**Department of Computing**

**Digital Image Processing**

**Class: BSCS-9ABC**

**Lab 5**

**Submitted By: Fatima Seemab**

**CMS ID: 291310**

**BSCS 9B**

Lab 5

Connected Components Labelling

**Introduction**

This lab is to introduce connected components labeling.

**Objectives**

This lab will provide the concepts of connected components and its significance in image processing.

**Tools/Software Requirement**

Python 3.X

**Description**

Connected components labeling scans an image and groups its [pixels](http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixel.htm) into components based on [pixel connectivity](http://homepages.inf.ed.ac.uk/rbf/HIPR2/connect.htm), i.e. all pixels in a connected component share similar [pixel intensity values](http://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm) and are in some way connected with each other. Once all groups have been determined, each pixel is labeled with a Gray-level or a color (color labeling) according to the component it was assigned to.

Extracting and labeling of various disjoint and connected components in an image is central to many automated image analysis applications.

**Lab Tasks**

Implement the connected component labeling algorithm discussed in class on the image given in the lab.

**Code for the given array**

import numpy as np

Dict = {}

img = np.array(

[[1, 1, 0, 1, 1, 1, 0, 1], [1, 1, 0, 1, 0, 1, 0, 1], [1, 1, 1, 1, 0, 0, 0, 1], [0, 0, 0, 0, 0, 0, 0, 1],

[1, 1, 1, 1, 0, 1, 0, 1], [0, 0, 0, 1, 0, 1, 0, 1], [1, 1, 1, 1, 0, 0, 0, 1], [1, 1, 1, 1, 0, 1, 1, 1]],

np.int32)

#dimensions of an array

x,y=img.shape

#initiating an array of equal size

label=np.zeros((x,y))

# counter is set to 1

count=1

for i in range(x):

for j in range(y):

#condition for matching criteria

if(img[i,j]==1):

#condition for the borders

if(i==0 and j==0):

label[i][j]=img[i][j]

elif(i==0 and j!=0):

if(img[i][j-1]==0):

count=count+1

label[i][j]=count

else:

label[i][j]=label[i][j-1]

elif(j==0 and i!=0):

if(img[i-1][j]==0):

count=count+1

label[i][j]=count

else:

label[i][j]=label[i-1][j]

else:

if(img[i][j-1]==1 or img[i-1][j]==1):

if (min(label[i-1][j],label[i][j-1])==0):

label[i][j]=max(label[i-1][j],label[i][j-1])

else:

if(label[i-1][j]!=label[i][j-1]):

#in case of more than one value in neighbors minimum value is preferred and #added to dictionary

Dict[max(label[i-1][j],label[i][j-1])] = min(label[i-1][j],label[i][j-1])

label[i][j]=min(label[i-1][j],label[i][j-1])

else:

label[i][j]=label[i-1][j]

else:

count=count+1

label[i][j]=count

index=list(Dict)

for i in range(x):

for j in range(y):

#for replacing the values those who have some already preferred one exists

abc=label[i][j]

k=0

for value in Dict.values():

if abc == index[k]:

label[i][j] = value

k=k+1

print(label)

**Output**

A picture containing text

Description automatically generatedText

Description automatically generated with medium confidence

Qr code

Description automatically generated

**Code for the RGB Image:**

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

import cv2

#read image

img\_grey = cv2.imread('Lab5-image.png', cv2.IMREAD\_GRAYSCALE)

# define a threshold, 128 is the middle of black and white in grey scale

thresh = 128

# threshold the image

img\_binary = cv2.threshold(img\_grey, thresh, 255, cv2.THRESH\_BINARY)[1]

#save image

cv2.imwrite('my.png',img\_binary)

im=Image.open('my.png')

Dict = {}

#converting image to array

img = np.array(im)

#To get the dimensions of an array

x,y=img.shape

#Initiating an array of equal size with zeros

label=np.zeros((x,y))

count=1

for i in range(x):

for j in range(y):

if(img[i][j]==0):

#condition for filtering the dark values

if(i==0 and j==0):

label[i][j]=img[i][j]

elif(i==0 and j!=0):

if(img[i][j-1]==0):

count=count+1

label[i][j]=count

else:

label[i][j]=label[i][j-1]

elif(j==0 and i!=0):

if(img[i-1][j]==0):

count=count+1

label[i][j]=count

else:

label[i][j]=label[i-1][j]

else:

if(img[i][j-1]==0 or img[i-1][j]==0):

if (min(label[i-1][j],label[i][j-1])==0):

label[i][j]=max(label[i-1][j],label[i][j-1])

else:

if(label[i-1][j]!=label[i][j-1]):

if (len(Dict)>0):

#searching in dictionary if already have the value

for value,key in list(Dict.items()):

if(value!=min(label[i-1][j],label[i][j-1])):

#Assigning values to dictionary

Dict[max(label[i-1][j],label[i][j-1])] = min(label[i-1][j],label[i][j-1])

else:

print('dictionary is initiated')

Dict[max(label[i-1][j],label[i][j-1])] = min(label[i-1][j],label[i][j-1])

label[i][j]=min(label[i-1][j],label[i][j-1])

else:

label[i][j]=label[i-1][j]

else:

count=count+1

label[i][j]=count

#Second Pass

for i in range(x):

for j in range(y):

#iterating over the whole picture and finding those pixels to whom there exists a #preferred value

abc=label[i][j]

for value in Dict:

if abc == value:

#the value is replaced with the preferred one

label[i][j] = Dict[value]

#for coloring the connected component and their display

plt.imshow(label,cmap="nipy\_spectral")

plt.show()

**Output:**

Graphical user interface, application

Description automatically generated

**Note :- You only have to do it using 4 connectivity.**

**Deliverable**

Upload a lab reporting code and screenshots of output on lms.