1]:
```
# Import required libraries
import cv2
from IPython.display import display, HTML
import base64
```

In [2]:
```
# Define function to show images properly in matplotlib inside Jupyter Lab
def imshow(name, imageArray):
    _, png = cv2.imencode('.png', imageArray)
    encoded = base64.b64encode(png)
    return HTML(data='''<img alt="{0}" src=" {1}"/>'''.format(name, encoded.decode('ascii')))
```

In [3]:
```
# Read image file
img = cv2.imread("coins.jpeg")

# display the original image
imshow("Original Image", img)
```

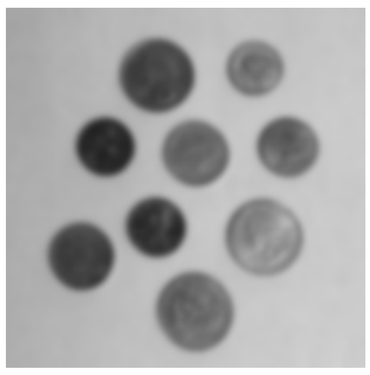
Out[3]:

In [4]:
```
# Convert image in grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imshow("Gray Scaled Image", gray)
```

Out[4]:

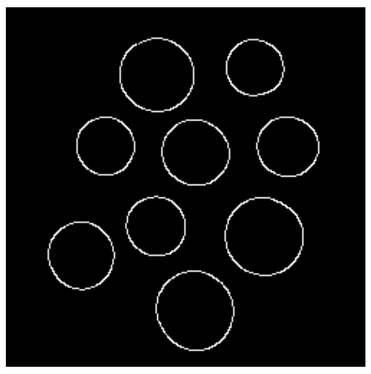
In [5]:
```
# Applying Guassian Blur (a low pass filter) to remove noise and reduce high-frequency components
blur_image = cv2.GaussianBlur(gray, (13, 13), 0) # kernel value 13 is used so that only outer contours are detected la
ter.
imshow("Blurred Image", blur_image)
```

Out[5]:



In [6]:
```
# Edge detection using Canny function. we'll set lower threshold @ 50 and upper threshold @ 150
edged_image = cv2.Canny(blur_image, 50, 150)
imshow("Edged Image", edged_image)
```

Out[6]:



In [7]:
```
# Finding contours using edged image and find contour feature
ret ,contours, hierarchy = cv2.findContours(edged_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE) # Retrieval method
used is external and contour drawing method is continuour chain (line)

# Superimpose contours on original image in red color
cv2.drawContours(img, contours, -1, (0,0,255), 5) # line thickness is 5
imshow("Original Image with contours", img)
```

Out[7]:

In [8]:
```
# Finding the count of coins detected
print ("Number of coins in the image are: %d." % (len(contours)))
```

```
Number of coins in the image are: 9.
```