MINI PROJECT

(COMPUTER VISION)

Submitted To: Dr. Muhammad Jawad Khan

Submitted By: Usman Zaheer, 327700

<u>Problem Statement</u>

For this case, you must recognize the shape and size of an object placed on (A3, Letter, A4) paper sheet. You must write a code that can first detect a blank paper to identify the dimensions based on the no. of pixels contained. Next, recognize the shape of the object placed on the blank paper. For this you must use real-time edge detection. The object can be placed at any angle/ posture within your boundary. You may have to use geometric transforms for detection. Once the shape is recognized map, the size of the object along all corners using the pixel ration with a standard paper size. To make difficulty easier, use only square, rectangular, triangular, and circular shaped objects for the assignment.

<u>Solution to the problem</u>

The below mentioned code contains all the necessary steps to detect an A4 paper. User can change the size of paper (A4, A3, Letter) as per the requirement. I have written all the necessary functions in the python file named “essentials.py”. This file is imported in the main file (jupyter notebook given) where the detection of paper, shape recognition and their dimension are shown. The input image on which the code is run is given below but it can also be run on the video.

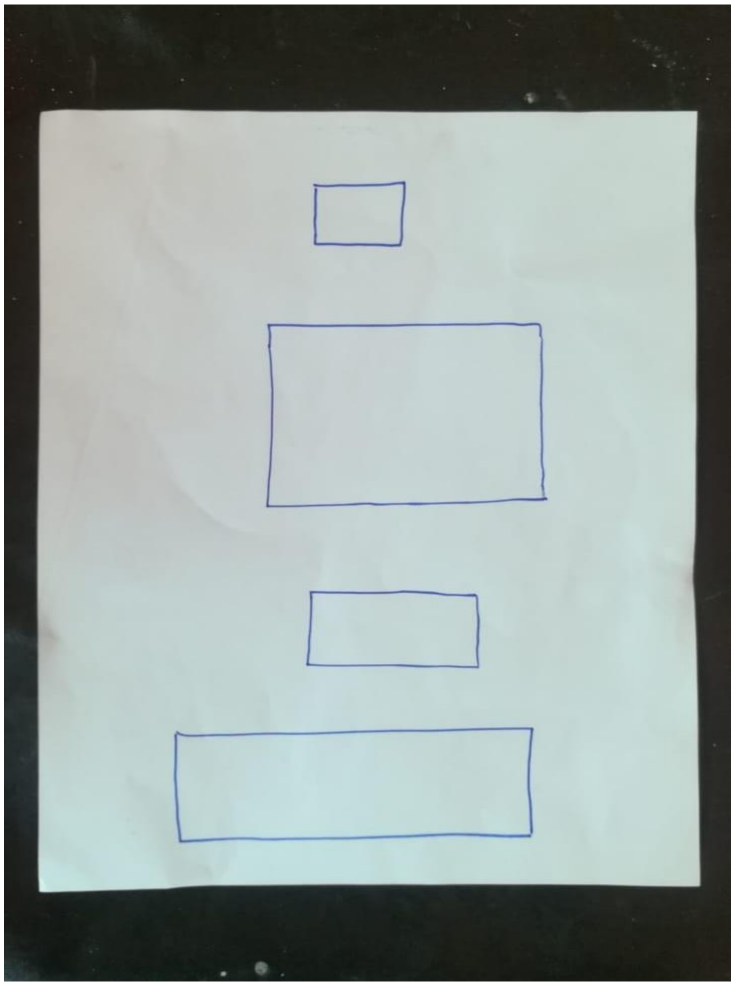


Figure 1: Input Image

PYTHON CODE

The code given below contains all the necessary functions which are called in the object measurement file.

“essentials.py”

Step 1: Importing Useful Libraries

```
import cv2
import numpy as np
```

Step 2: Defining the get contours functions which will contours in the image/video.

```
#Applying basic image processing techniques on the image.
def getContours(img,cThr=[100,100],showCanny=False,minArea=1000,filter=0,draw =False):
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray,(5,5),1)
    imgCanny = cv2.Canny(imgBlur,cThr[0],cThr[1])
    kernel = np.ones((5,5))
    imgDial = cv2.dilate(imgCanny,kernel,iterations=3)
    imgThre = cv2.erode(imgDial,kernel,iterations=2)
    if showCanny:cv2.imshow("Canny",imgThre)
```

Step 3 : Getting the contours and saving their parameters like area etc.

```
#Getting external contours as we need the external boundaries and saving them in the form of list.
 contours,hierarchy=
cv2.findContours(imgThre,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    finalCountours = []
    for i in contours:
        area = cv2.contourArea(i)
        if area > minArea:
            peri = cv2.arcLength(i,True)
            approx = cv2.approxPolyDP(i,0.02*peri,True)
            bbox = cv2.boundingRect(approx)
            if filter > 0:
                if len(approx) == filter:
                    finalCountours.append([len(approx),area,approx,bbox,i])
            else:
                finalCountours.append([len(approx),area,approx,bbox,i])
    finalCountours = sorted(finalCountours,key = lambda x:x[1] ,reverse= True)
    if draw:
        for con in finalCountours:
            cv2.drawContours(img,con[4],-1,(0,0,255),3)
    return img, finalCountours
```

Step 4: This function reorder the points so that the geometric shape can be identified.

```
def reorder(myPoints):
    #print(myPoints.shape)
    myPointsNew = np.zeros_like(myPoints)
    myPoints = myPoints.reshape((4,2))
    add = myPoints.sum(1)
    myPointsNew[0] = myPoints[np.argmin(add)]
    myPointsNew[3] = myPoints[np.argmax(add)]
    diff = np.diff(myPoints,axis=1)
    myPointsNew[1]= myPoints[np.argmin(diff)]
    myPointsNew[2] = myPoints[np.argmax(diff)]
    return myPointsNew
```

Step 5: This function is used to get the perspective transform of the paper, so that it can be warped correctly.

```
def warpImg (img,points,w,h,pad=10):
    # print(points)
    points =reorder(points)
    pts1 = np.float32(points)
    pts2 = np.float32([[0,0],[w,0],[0,h],[w,h]])
    matrix = cv2.getPerspectiveTransform(pts1,pts2)
    imgWarp = cv2.warpPerspective(img,matrix,(w,h))
    imgWarp = imgWarp[pad:imgWarp.shape[0]-pad,pad:imgWarp.shape[1]-pad]
    return imgWarp
```

Step 6: This function is used to get the dimensions of the detected shapes.

```
def findDis(pts1,pts2):
    return ((pts2[0]-pts1[0])**2 + (pts2[1]-pts1[1])**2)**0.5
```

This code is the main code where all the above-mentioned functions are being called.

Shape Detection and Messurement

Step 1: Importing Useful Libraries

```
#Importing the opencv and the above-mentioned functions module named "essentials"

import cv2
import essentials
```

Step 2: Setting the image path in case of image and camera parameters for video.

```
# For opening webcam, webcam = "True"
webcam = False
# Path of image
path = "myshapes.jpeg"
cap = cv2.VideoCapture(0)
cap.set(1,1000)
cap.set(3,1280)
cap.set(4,720)
scale = 2
#User can change the dimensions of paper as per the requirement.
#For A3 (wP=297 ,hP=420 ).
#For Letter (wP=215 ,hP=279)
#For A4 (wP=210 ,hP=297 ).

# Dimensions of A4 paper. Width = 210, Height = 297
wP = 210 *scale
hP= 297 *scale
```

Step 3: Setting the loop.

```
# Setting the loop for video and image.
while True:
    if webcam:success,img = cap.read()
    else: img = cv2.imread(path)
```

Step 4: Getting the contours on the basis of given image/video.

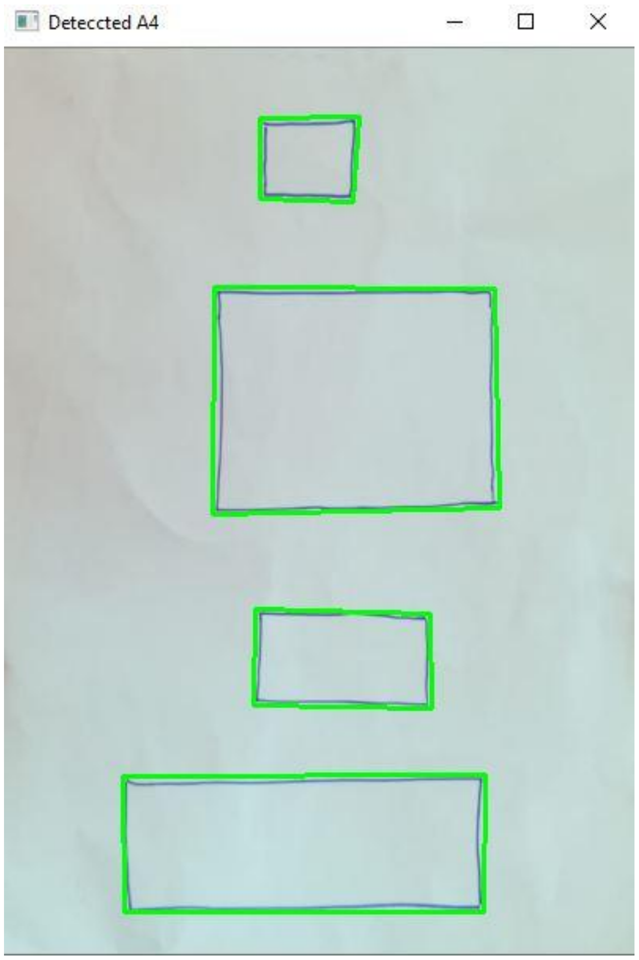
```
#Extracting the contours from the image or video. Dectecting the A4 paper as the biggest
contour.
    imgContours , conts = essentials.getContours(img,minArea=50000,filter=4)
    if len(conts) != 0:
        biggest = conts[0][2]
        #Dectecting the A4 paper as the biggest contour.
        imgWarp = essentials.warpImg(img, biggest, wP,hP)
```

<u>Step 4: Getting the smaller contours on the basis of biggest contour A4</u>

```
#Detecting the smaller contour shapes and calculating the dimensions of objects.
        imgContours2, conts2 = essentials.getContours(imgWarp,
                                            minArea=1000, filter=4,
                                            cThr=[50,50],draw = False)
```

<u>Step 5: Getting the smaller contours and creating bounding boxes</u>

```
        if len(conts) != 0:
            for obj in conts2:
                cv2.polylines(imgContours2,[obj[2]],True,(0,255,0),2)
```



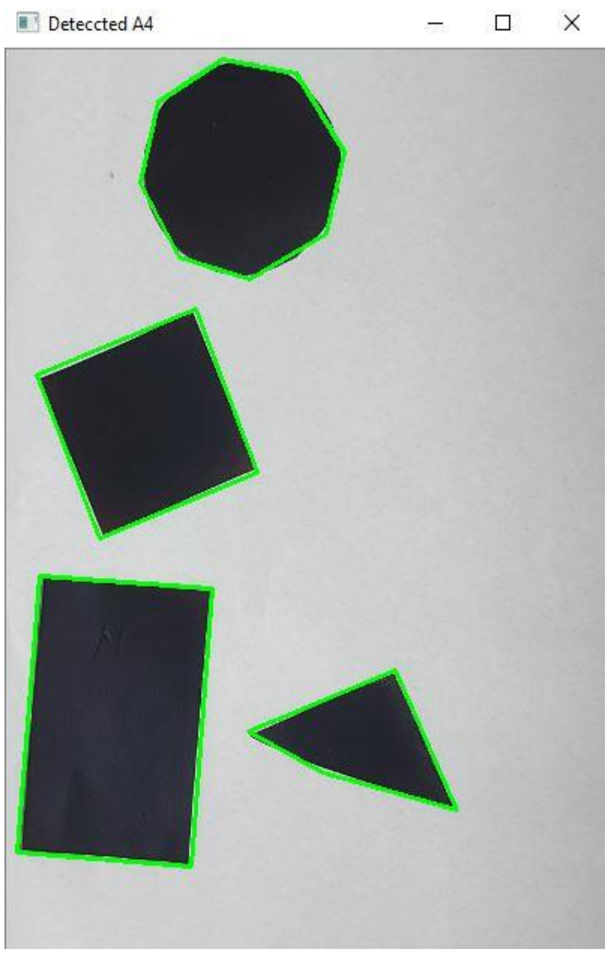


Figure 2: Bounding boxes around the detected shapes on detected A4 paper for different images.

Step 6: Creating arrows and displaying text of the dimensions for shapes

```
    nPoints = essentials.reorder(obj[2])
      nW = round((essentials.findDis(nPoints[0][0]//scale,nPoints[1][0]//scale)/10),1)
                  nH = round((essentials.findDis(nPoints[0][0]//scale,nPoints[2][0]//scale)/10),1)

                cv2.arrowedLine(imgContours2, (nPoints[0][0][0], nPoints[0][0][1]),
(nPoints[1][0][0], nPoints[1][0][1]),
                                (0, 255, 0), 3, 8, 0, 0.05)
                cv2.arrowedLine(imgContours2, (nPoints[0][0][0], nPoints[0][0][1]),
(nPoints[2][0][0], nPoints[2][0][1]),
                                (0, 255, 0), 3, 8, 0, 0.05)
                x, y, w, h = obj[3]
                cv2.putText(imgContours2, "{}cm".format(nW), (x + 30, y - 10),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1,
                            (0, 0, 255), 1)
                cv2.putText(imgContours2, "{}cm".format(nH), (x - 70, y + h // 2),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1,
                            (0, 0, 255), 1)
        #Displaying the detected A4 paper along with the detected shapes and dimensions.
        cv2.imshow("Deteccted A4", imgContours2)

    img = cv2.resize(img,(0,0),None,0.5,0.5)
    cv2.imshow("Original Input Image",img)
    cv2.waitKey(1)
```

Results on Image:

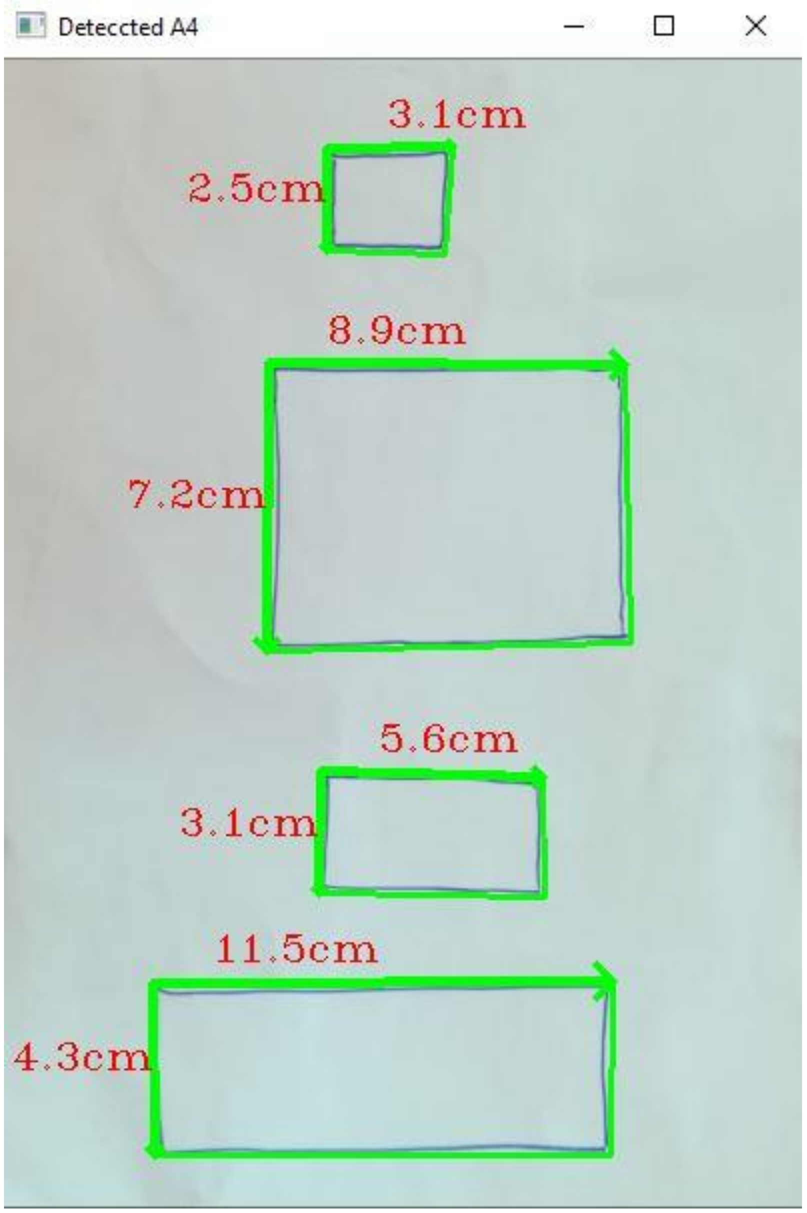
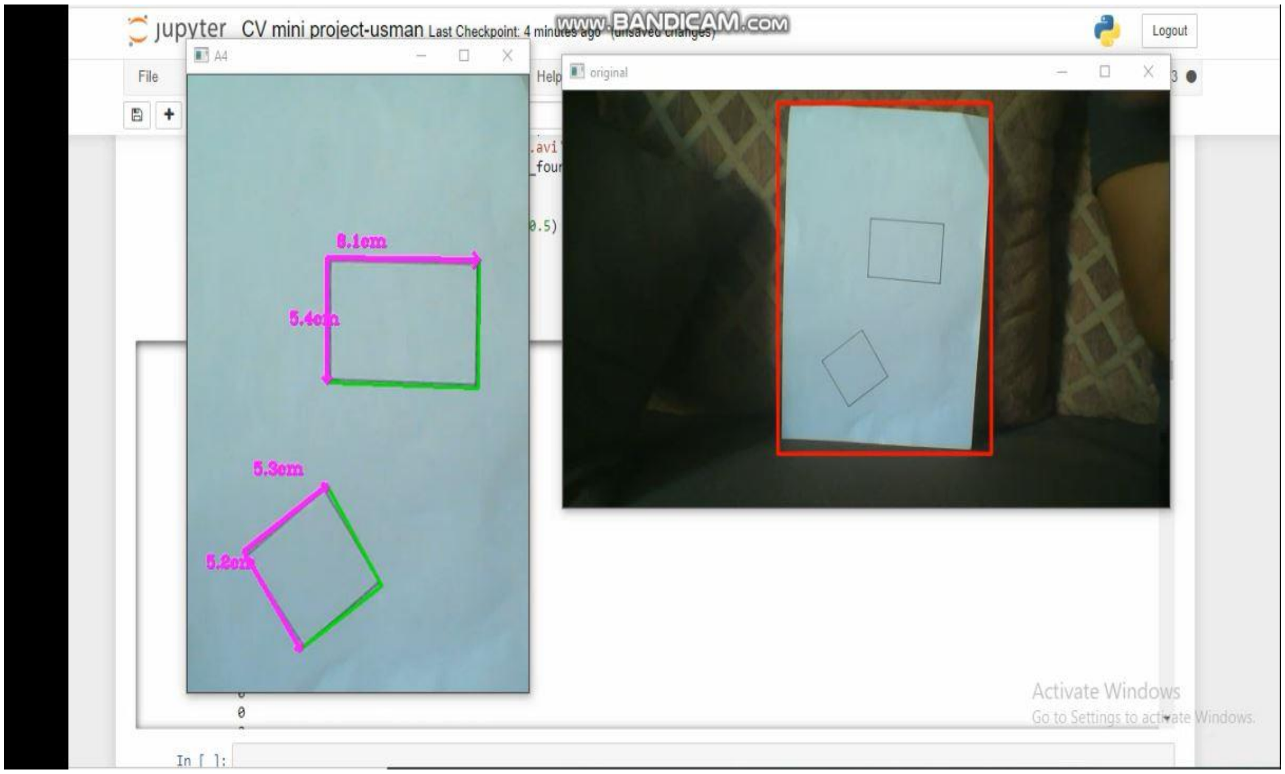


Figure 3: Detected shapes with dimensions on Detected A4 paper.

<u>Results on Realtime Video:</u>



jupyter CV mini project-usman Last Checkpoint: 4 minutes ago (unsaved changes)
www.BANDICAM.com
Logout
File
Help
A4
original
8.1cm
5.4cm
5.3cm
5.2cm
Activate Windows
Go to Settings to activate Windows.
In [ ]:

Complete Codes

Instructions on running the code:

Essentials.py should be imported in the given jupyter notebook (Mini Project_CV_Usman_Zaheer) or Just run the book as per the instructions in the book.
Both files are provided.

"Essentials.py"

```
import cv2
import numpy as np

def getContours(img,cThr=[100,100],showCanny=False,minArea=1000,filter=0,draw =False):
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray,(5,5),1)
    imgCanny = cv2.Canny(imgBlur,cThr[0],cThr[1])
    kernel = np.ones((5,5))
    imgDial = cv2.dilate(imgCanny,kernel,iterations=3)
    imgThre = cv2.erode(imgDial,kernel,iterations=2)
    if showCanny:cv2.imshow("Canny",imgThre)
    contours,hiearchy = cv2.findContours(imgThre,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    finalCountours = []
    for i in contours:
        area = cv2.contourArea(i)
        if area > minArea:
            peri = cv2.arcLength(i,True)
            approx = cv2.approxPolyDP(i,0.02*peri,True)
            bbox = cv2.boundingRect(approx)
            if filter > 0:
                if len(approx) == filter:
                    finalCountours.append([len(approx),area,approx,bbox,i])
            else:
                finalCountours.append([len(approx),area,approx,bbox,i])
    finalCountours = sorted(finalCountours,key = lambda x:x[1] ,reverse= True)
    if draw:
        for con in finalCountours:
            cv2.drawContours(img,con[4],-1,(0,0,255),3)
    return img, finalCountours

def reorder(myPoints):
    #print(myPoints.shape)
    myPointsNew = np.zeros_like(myPoints)
    myPoints = myPoints.reshape((4,2))
    add = myPoints.sum(1)
    myPointsNew[0] = myPoints[np.argmin(add)]
    myPointsNew[3] = myPoints[np.argmax(add)]
    diff = np.diff(myPoints,axis=1)
    myPointsNew[1]= myPoints[np.argmin(diff)]
    myPointsNew[2] = myPoints[np.argmax(diff)]
    return myPointsNew
```

```
def warpImg (img,points,w,h,pad=10):
    # print(points)
    points =reorder(points)
    pts1 = np.float32(points)
    pts2 = np.float32([[0,0],[w,0],[0,h],[w,h]])
    matrix = cv2.getPerspectiveTransform(pts1,pts2)
    imgWarp = cv2.warpPerspective(img,matrix,(w,h))
    imgWarp = imgWarp[pad:imgWarp.shape[0]-pad,pad:imgWarp.shape[1]-pad]
    return imgWarp

def findDis(pts1,pts2):
    return ((pts2[0]-pts1[0])**2 + (pts2[1]-pts1[1])**2)**0.5
```

“Mini Project_CV_Usman_Zaheer.ipynb”

```
import numpy as np
import cv2
import essentials

# For opening webcam, webcam = "True"
webcam = True
# Path of image
path = "myshapes.jpeg"
cap = cv2.VideoCapture(0)
cap.set(1,1000)
cap.set(3,1280)
cap.set(4,720)
scale = 2

#User can change the dimensions of paper as per the requirement.
#For A3 (wP=297 ,hP=420 ).
#For Letter (wP=215 ,hP=279)
#For A4 (wP=210 ,hP=297 ).

# Dimensions of A4 paper. Width = 210, Height = 297
wP = 210 *scale
hP= 297 *scale

while True:
    if webcam: success,img = cap.read()
    else: img = cv2.imread(path)
    if not success: continue; print ('fff')

# BGR to GRAY
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Binary Threshold
    _, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
    # Binary Fill
    h, w = thresh.shape
    mask = np.zeros([h+2, w+2], np.uint8)
    img1 = cv2.floodFill(thresh.copy(), mask, (0, 0), 0)[1]

    conts,_ = cv2.findContours(img1,cv2.CHAIN_APPROX_SIMPLE, cv2.RETR_TREE)
    if len(conts) != 0:
        max_area = 0
        for c in conts:
            M = cv2.moments(c)
            area = M['m00']
            if area >1e5:
                continue
            if area<10000:
                continue
            if area > max_area:
                max_area = area
                biggest = c
        print (max_area)
        x,y,w,h = cv2.boundingRect(biggest)
        cv2.rectangle(img,(x,y),(x+w,y+h),(0, 0, 255),5)
    #Extracting the contours from the image or video. Dectecting the A4 paper as the biggest contour.
    imgContours, conts = essentials.getContours(img,minArea=10,filter =4)

    #print (len(conts))
    if len(conts) != 0:

        biggest = conts[0][2]

        #print(biggest)
        imgWarp = essentials.warpImg(img, biggest, wP,hP)
 #Detecting the smaller contour shapes within the biggest contour.
        imgContours2, conts2 = essentials.getContours(imgWarp,minArea=2000,filter =4,cThr=[50,50],draw=False)
#Detecting the smaller contour shapes and calculating the dimensions of objects.
        if len(conts2) != 0:
            for obj in conts2:
                cv2.polylines(imgContours2,[obj[2]],True,(0,255,0),2)
                nPoints = essentials.reorder(obj[2])
                nW = round((essentials.findDis(nPoints[0][0]//scale,nPoints[1][0]//scale)/10),1)
                nH = round((essentials.findDis(nPoints[0][0]//scale,nPoints[2][0]//scale)/10),1)
                cv2.arrowedLine(imgContours2, (nPoints[0][0][0], nPoints[0][0][1]), (nPoints[1][0][0], nPoints[1][0][1]),
                                (0, 255, 0), 3, 8, 0, 0.05)
                cv2.arrowedLine(imgContours2, (nPoints[0][0][0], nPoints[0][0][1]), (nPoints[2][0][0], nPoints[2][0][1]),
                                (0, 255, 0), 3, 8, 0, 0.05)
                x, y, w, h = obj[3]
                cv2.putText(imgContours2, "{}cm".format(nW), (x + 10, y - 10),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.75,
                            (0, 0, 255), 1)
                cv2.putText(imgContours2, "{}cm".format(nH), (x - 40, y + h // 2),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.75,
                            (0, 0, 255), 1)
                if abs(nW-nH) <=0.5:
                    cv2.putText(imgContours2, 'A Square', (x + 10, y - 60), cv2.FONT_HERSHEY_COMPLEX_SMALL,
1,
                           (255, 0, 0), 1)
                if abs(nW-nH) >0.5:
                    cv2.putText(imgContours2, 'A Rectangle', (x + 10, y - 60),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1.5,
                            (255, 0, 0), 1)
                area = round (nW*nH)
                cv2.putText(imgContours2, "area = {} Sq.cm".format(area), (x + 10, y - 35),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.75,
                            (0,255, 255), 2)
        #cv2.circle(imgContours2,(100,100),20,(255, 0, 0),2)
        cv2.imshow('A4',imgContours2)
        cv2.imwrite('result1.jpg', imgContours2)
        frame_width = int(cap.get(3))
        frame_height = int(cap.get(4))
        size = (frame_width, frame_height)
        result = cv2.VideoWriter('result.avi',
                     cv2.VideoWriter_fourcc(*'MJPG'),
                     10, size)
    result.write(imgContours2)
#Displaying the detected A4 paper along with the detected shapes and dimensions .
img = cv2.resize(img,(0,0),None,0.5,0.5)

cv2.imshow('original',img)

if cv2.waitKey(1)==27:
    break
```