Mujtaba Shahid Faizi

BSCS-5A

#131818

Lab 11 of DIP

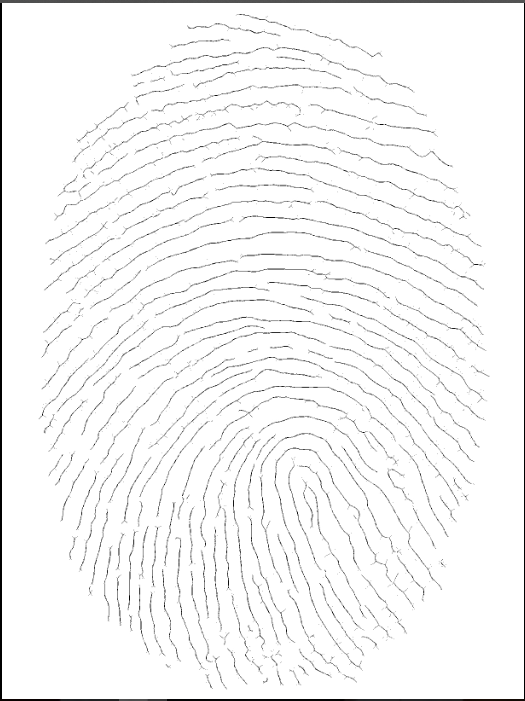
**Task A**

**Code**

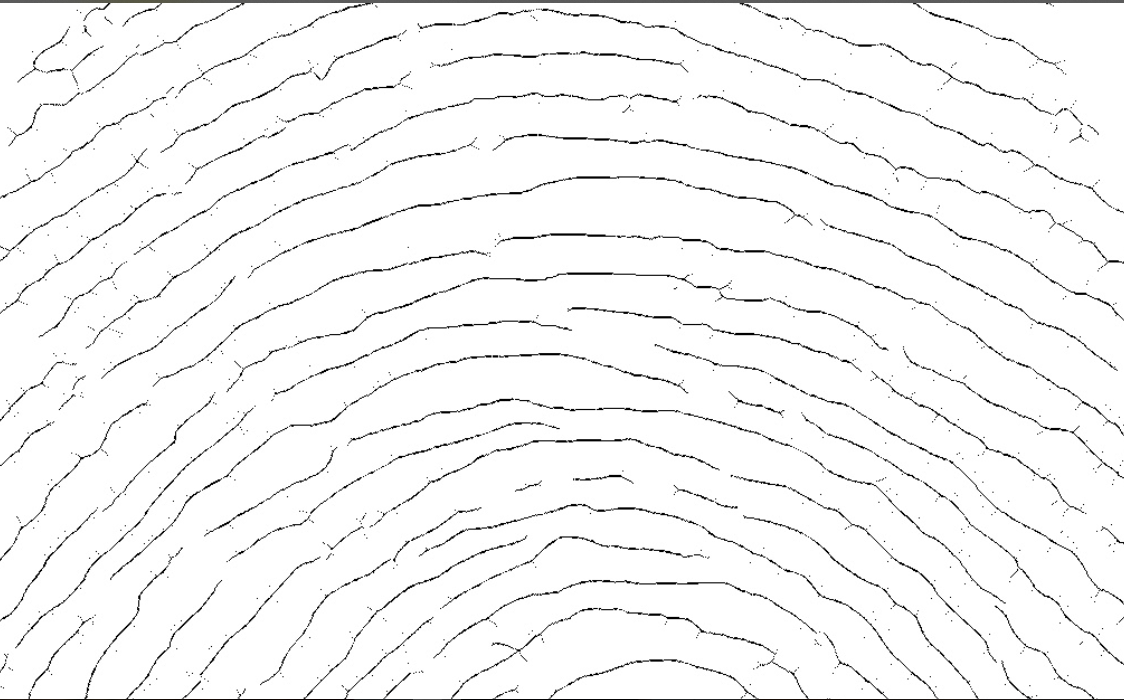
**import** cv2  
**import** numpy **as** np  
  
img=cv2.imread(**"ThumbImpression.png"**,0)  
img=255-img *#negative*size = np.size(img)  
skel = np.zeros(img.shape, np.uint8)  
  
ret, img = cv2.threshold(img, 127, 255, 0)  
element = cv2.getStructuringElement(cv2.MORPH\_CROSS, (3, 3))  
done = **False  
  
while** (**not** done): *#skeletanization* eroded = cv2.erode(img, element)  
 temp = cv2.dilate(eroded, element)  
 temp = cv2.subtract(img, temp)  
 skel = cv2.bitwise\_or(skel, temp)  
 img = eroded.copy()  
  
 zeros = size - cv2.countNonZero(img)  
 **if** zeros == size:  
 done = **True**kernel = np.ones((1,1), np.uint8)  
  
skel = cv2.dilate(skel, kernel, iterations=4) *#dilation*skel=255-skel *#negative*cv2.imwrite(**"skel.jpg"**, skel)  
  
  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Screenshots**

So, I basically performed skeletanization of the image and dilated it to achieve the best result, to give the following figure.



**Zoomed-in image**



**Task B**

**Code**

**import** cv2  
**import** numpy **as** np  
**import** math  
  
**def** b2wtrans(im): *# finding black to white transitions*  
 arr=np.asarray(im)  
 prev = arr[0, 0]  
 n = 0  
 width = arr.shape[0]  
 height = arr.shape[1]  
 **for** x **in** range(1, width):  
 **for** y **in** range(1, height):  
 curr = arr[x,y]  
 **if** curr == 255 and prev == 0:  
 n= n + 1  
 prev = curr  
 **return** n  
  
img=cv2.imread(**"Signature.png"**,0)  
ret3, segmented = cv2.threshold(img, 0, 255, cv2.THRESH\_BINARY)  
cv2.imshow(**"original"**,segmented)  
segmented = cv2.medianBlur(segmented,3)  
*# find where the signature is and make a cropped region*points = np.argwhere(segmented == 0) *# find where the black pixels are*points = np.fliplr(points) *# store them in x,y coordinates instead of row,col indices*x = min(points[:, 0]) *# finding the first & last of black pixels of both axes*y = min(points[:, 1])  
w = max(points[:, 0])  
h = max(points[:, 1])  
crop = segmented[y:h, x:w] *# create a cropped region of the gray image  
# get the thresholded crop*retval, crop = cv2.threshold(crop, thresh=200, maxval=255, type=cv2.THRESH\_BINARY)  
arr=np.asarray(crop)  
width=arr.shape[0]  
height=arr.shape[1]  
cv2.imshow(**"Cropped"**,crop)  
cx = 0  
cy =0  
n = 0  
**for** a **in** range(0, width): *#finding the centroid* **for** b **in** range(0, height):  
 **if** arr[a,b]==0:  
 cx = cx + a  
 cy = cy + b  
 n = n + 1  
cx = math.floor(cx / n)  
cy = math.floor(cy / n)  
print(**"left : "**,x,**", top : "**,y,**", right : "**,w,**", bottom : "**,h,**", x-centroid : "**,cx,**", y-centroid : "**,cy)  
topleft = segmented[y:x+cy, x:y+cx] *# create a cropped region of the image*topright = segmented[y:cy+x, cx+y:w] bottomleft=segmented[cy+x:h,x:y+cx]  
bottomright=segmented[cy+x:h, cx+y:w]  
cv2.imshow(**"topleft"**,topleft)  
cv2.imshow(**"topright"**,topright)  
cv2.imshow(**"bottomleft"**,bottomleft)  
cv2.imshow(**"bottomright"**,bottomright)  
print(**"topleft transitions : "**,b2wtrans(topleft))  
print(**"topright transitions : "**,b2wtrans(topright))  
print(**"bottomleft transitions : "**,b2wtrans(bottomleft))  
print(**"bottomright transitions : "**,b2wtrans(bottomright))  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Screenshots**

