CMPEN/EE455: Digital Image Processing I

Computer Project # 2
Connected Component Labelling and Set Operations of Images

Pavan Gurudath
Date: 09/22/2017

OBJECTIVES

This project is aimed at performing simple image-processing operations on the images, such as connected-component labeling and logical operations. The main objectives of this project include the following:

1. **Bright-Region Extraction**
   - To choose a threshold for a noisy image such that the bright regions of the given image could be extracted.
   - To obtain the connected components of the image and therein label each pixel using an inbuilt function in MATLAB.
   - To obtain the four largest connected components.
2. **Logical (Set) Operations**
   - To build functions that would perform AND, OR and XOR binary-image operators and NOT unary-image for two input images.
   - To create a minimum operator function which would process two images and gives an output image assigned with minimum pixels of the processed images.

METHODS

This project consists of two main objectives, i.e. bright region extraction and logical set operations. The following subsections details the methods in which these project statement have been solved.

1. **Bright Region Extraction:**
   In order to perform the objectives under this category, an input image named ***'wheelnoise.gif'*** has been utilized. This image is an image induced with noise. The main task is to obtain the major components of the wheel from the noisy image. The four main components can be observed in the original image 'wheel.gif'. The following set of steps are carried out to obtain the bright regions.

   1.1. To choose a threshold for a noisy image such that the bright regions of the given image could be extracted.

   - The image ***'wheelnoise.gif'*** is scanned through every pixel and the image is thresholded.
   - The thresholding of an image can be expressed as follows:

$$f(x,y) = \begin{cases} 0, & f(x,y) < 150 \\ f(x,y), & otherwise \end{cases}$$

   - The resultant thresholded image is stored as ***'fthresh.gif'*** and undergoes further processing.

   1.2. To obtain the connected components of the image and therein label each pixel using an inbuilt function in MATLAB.

   - The connected components of the thresholded image 'fthresh.gif' is found out by using an inbuilt MATLAB function named bwlabel whose syntax is as follows:

$$[\boldsymbol{flabel}, \boldsymbol{num}] = \boldsymbol{bwlabel}[\, \boldsymbol{fthresh}, \boldsymbol{conn}]$$

   - The aforementioned functioned creates the labelled image and stores it in the variable flabel. 'num' holds the value of the number of labelled components present in the image. 'fthresh' is the input image, while 'conn' mentions the type of connectivity, i.e. either 4-connectivity or 8-connectivity.
   - In order to display the labeled image with colored components, the following MATLAB inbuilt function is utilized:

$$\boldsymbol{fRGB} = \boldsymbol{label2rgb}(\boldsymbol{flabel})$$

   1.3. To obtain the four largest connected components.

   - To obtain the four largest connected components, we need a distribution of the connected component label and its frequency. To achieve this, an array named 'count' is defined whose first row consists of the frequency of the corresponding label number in the second row.
   - A loop is assigned to go through every pixel in order to obtain the frequency of every labelled component from the 'flabel' image and store it in the 'count' array.
   - The 'count' array is sorted using selection sort algorithm to obtain the largest four components and is stored in another array of size 2x4 named 'big'.

- The label number is passed to a function named 'component.m' which takes the input label number along with the flabel image. This function retains the label number and converts all the other components to be zero.
- The four largest components are displayed individually and together as a whole.

2. **Logical (Set) Operations**

In order to perform the objectives under this category, input images namely ***'match1.gif', 'match2.gif', 'mandrill_gray.tif'*** and ***'cameraman.tif'*** has been utilized. These images are used to perform the binary operation of AND, OR and XOR as well as the minimum of the two images that would lead to an operation otherwise known as erosion. The following set of steps are carried out to achieve the objectives mentioned:

2.1. To build functions that would perform AND, OR and XOR binary-image operators and NOT unary-image for two input images.

- A function for each of the abovementioned operator is built.
- The input images namely ***'match1.gif'*** and ***'match2.gif'*** are sent to these functions.
- The functions are coded such that they satisfy the set union, intersection and complements as per the following definitions:
  - If the pixel intensities are both found to be high, the resulting output pixel intensity is taken to be 1 in case of the AND operator.

$$\boldsymbol{A\ AND\ B = A.B = A \cap B}$$

It is the intersection of all the elements of set A and set B.

  - If the pixel intensities are found to be either both high or even one is high, the resulting output pixel intensity is taken to be 1 in case of the OR operator.

$$\boldsymbol{A\ OR\ B = A + B = A \cup B}$$

It is the set of all elements that belong to set A as well as set B

  - If the pixel intensities are both found to be high or both found to be low, the resulting output pixel intensity is taken to be 0 in case of the XOR operator.

$$\boldsymbol{A\ XOR\ B = A\bar{B} + \bar{A}B = A \oplus B}$$

  - In case of the NOT operator, if the pixel intensity if high, the output is set to low and if the intensity is read as low, the output image intensity is set to high.

$$\boldsymbol{NOT(A) = \bar{A}}$$

Note: The above definition answers the question asked in 2(a).

- In order to perform the AND, OR, XOR and NOT operation, the functions 'andfn.m', 'orfn.m', 'xorfn.m', and 'notfn.m' are called respectively.
- These functions compute pixel by pixel and their calculations are done as shown in the below table.

INPUT		**AND**	**OR**	**XOR**	**NOT(A)**
A	**B**	$\boldsymbol{A \cap B}$	$\boldsymbol{A \cup B}$	$\boldsymbol{A \oplus B}$	$\boldsymbol{\bar{A}}$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

- The output images of the above operations are displayed and stored appropriately.

2.2. To create a minimum operator function which would process two images and gives an output image assigned with minimum pixels of the processed images.

- A function named minfn is created that calculates the minimum pixel intensity of the two input images, namely ***'mandrill_gray.tif'*** and ***'cameraman.tif'*.**
- The function is fed with the two images, and the algorithm compares every pixel intensity value of image C with its corresponding pixel intensity value of image D and stores the value of the lesser intensity level in the resultant image.
- If pixel intensity of image C is $f_c(x, y)$ and that of image D is $f_d(x, y)$, then the function 'minfn.m' calculates the result as follows:

$$\min(C, D) = \begin{cases} f_c(x, y) & if\ f_c(x, y) < f_d(x, y) \\ f_d(x, y) & if\ f_d(x, y) \leq f_c(x, y) \end{cases}$$

- The resultant image is displayed and thus stored.

RESULTS

This section details all the images that are obtained after completing each of the objective. Firstly, the image that is considered as the input image for the extraction of bright region i.e. 'wheelnoise.gif' is as shown in Fig 1.

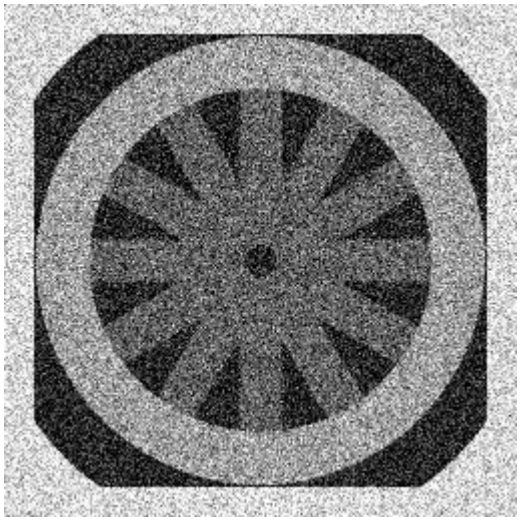

Figure 1: Input Image – 'wheelnoise.gif'

The original image that consists of four components and has no noise unlike fig 1 is as shown in Fig 2.

Figure 2: Without noise original image 'wheel.gif'

Once the image in figure 1 undergoes suitable thresholding as defined in the methodology, the thresholded image is stored as 'fthresh.gif'.

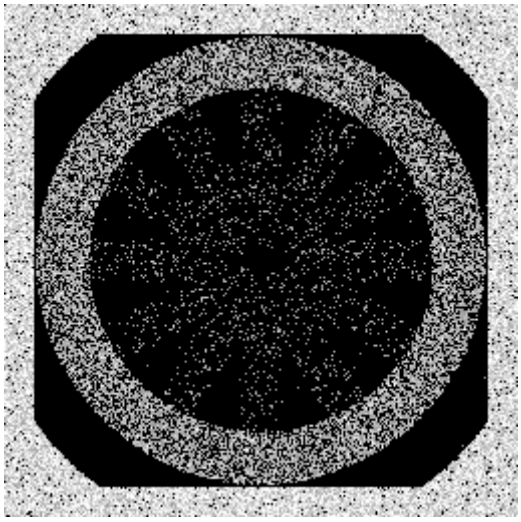

Figure 3: Image thresholded

The thresholded image undergoes labelling after being subjected to the bwlabel inbuilt command. The 8-connected component image is stored in flabel and fig 4 depicts the image after every label is represented with a colored component.

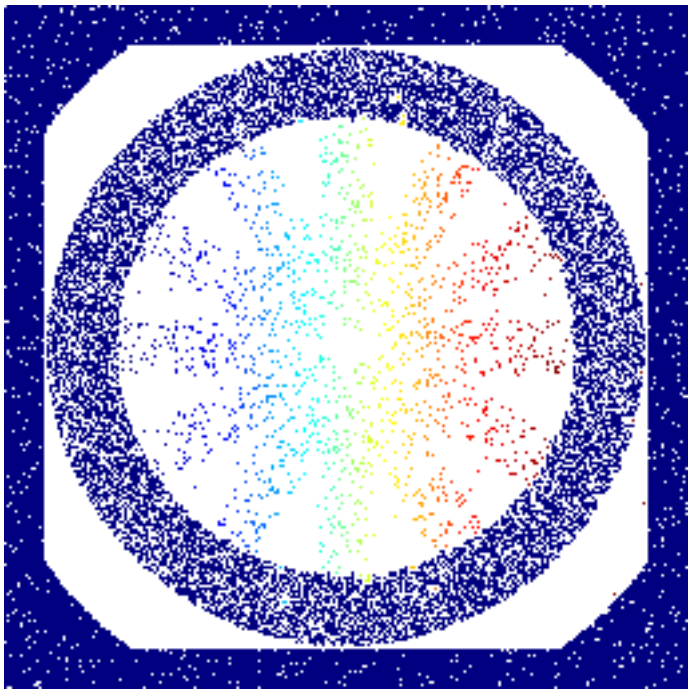

Figure 4: Labeled image with colored components

The four largest components are as shown from Fig 5 through Fig 8. Fig 9 depicts the combination of the four large components.

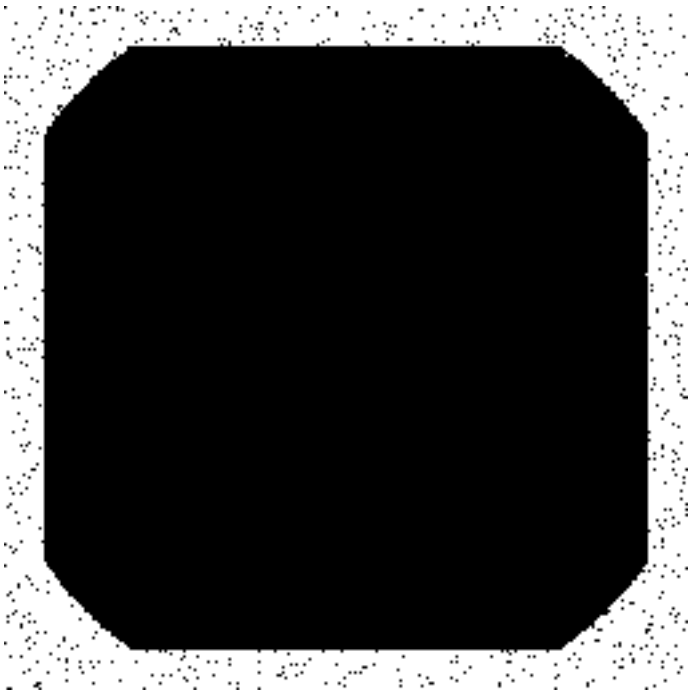

Figure 5: First Component

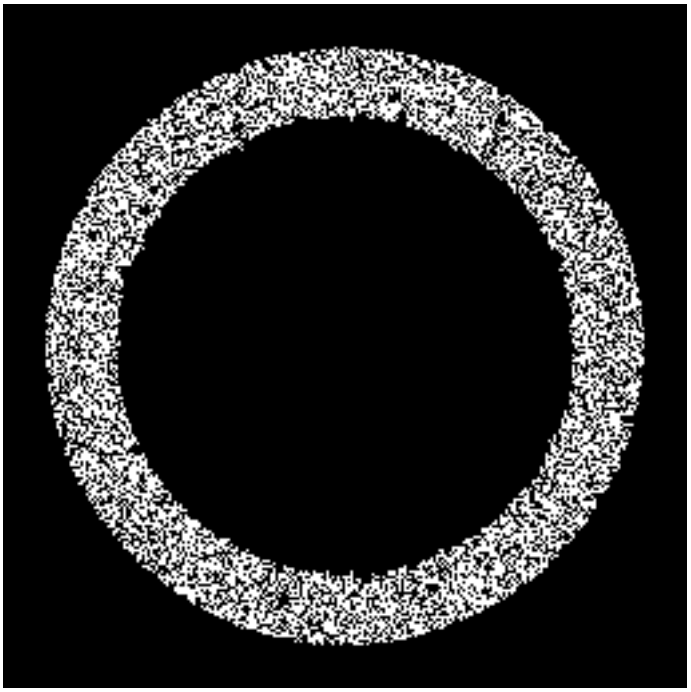

Figure 6: Second component

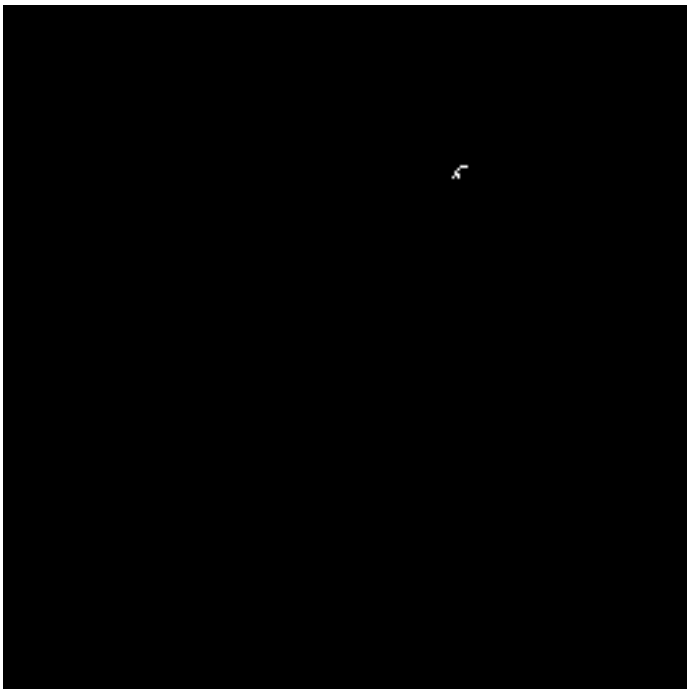

Figure 7: Third component

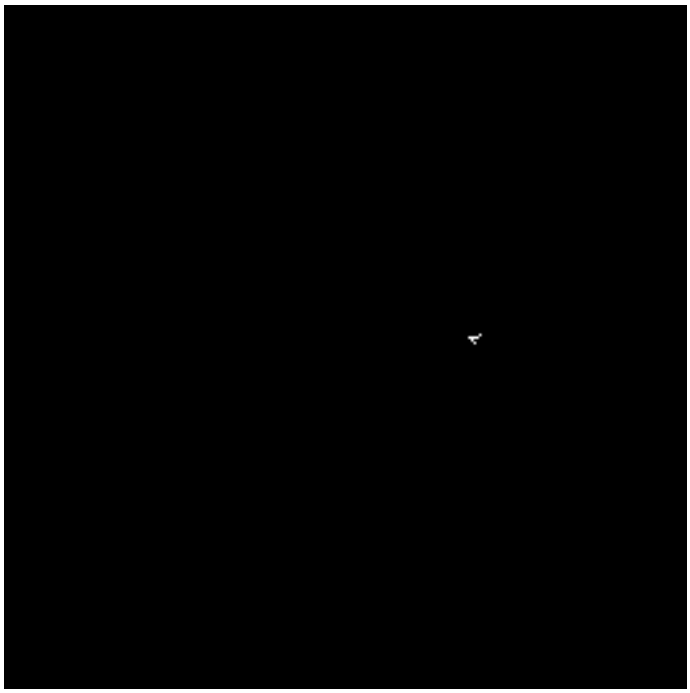

Figure 8: Fourth component

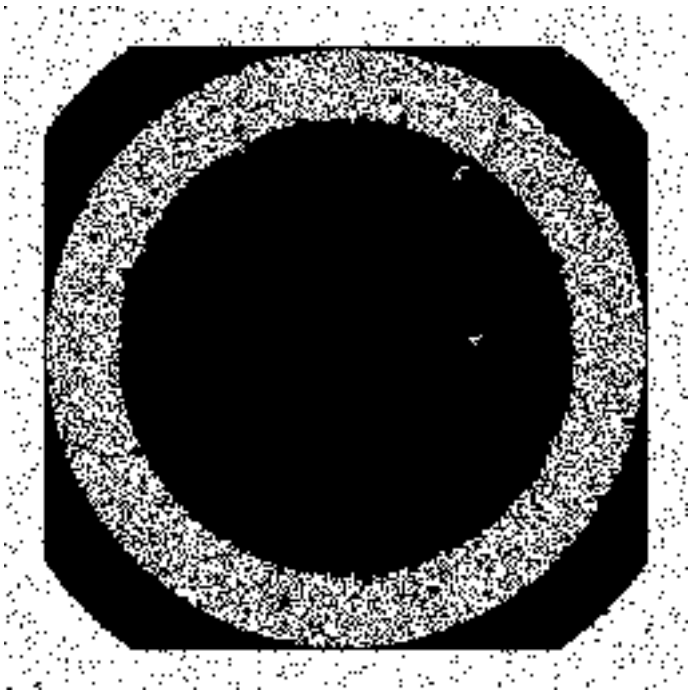

Figure 9: Four largest components

Here, we can see that the Fig 9 in comparison to the original image Fig 1has lost some of the information i.e. the spokes of the wheel seem to have disappeared. This is due to the fact that while thresholding, the intensity of the spokes which are less than the thresholded value, become zero and therefore its information completely. While tinkering with the threshold value, we notice that if the threshold value is reduced or if multiple threshold values are used in order to try to achieve the 'wheel spokes' component, the outer boundary as shown in Fig 5 and the wheel frame in Fig 6 get 8 connected and thereby becomes one frame. This would be the reason that the third and fourth component do not have as much area. One way to rectify this situation would be to reduce the noise present in the original image.

Overall, we observe how every pixel in an image is associated with the surrounding pixel and thereby understand how region extraction can be done using the concept of connectivity.

For the second set of objectives, i.e. to perform the logical operations the input images that were considered for the AND, XOR and OR functions are:

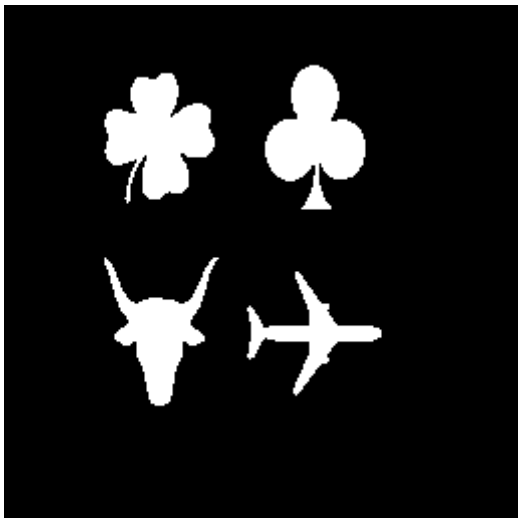

Figure 10: Input image A: 'Match1.gif'

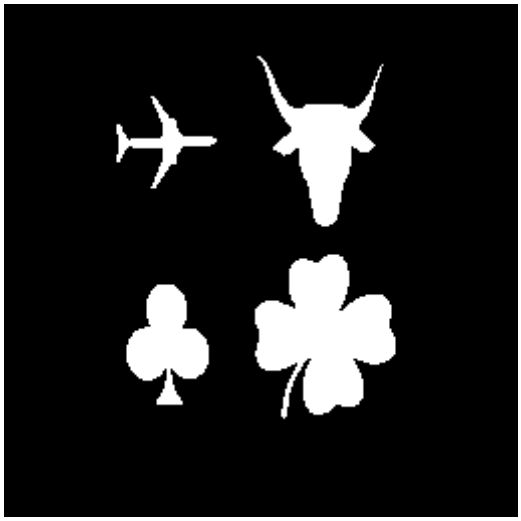

Figure 11: Input image B: 'Match2.gif'

The output images after performing the (A AND B), (A OR B), (A XOR B) and NOT(A) operations are:

Figure 12: A AND B

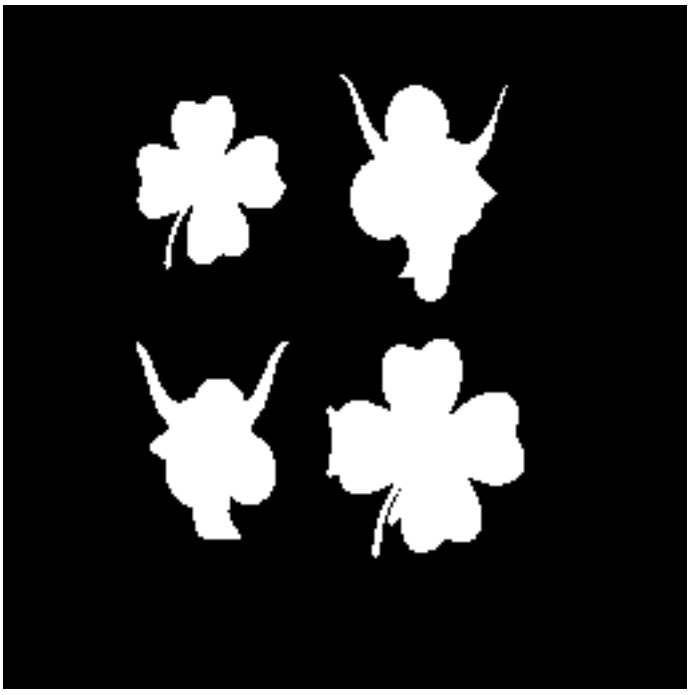

Figure 13: A OR B

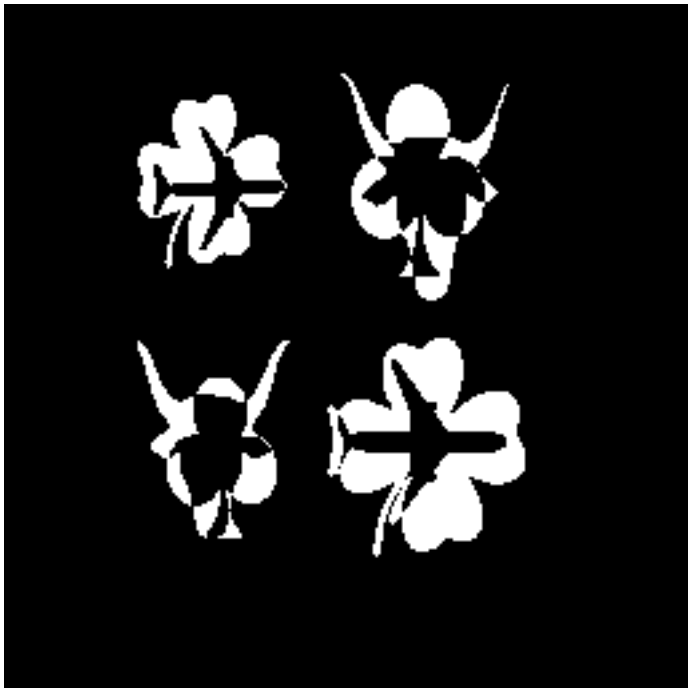

Figure 14: A XOR B

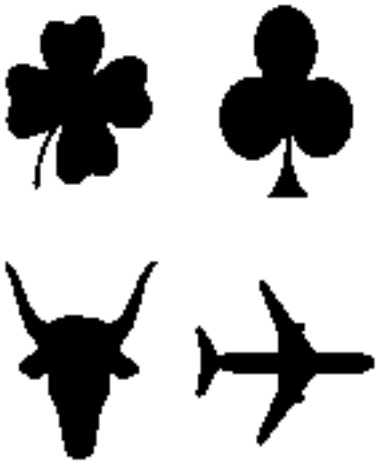

Figure 15: NOT(A)

The input images for the minimum of ‘mandrill_gray.tif’ and ‘cameraman.tif’ are as shown below in figure 11 and 12.

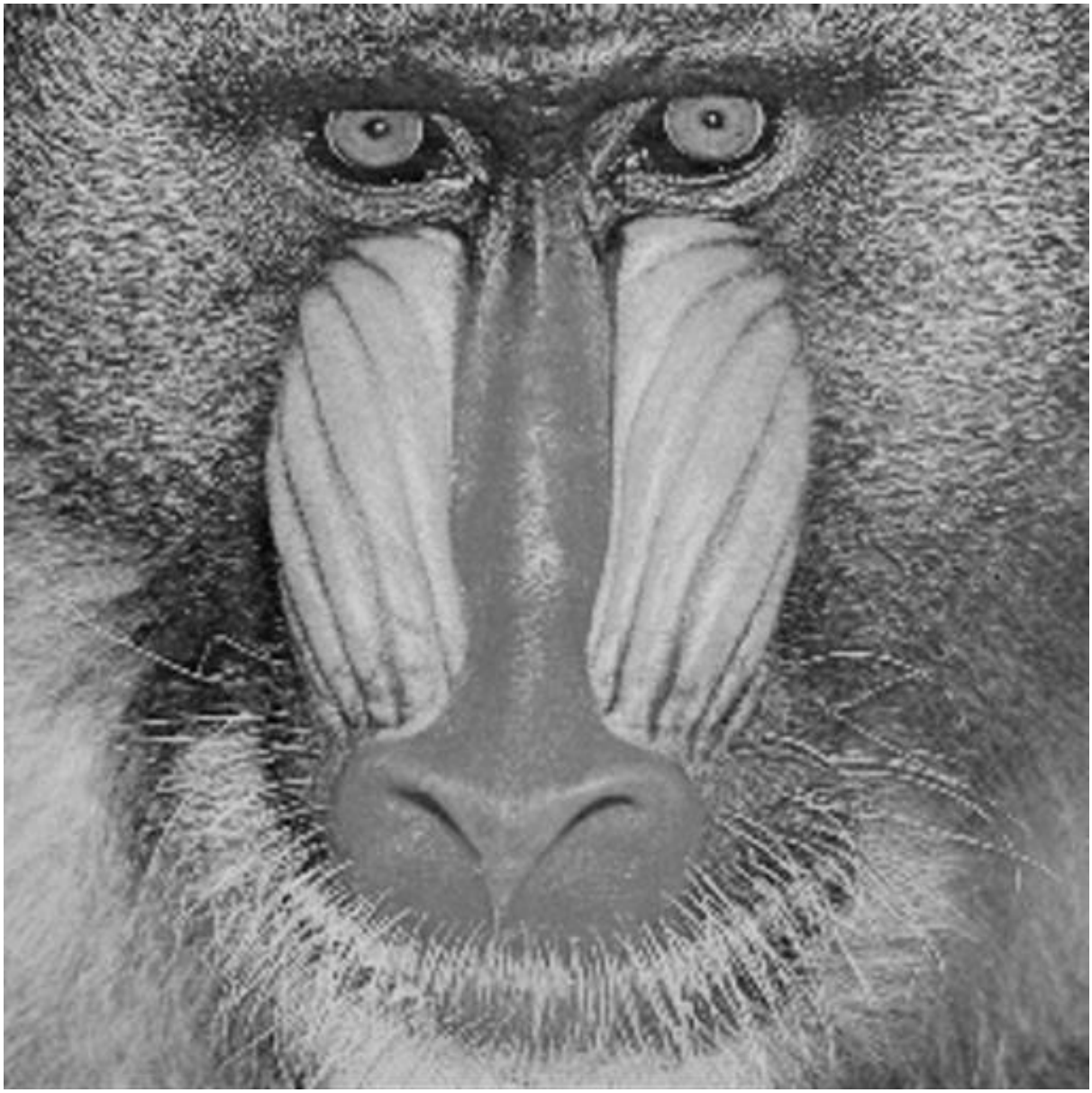

Figure 16: Input image C: 'Mandrill_gray.tif'

Figure 17: Input image D: 'Cameraman.tif'

The operation minimum of C and D results in the following image:

Figure 18: minimum of image C and image D

By performing the processing for the second objective, we observe the logical operations that take place in the case of images.

CONCLUSION:

The problem statement in this project allows us to make certain conclusion based on region extraction and logical operation on images. We can conclude that the regions from an image can be extracted using the concept of connectivity to separate different features of an image from one another. These extracted regions could be further used to examine in detail.

It can also be concluded that the AND operation of images is equivalent to the intersection of sets where every image is a set of all pixel's intensity values. Similarly, the OR and NOT operation is equivalent to the union and complement of sets respectively, where every image is a set of all pixel's intensity values. These operations could be used in order to dilate and/or erode certain regions of an image.

While computing the minimum of two images, it is observed that there is a tendency for one of the images to appear in foreground while the other tends to appear as the background or a screen.