**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |
| **Course/Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |

**CS-477 Computer Vision**

**Lab#6: Pixel connectivity and noise reduction filters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **PLO4-CLO4** | **PLO5-CLO5** | **PLO8-CLO6** | **PLO9-CLO7** |
| **Name** | **Reg. No** | **Investigation**  **(5 marks)** | **Modern Tool Usage**  **(5 marks)** | **Ethics**  **(5 marks)** | **Individual and Team Work**  **(5 marks)** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Lab#6: Pixel connectivity and noise reduction filters**

**Objectives:**

The objective of this lab is:

* To perform connected component labeling of a binary image.
* Learn how to implement the arithmetic mean filter, as well as some of its variations, such as the contra-harmonic mean, the harmonic mean, and the geometric mean filters.
* Learn how to perform order statistic filtering, including median, min, max, midpoint, and alpha-trimmed mean filters!

**Lab Instructions**

* This lab activity comprises of following parts: Lab Exercises, and Post-Lab Viva/Quiz session.
* The lab report shall be uploaded on LMS.
* Only those tasks that are completed during the allocated lab time will be credited to the students. Students are however encouraged to practice on their own in spare time for enhancing their skills.

**Lab Report Instructions**

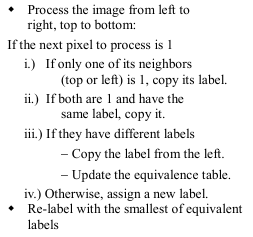
All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

* Lab objectives
* Python codes
* Results (graphs/tables) duly commented and discussed
* Conclusion

# Lab6: Connected component analysis.

## Theory:

**Connected Component Analysis** or Labeling 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 labeling procedure is as follows:



Algorithm 1: simplified two pass algorithm.

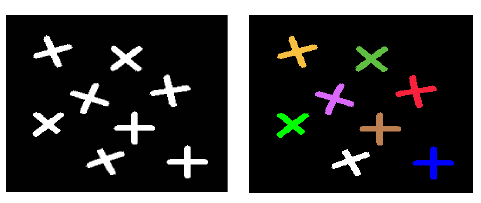


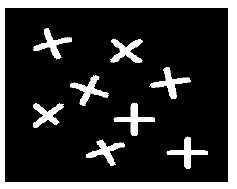
Figure 3: Labeling of each connected component to a different color (for visualization only)

### Task 1:

* Read “cc.png” from directory.
* Apply connected component labelling using 4 connectivity and count total number of objects in the list. (**which means write your own function from scratch!**)
  + HINT1: the image is a binary image with background (black portion) has numeric value of 1 while the white objects have numeric values of 255
  + HINT2: you can use **two-pass algorithm** from the Algorithm 1 (which doesn’t use a specialized data structure and simply use two for loops!)
* Use Opencv’s **cv2.connectedComponents** function and admire the underlying workings.

**Note: -** Before getting to test your code on an image it is highly recommended to test it on a 2d-array. It will help you in following ways:-

* As a good measure of cross check i.e. you can quickly figure out which condition is going wrong because you have a standard output to compare it with and you can apply corrective measures where you find conditions going wrong.
* Debugging is easier with a smaller size 2d-arrray than a huge size image.



### Penalty for copying code from internet: -

You are strictly not permitted to copy code from the internet because this lab will be evaluated on the base of quiz/viva (in next lab) so anyone who has poor understanding of procedure overall will get a penalty in lab submission part as well.

## Task #2:-

Corrupt the input images with different types of noise models such as:-

1. 'Gaussian'
2. 'poisson'
3. 'salt & pepper'
4. 'speckle'
5. Salt only noise
6. Pepper only noise

### Note:-

You may use the skimage.util.random\_noise function from scikit image library.

### Example:-

guassiannoise=skimage.util.random\_noise(img,"gaussian")

peppernoise=skimage.util.random\_noise(img,"pepper")

You can incorporate all the above mentioned 6 types of noise models by playing

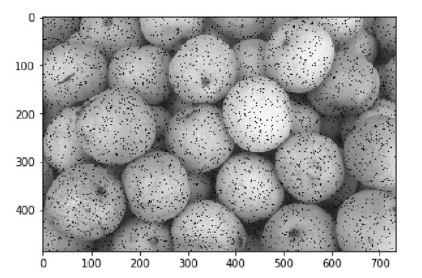
around with the second argument.

For further details you may consult the following link:-

<https://scikit-image.org/docs/stable/api/skimage.util.html#random-noise>

### image with gaussian noiseImage with

### image with paper noise

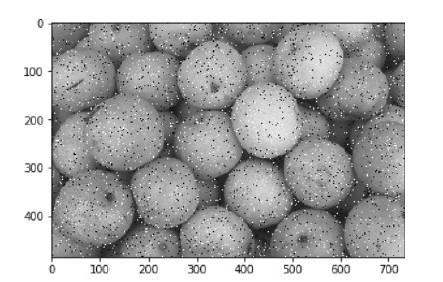


### Image with Poisson noise

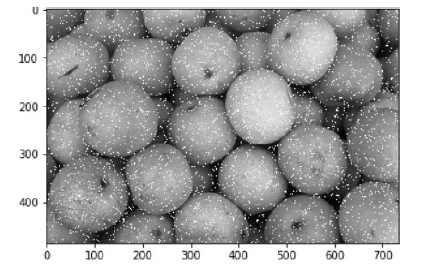
A close-up of a pile of fruit

Description automatically generated

### Image with salt and paper noise



### Image with salt Noise



### Image with speckle noise

A close-up of a pile of fruit

Description automatically generated

## Task #3: -Apply different kinds of noise removal filters as given below.

1. Arithmetic mean
2. Geometric mean
3. Harmonic mean
4. Contra harmonic mean, the contra harmonic mean filter is used for filtering
5. an image with either salt or pepper noise (but not both).
6. Max filters
7. Min filters

**Note:-**

**1):-**in order to find out how a filter is affecting a particular type of noise you can save the resultant image using cv2.imwrite because just plotting would might not give you a precise intuition.

**2):-** You can use **filters** class from **scikit image** library. It contains very handy and easy to use builtin functions for filters.

You can import it as follows:-

o import skimage

o from skimage import filters

As it is a well renowned library for image processing so it has vast variety of functions. But for today’s lab as we are using filtering so **scipy.ndimage** package is useful for you.

You may visit the following link to get an insight about most of the filters for task2.

<https://docs.scipy.org/doc/scipy-0.14.0/reference/ndimage.html>

**1):- Arithmetic mean**

Suppose you have a 3\*3 kernel as follows:-

kernel = np.array([ 1, 1, 1, 1, 1 , 1, 1, 1, 1]).reshape(3, 3)/9

You can apply it using cv2.filter2D function as used in previous labs.

**2):- Min filter**

You may use the minimum\_filter function from scikit image.

https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.minimum\_filter.html

**3):- Max filter**

For max filter of scikit image visit the following link.

<https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.maximum_filter.html#scipy.ndimage.filters.maximum_filter>

**4):- Median filter**

For median filter visit the following link.

https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.filters.median\_filter.html#scipy.ndimage.filters.median\_filter

**5):- Geometric mean**

Geometric mean filter can be found in rank class of skimage.filters

Visit the following link to read further.

https://scikit-image.org/docs/dev/api/skimage.filters.rank.html#skimage.filters.rank.geometric\_mean