**Name: Ahmad Amjad Mughal**

**Reg No: 121672**

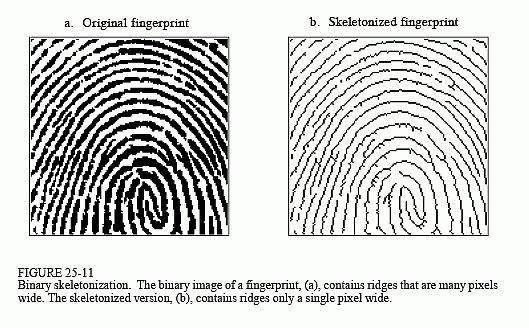
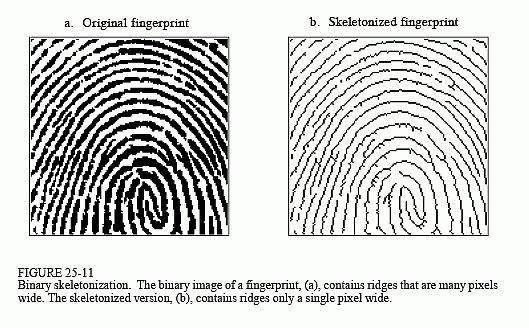
**Class: BSCS6C**

Lab-12: Applications of DIP

**Task (A):**

Considering that you are a digital Image processing expert, can you suggest the steps (and, of course, show the results experimentally) for moving from image “a” to image “b” as given below. You already have studied a compendium of approaches that can help you out here. Trying to be unique and still getting the results – may help you get good marks.

Note: You can apply your algorithm on the image (ThumbImpression.png) given to you separately in this lab, but the results should be depicting the similar concept as provided in image “b” below.

“a” “b”

**Task (A) Code:**

**import** cv2  
**from** PIL **import** Image

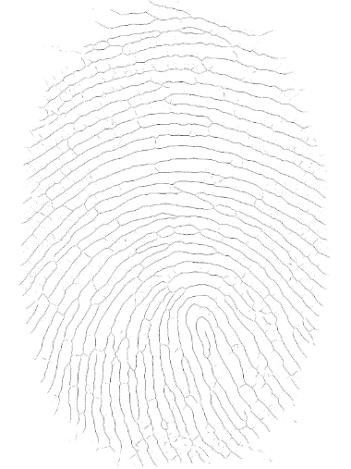
**import** numpy **as** np  
**import** PIL.ImageOps  
  
*#reading an image*original\_image = cv2.imread(**'ThumbImpression.png'**, 0)  
  
kernel = np.zeros((6, 6), np.uint8)  
size = np.size(original\_image)  
skel = np.zeros(img.shape, np.uint8)

*#Defining a structuring element*  
structuring\_element = cv2.getStructuringElement(cv2.MORPH\_CROSS, (3, 3))  
ok = False

*#Morphological Operations*  
**while** (**not** ok):  
 eroded = cv2.erode(original\_image,structuring\_element)  
 temp = cv2.dilate(eroded, structuring\_element)  
 temp = cv2.subtract(original\_image, temp)  
 skel = cv2.bitwise\_or(skel, temp)  
 original\_image = eroded.copy()  
  
 zeros = size - cv2.countNonZero(original\_image)  
 **if** zeros == size:  
 done = True  
  
skel = Image.fromarray(skel)  
output\_image = PIL.ImageOps.invert(skel)  
*#output\_image.save('hello.png')*output\_image.show()

**Task (A) Output**

**Original Resultant**

****

**Task (B): Application**

Take image of a handwritten signature and perform the following tasks: (Input signature is provided as a single image separately – Signature.png)

1. Develop a bounding box around the signature content.
2. Find out the centroid of the signature.
3. Segment signature from centroid vertically and horizontally (the signature will be divided into four pieces)
4. Calculate black to white transitions for each of the four segments.

**Task (B) Code:**

import cv2

#reading image

input\_image = cv2.imread("Signature.png", 0)

# Binarize the image

ret,binary = cv2.threshold(input\_image,10,255,cv2.THRESH\_BINARY)

#to find the object in image

**Task B (Part 1)**

def get\_object(image):

height, width = image.shape[0], image.shape[1]

left, top = width, height

right, bottom = 0, 0

for x in xrange(height):

for y in xrange(width):

if (image[x,y] == 0):

right = x if x > right else right

left = x if x < left else left

bottom = y if y > bottom else bottom

top = y if y < top else top

return (left, right, top, bottom)

**Task B (Part 2)**

def get\_centroid(image):

height, width = image.shape[0], image.shape[1]

cx, cy, n = 0, 0, 0

for x in xrange(height):

for y in xrange(width):

if (image[x,y] == 0):

cx += x

cy += y

n += 1

cx /= n

cy /= n

return (cx, cy)

**Task B (Part 3)**

def get\_segments(image, cent):

height, width = image.shape[0], image.shape[1]

cx, cy = cent

image1 = image[0:cx, 0:cy]

image2 = image[0:cx, cy:width]

image3 = image[cx:height, 0:cy]

image4 = image[cx:height, cy:width]

return [image1, image2, image3, image4]

**Task B (Part 4)**

# calculating black white transition

def calc\_black\_white\_transition(image):

height, width = image.shape[0], image.shape[1]

previous = image[0,0]

var\_1 = 0

for x in range(1, height):

for y in range(1, width):

current = image[x,y]

# check if there is black to white transition

if (current == 255 and previous == 0):

var\_1 = var\_1 + 1

previous = current

return var\_1

boundingBox = get\_object(binary);

crop\_image = binary[boundingBox[0]:boundingBox[1], boundingBox[2]:boundingBox[3]]

centroid = get\_centroid(crop\_image)

# diving image into four segments at centroid point

segments = get\_segments(crop\_image, centroid

#calculating transitions

transitions = [calc\_black\_white\_transition(seg) for seg in segments]

print ("Top\_Left||Top-Right||Bottom-Left||Bottom-Right")

print ("Transistions are: ", transitions)

cv2.imshow("TopLeft", segments[0])

cv2.imwrite("TopLeft.png", segments[0])

cv2.imshow("TopRight", segments[1])

cv2.imwrite("TopRight.png", segments[1])

cv2.imshow("BottomLeft", segments[2])

cv2.imwrite("BottomLeft.png", segments[2])

cv2.imshow("BottomRight", segments[3])

cv2.imwrite("BottomRight.png", segments[3])

cv2.waitKey(0)

**Output**

**Bottom Left**



Bottom Right



Top Left



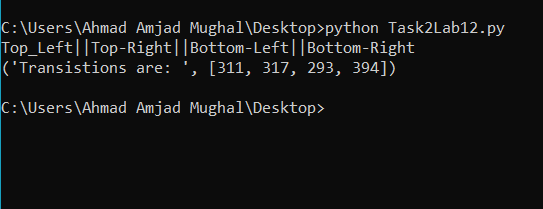
Top Right



Signature



**Output Screenshot**

****