**Document Scanner**

Basic Overview:

* Preprocess the image using standard functions like color – gray conversion, blurring, edge detection, erosion and dilation
* Find the biggest Contour using findContour() function. An approximate for the biggest contour are passed on to the warping function’
* The warping function uses the approximate coordinates to apply perspective transform to the required area. But since it expects the points in a specific order, we use a re-ordering function to use that the points are indeed in the expected order

**Preprocessing function**

kernel = np.ones((5,5))

def preProcessing(img):

    imgGray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

    imgBlur = cv2.GaussianBlur(imgGray,(5,5),sigmaX=1)

    imgCanny = cv2.Canny(imgBlur,200,200)

    imgDial = cv2.dilate(imgCanny,kernel,iterations=2)

    imgErode = cv2.erode(imgDial,kernel,iterations=1)

    return imgErode

**Contour function**

Returns the coordinates of corners of biggest contour found within the image.

def getContours(img):

    maxArea = 0

    biggest = np.array([])

    contours, hierarchy = cv2.findContours(img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

    for cnt in contours:

        area = cv2.contourArea(cnt)

        if area > 1000:

            #cv2.drawContours(imgContour,cnt,-1,(255,0,0),20)

            peri = cv2.arcLength(cnt,closed=True)

            approx = cv2.approxPolyDP(cnt,0.02\*peri,closed=True)

            if area > maxArea and len(approx) == 4:

                biggest = approx

                maxArea = area

    cv2.drawContours(imgContour,biggest,-1,(255,0,0),20)

    return biggest

**Warping function**

* Now that we have our biggest contour , we need to warp it and get its' birds eye view
* The main thing we need to contend with is the order in which the getPerspectiveTransform() expects the points. If the points obtained via biggest don't follow the format , it might lead to unexpected / unwanted results
* We reorder the given coordinates and then use it to in getPerspectiveTransform() and warpPerspective()

def getWarp(img,biggest):

    biggest = reorder(biggest)

    pts1 =  np.float32(biggest)

    pts2 = np.float32([[0,0],[widthImage,0], [0,heightImage],[widthImage,heightImage]])

    matrix = cv2.getPerspectiveTransform(pts1,pts2)

    imgOutput = cv2.warpPerspective(img,matrix,(widthImage,heightImage))

    imgCropped  = imgOutput[20:imgOutput.shape[0]-20,20:imgOutput.shape[1]-20]

    imgCropped = cv2.resize(imgCropped,(widthImage,heightImage))

    return imgCropped

**Reordering function**

Ensures that the coordinates of the biggest contour obtained via getContours() function are in the order expected by the perspective function.

* The following function will take the coordinates of the biggest contour and rearrange them in the expected order
* The way we do this is by adding the coordinates of the corners.(rowwise addition)
* The one with the least sum would be the origin while the one with the highest would be the extreme corner(diagonal opposite of origin)
* Next ,take rowwise difference . The onw wiith the least difference will be the second point, and the one with the max difference will be the third point

def reorder(myPoints):

    myPoints = myPoints.reshape((4,2))

    myPointsNew = np.zeros((4,1,2),np.int32)

    add = myPoints.sum(1)

    #print('add',add)

    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)]

    #print('New Points ',myPointsNew)

    return myPointsNew

**Driver code**

**For test image**

img = cv2.imread('D:\\Courses\\OpenCV\\Resources\\document.jpg')

img = cv2.resize(img,(widthImage,heightImage))

imgContour = img.copy()

imgThres = preProcessing(img)

biggest = getContours(imgThres)

if biggest.size != 0:

    imgWarped = getWarp(img,biggest)

    imgArray = [[img,imgContour],

            [imgThres,imgWarped]]

else:

    imgArray = [[img,imgContour],

            [imgThres,img]]

stackedImages = stackImages(0.6,imgArray)

cv2.imshow('Stacked Images',stackedImages)

cv2.waitKey(0)

**For webcam feed**

cap = cv2.VideoCapture(0)

cap.set(3,widthImage)

cap.set(4,heightImage)

while True:

    success, img = cap.read()

    imgContour = img.copy()

    img = cv2.resize(img,(widthImage,heightImage))

    imgThres = preProcessing(img)

    biggest = getContours(imgThres)

    if biggest.size != 0:

        imgWarped = getWarp(img,biggest)

        imgArray = [[img,imgContour],

                    [imgThres],imgWarped]

    else:

        imgWarped = getWarp(img,biggest)

        imgArray = [[img,imgContour],

                    [imgThres],img]

    stackedImages = stackImages(0.6,imgArray)

    cv2.imshow('Video',stackedImages)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cv2.destroyAllWindows()

cap.release()