Lab 5

Group Members:

Hassan Shahzad (211798)

Muhammad Sulayman Khan (211291)

Task 1,2,3,4:

Code:

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

#transition fuction

def transition(x1,x2,y1,y2,final):

  prev = final[y1,x1]

  n = 0

  for x in range(x1, x2-1):

    for y in range(y1, y2-1):

      curr = final[y,x]

      if curr == 255 and prev == 0: #count every white pixel after balck pixel

        n = n + 1

      prev = curr

  return n

#ratio function

def findRatio(left,  right,  top,  bottom):

  return ((right-left)/(bottom-top)) #ratio=width/height

#

def findBlacks(image,left,right,top,bottom):

  number = 0

  for x in range(top, bottom-1):

    for y in range(left, right-1):

      if image[x,y]==0:

        number = number + 1

  return number

def findNormSize(left, right, top, bottom, b):

  normalize = 0

  width = right - left

  height = bottom - top

  if b != 0:

    normalize = (width \* height)/b

  return normalize

def findCentAngle(cx, cy, bottom, left):

  a = 0

  dx = cx - left

  dy = bottom - cy

  if dx != 0:

    a = np.arctan(dy/dx)

  return a

def findNormAngles(image, left, right, top, bottom):

  number = 0

  a = 0

  value = 0

  for x in range(top, bottom-1):

    for y in range(left, right-1):

      if image[x, y] == 0:

        number = number + 1

        dx = x - left

        dy = bottom - y

        if dx != 0:

          a = np.arctan(dy/dx)

          value = value + a

  if number != 0:

    value = value / number

  return value

#Centroid function

def findCentroid(image,  left,  right,  top,  bottom):

  cx = left

  cy = top

  n = 0

  for x in range(left,right-1):

    for y in range(top, bottom-1):

      if cropImg[y,x] == 0:

        cx = cx + x

        cy = cy + y

        n = n + 1

  if n !=0:  #if section isnt empty

    cx = cx/n

    cy = cy/n

   #write centroid to text file

  f= open("centroid.txt","a+")

  f.write("%f , %f \n" % (cy , cx))

  f.close()

  cImg = cropImg

  for x in range(left,right-1):

    for y in range(top, bottom-1):

      cImg[int(cy),x] = 50  #print boundaries

      cImg[y,int(cx)] = 50

  cv2\_imshow(cImg)

  return int(cx), int(cy)

#split function

def split(image,left,right,top,bottom, depth = 0):

  cx,  cy  =  findCentroid(image,  left,  right,  top,  bottom)

  if  cx != 0 and cy != 0:

    if depth  <  3:

      split(image,  left,  cx,  top,  cy,  depth  +  1)  #called recursively for four sectors around each centroid

      split(image,  cx,  right,  top,  cy,  depth  +  1)

      split(image,  left,  cx,  cy,  bottom,  depth  +  1)

      split(image,  cx,  right,  cy,  bottom,  depth  +  1)

    else:  #if sector cant be divided further

      t  =  transition(left,  right,  top,  bottom,image)

      r  =  findRatio(left,  right,  top,  bottom)

      b = findBlacks(image,left,right,top,bottom)

      s = findNormSize(left, right, top, bottom, b)

      a = findCentAngle(cx, cy, bottom, left)

      A = findNormAngles(image, left, right, top, bottom)

      #write transitions to text file

      f1= open("transitions.txt","a+")

      f1.write("%f\n" % t)

      f1.close()

      #write aspect ratio to text file

      f2= open("aspectRatio.txt","a+")

      f2.write("%f\n" % r)

      f2.close()

      #write number of blacks to text file

      f2= open("blacks.txt","a+")

      f2.write("%f\n" % b)

      f2.close()

      #write normalized size to text file

      f2= open("normalized.txt","a+")

      f2.write("%f\n" % s)

      f2.close()

      #write inclination of centroid to text file

      f2= open("inclination.txt","a+")

      f2.write("%f\n" % a)

      f2.close()

      #write normalized inclination of blacks to text file

      f2= open("normalizedInclination.txt","a+")

      f2.write("%f\n" % A)

      f2.close()

#main

img = cv2.imread('signature.jpg',0) #read image

(thresh, bWImg) = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY) #conert to binary

print("Black and White Image:")

cv2\_imshow(bWImg)

hei,wid = bWImg.shape #find height and width of image

left = wid

right = 0

top = hei

bottom = 0

#crop image

for x in range(0,wid-1):

  for y in range(0, hei-1):

    color = bWImg[y,x]

    if color==0:

      if x > right:

        right = x

      if x < left:

        left = x

      if y > bottom:

        bottom = y

      if y < top:

        top = y

cropImg = bWImg[top:bottom, left:right]

print("Cropped Image:")

cv2\_imshow(cropImg)

cImg = cropImg

for x in range(0,right-left-1): #assign bounding box to cropped image

  for y in range(0, bottom-top-1):

    cImg[0,x] = 50

    cImg[y,0] = 50

    cImg[bottom-top-1,x] = 50

    cImg[y,right-left-1] = 50

#split image

split(cropImg,0,right-left,0,bottom-top)

Output:

Files:



Task-1 Output:



Task-2 Output:



Task-3 Output:



Task-4 Output:

