LAB # 02:Basic Image Processing & Connected Component Analysis

Lab Objective:

The objective of this lab is to introduce the student with some transformation especially with respect to image processing and perform connected component labelling in images and to get an understanding of intensity level resolution.

Lab Description:

1. Rotation:

Rotation of an image for an angle θ is achieved by the transformation matrix of the form

$$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

But OpenCV provides scaled rotation with adjustable center of rotation so that you can rotate at any location you prefer. Modified transformation matrix is given by

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

where:

$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

To find this transformation matrix, OpenCV provides a function, **cv2.getRotationMatrix2D**.

Fig:rotation

2. Connected Component Analysis

Connected Component Analysis or Labelling enables us to detect different objects from a binary image. Once different objects have been detected, we can perform a number of operations on them: from counting the number of total objects to counting the number of objects that are similar, from finding out the biggest object of the bunch to finding out the smallest and from finding out the closest pair of objects to finding out the farthest etc.

Connected Component labelling procedure is as follows:

- Process the image from left to right, top to bottom:

  If the next pixel to process is 1

  i.) If only one from top or left is 1, copy its label.

  ii.) If both top and left are one and have the same label, copy it.

  iii.) If top and left they have different labels

  - Copy the smaller label
  - Update the equivalence table.

  iv.) Otherwise, assign a new label.
- Re-label with the smallest of equivalent labels

Video explanation: https://www.youtube.com/watch?v=ticZclUYy88

Some Useful Commands:

i) arr = np.array([[1, 2, 3] , [6, 5, 4]])

arr + 2 will add 2 in each element of arr

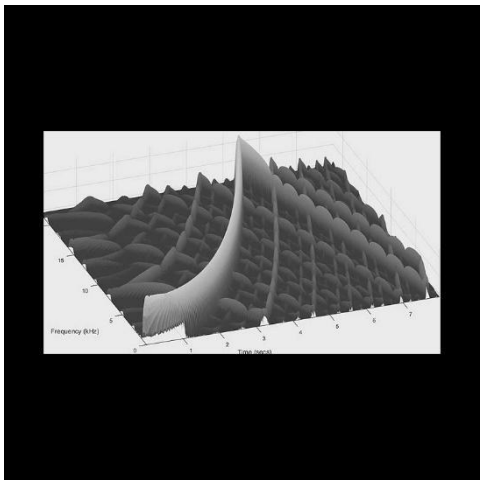
ii) arr = np.array([[1, 2, 3] , [6, 5, 4]])

arr == 2 will return following bolean array

array([[False, True, False] , [False, False, False]])

LAB TASK:

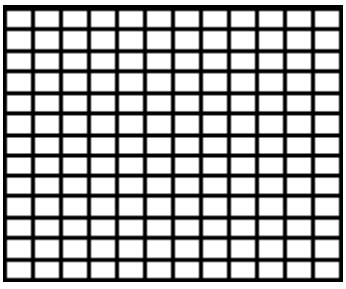
1:Create a generic code that create a border around any landscape image as shown below. The length of right and left borders must be 10% of the original horizontal length of the image. The length of upper and lower border must be such that the image now have same number of rows as columns. Save the image.



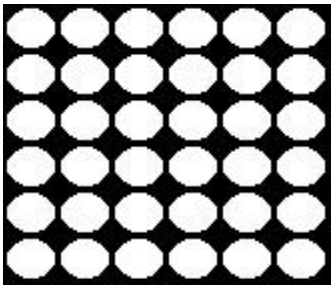
2:Read any image that you want and save it in gray scale. Now rotate the image that you have read. Write the image to the disk.

Hint: you can use built in functions of opencv

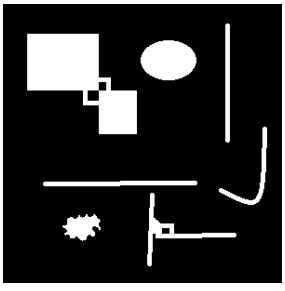
3: For the image given below (provided with the lab handout), apply the connectected component labelling and count the total number of white objects. First threshold the images and then do connected component analysis.



(a)



(b)



(c)

Distance Measures:

If we have 3 pixels p, q and z respectively p with (x,y) ,q with (s,t) and z with (v,w)

Then D is a distance function or metric if

a. $D(p,q) \geq 0$,$D(p,q) = 0$ iff $p = q$

b. $D(p,q) = D(q,p)$

c. $D(p,z) \leq D(p,q) + D(q,z)$

Euclidean distance between p and q is defined as:

$$De(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$$

The pixels having distance less than or equal to some value r from (x,y) are the points contained in a disk of radius r centered at (x,y).

D4 distance (also called **city-block distance**):

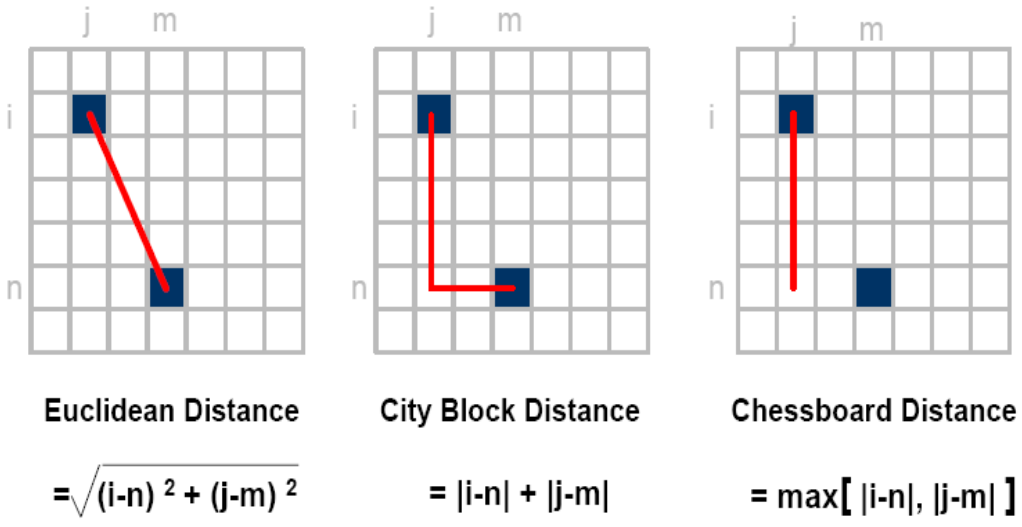
$$D4(p,q) = |x-s| + |y-t|$$

The pixels having a D4 distance less than some r from (x,y) form a diamond centered at (x,y) .

D8 distance (also called **chessboard distance**):

$$D8(p,q) = \max(|x-s| , |y-t|)$$

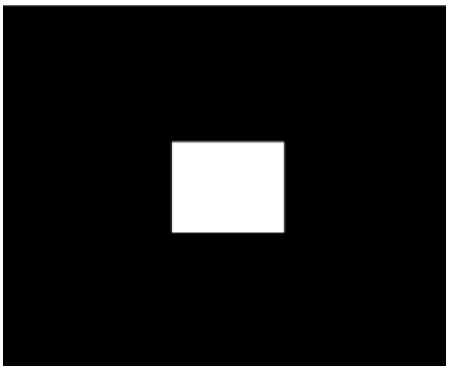
The pixels having a D8 distance less than some r from (x,y) form a square centered at (x,y)



Home Task

- Write a function which takes two point of image as input argument and calculate above distances.

Calculate distance(p1,p2,choice). Choice variable is input integer. If its value is 1 calculate Euclidean distance , 2 for city block distance and 3 for chessboard distance.



- Write a function which take maze binary image as input argument and should be able to solve given maze image using connected component analysis technique.

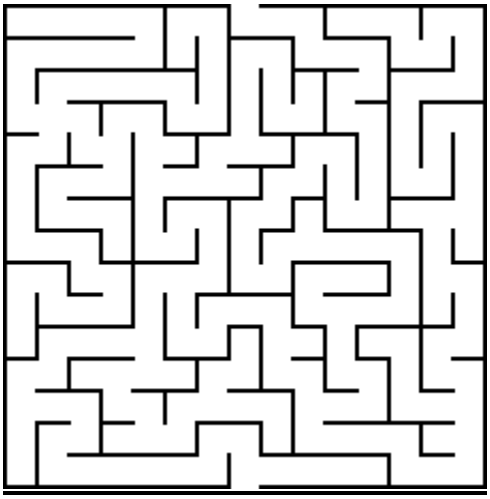


Fig maze

- As we discussed in the class about zoom out and zoom in features and behind these, there are decimation (down sample) and interpolation (up sample) technique.

Interpolation techniques:

i. Bilinear Interpolation Technique
ii. Cubic interpolation Technique

Your task is to read image of size 500*500. Down sample the image by 2 so its size will be of 250*250. Apply Bilinear interpolation technique for up sampling by 2.

You can use built in functions for interpolation and decimation techniques other than implementing the actual algorithm.

Applications of Connected Component Analysis:

Following are some applications of connected component analysis. We can use this analysis in many real problems such as,

I. Binary object classification
II. Detect multiple objects in image

References:

- ✓ https://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm
- ✓ https://www.pyimagesearch.com/2016/10/31/detecting-multiple-bright-spots-in-an-image-with-python-and-opencv/
- ✓ https://www.imageeprocessing.com/2012/11/bit-plane-slicing.html

THINK!!

1. What will be the number of dimension of a grayscale image if opened as colored?
2. Is image a list, tuple or an array?
3. A black and white image can only have what gray levels?
4. Is it possible to count objects which are inside an object using connected component analysis?
5. Can we apply connected component analysis any image without doing any preprocessing