CSCI-B456 – Image Processing
Spring 2018

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Project 1

Due by 1/26/2018, Friday Midnight through Canvas

Instructions:

Please complete the following projects. Submit all functions and test programs (m files) on Canvas. Provide a readme file if any special instructions are needed to test them.

Caution: You are NOT allowed to use any internal ‘connected component related’ functions from MATLAB.

Part 1:

Write a MATLAB function to find all the connected components in a given binary matrix. Your function will have two inputs, the matrix X and the connectivity (i.e., the number of neighbors to be considered) e.g., 4 or 8 neighbors. Test your program by using a 128x128 binary matrix which includes ones at random locations.

Here is how you can write a MATLAB function

```
function resulting_image = find_connected_components (input_matrix, connectivity)
% Your program goes here
end
```

Expected Inputs or Arguments:

- *input_matrix* : Matrix X described above , which can be generated randomly using the following command:
  input_matrix = randi([0,1], 128, 128)
- *connectivity* : This argument should accept either 4 or 8 as input to mean either 4-neighbor or 8-neighbor traversal.

Expected Output:

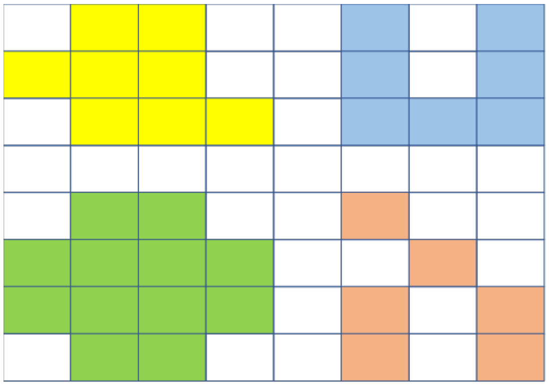
- *resulting_image* : Should be an image which distinguishes each component with a different color.

Deliverables:

Please submit a file named find_connected_components.m which should contain a function to find connected components of a given matrix. Also submit a test program which tests your function and displays the connected components (Please visualize different components by coloring them differently).

Example:
Here is an example that demonstrates this idea using an 8x8 binary matrix as shown below: Goal is to go over all the elements in the matrix and identifying the regions which have 1's in a connected manner. The matrix (on the left) shown below has random ones and zeros in an 8x8 matrix and your connected component function was able to label 4 connected regions shown below (on the right).

0	1	1	0	0	1	0	1
1	1	1	0	0	1	0	1
0	1	1	1	0	1	1	1
0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0
1	1	1	1	0	0	1	0
1	1	1	1	0	1	0	1
0	1	1	0	0	1	0	1



Part2:

Write a program to calculate the distance between two elements in a matrix X using either Euclidean, Manhattan or Chessboard distances, and display the calculated distance on the matrix shown as an image (See example below).

Here is the **expected** MATLAB function:

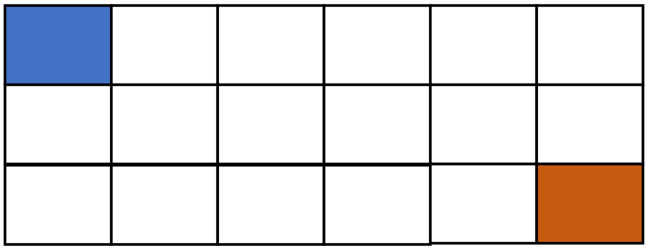
- *function find_show_distance(input_matrix, location_1, location_2, distance_measure)*

  Where:
- *input_matrix*: Should accept a rectangular matrix
- *location_1*: [row, column] of the first element
- *location_2*: [row, column] of the second element
- *distance_measure*: Should accept values 0, 1 or 2 where
  - 0 – Euclidean distance
  - 1 – Manhattan distance
  - 2 – Chessboard distance

Expected Output:
It displays an image with a line drawn between the two given elements (locations) for which the distance has been calculated and shows the calculated distance along the line.

Deliverables:
Submit a file named find_show_distance.m which contains a function to find and show the distance between two elements in matrix.

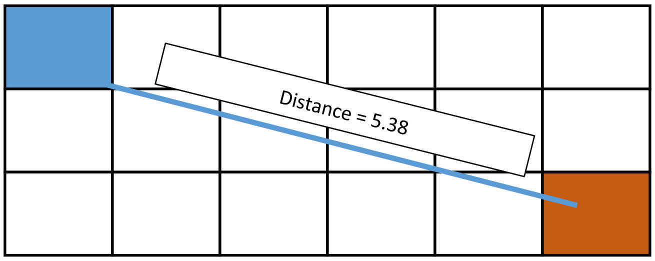
Example:
Consider the matrix shown below in a grid format. There are 3 rows and 6 columns in this matrix. Suppose we want to find the distance between the blue element and orange element using Euclidean distance.



Then the input to the function find_show_distance would be as follows:

input_matrix = 3x6 matrix
location_1: [1,1]
location_2: [3,6]
distance_measure = 0

Expected output as follows:


Distance = 5.38