

# **Auto-Segmentation using Mean-shift Algorithm and Entropy Analysis**

A major project report submitted in the partial fulfillment of the  
requirements for the award of the degree of

**MASTER OF TECHNOLOGY**  
(INFORMATION SYSTEMS)

Submitted By:

**ANKIT KUMAR** (Roll  
No. 2K13/ISY/04)

Under the esteemed guidance of

**Dr. SEBA SUSAN**  
( Assistant Professor )



**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**  
**DELHI TECHNOLOGICAL UNIVERSITY BAWANA  
ROAD, DELHI-110042**  
**SESSION: 2013-2015**

## **CERTIFICATE**

This is to certify that work entitled “**Auto-segmentation using Mean – Shift Algorithm And Entropy Analysis**” submitted by **Ankit kumar (2k13/ISY/04)** to Delhi Technological University, Delhi for the award of the degree of Master of Technology is a bonafide record of research work carried out by him under my supervision.

The content of this thesis , in full or parts , have not been submitted to any other institute or university for the award of any degree or diploma.

**Dr. SEBA SUSAN**

Project Guide

Assistant Professor

Department of Computer Science and Engineering

Delhi Technological University

Shahbad Daultpur, Bawana Road, Delhi-110042

Date:-----

## **ACKNOWLEDGEMENT**

I would like to thank my project guide, **Dr. Seba Susan** for her valuable guidance and wisdom in coming up with this project. I humbly extend my words of gratitude to **Dr. O. P. Verma .**, Head of Department, and other faculty members of Computer Science and Engineering department for providing their valuable help and time whenever it was required. I thank all my friends at DTU who were constantly supporting me throughout the execution of this thesis.

Special thanks to the Almighty Lord for giving me life and the strength to persevere through this work. Last but not least, I thank my family and for believing in me and urging me. May you all be blessed.

**Ankit kumar**

**Roll No. 2k13/ISY/04**

M.Tech (Information Systems)

E-mail: **ankitsai11@gmail.com**

Department of Computer science and Engineering

Delhi Technological University

## ABSTRACT

The development in the segmentation of object from the images and video has reached to the great height .Image segmentation has been widely used to split image into smaller parts so that useful information can be derived from them. This subject has drawn a lot of attention because of its useful applications and also because it gives a wide range of ideas that are still be explored more. Segmentation is the process of splitting the image into smaller regions. Auto-segmentation can be used in detecting the objects and the further results can be used for many purposes like clustering , marketing purpose , medical purpose .Over the past years, many segmentation Algorithms have been introduced in order to retrieve the useful data.

In this thesis , we have proposed a new approach for auto-segmentation using mean –shift Algorithm to probe further in this arena. We have used different datasets including Drive Dataset and Berkeley Dataset to test our approach. In this, we came to know that the auto-segmentation algorithm gives better results at high value of entropy.

# Table of Contents

Table of Contents .....	
Certificate .....	i
Acknowledgement .....	ii
Abstract.....	iii
1. Introduction.....	1
1.1 Evolution of Segmentation .....	1
1.2 Different Methods of Segmentation .....	2
1.3 Aim Of Segmentation .....	2
1.4 Defects .....	3
1.5 Application of Segmentation.....	3
2. Literature Review .....	4
2.1 Image Segmentation Techniques .....	5
2.2 MeanShift Algorithm .....	7
2.3 Bandwidth Selection in MeanShift Algorithm .....	7
3. Mean Shift Algorithm.....	8
4. Proposed Auto-Segmentation Using Mean-Shift Algorithm.....	13
4.1 Automization of Mean-Shift .....	14
4.2 Proposed Algorithm .....	15
5. Results and discussion .....	16
6. Conclusion .....	92

# **CHAPTER- 1**

## **Introduction**

## 1.1 Evolution

As we are living in the era of 21 century, we require the different methods to split the image . so, that the processing can be applied effectively in order to retrieve any information from the images . **Segmentation** is a technique by which we can subdivide the image in small regions or images . The result obtained can be used for medical purpose like detecting cancer in brain etc, clustering purpose etc. It basically cluster the data into groups which are similar in whole image . Many algorithms were developed for segmentation and clustering.

Images are said to one of the most important part in human life as it is used to share the information and sending it to the other person . There has been many development in research of image segmentation , enhancement , edge detection in the last decades . There are many applications of image segmentation which is used for medical purpose . Many algorithms were proposed for co-clustering in medical images[1] . Some of the co-clustering techniques are based on bacterial foraging [1] .

There are different methods are used for segmentation purpose . and Many papers are proposed on this image segmentation .

- (1) k- means clustering
- (2) Region growing clustering
- (3) A Diffusion Approach to Seeded Image Segmentation
- (4) C- means algorithm

Many work has done on different algorithms . The image segmentation can be used for the multichannel images.

### **Image Segmentation can be classified on two different aspects of images**

#### (1) Differences in Intensities

In this case, the intensities of the pixels are compared with its neighbouring pixels intensity .The point where the intensity difference occur that means the segment of image changes and its is used in case of edge detection method

#### (2) Similarities in Regions

The image is divided into regions that are similar with set of criteria .It includes the thresholding, region growing, region splitting and merging.

## 1.2 Different Method for Image Segmentation :

Image segmentation can be done on different methods as explained in [2]

(a) Image Segmentation based on edge detection –

In this image segmentation is done on the basis of difference in contrast and where there is change in intensity the edge is detected . There may be a lot of edges present in the image and that edges may be discontinuous in nature .

If there is any object in the image then there will be closed edge around the boundary of the object . Thresholding technique is based on image space regions i.e. on characteristics of image

(b) Image Segmentation based on Thresholding –

This method is used where there is object of light color with dark background . Thresholding technique is based on image space regions . Thresholding operation convert a multilevel image into a gray channel image i.e., it is done for choosing a proper threshold  $T$ , it divide the pixels of image into several different regions and separate the light and more useful objects from background. Any pixel  $(x, y)$  is considered as a part of object if its intensity exceed or is in the range of the threshold value i.e.  $f(x, y) \geq T$ , otherwise the pixel belong to background . We have two type of thresholding in image .i.e, local and global thresholding . When the value of thresholding is kept same it is know as global thresholding , otherwise known as local thresholding .

(c) Image Segmentation based on region growing –

This is one of the simplest segmentation method in which the image is splitted into different region and then region growing concept is applied . and after applying the method the segmentation we start merging the region with similar values of intensities .

(d) Image Segmentation based on Clustering –

In this we basically start clustering the image on the basis of similarities in value of pixels and start making the clusters and after that we make the cluster with similar value of pixel values .

## 1.3 The Aim of Segmentation

The aim of segmentation is to divide the image into small parts so that the required information from the image can be retrieved. In this , we separate the homogenous group of data from the image and separate the unuseful information . In this , we cluster the same kind of data . and find the equally likely modes .



## **1.4 Defects in Segmentation**

There are the different methods available for image segmentation. Every methods have its benefits i.e some image segmentation techniques show good results for particular type of input images. Several algorithms and technologies has been developed but every algorithm have its specific application depending on input type of data .

Some Algorithms give better results for color image as input and some give better result for gray scale images.

## **1.5 Applications of Segmentation :**

It is a vigorous algorithm that has a lot of practical applications particularly in the computer vision field . An image can be represented by different regions. The regions are further differentiated by multiple dimensions. The following are the applications of mean shift which help to obtain the required results.

### **Clustering**

The most important application of Mean Shift is clustering. The fact that Mean Shift is non-parametric and does not make assumptions about the number of clusters or the shape of the cluster makes it a superlative method for handling clusters of arbitrary shape and number.

The stationary points obtained help in assessing the density function. All points associated with the same stationary point belong to the same cluster.

### **Computer Vision Applications**

Mean Shift is used in multiple tasks in Computer Vision like segmentation, tracking, discontinuity preserving smoothing etc. For more details see [2],[8].

### **Marketing**

To find the similar group of customer in market .

### **Medical**

To extract the required segment of the image for medical purpose .

### **Insurance**

To cluster the customers and to identify the frauds .

# **Chapter – 2**

## **Literature Review**

## 2.1 Image Segmentation Techniques

Segmentation is one of the image processing technique by which we can retrieve information from image . It is an area of research in academics and industry . A large number of segmentation algorithms have proposed in past years .Some image segmentation methods uses different algorithms . Some algorithms uses region growing for color image segmentation [3] . for the segmentation , different approaches are used like wrapper based approach [4] have also been used . For segmentation of images different operators can be used like sobel operator [5] which also explain about maximum entropy. Juyong Zhang et. al[6] gave a diffusion approach used in image segmentation [6]. Image Segmentation include Edge detection based method[9], Region growing Methods [10].

Image segmentation has been widely applied in biomedical imaging and the aim is to divide the image into smaller segments by using graph partitioning which is more effective for image segmentation [7] . There are also some approaches in which the user only give resolution as input only [8]. There has been a lot of development done in image segmentation field but still it can be explored more . The colour information is used to create histograms and where there is change in intensity of pixel values the histogram shows the change in intensity .

We also prove the convergence for discrete data for mean-shift procedure to the nearest stationary point [12] . Its main application to find the modes of density.

As explained in [12] ,there are some low level and high level tasks are required for the segmentation . If the result of low level is not correct then it may be misleading and will not produce the appropriate results . This algorithm do the two main tasks , persevering the discontinuity smoothing and image segmentation .The approach in [12] explain that low level should provide the enough representation of input and the feature extraction can be controlled by changing the parameters in input values . It provides the results for both color image and gray scale image. By putting the significant features together, It provide the excellent tolerance to noise level which can mislead the results . The disadvantage of the algorithm can be avoided by adding the appropriate parameters from the input domain .

As per the fuzzy co-clustering algorithms [1], is proposed for medical images. This algorithm basically works on segmenting the required region in histo-pathological images which consist of groups of similar cells indicating some form of abnormality in the animal tissue. This algorithm relies on improved colour clustering results when [1] algorithm is applied as compared to the other conventional clustering techniques.

In this different type of lesion have extracted from the images . In this , objective function is optimized by bacterial foraging algorithm which gives more specific values to the parameters involved . The algorithm produces more accurate and form distinct co-clusters depending on the membership values . In algorithm [1] is used for colour segmentation of the medical images to detect dark and bright lesions. This algorithm is independent of shapes as , it is color based method . The results are quite good but for the computational complexity and the number of iterations involved.

The algorithm [1] is ineffective for the segmentation of colour medical images using CILELAB features.

The algorithm is used successfully to detect the lesions even in not so distinct background. The number of clusters present in the images is judged by the validity function. The main advantage of this algorithm is that it can delineate clusters based on feature membership function by the use of rank and the corresponding centroids which identify particular shades of colour.

There is an algorithm[1] Fuzzy Co-Clustering of medical images using bacterial foraging . This is a modification of the Fuzzy Clustering for Categorical Multivariate data (FCCM) algorithm termed as dasia Fuzzy Co-Clustering Algorithm for Images (FCCI) psila is proposed for clustering of medical images. The main work is to segment regions of interest in histo-pathological images which consist of groups of similar cells indicating some form of abnormality in the animal tissue. In these proposed method relies on improved color clustering results when FCCI is applied on images as compared to the conventional clustering techniques. The method also categorizes different types of lesions based on the co-clustering results. The objective function is optimized using the bacterial foraging algorithm which gives image specific values to the parameters involved in the algorithm. The color segmentation results are found to be more accurate, producing well formed and valid clusters having ldquocrispsila values of membership function with lesser number of iterations. The algorithm results in distinct co-clusters ranked in the order of their memberships.

## 2.2 Mean-Shift Algorithm

This algorithm is used to find the densest region among the distribution of data. The main task in the algorithm is to find modes of equally likely hood in the distribution of data. In this, a kernel is selected where the distribution of data is more. After that it find the centroid or centre of mass and it shift towards the direction of maximum increase in density. It propagates similarly for the all distribution of data until it finds the mode of distribution.

## 2.3 Bandwidth selection in Mean-shift Algorithm

There are two method for selecting the bandwidth in mean-shift i.e fixed bandwidth and variable bandwidth. The variable bandwidth shows the superiority over the fixed bandwidth. When there is fixed bandwidth, then the effectiveness and correctness cannot be checked. By fixing the value of mean-shift, the optimum value of bandwidth cannot be found where it shows the better results. [13]. We obtain the best equally likelihood modes using variable bandwidth.

There are many people who fix the bandwidth at 0.8 and obtained the results for the same bandwidth [14] but some people optimized the bandwidth to obtain the better results.

There is some cases in mean shift-based approach for local bandwidth selection in the multimodal, multivariate case where we found that the variable bandwidth is used. It is found that the best bandwidth maximizes the objective function[15]. There are some applications of Mean Shift algorithm such as object tracking, image and video segmentation which need a variable bandwidth for their application [16].

# **Chapter-3**

## **Mean-shift**

## **Algorithm**

## **Mean –shift Mode :**

It locates the densest region among the whole distribution, therefore it is known as Mode seeking Algorithm.

## **Mean Shift Algorithm :**

It is a tool of finding modes in set of data samples. It can be used for PDF (Probability Density Function). In this, algorithm find the region of high density of distribution and it find the centroid or centre of mass of that distribution. It shift to centre of mass and similarly do again and again, to find the densest regions of the given distribution. It basically separate the clusters present in the This is proved for discrete data the convergence of recursive Mean –Shift procedure to the nearest stationary point of underlying density function. It performs two vision tasks, discontinuity preserving smoothing and image segmentation.

It includes the various terms which are explained below:

## **Mean-Shift Vector :**

It's toward the direction of maximum increase of density, Data Point & approximate the location of Mean of data. It basically in the direction of density estimated. It computes the density iteratively.

## **Task :**

Estimate the exact location of mean of data by determining the shift vector from the initial mean.

$$M(y) = \left[ \frac{1}{n} \sum_{i=1}^n x_i \right] - y_o \quad \text{Where } y_o = \text{Initial estimate value}$$

## **Mean-Shift (Weighted) :**

Weights are determined using kernels(Masks) : Uniform, Gaussian or Epanechnikov

$$M_h(y_o) = \frac{\sum_{i=1}^{n_x} w_i(y_o) x_i}{\sum_{i=1}^{n_x} w_i(y_o)} - y_o$$

$n_x$  : No of data points

$y_o$  : Initial mean Location

$x_i$  : Data points

$h$  : Kernel Radius

The steps involved in the Mean-Shift algorithm:

- (a) Find the densest region of distribution.
- (b) Find Centroid or Centre of Mass .
- (c) Then shift to the centre of mass and similarly do again and again , as to find densest region .
- (d) This shows how the mean is shifting

### **Parametric Density Estimation (PDF(x))**

Data points are sampled from underlying PDF(x)

$$PDF(x) = \sum_i c_i e^{-\frac{(x-x_i)^2}{2\sigma_i^2}}$$

### **Kernel Density Estimation**

A function of some finite number of data points  $x_1, \dots, x_n$

$$P(x) = \frac{1}{n} \sum_{i=1}^n k(x - x_i)$$

### **Different type of kernels**

#### **(i) Epanechnikov Kernel**

$$K_e(x) = \begin{cases} c(1-|x|^2) & |x| \leq 1 \\ 0 & \text{Otherwise} \end{cases}$$

#### **(ii) Uniform Kernel**

$$K_u(x) = \begin{cases} c & |x| \leq 1 \\ 0 & \text{Otherwise} \end{cases}$$

#### **(iii) Normal Kernel**

$$K_N(x) = c \cdot \exp\left(-\frac{1}{2}|x|^2\right)$$



## Relation between kernel Density Estimation and Meanshift

### Radially symmetric kernel

$$K(x) = ck(\|x\|^2) \quad \text{where } k = \text{Profile}$$

$$P(x) = \frac{1}{n} \sum_{i=1}^n k(\|x - x_i\|^2)$$

### Kernel Density Estimate

$$\nabla P(x) = \frac{1}{n} \sum_{i=1}^n \nabla k(\|x - x_i\|^2)$$

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n (x - x_i) k'(\|x - x_i\|^2) \dots\dots\dots(1)$$

$$\because k'(x) = g(x)$$

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n x_i g(\|x - x_i\|^2) - \frac{1}{n} 2c \sum_{i=1}^n x g(\|x - x_i\|^2) \dots\dots\dots(2)$$

$$\nabla P(x) = \frac{1}{n} 2c \sum_{i=1}^n g(\|x - x_i\|^2) \left[ \frac{x_i g(\|x - x_i\|^2)}{g(\|x - x_i\|^2)} - x \right] \dots\dots\dots(3)$$

$$\nabla P(x) = \frac{c}{n} \sum_{i=1}^n \nabla k_i \dots\dots\dots(4)$$

$$\nabla P(x) = \frac{c}{n} \sum_{i=1}^n g_i \dots\dots\dots(5)$$

$$\nabla P(x) = \left[ \frac{\sum_{i=1}^n x_i g_i}{\sum_{i=1}^n g_i} - x \right] \dots\dots\dots(6)$$

$$\cdot \nabla P(x) = \frac{c}{n} \sum_{i=1}^n g_i \times m(x) \dots\dots\dots (7)$$

Therefore, Meanshift Can be expressed as

$$m(x) = \frac{\nabla P(x)}{\frac{c}{n} \sum_{i=1}^n g_i} \dots\dots\dots(8)$$

# **Chapter-4**

## **Proposed Auto-Segmentation Using Mean-Shift Algorithm**

We have proposed the Auto-segmentation using Mean-shift Algorithm , one of the modified version of Mean-Shift algorithm

These algorithm are modified in different aspects :

- We have passed the images
- Mean-shift algorithm is applied on the image.
- Segmentation of image is performed.

#### 4.1 Automization of Mean-Shift

In the automization of Mean-Shift the program is automized and it includes the following steps

Initialize search with BW= 0.01, clusters= 02

REPEAT

Find mean-shift modes

Find entropy of modes as per formula

$$H_1 = -\sum_{i=1}^n -p_i \log p_i \dots\dots\dots (9)$$

$$H_2 = \sum_{i=1}^n -p_i e^{-p_i} \dots\dots\dots (10)$$

These equations are provided in [16]

UNTIL max-limit of BW= 1.5 clusters = 10 is reached for small increment of BW = 0.01, cluster = 1

## 4.2 Proposed Algorithm:

In this, we proposed an algorithm by using mean shift mode algorithm and we apply the algorithm on the image data which has been applied on linear data earlier .

These algorithms include the following steps:

- (i) Initially image is read.
- (ii) To apply the code , we convert the image into one –dimensional array .After converting into one dimensional array ,the mean shift mode algorithm is applied on the that one-dimensional image .
- (iii) In this step, the result obtained after applying the mean-shift algorithm. We just pass that result obtain in order to compare with the ground truth .
- (iv) For comparing the images with ground truth, we convert the both ground truth result and results obtained into one –d and then compare the image.
- (v) After comparing the image we calculate the precision , recall and f-score value for that images

By applying the algorithm, we came to know that it shows better results .

# **Chapter – 5**

## **RESULTS**

**&**

## **DISCUSSION**

Initially , Meansshift algorithm is used to cluster the linear data and it form the cluster and separate the data using the algorithm .

### (a) On Linear Data

We have applied the mean shift mode code on linear data and we found that the data get separated on the basis of different label . In this , the clusters are formed on the basis of seeds and the one cluster get merged with other cluster in order to obtain the single cluster on the basis of the label

#### Result :

➤ [a,b,c]= findModes(pts,2,.1)

Visiting cluster 1

Merged cluster 1 with 1

Merged cluster 2 with 1

a = 100.9998 101.9998

b = 1

c = 1 1 1 1 1

➤ [a,b,c]= findModes(pts,3,.1)

Visiting cluster 1

Merged cluster 1 with 1

Merged cluster 2 with 1

Merged cluster 3 with 1

a = 2.6412 2.9200

b = 1

c = 1 1 1 1 1

➤ [a,b,c]= findModes(pts,4,.1)

Visiting cluster 1

Merged cluster 1 with 1

Merged cluster 2 with 1

Merged cluster 4 with 1

Visiting cluster 3

Merged cluster 3 with 3

a = 2.6405 2.9192

100.9998 101.9998

b = 0.6000

0.4000

c = 1 1 2 2 1

## (b) On Image Data

This `findmodes` function shows different result for different value of points , and no of seeds and threshold .

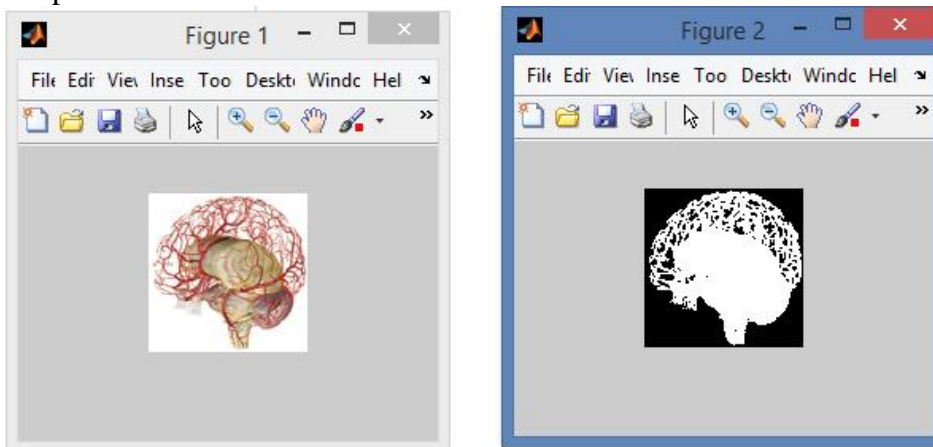
Points are provided by reading the image using `imread` function

Image is kept same for different value of no seeds and threshold values .

## RESULTS OF MEAN-SHIFT FOR DIFFERENT IMAGES

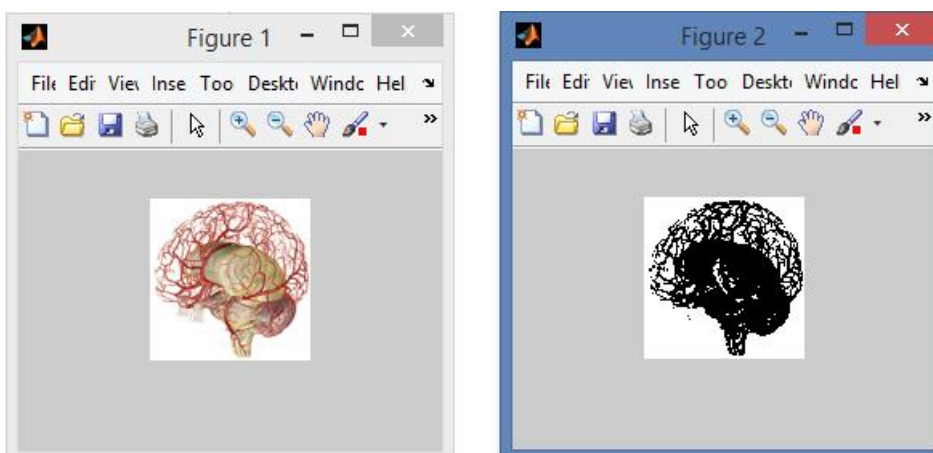
1. `[a,b,c ]= findModes(pts,2,.2);`

Output :



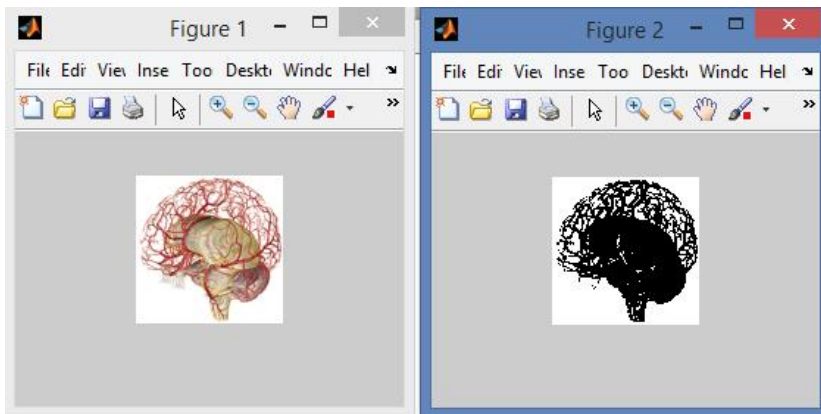
In this case the two value of  $c = 1, 2$  is created .

2. `[a,b,c ]= findModes(pts,2,.3);`



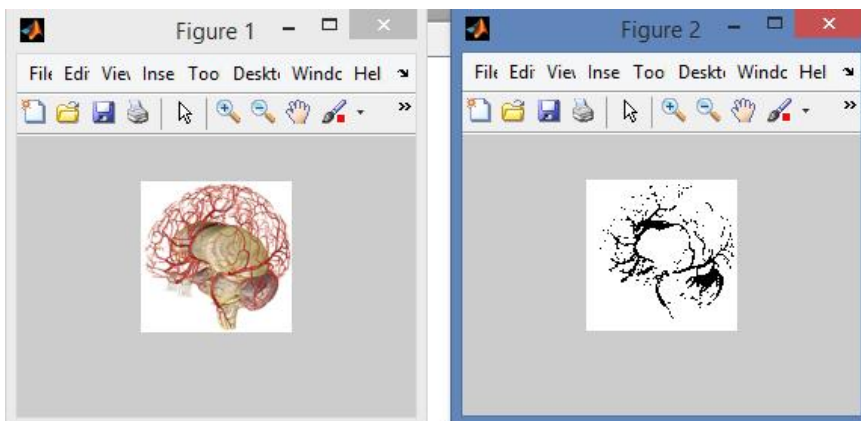


3. `[a,b,c]= findModes(pts,2,.4);`



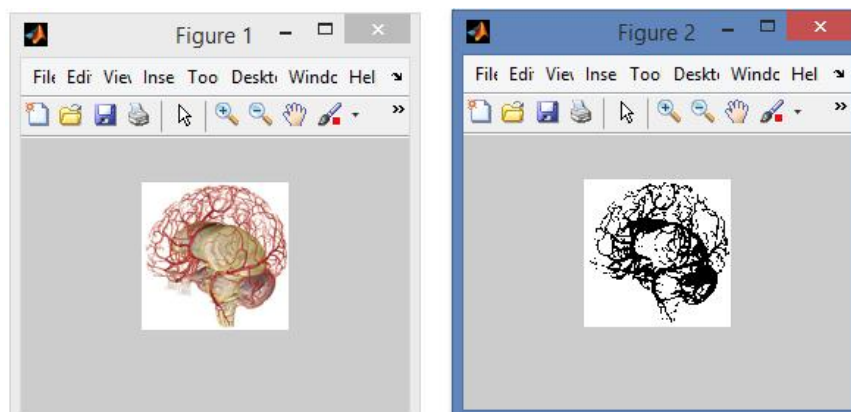
In this case the two value of  $c = 1, 2$  is created

4. `[a,b,c]= findModes(pts,2,.6);`

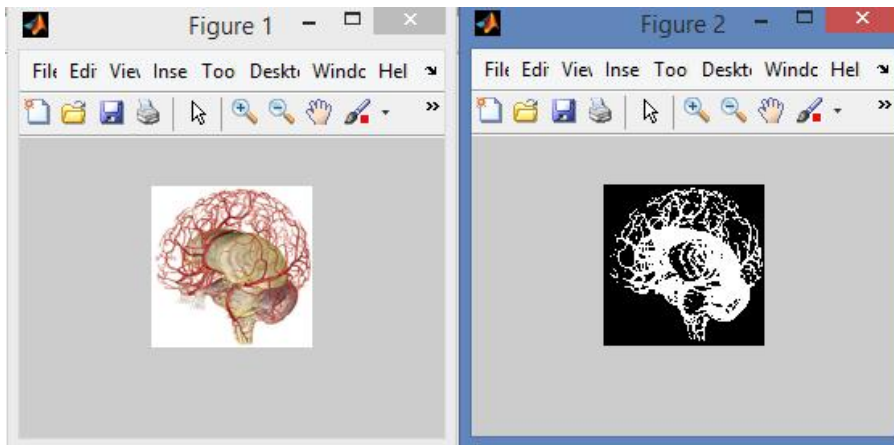


In this case the two value of  $c = 1, 2$  is created

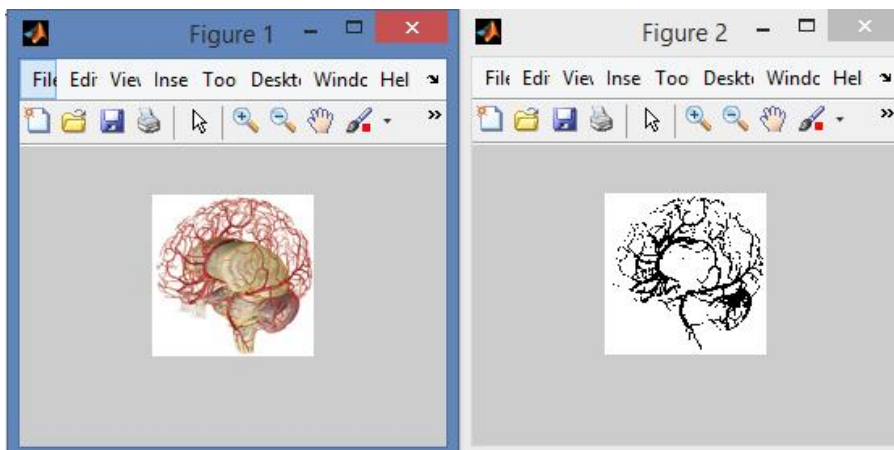
5. `[a,b,c]= findModes(pts,2,.7);`



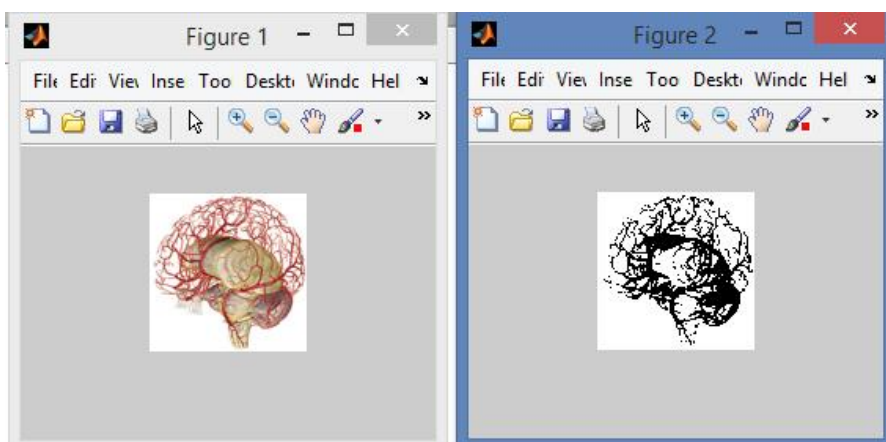
6. `[a,b,c]= findModes(pts,2,.12);`



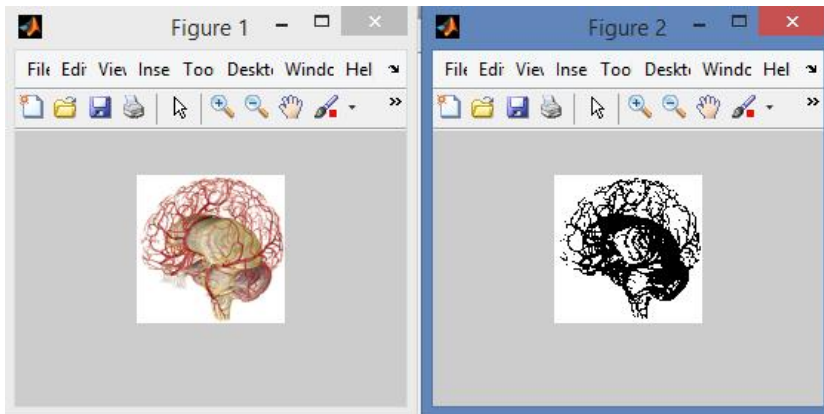
7. `[a,b,c]= findModes(pts,2,.25);`



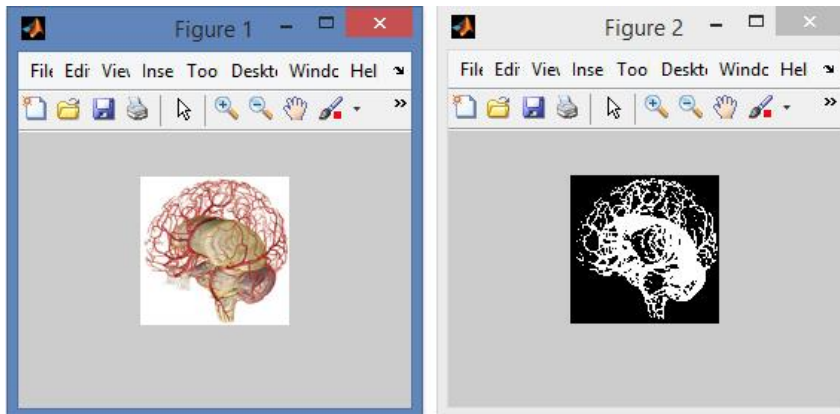
8. `[a,b,c]= findModes(pts,2,.30);`



9. `[a,b,c ]= findModes(pts,2,.60);`

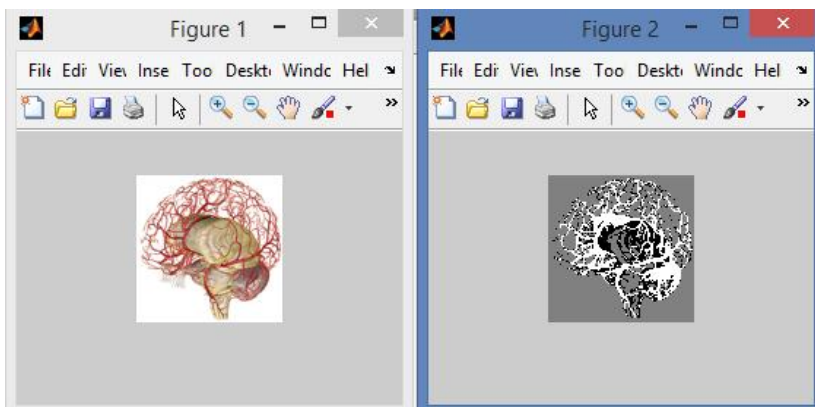


10. `[a,b,c ]= findModes(pts,3,.3);`



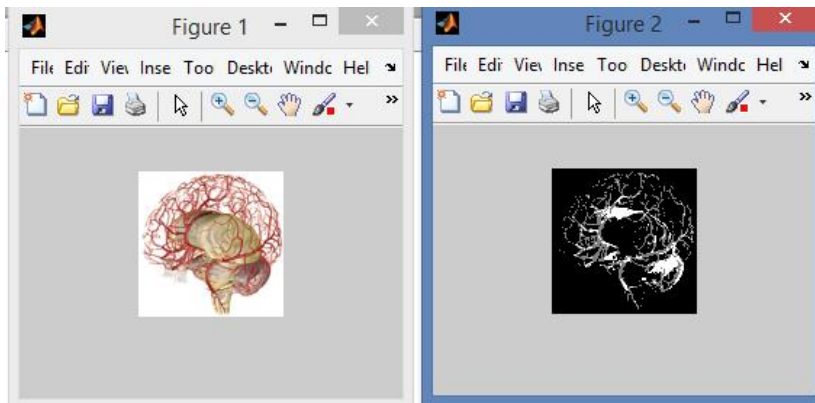
In this case the two value of  $c = 1, 2$  is created

11. `[a,b,c ]= findModes(pts,3,.4);`



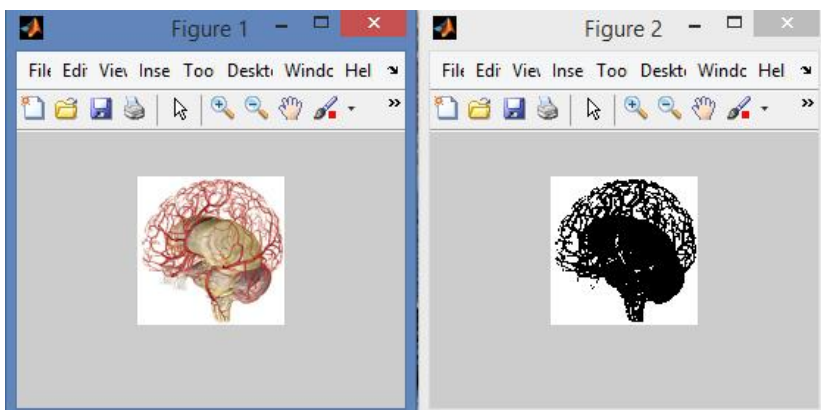
In this case the two value of  $c = 1, 2, 3$  is created

12. `[a,b,c]= findModes(pts,3,.5);`



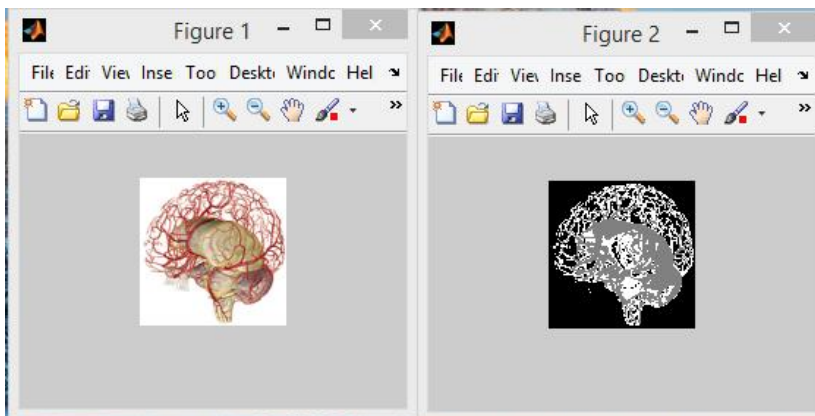
In this case the two value of  $c = 1, 2, 3$  is created

13. `[a,b,c]= findModes(pts,3,.6);`



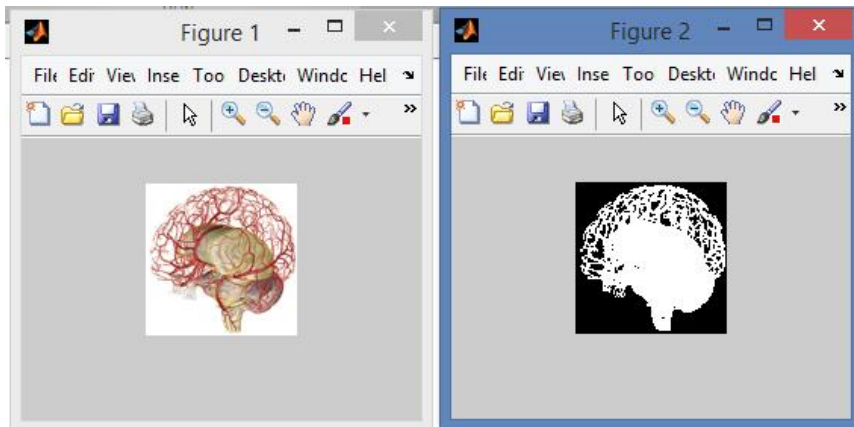
In this case the two value of  $c = 1, 2$  is created

14. `[a,b,c]= findModes(pts,3,.7);`



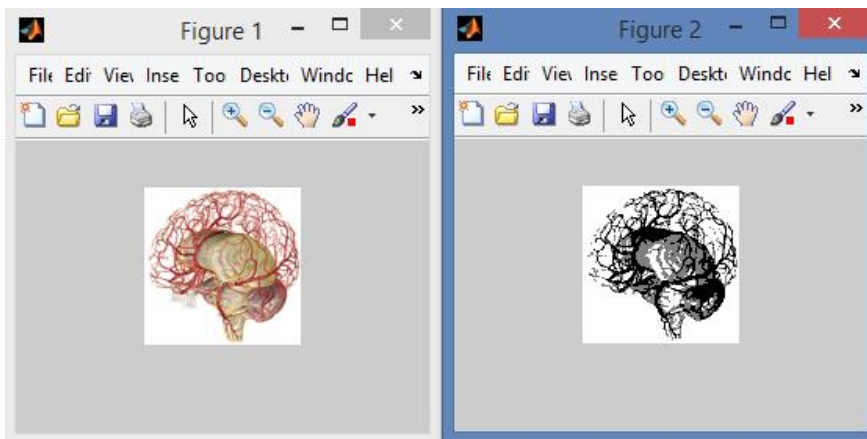
In this case the two value of  $c = 1, 2, 3$  is created

15. `[a,b,c]= findModes(pts,3,.8);`



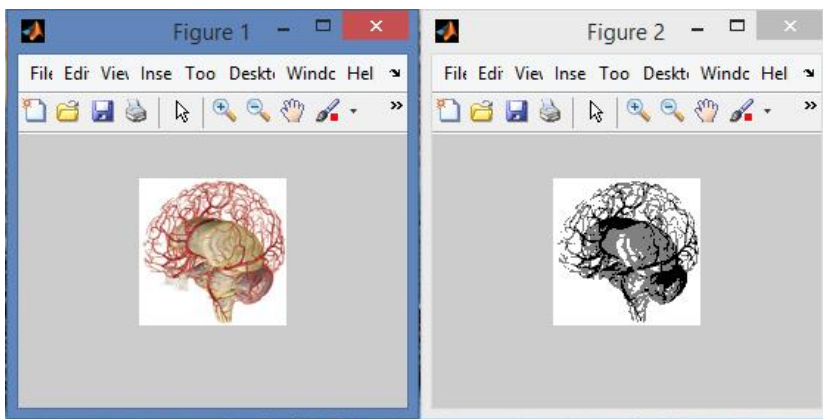
In this case the two value of  $c = 1, 2$  is created

16. `[a,b,c]= findModes(pts,3,.9);`



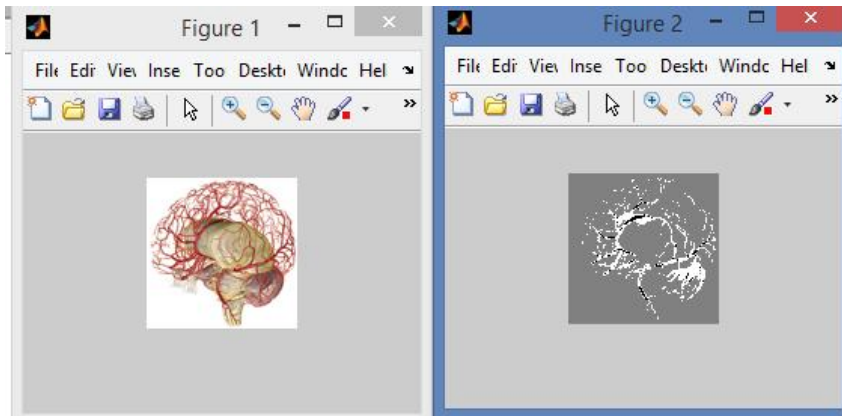
In this case the two value of  $c = 1, 2, 3$  is created

17. `[a,b,c]= findModes(pts,3,.10);`



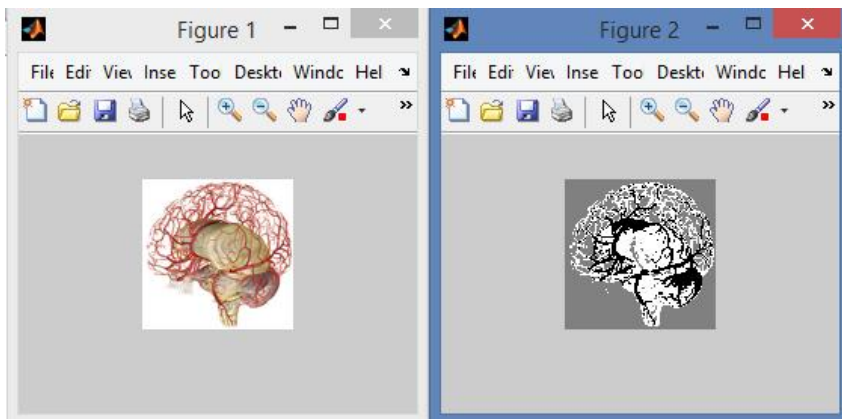
In this case the two value of  $c = 1, 2, 3$  is created

18. `[a,b,c]= findModes(pts,3,.20);`



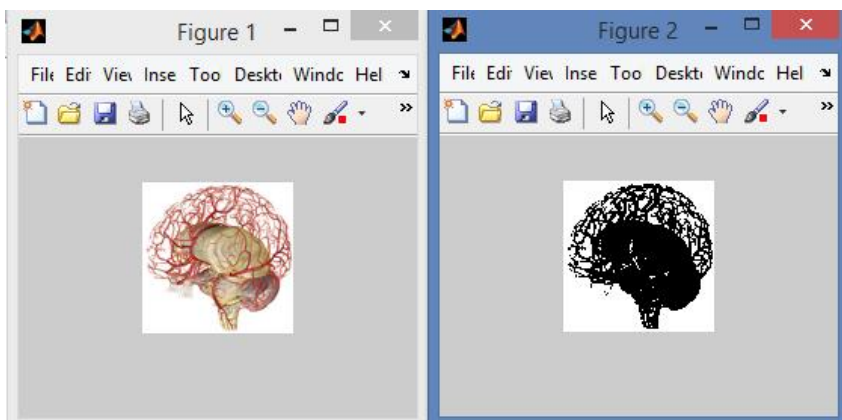
In this case the two value of  $c = 1, 2, 3$  is created

19. `[a,b,c]= findModes(pts,3,.30);`



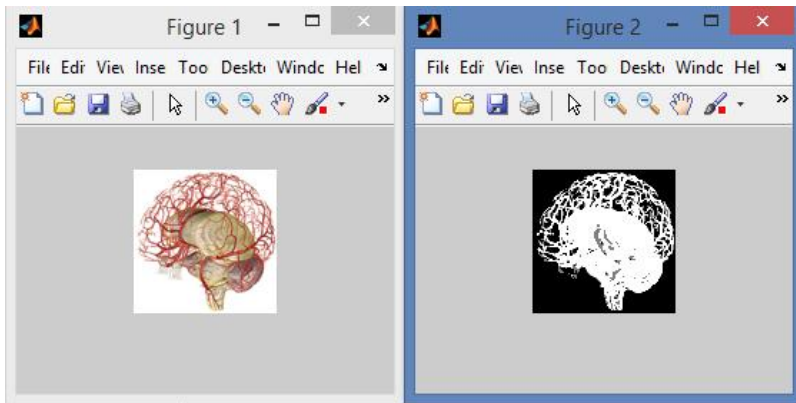
In this case the two value of  $c = 1, 2, 3$  is created

20. `[a,b,c]= findModes(pts,3,.40);`



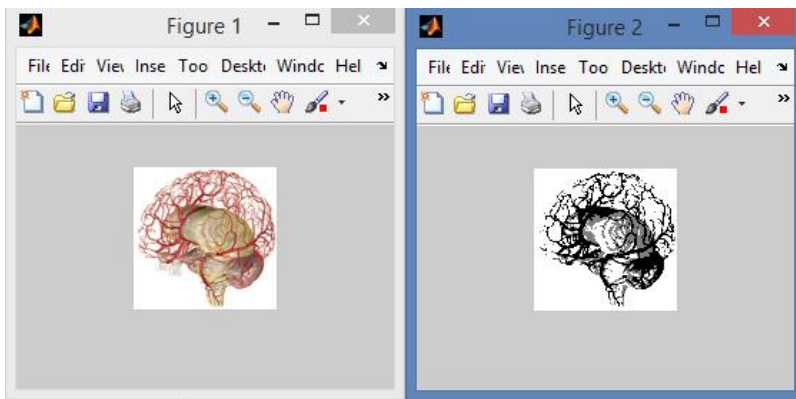
In this case the two value of  $c = 1, 2$  is created

21. `[a,b,c]= findModes(pts,3,.60);`



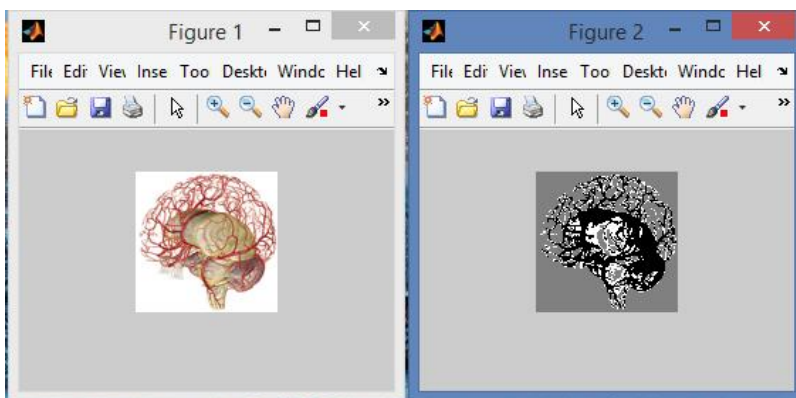
In this case the two value of  $c = 1, 2, 3$  is created

22. `[a,b,c]= findModes(pts,3,.90);`



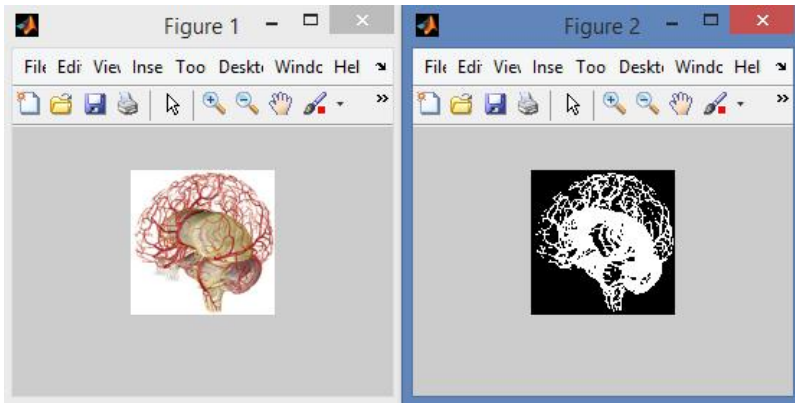
In this case the two value of  $c = 1, 2, 3$  is created

23. `[a,b,c]= findModes(pts,3,.200);`



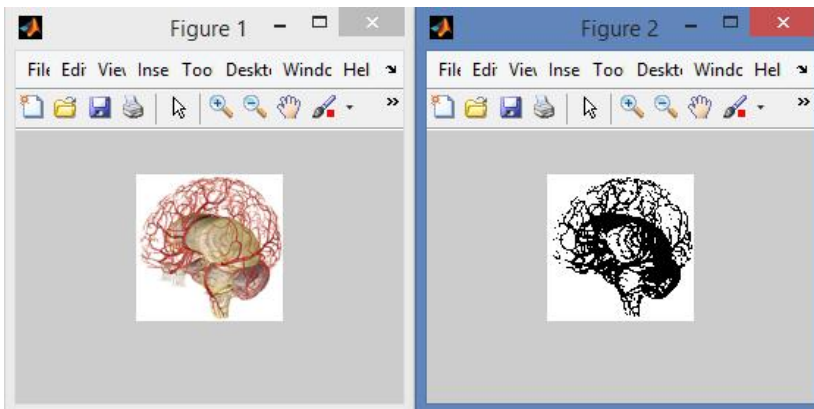
In this case the two value of  $c = 1, 2, 3$  is created

24. `[a,b,c]= findModes(pts,3,.300);`



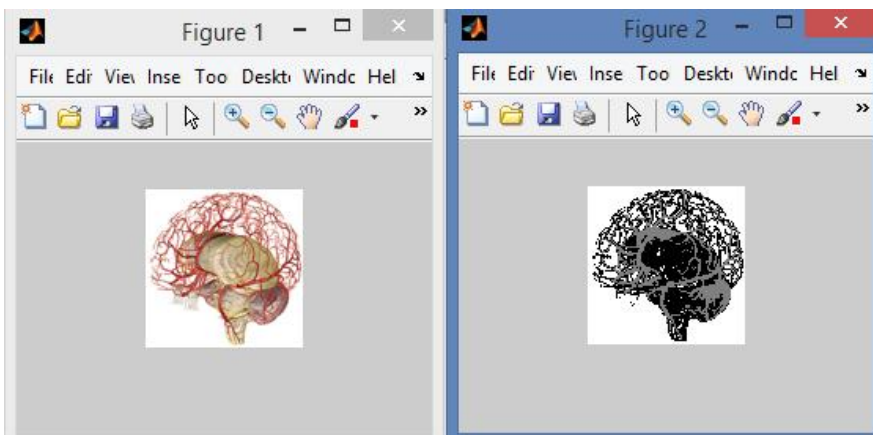
In this case the two value of  $c = 1, 2$  is created

25. `[a,b,c]= findModes(pts,3,.400);`



In this case the two value of  $c = 1, 2$  is created

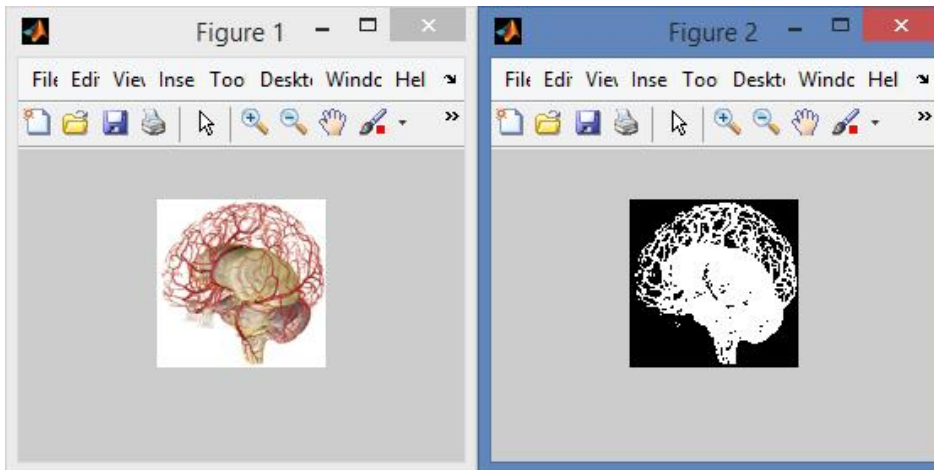
26. `[a,b,c]= findModes(pts,3,.700);`



In this case the two value of  $c = 1, 2, 3$  is created

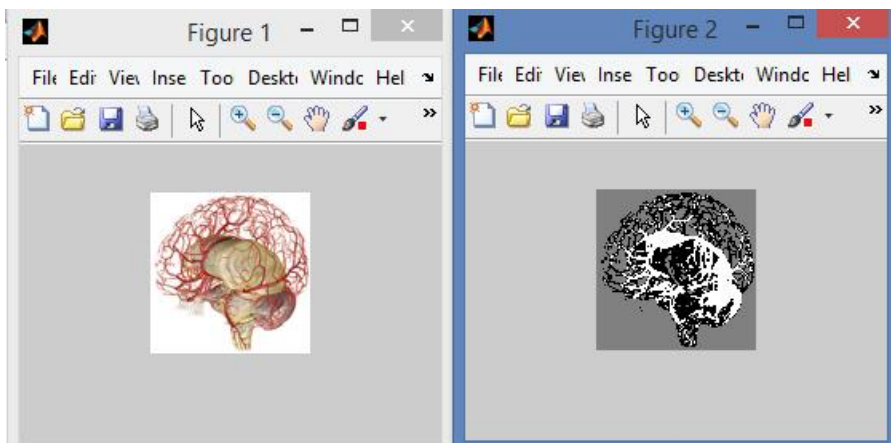


27. `[a,b,c]= findModes(pts,4,.3);`



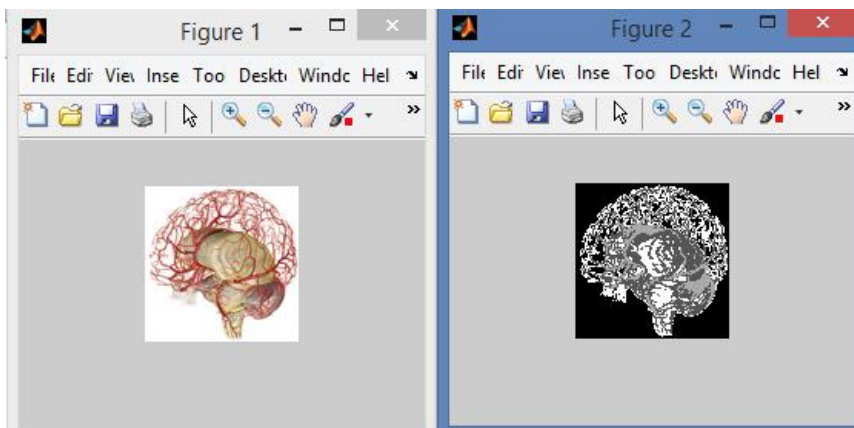
In this case the two value of  $c = 1, 2, 3, 4$  is created

28. `[a,b,c]= findModes(pts,4,.90);`



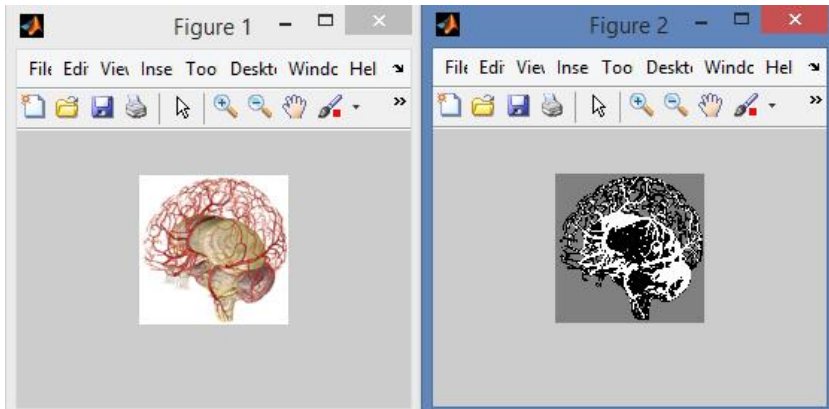
In this case the two value of  $c = 1, 2, 3$  is created

29. `[a,b,c]= findModes(pts,4,.400);`



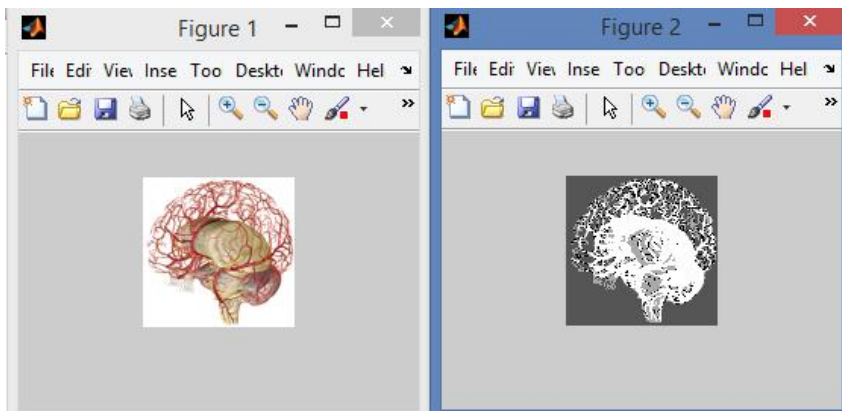
In this case the two value of  $c = 1, 2, 3, 4$  is create

30. `[a,b,c]= findModes(pts,5,.100);`



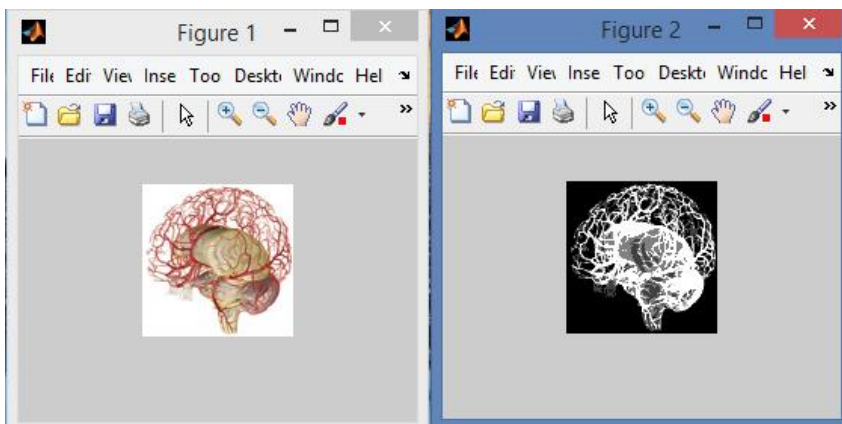
In this case the two value of  $c = 1, 2, 3$  is created

31. `[a,b,c]= findModes(pts,6,.100);`



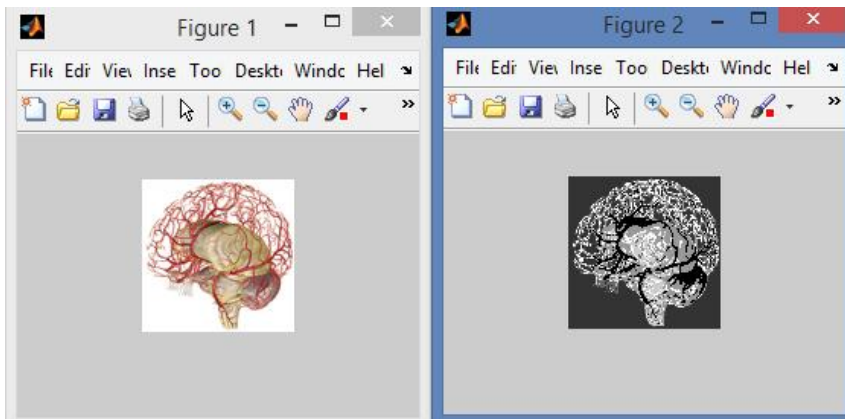
In this case the two value of  $c = 1, 2, 3, 4, 5, 6, 7$  is created

32. `[a,b,c]= findModes(pts,7,.100);`



In this case the two value of  $c = 1, 2, 3, 4, 5$  is created

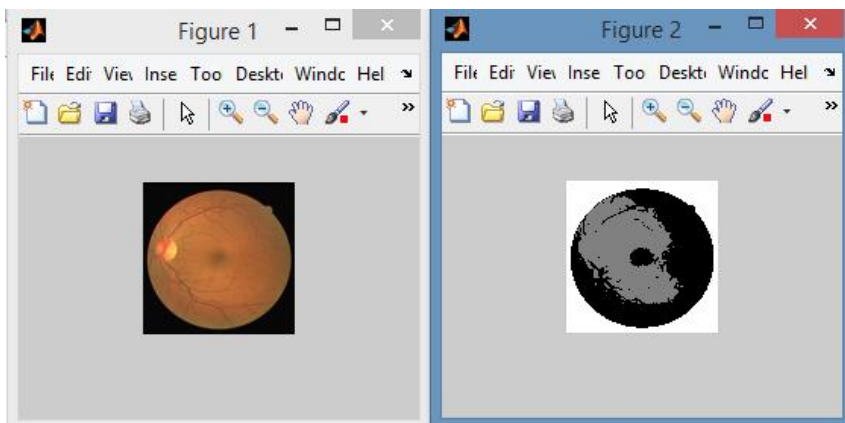
32. `[a,b,c]= findModes(pts,7,.100);`



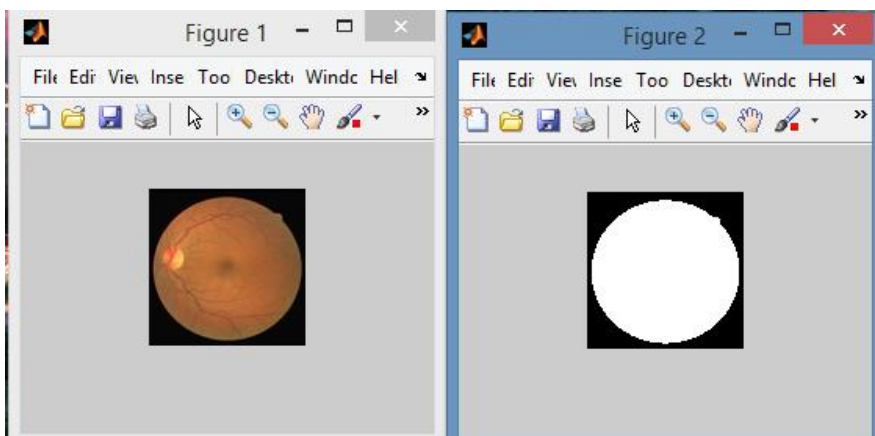
### (c) On Drive Database

When the findmodes function is used on drive database with different value of data

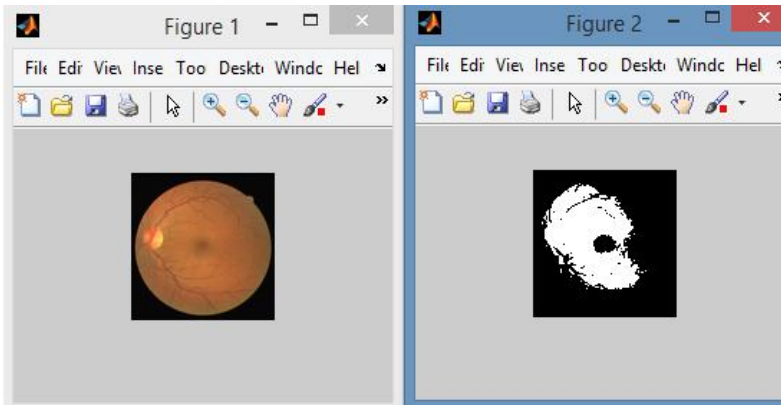
1. `[a,b,c]= findModes(pts,4,.4);`



2. `[a,b,c]= findModes(pts,5,.6);`



3. `[a,b,c]= findModes(pts,3,0.8 );`

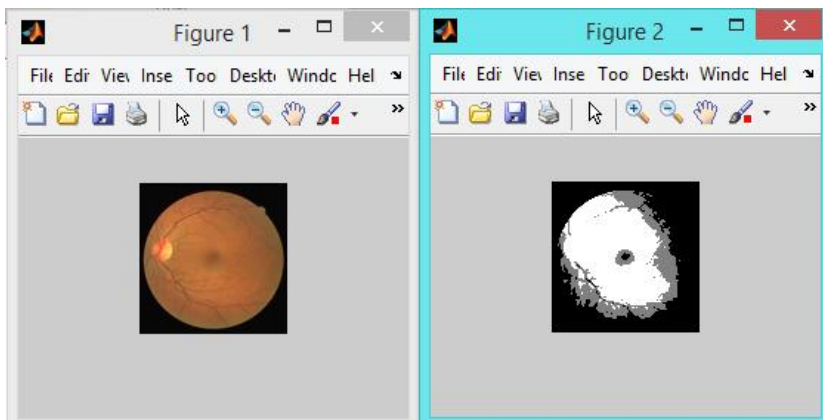


The best result for drive datasets are obtained at threshold value of 0.8 bandwidth which is the optimum value of bandwidth for the drive database. and this shows that the extraction of required information is maximum at this value of bandwidth . As, this shows the better results at bandwidth of 0.8 therefore , it is the optimum value of bandwidth for this database .

**d) Red.m file is compiled to extract the red component and various results are obtained for different value of seeds and threshold**

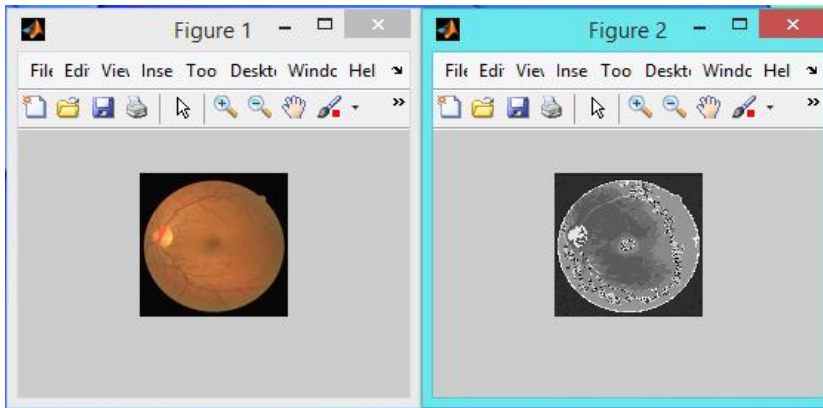
This is the result of red.m

1. `[a,b,c]= findModes(pts,3,.009);`



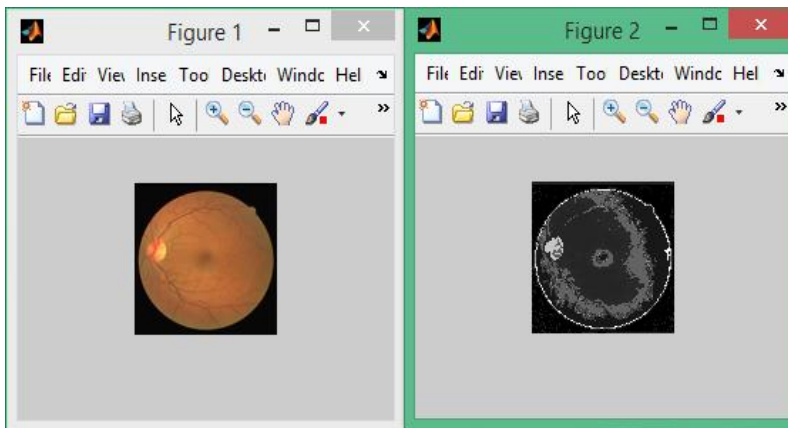
Min value of c is 1 & 3

2. `[a,b,c ]= findModes(pts,330,.0009);`

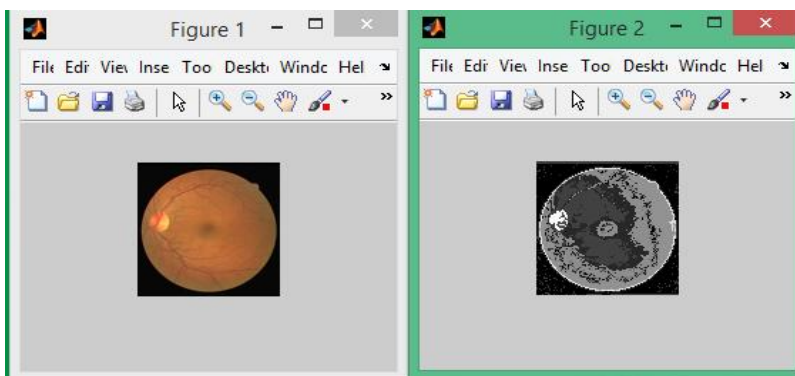


Min value of c is 2 and 41

2. `[a,b,c ]= findModes(pts,5000,.0009);`



3. `[a,b,c ]= findModes(pts,10000,.0009);`

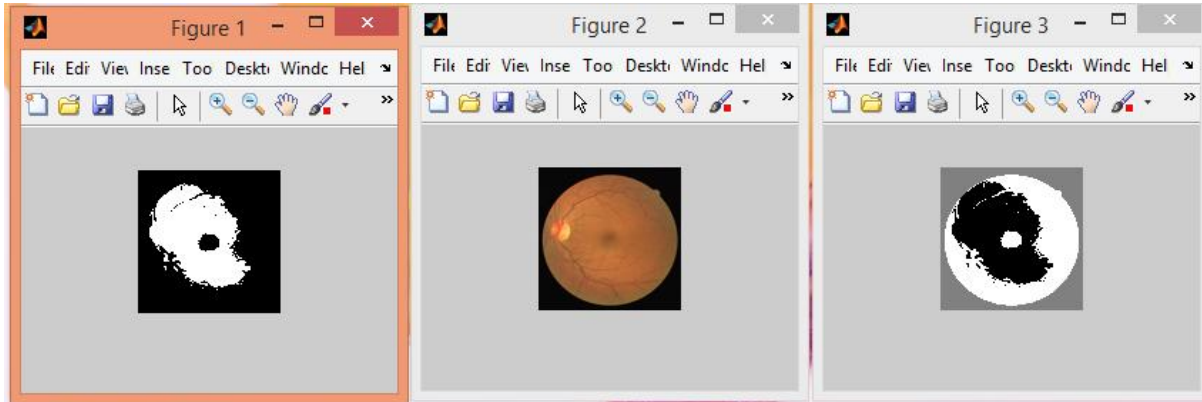


Min value of c is 1 & 71

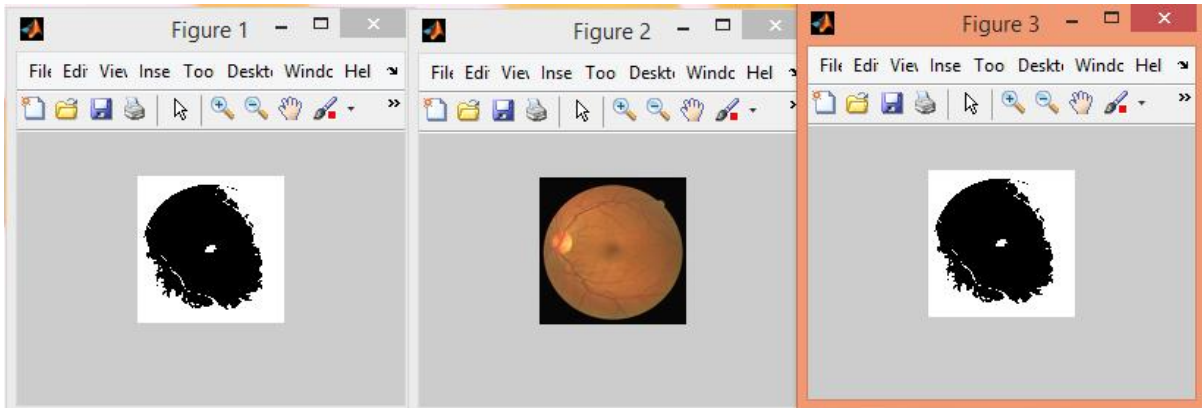
When it is compared for different value for different value of  $D$ .

1.

```
[a,b,c]= findModes(pts,3,.009);  
FOR D==1
```

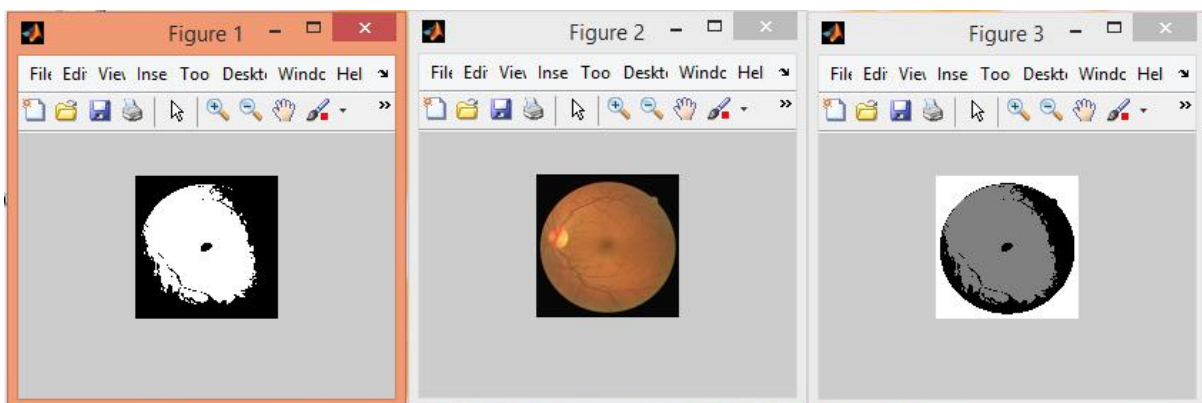


```
SE1=1  
D==2
```

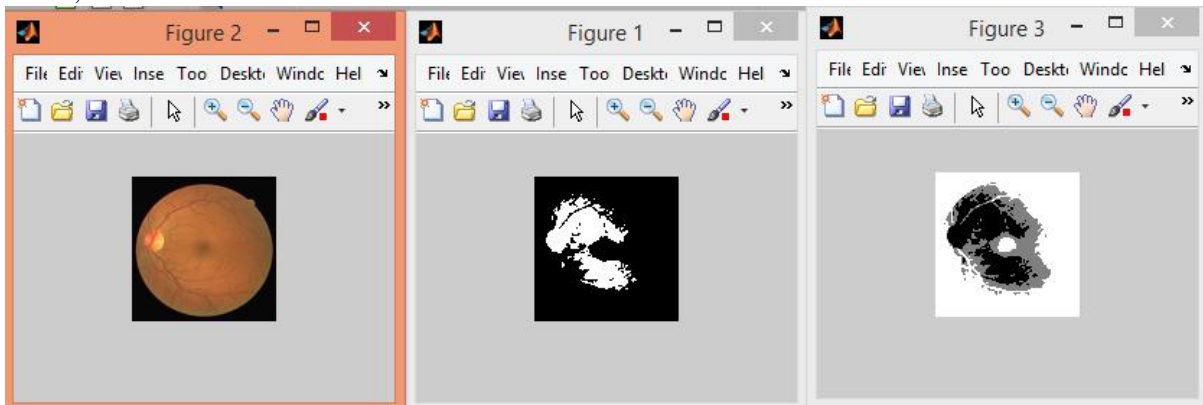


```
SE1=1
```

```
2. [a,b,c]= findModes(pts,3,.009);  
D==2 ,SE1=1
```

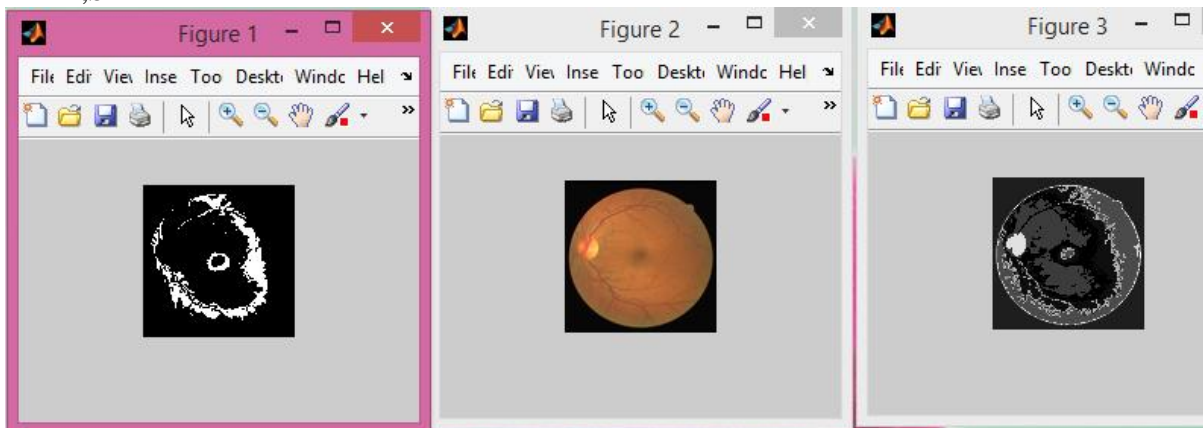


D==1,SE1=1

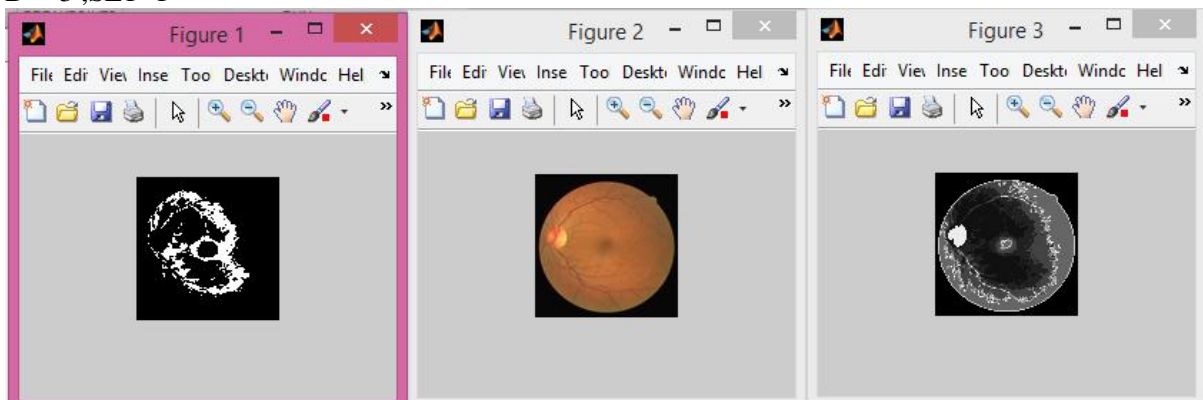


3. [a,b,c]= findModes(pts,330,.008);

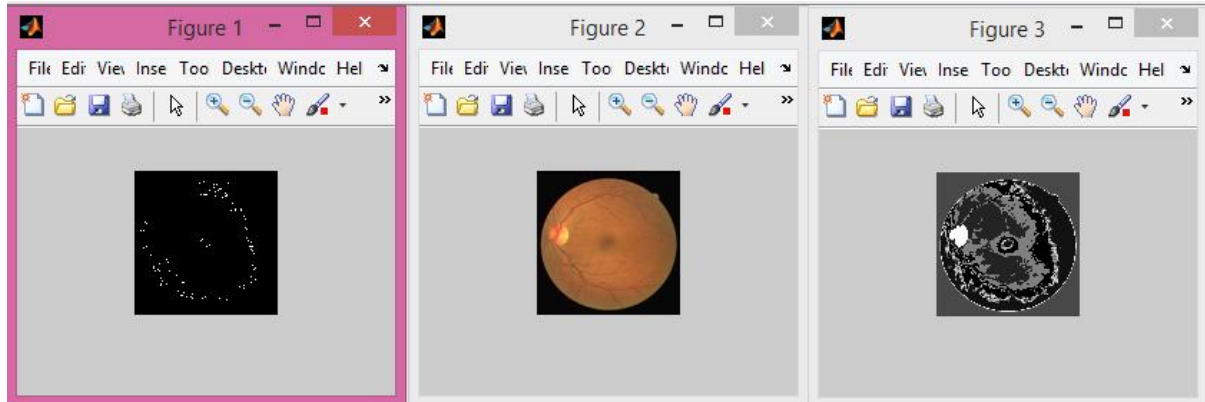
D==2,SE1=1



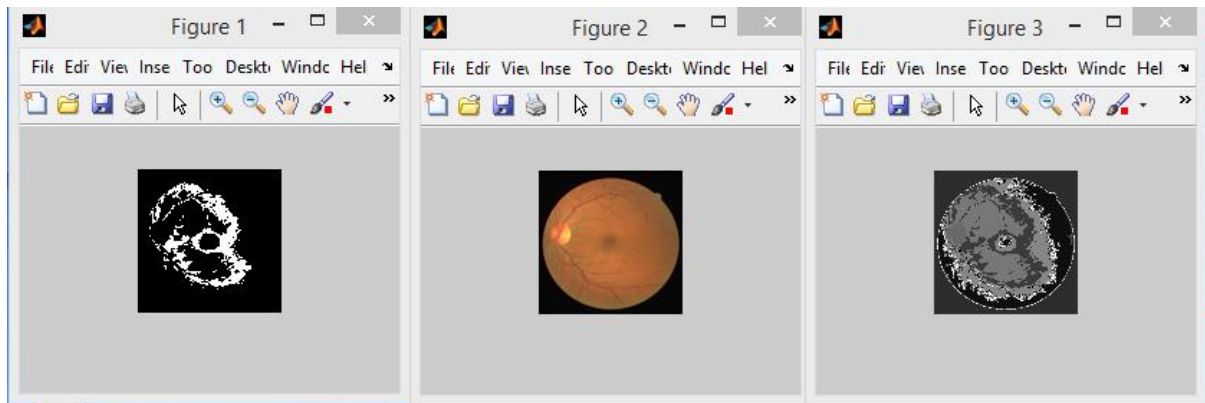
D==3,SE1=1



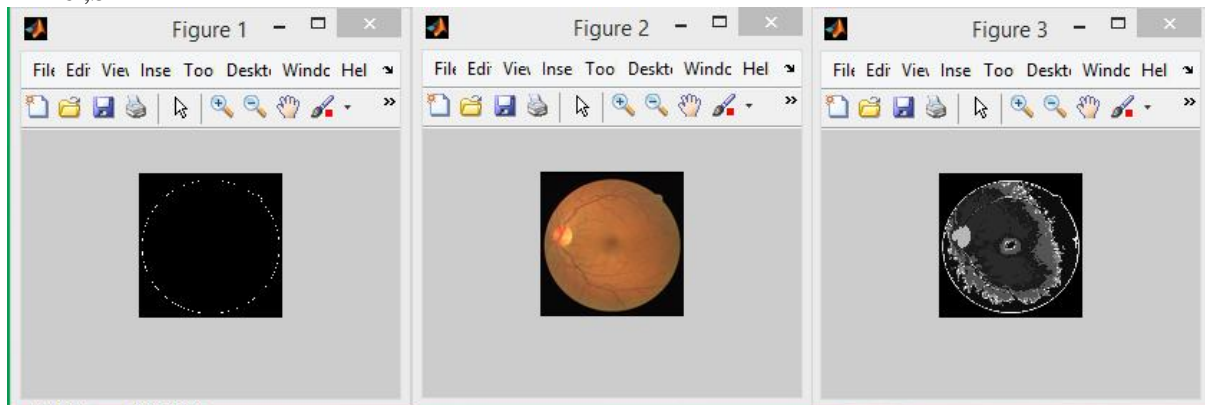
D==13,SE1=1



2. [a,b,c]= findModes(pts,330,.008);  
D==5,SE1=0

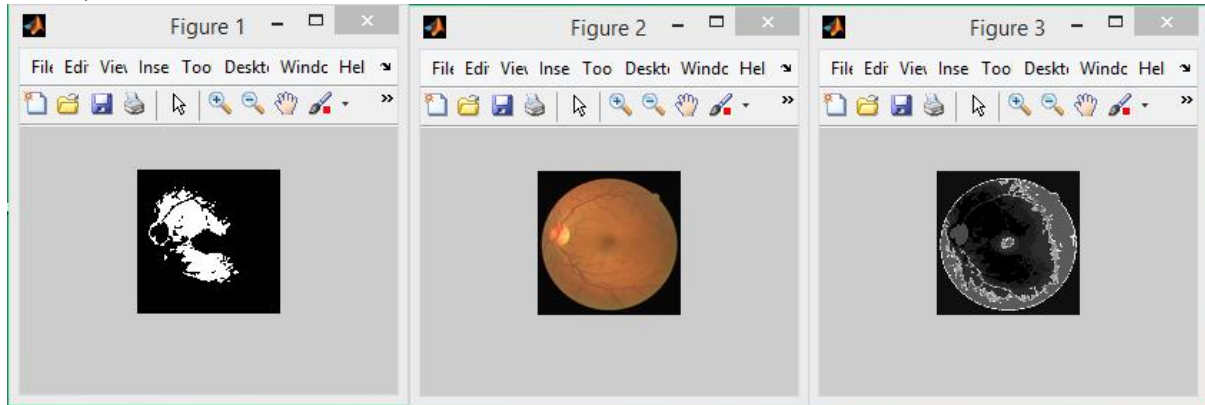


D==6,SE1=4

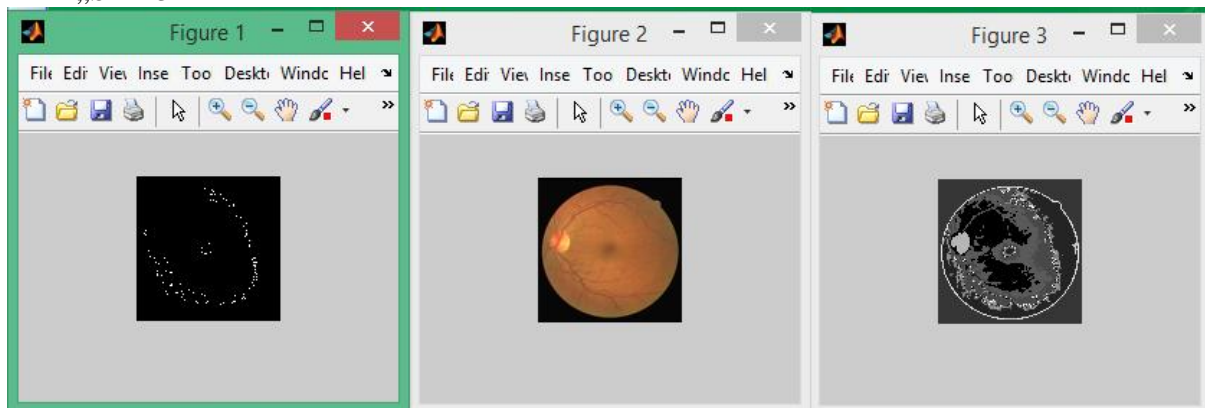




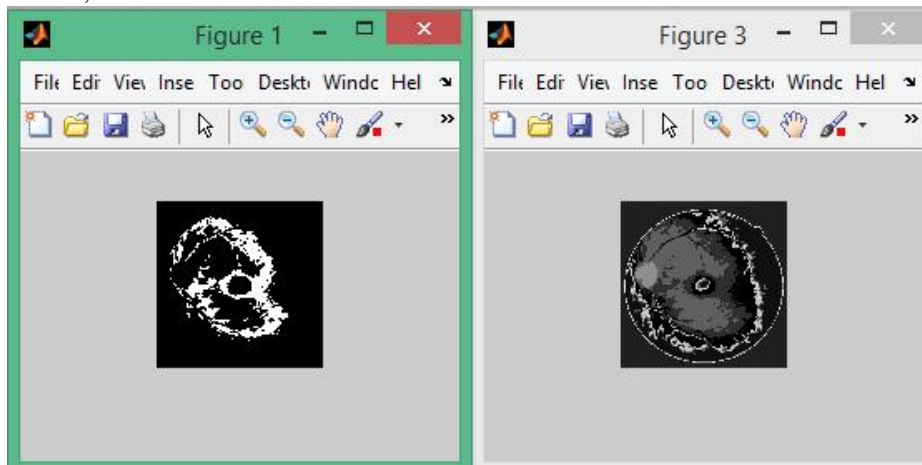
D==1 , SE1=0



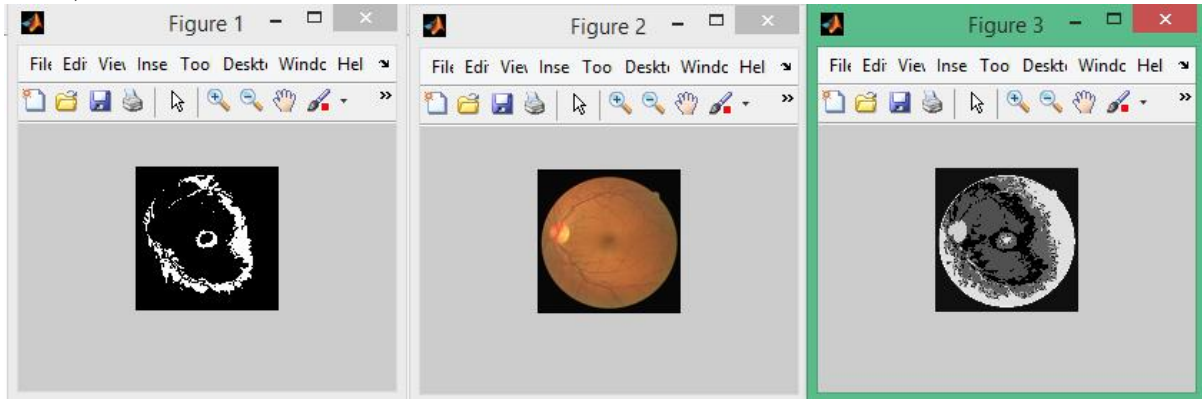
D==2,,SE1=3



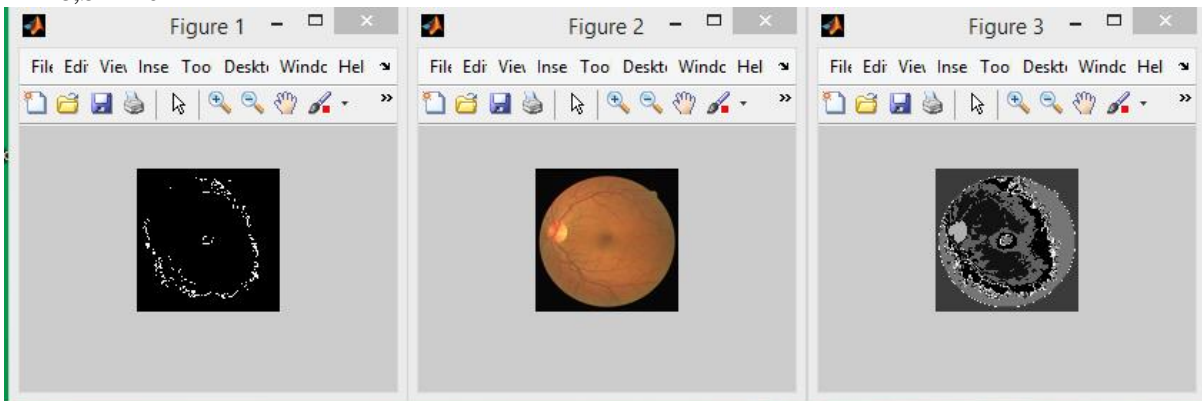
D==4,SE1=0



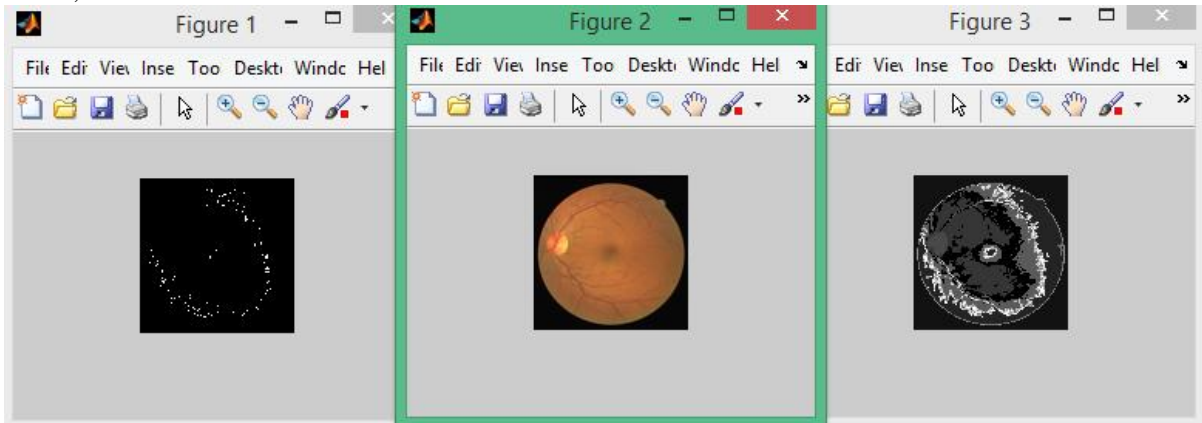
D==7,SE1=0



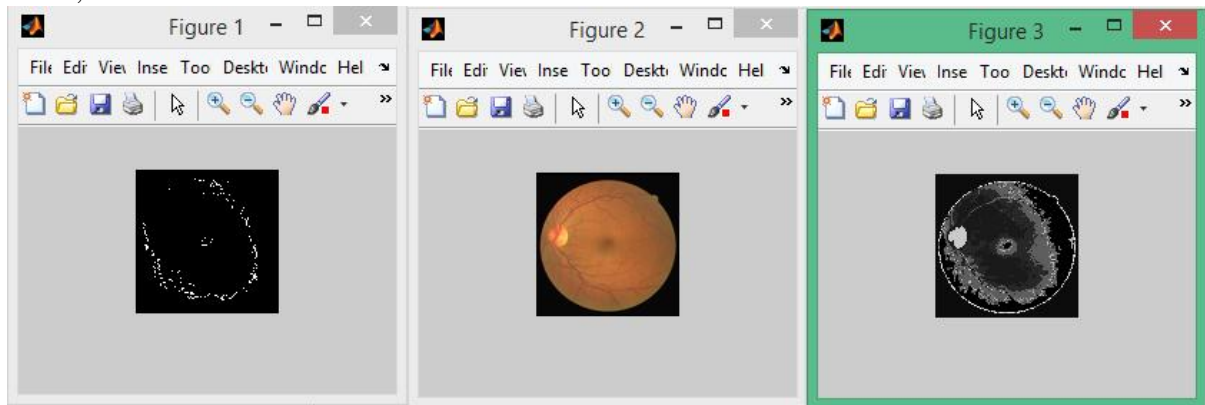
D==8,SE1=0



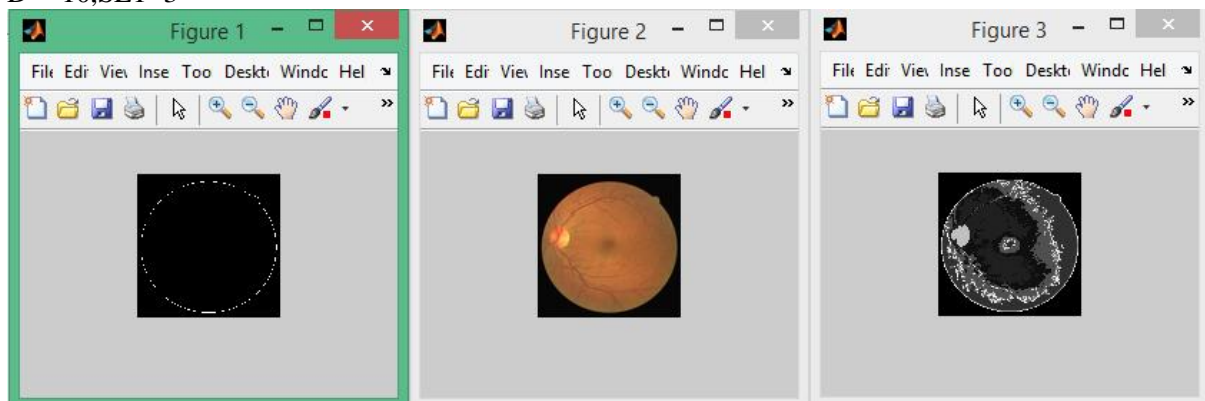
D==8,SE1=3



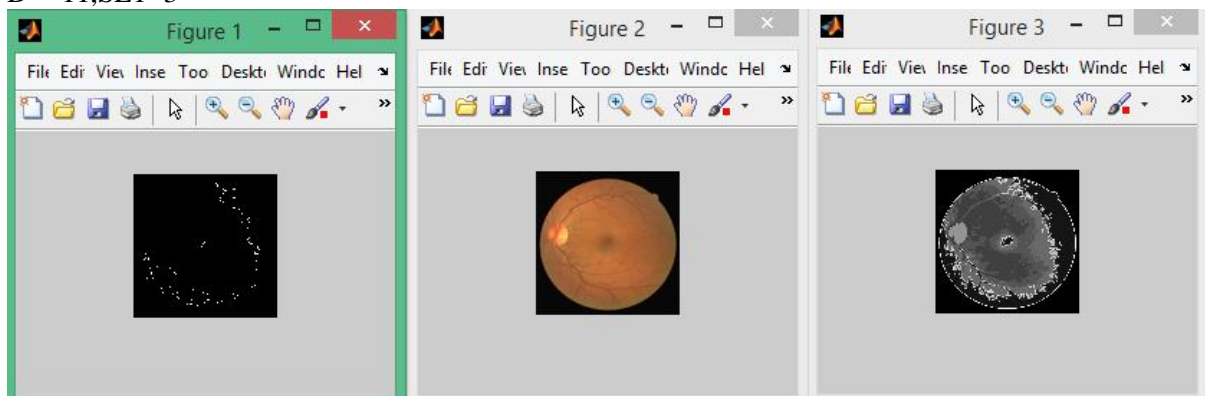
D==9,SE1=1



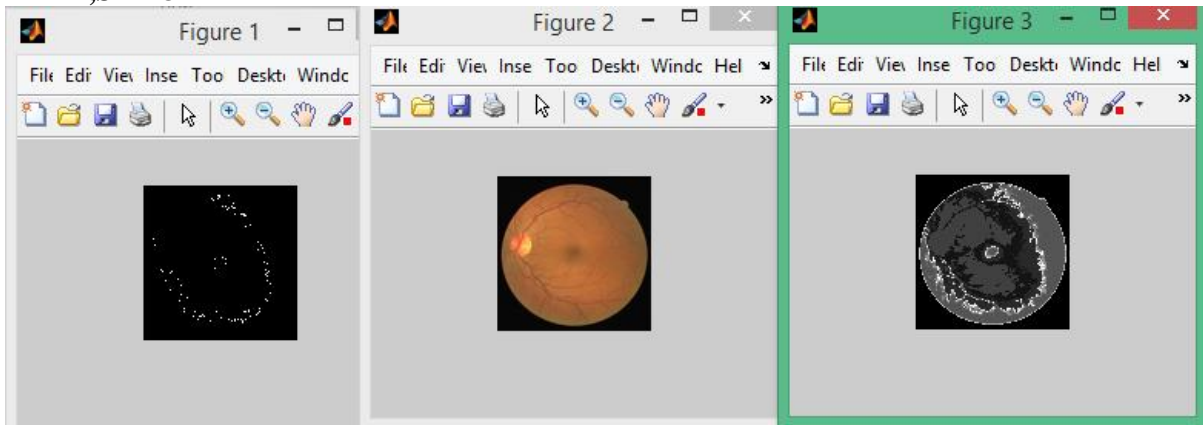
D==10,SE1=3



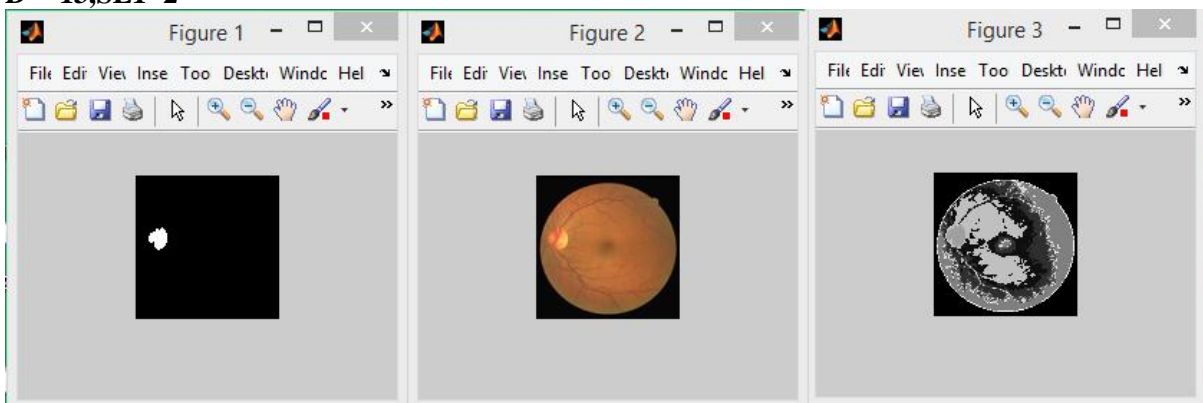
D==11,SE1=3



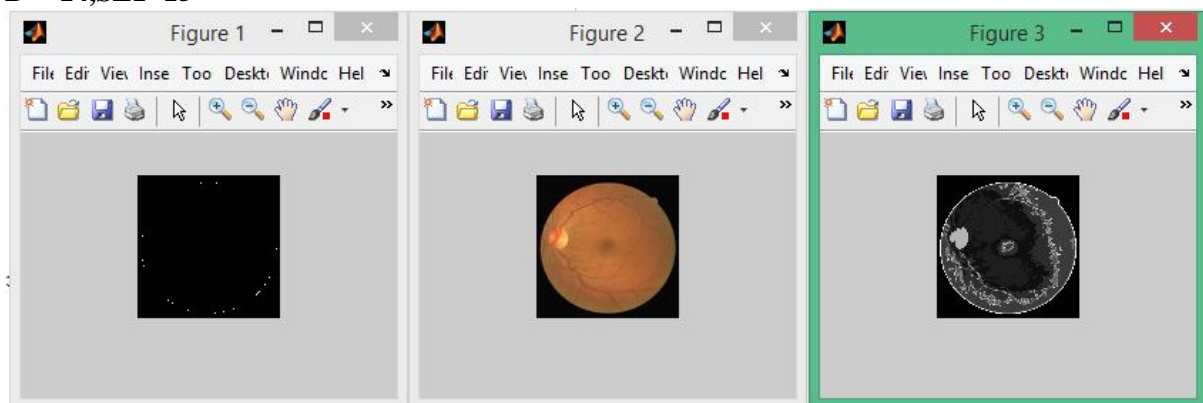
**D==12,SE1=3**



**D==13,SE1=2**



**D==14,SE1=15**



In this case , we extracted some of all color components (Red, Green, Blue) .

## CIELAB

This is also a colour model in which L, A, B are different components of the image. The vertical L\* axis represents Lightness, ranging from 0-100. These are at right angles to each other and cross each other in the centre, which is neutral (grey, black or white). They are based on the principal that a colour cannot be both red and green, or blue and yellow. The a\* axis is green at one extremity (represented by -a), and red at the other (+a). The b\* axis has blue at one end (-b), and yellow (+b) at the other. The centre of each axis is 0. A value of 0, or very low numbers of both a\* and b\* will describe a neutral or near neutral.

In theory there are no maximum values of a\* and b\*, but in practice they are usually numbered from -128 to +127 (256 levels).

The different components are extracted as follow

```
cform = makecform('srgb2lab');  
T = applycform(A,cform);  
%figure, imshow(T);  
L = T(:, :, 1); % Extract the L image.  
A_cie = T(:, :, 2); % Extract the A image.  
B = T(:, :, 3); % Extract the B image.
```

Then this A component is passed through the code to obtain the result

**In this we work on A instead of R component .**

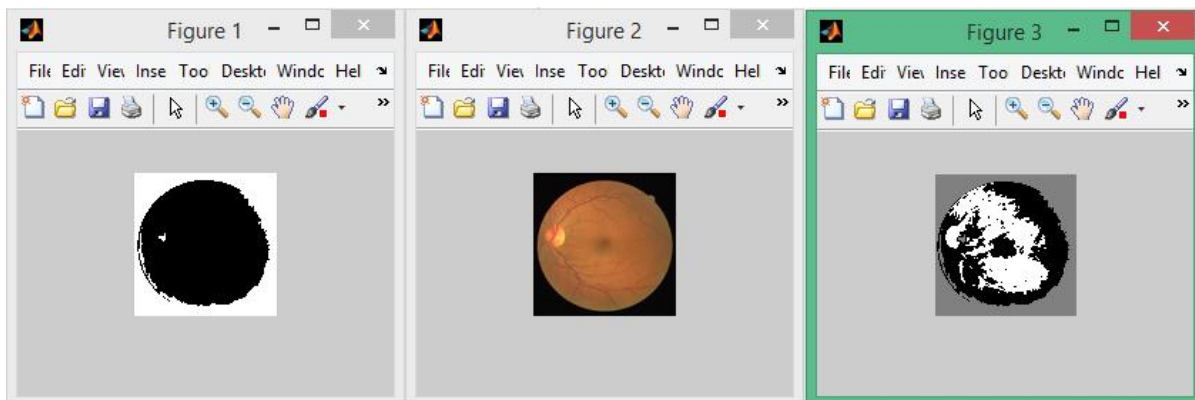
## CIELAB RESULTS :

### Results for Different value of D in CIELAB

[a,b,c]= findModes(pts,330,.008);

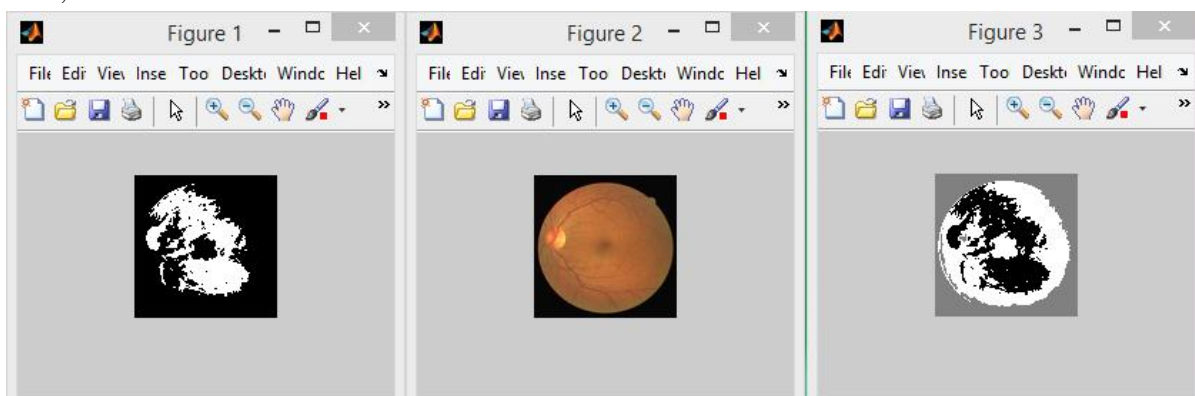
1.

D==2 , SE1=0



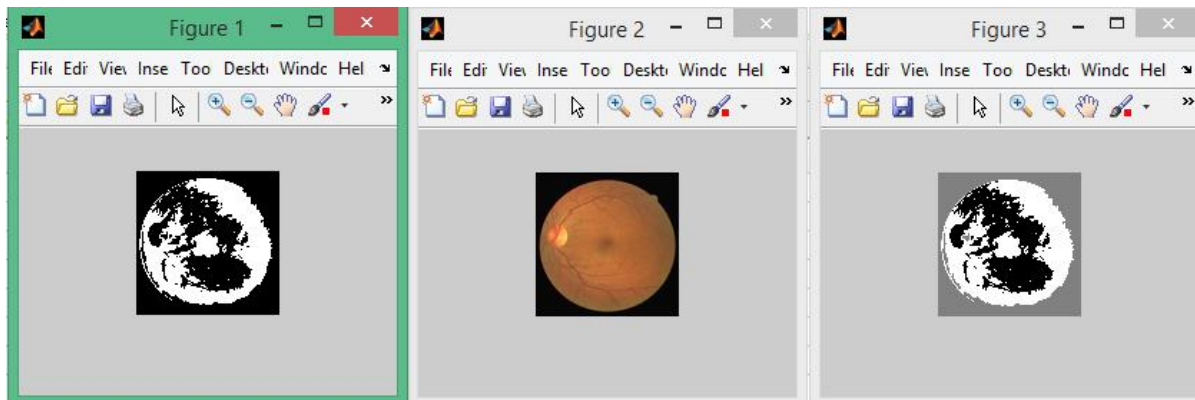
2.

D==1, SE1=0



3.

D==3, SE1=0

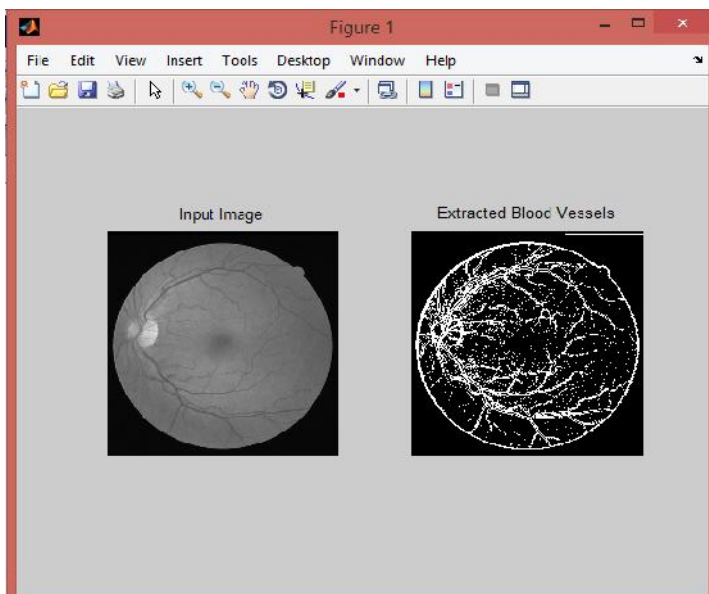


By applying the different operations on images , we found that the results obtained by A components of CIELAB are much better than rgb model and blood vessels are more visible than vessels extracted by rgb model . so , after working on different values of seeds and threshold we can say that cielab is much better than RGB .

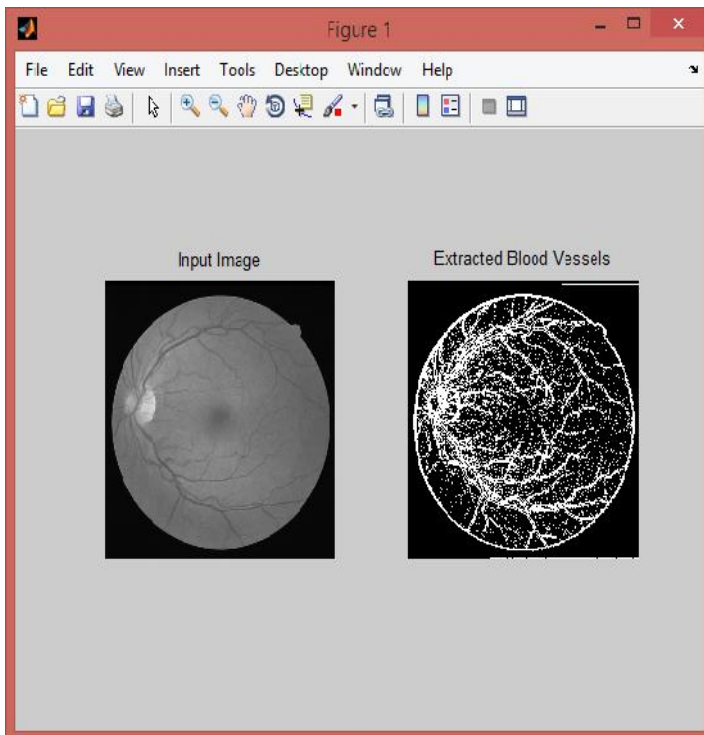
### (f) Retinal Blood Vessel Extraction

The code contain three file defect,vesselextract,main file . and different result is obtain for different value of thresholds .

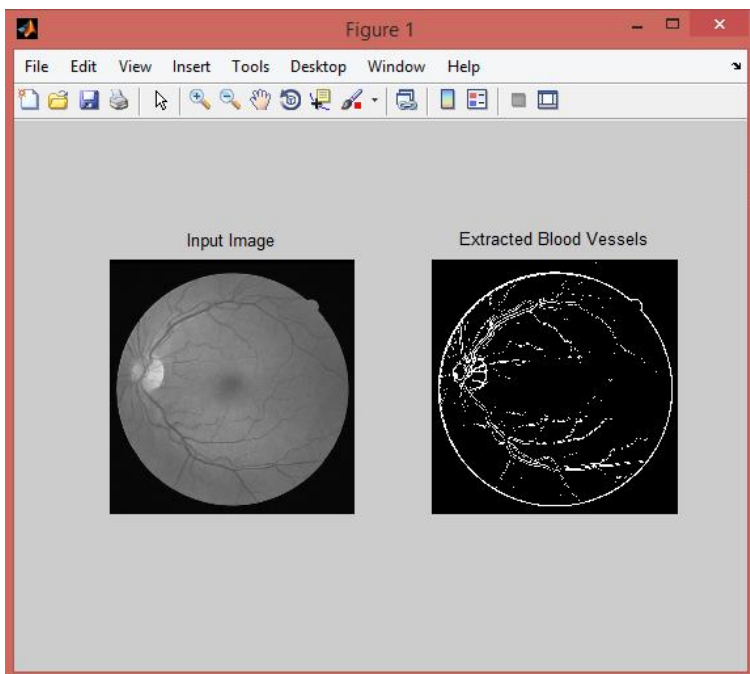
- For the Threshold value = 5



➤ For the Threshold value = 4



➤ For the Threshold value = 10



The above results show that the extraction is better in case of low value of threshold .



### (g) Mean-shift code is applied on Berkeley segmentation dataset

In this the mean-shiftcode is applied on Berkeley dataset on 100 images and then obtained result is then compared with ground truth .

In this the mean-shift code is applied on the original image of dataset after obtaining the result canny edge detection method is applied on the image then the result is compared with the ground truth .

Then , the value of precision and recall and fscore is calculated

Using the formula

$$Pre = t_p / (t_p + f_p);$$

$$Rec = t_p / (t_p + f_n);$$

$$fscore = 2(Pre * Rec) / Pre + Rec);$$







In this case the entropy of the image is also calculated and it has been observed that the segmentation results are much better for low value of entropy .

In this , we have calculated the values of Precision , Recision and f-score at different threshold Values and we found that the **segmentation results are much better at the value 0.8** instead of other values .










Results obtained after applying the results on 100 images used in Berkeley data set and then values are obtained and some of results are shown in table .










The table showing the result obtained after applying the pure MeanShift algorithm code on the different images is shown below .







Table 4.1










NO OF IMAGES	ORIGINAL IMAGE	RESULT	CANNY	Pre,rec,fscore 1	Pre,rec,fscore 2	Pre,rec,fscore 3	Pre,rec,fscore 4	Pre,rec,fscore 5
1.	100098.jpg 			0.8647, 0.8712, 0.7533	0.8738 0.8799 0.7689	0.9077 0.9123 0.8280	0.9645 0.9664 0.9320	0.9061 0.9018 0.8253
2.	112082.jpg 			0.9337 0.9032 0.8433	0.9409 0.9134 0.8595	0.9379 0.9092 0.8527	0.9394 0.9112 0.8560	0.9400 0.9121 0.8574
3.	227092.jpg 			0.9535 0.9328 0.8894	0.8725 0.8225 0.7177	0.9430 0.9181 0.8567	0.9430 0.9293 0.8837	0.9491 0.9266 0.8794
4.	304034.jpg 			0.9576 0.9361 0.8964	0.9575 0.9360 0.8962	0.9598 0.9393 0.9015	0.9526 0.9288 0.8848	0.9494 0.9420 0.8773
5.	15004.jpg 			0.9088 0.8670 0.7880	0.9089 0.8670 0.7880	0.9399 0.9109 0.8562	0.9071 0.8646 0.7843	0.9048 0.8614 0.7794
6.	242078.jpg 			0.9469 0.9225 0.8735	0.8858 0.8383 0.7426	0.9213 0.8866 0.8167	0.9162 0.8795 0.8058	0.9022 0.8604 0.7762










7.	23084.jpg 			0.9372 0.87 0.17	0.9427 0.9165 0.8639	0.9500 0.9269 0.8806	0.9141 0.8766 0.8012	0.9403 0.9131 0.8585
8.	35070.jpg 			0.9353 0.9068 0.8482	0.9386 0.9114 0.8554	0.9354 0.9069 0.8482	0.9357 0.9073 0.8489	0.9353 0.9068 0.8481
9.	38092.jpg 				0.9056 0.8647 0.7830	0.9249 0.8913 0.8233	0.9145 0.8769 0.8020	0.9119 0.8733 0.7964
10.	42078.jpg 			0.9526 0.94 0.92	0.9510 0.9301 0.8846	0.9453 0.9221 0.8716	0.9520 0.9315 0.8867	0.9446 0.9211 0.8701
11.	58060.jpg 			0.9574 0.9367 0.8968	0.9328 0.9012 0.8406	0.9555 0.9338 0.8923	0.9316 0.8995 0.8380	0.9048 0.8620 0.7799
12.	62096.jpg 			0.9471 0.9237 0.8748	0.9451 0.9209 0.8703	0.9318 0.9023 0.8408	0.9473 0.9239 0.8753	0.9471 0.9237 0.8748







13.	147062.jpg 			0.9391 0.9100 0.8546	0.9331 0.9014 0.8411	0.9165 0.8779 0.8046	0.9358 0.9052 0.8471	0.9567 0.9353 0.8948
14.	147091.jpg 			0.9366 0.9096 0.8519	0.9280 0.8977 0.8331	0.9441 0.9200 0.8685	0.9280 0.8977 0.8331	0.9349 0.9073 0.8483
15.	163014.jpg 			0.9272 0.8950 0.8299	0.9291 0.8975 0.8339	0.9307 0.8998 0.8374	0.9274 0.8953 0.8303	0.9236 0.8899 0.8219



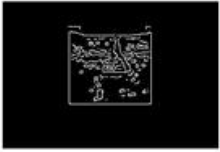






16.	183087.jpg 			0.9608 0.9431 0.9062	0.9212 0.8876 0.8176	0.9571 0.9378 0.8975	0.9490 0.9293 0.8791	0.9571 0.9378 0.8975
17.	187029.jpg 			0.9392 0.9121 0.8566	0.9439 0.9186 0.8544	0.9534 0.9231 0.8886	0.9401 0.9133 0.8586	0.9254 0.8929 0.8263
18.	208001.jpg 			0.8925 0.8454 0.7545	0.9274 0.8939 0.8290	0.8459 0.7834 0.6627	0.8822 0.8314 0.7334	0.9139 0.8749 0.7996







19.	247085.jpg 			0.9439 0.9194 0.8678	0.9222 0.8893 0.8201	0.9484 0.9257 0.8779	0.9099 0.8725 0.7939	0.8946 0.8518 0.7620
20.	291000.jpg 			0.9541 0.9315 0.8888	0.9409 0.9123 0.8584	0.9448 0.9180 0.8674	0.9419 0.9138 0.8607	0.9323 0.9000 0.8390
















21.	376001.jpg 			0.8916 0.8430 0.7517	0.8570 0.7966 0.6827	0.8727 0.8127 0.7133	0.8881 0.8383 0.7445	0.8830 0.8313 0.7341
22.	353013.jpg 			0.9421 0.9145 0.8615	0.9485 0.9237 0.8762	0.9368 0.9069 0.8495	0.9371 0.9073 0.8502	0.9193 0.8182 0.8110
23.	309004.jpg 			0.9495 0.9238 0.8771	0.9515 0.9266 0.8817	0.9509 0.9258 0.8803	0.9296 0.8949 0.8319	0.9493 0.9234 0.8766

24.	66053.jpg 			0.9457 0.9228 0.8727	0.9431 0.9191 0.8668	0.9255 0.8950 0.8283	0.9124 0.8771 0.8003	0.9271 0.8970 0.8316
25.	35008.jpg 			0.9508 0.9583 0.9111	0.9024 0.9167 0.8272	0.9075 0.9211 0.8359	0.9487 0.9566 0.9076	0.9061 0.9199 0.8335
26.	181018.jpg 			0.9419 0.9157 0.8625	0.9145 0.8776 0.8026	0.9116 0.8736 0.7963	0.9276 0.8956 0.8307	0.9183 0.8823 0.8106



















27.	35058.jpg 			0.9426 0.9162 0.8636	0.9346 0.9050 0.8458	0.9407 0.9135 0.8592	0.9436 0.9176 0.8659	0.9436 0.9176 0.8659
28.	41025.jpg 			0.9260 0.8911 0.8251	0.9472 0.9214 0.8727	0.9549 0.9326 0.8905	0.9255 0.8904 0.8240	0.9255 0.8904 0.8240





29.	76053.jpg 			0.9548 0.9341 0.8918	0.8931 0.8486 0.7579	0.9474 0.9235 0.8749	0.9507 0.9282 0.8824	0.9268 0.8947 0.8292
30.	108082.jpg 			0.9573 0.9369 0.8969	0.9512 0.9282 0.8829	0.9508 0.9276 0.8819	0.9501 0.9266 0.8803	0.9559 0.9345 0.8889
31.	134008.jpg 			0.9603 0.9403 0.9030	0.9597 0.9395 0.9016	0.9602 0.9402 0.9027	0.9594 0.9390 0.9009	0.9559 0.9339 0.8927














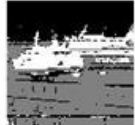




32.	181079.jpg 			0.9342 0.9044 0.8449	0.9347 0.9050 0.8460	0.9341 0.9042 0.8446	0.9354 0.9060 0.8474	0.9354 0.9060 0.8474
33.	317080.jpg 			0.9345 0.9062 0.8468	0.9492 0.9268 0.8797	0.9401 0.9140 0.8593	0.9191 0.8851 0.8136	0.9090 0.8713 0.7920



















34.	271031.jpg 			0.9667 0.9520 0.9202	0.9670 0.9524 0.9210	0.9587 0.9406 0.9018	0.9671 0.9525 0.9212	0.9670 0.9523 0.9208
35.	176039.jpg 			0.9576 0.9382 0.8984	0.9708 0.9572 0.9293	0.9576 0.9381 0.8983	0.9576 0.9382 0.8984	0.9529 0.9315 0.8877
36.	35010.jpg 			0.9009 0.8577 0.7728	0.9063 0.8651 0.7840	0.8855 0.8367 0.7409	0.8855 0.8367 0.7409	0.8867 0.8384 0.7434
37.	35091.jpg 			0.8948 0.8460 0.7570	0.8995 0.8525 0.7669	0.9134 0.8720 0.7965	0.9447 0.9169 0.8662	0.8900 0.8394 0.7471
38.	124084.jpg 			0.8906 0.8479 0.7552	0.9319 0.9036 0.8421	0.8974 0.8569 0.7960	0.9319 0.9036 0.8421	0.9353 0.9082 0.8494















39.	138078.jpg 			0.9516 0.9288 0.8839	0.9476 0.9231 0.8747	0.9519 0.9293 0.8846	0.9518 0.9292 0.8844	0.8829 0.8335 0.7359
40.	37073.jpg 			0.8975 0.8566 0.7688	0.9514 0.9304 0.8851	0.9007 0.8610 0.7755	0.9754 0.9643 0.9406	0.8963 0.8551 0.7661
41.	66075.jpg 			0.9536 0.9332 0.8899	0.9554 0.9358 0.8941	0.9372 0.9103 0.8531	0.9550 0.9351 0.8930	0.9557 0.9362 0.8947
42.	89072.jpg 			0.9328 0.9028 0.8422	0.9198 0.8847 0.8138	0.9261 0.8933 0.8273	0.9382 0.9103 0.8540	0.9309 0.9001 0.8380
43.	161062 			0.9484 0.9266 0.8788	0.9496 0.9283 0.8815	0.9503 0.9293 0.8832	0.9476 0.9254 0.8769	0.9449 0.9218 0.8710
44.	187039 			0.9514 0.9297 0.8845	0.9420 0.9165 0.8633	0.9420 0.9219 0.8720	0.9504 0.9283 0.8822	0.9473 0.9238 0.8751




45.	376043.jpg 			0.9494 0.9244 0.8777	0.9432 0.9154 0.8634	0.9215 0.8844 0.8150	0.9489 0.9236 0.8764	0.9318 0.8990 0.8377
46.	97033.jpg 			0.9477 0.9230 0.8747	0.9160 0.8782 0.8044	0.9425 0.9155 0.8628	0.9159 0.8781 0.8043	0.9170 0.8796 0.8065
47.	134052.jpg 			0.9431 0.9171 0.8649	0.9477 0.9237 0.8754	0.9525 0.9305 0.8862	0.9320 0.9015 0.8401	0.9433 0.9174 0.8654
48.	326038.JPG 			0.9547 0.9345 0.8922	0.9457 0.9217 0.8716	0.9576 0.9386 0.8989	0.9451 0.9209 0.8703	0.9506 0.9286 0.8827
49.	365025.jpg 			0.9402 0.9140 0.8593	0.9348 0.9066 0.8475	0.9198 0.8858 0.8148	0.9198 0.8857 0.8147	0.9397 0.9133 0.8583
















50.	15088.jpg 			0.9591 0.9413 0.9028	0.9597 0.9420 0.9040	0.9604 0.9431 0.9057	0.9620 0.9453 0.9093	0.9621 0.9455 0.9096
51.	22013.jpg 			0.9555 0.9346 0.8930	0.9463 0.9215 0.8720	0.9493 0.9257 0.8787	0.9455 0.9203 0.8701	0.8698 0.8165 0.7102
52.	25098.jpg 			0.9418 0.9148 0.8616	0.9404 0.9128 0.8584	0.9115 0.8723 0.7951	0.9046 0.8628 0.7805	0.9023 0.8597 0.7757
53.	43074.jpg 			0.9518 0.9291 0.8843	0.9534 0.9312 0.8878	0.9498 0.9262 0.8797	0.9499 0.9263 0.8799	0.9499 0.9263 0.8799
54.	78019.jpg 			0.9088 0.8713 0.7918	0.8906 0.8469 0.7543	0.9175 0.8832 0.8103	0.9044 0.8654 0.7827	0.9127 0.8765 0.8000
55.	145053.jpg 			0.9470 0.9228 0.8740	0.9151 0.8781 0.8038	0.9111 0.8727 0.7950	0.8941 0.8496 0.7597	0.8941 0.8496 0.7597










56.	145086.jpg 			0.9465 0.9222 0.8729	0.9446 0.9195 0.8686	0.9165 0.8803 0.8067	0.9266 0.8942 0.8286	0.9266 0.8942 0.8286
57.	148089.jpg 			0.9287 0.8944 0.8307	0.9034 0.8587 0.7757	0.9456 0.9186 0.8686	0.9357 0.9043 0.8461	0.9484 0.9277 0.8750
58.	181059.jpg 			0.9347 0.9062 0.8469	0.9392 0.9125 0.8571	0.9159 0.8802 0.8061	0.9285 0.8976 0.8334	0.9306 0.9005 0.8380
59.	188063.jpg 			0.9381 0.9113 0.8550	0.9523 0.9312 0.8868	0.9495 0.9272 0.8803	0.9433 0.9186 0.8665	0.9521 0.9309 0.8862
60.	196073.jpg 			0.9614 0.9451 0.9086	0.9626 0.9469 0.9115	0.9692 0.9561 0.9266	0.9689 0.9556 0.9259	0.9689 0.9556 0.9259
61.	225017.jpg 			0.9506 0.9264 0.8807	0.9439 0.9168 0.8654	0.9479 0.9225 0.8744	0.9396 0.9105 0.8555	0.9327 0.9005 0.8400



62.	260058.jpg 			0.9294 0.8991 0.8356	0.9326 0.9035 0.8427	0.9665 0.9513 0.9195	0.9490 0.9264 0.8791	0.9539 0.9333 0.8902
63.	302003.jpg 			0.8972 0.8582 0.7700	0.9102 0.8754 0.7968	0.8822 0.8384 0.7396	0.9072 0.8714 0.7906	0.9022 0.8648 0.7802
64.	159029.jpg 			0.9287 0.8946 0.8308	0.9421 0.9139 0.8609	0.9433 0.9157 0.8638	0.9429 0.9150 0.8627	0.9429 0.9150 0.8627







65.	285079.jp 			0.9429 0.8607 0.7798	0.9023 0.8556 0.7720	0.9343 0.9013 0.8421	0.9405 0.9102 0.8560	0.9137 0.8717 0.7964
-----	--	---	--	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------


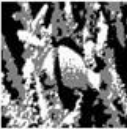







66.	118035.jpg 			0.9422 0.9171 0.8641	0.9471 0.9238 0.8749	0.9067 0.8682 0.7872	0.9515 0.9031 0.8851	0.9356 0.9078 0.8984
-----	---	---	--	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------

67.	19021.jpg 			0.9411 0.9135 0.8597	0.9422 0.9150 0.8621	0.9422 0.9150 0.8622	0.9211 0.8853 0.8155	0.9253 0.8911 0.8246
68.	46076.jpg 			0.9439 0.9197 0.8680	0.9352 0.9077 0.8489	0.9333 0.9050 0.8446	0.9303 0.9009 0.8381	0.9406 0.9152 0.8608
69.	86000.jpg 			0.9425 0.9163 0.8636	0.9267 0.8941 0.8285	0.9492 0.9258 0.8787	0.9382 0.9102 0.8539	0.8895 0.8431 0.7500
70.	48055.jpg 			0.9451 0.9208 0.8702	0.9322 0.9028 0.8416	0.9505 0.9284 0.8824	0.9542 0.9337 0.8910	0.9220 0.8887 0.8193
71.	118020.jpg 			0.9421 0.9154 0.8623	0.9362 0.9070 0.8491	0.8924 0.8465 0.7554	0.9137 0.8756 0.8000	0.9145 0.8768 0.8018
72.	8049.jpg 			0.8861 0.8390 0.7435	0.8750 0.8242 0.7212	0.9401 0.9132 0.8585	0.9221 0.8880 0.8180	0.9221 0.8880 0.8188



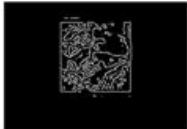






73.	135069.jpg 			0.9647 0.9494 0.9159	0.9658 0.9509 0.9184	0.9657 0.9509 0.9183	0.9657 0.9509 0.9183	0.9657 0.9509 0.9182
74.	145014.jpg 			0.9416 0.9146 0.8612	0.8819 0.8321 0.7338	0.9429 0.9165 0.8642	0.9429 0.9165 0.8642	0.9261 0.8927 0.8268
75.	166081.jpg 			0.9396 0.9113 0.8563	0.9495 0.9254 0.8787	0.9174 0.8801 0.8074	0.9151 0.8768 0.8023	0.9257 0.8917 0.8255


76.	167083.jpg 			0.9586 0.9390 0.9001	0.9150 0.8773 0.8027	0.9578 0.9378 0.8983	0.9563 0.9357 0.8948	0.9589 0.9394 0.9008
77.	159091.jpg 			0.9467 0.9212 0.8721	0.9574 0.9367 0.8967	0.9428 0.9197 0.8634	0.9428 0.9157 0.8634	0.9428 0.9157 0.8634
78.	172032.jpg 			0.9459 0.9228 0.8728	0.9325 0.9042 0.8432	0.9458 0.9226 0.8726	0.9490 0.9271 0.8798	0.9032 0.8645 0.7808



79.	175032.jpg 			0.9197 0.8803 0.8096	0.9393 0.9086 0.8534	0.8778 0.8218 0.7214	0.9427 0.9135 0.8612	0.9176 0.8772 0.8049
80.	182053.jpg 			0.9395 0.9116 0.8564	0.9419 0.9151 0.8619	0.8977 0.8537 0.7664	0.8946 0.8494 0.7599	0.9419 0.9151 0.8619

81.	209070.jpg 			0.9395 0.9099 0.8548	0.8865 0.8356 0.7407	0.8822 0.8298 0.7321	0.9195 0.8104 0.8814	0.9195 0.8814 0.8104
82.	225017.jpg 			0.9506 0.9273 0.8815	0.9439 0.9178 0.8663	0.9479 0.9234 0.8753	0.9396 0.9116 0.8565	0.9327 0.9018 0.8411
83.	232038.jpg 			0.9323 0.9028 0.8417	0.9379 0.9106 0.8540	0.9213 0.8876 0.8178	0.9262 0.8944 0.8284	0.9262 0.8944 0.8284



84.	123074.jpg 			0.9476 0.9221 0.8738	0.9464 0.9204 0.8711	0.9491 0.9244 0.8774	0.9527 0.9296 0.8856	0.9722 0.9582 0.9315
85.	268002.jpg 			0.9473 0.9228 0.8741	0.9463 0.9214 0.8719	0.9463 0.9213 0.8718	0.9429 0.9166 0.8643	0.9445 0.9188 0.8677
86.	285036.jpg 			0.8974 0.8547 0.7670	0.9296 0.8988 0.8355	0.9486 0.9254 0.8779	0.9118 0.8742 0.7972	0.9226 0.8889 0.8201

87.	368016.jpg 			0.9465 0.9229 0.8735	0.9409 0.9150 0.8609	0.9485 0.9256 0.8779	0.9155 0.8798 0.8054	0.9252 0.8931 0.8262
88.	41004.jpg 			0.9426 0.9172 0.8645	0.9332 0.9041 0.8437	0.9449 0.9205 0.8698	0.9418 0.9161 0.8628	0.9391 0.9123 0.8567
89.	41033.jpg 			0.9469 0.9227 0.8737	0.9430 0.9172 0.8650	0.9517 0.9295 0.8845	0.9430 0.9172 0.8650	0.9359 0.9071 0.8489

90.	183055.jpg 			0.9545 0.9342 0.8917	0.8941 0.8510 0.7609	0.9555 0.9357 0.8941	0.9558 0.9360 0.8946	0.9449 0.9207 0.8700
91.	241004.jpg 			0.9302 0.9011 0.8382	0.9493 0.9275 0.8804	0.9317 0.9031 0.8415	0.9311 0.9023 0.8401	0.9317 0.9031 0.8414
92.	311081.jpg 			0.9296 0.8496 0.8317	0.9280 0.8923 0.8281	0.9744 0.9067 0.9361	0.9362 0.9041 0.8465	0.9319 0.8979 0.8367
93.	385028.jpg 			0.9208 0.8862 0.8160	0.9440 0.9186 0.8671	0.9451 0.9202 0.8697	0.9203 0.8855 0.8149	0.9335 0.9039 0.8438
94.	388016.jpg 			0.9010 0.8590 0.7740	0.9582 0.9388 0.8996	0.9321 0.9018 0.8405	0.9134 0.8759 0.8000	0.8986 0.8557 0.7689
95.	135037.jpg 			0.9593 0.9408 0.9025	0.9614 0.9438 0.9074	0.9508 0.9288 0.8831	0.9520 0.9305 0.8858	0.9102 0.8725 0.7941










96.	106020.jpg 			0.9532 0.9300 0.8865	0.9532 0.9316 0.8890	0.9431 0.9154 0.8633	0.9432 0.9156 0.8636	0.9389 0.9093 0.8537
97.	323016.jpg 			0.9478 0.9240 0.8757	0.9348 0.9058 0.8467	0.9126 0.8750 0.7986	0.8958 0.8521 0.7633	0.9360 0.9075 0.8495
98.	246016.jpg 			0.9184 0.8839 0.8118	0.9317 0.9022 0.8406	0.9311 0.9013 0.8393	0.9345 0.9060 0.8466	0.9397 0.9132 0.8581
99.	92059.jpg 			0.9566 0.9364 0.8958	0.9472 0.9228 0.8741	0.9554 0.9346 0.8929	0.9414 0.9147 0.8611	0.9332 0.9031 0.8428
100	122048.jpg 			0.9343 0.9053 0.8458	0.9420 0.9160 0.8628	0.9599 0.9415 0.9038	0.9563 0.9363 0.8953	0.9275 0.8958 0.8309






















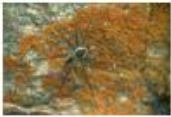


## Results of improved Mean-Shift :






















The table given below shows the result of improved Mean-shift Algorithm .



















(i) Result of Auto-Segmentation Using Shannon Entropy shown below in table 4.2










Table 4.2







NO OF IM AG ES	ORIGINAL IMAGE	RESULT	CANNY	PR,RE,FS	PR,RE,FS	PR,RE,FS	PR,RE,FS	PR,RE,FS	BW	CLUSTER
				1	2	3	4	5		
1.				0.9418 0.9856 0.9283	0.9404 0.9849 0.9263	0.9115 0.9740 0.8878	0.9046 0.9787 0.8854	0.9023 0.9725 0.8775	1(0.1400) 2(0.0800) 3(0.9900) 4(0.3200) 5(0.5500)	10 10 10 10
2.	8049 			0.8861 0.9778 0.8665	0.8750 0.9757 0.8538	0.9401 0.9908 0.9314	0.9221 0.9846 0.9079	0.9621 0.9935 0.9558	1(0.0100) 2(0.0100) 3(0.0300) 4(0.0400) 5(0.0200)	9 10 10 10 10
3.	15088 			0.9591 0.9927 0.9521	0.9597 0.9922 0.9522	0.9604 0.9915 0.9523	0.9620 0.9927 0.9550	0.9621 0.9907 0.9531	1(0.0100) 2(0.0600) 3(0.0600) 4(0.1000) 5(0.0700)	8 10 9 7 7

4.	19021 			0.9411 0.9848 0.9268	0.9422 0.9839 0.9271	0.9422 0.9820 0.9253	0.9211 0.9777 0.9006	0.9253 0.9765 0.9036	1(0.0800) 2(0.0400) 3(0.1700) 4(0.0400) 5(0.1700)	9 9 9 9 9
5.	22013 			0.9555 0.9934 0.9492	0.9463 0.9920 0.9387	0.9493 0.9891 0.9389	0.9455 0.9896 0.9356	0.8698 0.9697 0.8434	1(0.2900) 2(0.2900) 3(0.0300) 4(0.0600) 5(0.7900)	10 10 10 10 10
6.	23084 			0.9372 0.9875 0.9254	0.9427 0.9872 0.9306	0.9500 0.9867 0.9374	0.9141 0.9792 0.8950	0.9403 0.9864 0.9274	1(1.2300) 2(0.6600) 3(1.1400) 4(0.2000) 5(0.4900)	10 10 10 10 10
7.	35008 			0.9508 0.9910 0.9422	0.9024 0.9827 0.8868	0.9075 0.9793 0.8888	0.9488 0.9919 0.9423	0.8867 0.9736 0.8632	1(0.1000) 2(0.4400) 3(0.0400) 4(0.0500) 5(0.1300)	10 10 10 10 10
8.	35010 			0.9009 0.9709 0.8747	0.9063 0.9778 0.8862	0.8855 0.9655 0.8549	0.8855 0.9771 0.8652	0.8867 0.9716 0.8616	1(0.6800) 2(0.2000) 3(0.1300) 4(0.0900) 5(0.0200)	10 10 10 10 10
9.	35058 			0.9426 0.9875 0.9308	0.9346 0.9857 0.9212	0.9407 0.9866 0.9280	0.9436 0.9894 0.9336	0.9436 0.9885 0.9327	1(0.0100) 2(0.0100) 3(0.0200) 4(0.0500) 5(0.2100)	8 8 8 9 7
10.	35070 			0.9353 0.9851 0.9214	0.9386 0.9881 0.9274	0.9354 0.9858 0.9220	0.9357 0.9859 0.9225	0.9353 0.9825 0.9189	1(0.6300) 2(0.1800) 3(0.1300) 4(0.7000) 5(0.5400)	8 9 10 9 9
11.	35091 			0.8948 0.9586 0.8578	0.8995 0.9774 0.8792	0.9134 0.9659 0.8822	0.9447 0.9811 0.9269	0.8900 0.9585 0.8530	1(0.1700) 2(0.5800) 3(0.0500) 4(0.1900) 5(0.0300)	10 10 9 10 9



















12.	38092 			0.9236 0.9860 0.9107	0.9056 0.9882 0.8949	0.9249 0.9841 0.9102	0.9145 0.9820 0.8981	0.9119 0.9850 0.8982	1(1.3500) 2(1.2000) 3(0.2500) 4(1.1800) 5(0.5100)	10 10 10 10
13.	41004 			0.9426 0.9887 0.9319	0.9332 0.9857 0.9198	0.9449 0.9887 0.9343	0.9418 0.9867 0.9292	0.9391 0.9877 0.9275	1(0.5100) 2(0.0700) 3(0.0200) 4(0.1100) 5(0.0300)	10 10 10 10 10
14.	41025 			0.9260 0.9797 0.9083	0.9472 0.9850 0.9330	0.9549 0.9894 0.9447	0.9255 0.9784 0.9055	0.9255 0.9795 0.9065	1(0.8400) 2(1.4100) 3(0.2700) 4(1.1000) 5(0.8500)	10 10 9 10 10
15.	41033 			0.9469 0.9904 0.9378	0.9430 0.9900 0.9336	0.9517 0.9920 0.9440	0.9430 0.9877 0.9314	0.9359 0.9897 0.9263	1(0.0600) 2(0.0500) 3(0.1800) 4(0.1000) 5(0.1600)	8 10 10 9 8
16.	42078 			0.9526 0.9950 0.9479	0.9510 0.9944 0.9457	0.9453 0.9938 0.9394	0.9520 0.9945 0.9467	0.9446 0.9940 0.9389	1(1.0400) 2(0.8300) 3(1.3000) 4(1.4000) 5(0.7700)	9 9 9 10 10
17.	43074 			0.9518 0.9897 0.9420	0.9534 0.9873 0.9413	0.9498 0.9898 0.9401	0.9499 0.9882 0.9388	0.9499 0.9861 0.9367	1(0.1100) 2(0.1300) 3(0.0300) 4(0.0400) 5(0.1000)	10 10 9 9 10
18.	46076 			0.9439 0.9909 0.9353	0.9352 0.9905 0.9263	0.9333 0.9884 0.9225	0.9303 0.9908 0.9217	0.9406 0.9906 0.9318	1(0.0300) 2(0.0200) 3(0.0200) 4(0.0600) 5(0.0500)	9 10 10 10 10













19.	48055				0.9451 0.9910 0.9366	0.9322 0.9889 0.9219	0.9505 0.9917 0.9449	0.9542 0.9936 0.9481	0.9220 0.9869 0.9099	1(0.1100) 2(0.8200) 3(1.3000) 4(0.6800) 5(1.2600)	10 10 10 10
20.	58060				0.9574 0.9906 0.9484	0.9328 0.9870 0.9207	0.9555 0.9908 0.9467	0.9316 0.9866 0.9191	0.9048 0.9820 0.8885	1(0.0700) 2(0.4600) 3(0.4500) 4(0.4400) 5(0.2400)	10 10 8 10 7
21.	62096				0.9471 0.9889 0.9366	0.9451 0.9894 0.9351	0.9318 0.9846 0.9175	0.9473 0.9913 0.9391	0.9471 0.9873 0.9351	1(0.0200) 2(0.0100) 3(0.0300) 4(0.0100) 5(0.0300)	10 9 10 6 10
22.	66075				0.9536 0.9929 0.9468	0.9554 0.9945 0.9502	0.9372 0.9915 0.9292	0.9550 0.9935 0.9488	0.9557 0.9910 0.9471	1(0.1400) 2(0.1100) 3(0.0400) 4(0.0600) 5(0.0700)	9 10 10 9 10
23.	76053				0.9548 0.9897 0.9449	0.8931 0.9687 0.8652	0.9474 0.9848 0.9329	0.9507 0.9905 0.9416	0.9127 0.9810 0.8954	1(0.0400) 2(0.0100) 3(0.4400) 4(0.0100) 5(0.1600)	10 9 10 10 10
24.	78019				0.9088 0.9766 0.8875	0.8906 0.9728 0.8664	0.9157 0.9806 0.8997	0.9044 0.9808 0.8871	0.9127 0.9763 0.8910	1(0.0500) 2(0.0900) 3(1.0000) 4(1.3300) 5(0.1200)	10 10 10 10 10













25.	86000 			0.9425 0.9890 0.9322	0.9267 0.9837 0.9116	0.9492 0.9896 0.9394	0.9382 0.9869 0.9259	0.8895 0.9778 0.8697	1(0.8400) 2(0.4300) 3(1.1000) 4(1.0200) 5(1.4700)	10 10 10 10
26.	89072 			0.9328 0.9830 0.9170	0.9198 0.9812 0.9026	0.9261 0.9777 0.9054	0.9382 0.9861 0.9251	0.9309 0.9847 0.9167	1(0.0400) 2(0.0600) 3(0.1000) 4(0.6300) 5(0.1800)	10 10 10 10 10
27.	92059 			0.9566 0.9876 0.9448	0.9472 0.9878 0.9356	0.9554 0.9898 0.9457	0.9414 0.9879 0.9301	0.9332 0.9855 0.9197	1(0.0100) 2(0.1800) 3(0.1400) 4(0.1600) 5(0.0400)	09 10 9 10 7






















28.	97033 			0.9477 0.9889 0.9371	0.9160 0.9818 0.8993	0.9425 0.9901 0.9332	0.9159 0.9789 0.8966	0.9170 0.9791 0.8977	1(0.2100) 2(0.0500) 3(0.0500) 4(0.1200) 5(0.0600)	10 10 10 10 10
29.	100098 			0.8647 0.9593 0.8295	0.8738 0.9586 0.8376	0.9077 0.9785 0.8881	0.9645 0.9898 0.9546	0.9061 0.9654 0.8747	1(0.0100) 2(0.1600) 3(0.2100) 4(0.9900) 5(0.2600)	10 10 10 10 10



















































30.	106020				0.9532 0.9883 0.9420	0.9543 0.9860 0.9409	0.9431 0.9857 0.9296	0.9432 0.9824 0.9267	0.9389 0.9830 0.9929	1(0.1100) 2(0.1400) 3(0.1100) 4(0.1400) 5(0.0800)	10 10 10 10 10
31.	108082				0.9573 0.9892 0.9470	0.9512 0.9917 0.9434	0.9508 0.9914 0.9426	0.9501 0.9900 0.9406	0.9400 0.9864 0.9273	1(1.3300) 2(1.0200) 3(0.7100) 4(0.2900) 5(0.4700)	8 9 10 10 10
32.	112082				0.9337 0.9878 0.9223	0.9409 0.9881 0.9297	0.9379 0.9847 0.9235	0.9394 0.9836 0.9239	0.9400 0.9889 0.9296	1(0.0700) 2(1.3600) 3(0.0900) 4(0.2100) 5(0.1700)	10 10 10 10 10
33.	118020				0.9421 0.9859 0.9288	0.9362 0.9871 0.9241	0.8924 0.9781 0.8729	0.9137 0.9802 0.8956	0.9145 0.9804 0.8966	1(1.1200) 2(0.3800) 3(0.0100) 4(1.1000) 5(1.3500)	10 10 10 10 10
34.	118035				0.9442 0.9930 0.9357	0.9471 0.9931 0.9406	0.9067 0.9858 0.8938	0.9515 0.9931 0.9449	0.9356 0.9907 0.9269	1(0.0300) 2(0.0700) 3(0.0100) 4(0.0100) 5(0.0600)	8 9 10 8 10
35.	122048				0.9356 0.9907 0.9269	0.9343 0.9908 0.9258	0.9420 0.9901 0.9326	0.9563 0.9934 0.9499	0.9275 0.9883 0.9167	1(0.0600) 2(0.4900) 3(0.8100) 4(1.1100) 5(1.1700)	10 9 10 8 10

























36.	123074 			0.9476 0.9833 0.9318	0.9464 0.9840 0.9312	0.9491 0.9867 0.9365	0.9527 0.9848 0.9383	0.9722 0.9922 0.9646	1(0.9700) 2(0.2000) 3(1.2600) 4(0.1700) 5(0.7600)	10 10 10 10 10
37.	124084 			0.8906 0.9780 0.8710	0.9319 0.9858 0.9175	0.8974 0.9772 0.8725	0.9319 0.9856 0.9185	0.9353 0.9849 0.9211	1(1.1100) 2(0.0400) 3(0.0900) 4(0.7400) 5(0.4100)	10 10 10 9 9
38.	385028 			0.9208 0.9861 0.9079	0.9440 0.9903 0.9348	0.9451 0.9902 0.9358	0.9203 0.9836 0.9052	0.9318 0.9877 0.9204	1(0.0300) 2(0.0100) 3(0.0400) 4(0.0800) 5(0.0600)	9 10 10 9 10
39.	376043 			0.9494 0.9878 0.9378	0.9432 0.9846 0.9287	0.9215 0.9772 0.9005	0.9489 0.9881 0.9376	0.9318 0.9787 0.9186	1(1.4000) 2(0.7100) 3(0.0100) 4(0.0900) 5(0.0600)	10 10 10 10 10













40.	376001 			0.8916 0.9676 0.8627	0.8570 0.9543 0.8178	0.8727 0.9643 0.8416	0.8881 0.9679 0.8596	0.8830 0.9625 0.8499	1(0.3900) 2(0.2600) 3(0.0100) 4(0.4300) 5(0.5900)	10 10 10 10 10
41.	134008 			0.9603 0.9891 0.9498	0.9597 0.9897 0.9498	0.9602 0.9922 0.9527	0.9594 0.9917 0.9514	0.9559 0.9892 0.9455	1(0.5100) 2(0.1900) 3(0.4300) 4(0.3900) 5(0.3100)	8 9 9 9 9
42.	368016 			0.9465 0.9926 0.9395	0.9409 0.9861 0.9278	0.9485 0.9885 0.9376	0.9155 0.9784 0.8957	0.9252 0.9821 0.9086	1(0.0400) 2(0.2000) 3(0.1700) 4(0.0300) 5(0.0800)	10 10 10 9 9
43.	365025 			0.9402 0.9911 0.9318	0.9348 0.9867 0.9224	0.9198 0.9857 0.9067	0.9198 0.9861 0.9070	0.9397 0.9858 0.9264	1(0.6600) 2(0.1400) 3(0.4800) 4(0.0700) 5(0.0700)	10 10 10 10 10













44.	145014				0.9416 0.9894 0.9290	0.8819 0.9729 0.8580	0.9429 0.9869 0.9306	0.9429 0.9897 0.9304	0.9261 0.9806 0.9082	1(1.2800) 2(0.1600) 3(0.3100) 4(1.3400) 5(1.3900)	2 2 2 2 2
45.	353013				0.9421 0.9887 0.9315	0.9485 0.9884 0.9375	0.9368 0.9855 0.9232	0.9371 0.9861 0.9241	0.9193 0.9829 0.9036	1(0.1000) 2(0.2000) 3(0.0100) 4(0.0100) 5(0.0300)	9 10 10 10 9
46.	388016				0.9010 0.9802 0.8831	0.9010 0.9748 0.8783	0.9321 0.9852 0.9183	0.9134 0.9838 0.8986	0.8986 0.9688 0.8706	1(0.1500) 2(0.1000) 3(0.1500) 4(0.1800) 5(0.1200)	9 10 10 10 10
47.	317080				0.9345 0.9879 0.9232	0.9492 0.9904 0.9400	0.9401 0.9903 0.9309	0.9191 0.9862 0.9065	0.9090 0.9835 0.8940	1(0.9100) 2(0.8900) 3(0.0800) 4(0.3000) 5(1.4600)	9 7 10 10 10
48.	323016				0.9478 0.9905 0.9387	0.9348 0.9886 0.9242	0.8958 0.9788 0.8760	0.9360 0.9892 0.9260	0.9126 0.9837 0.8977	1(0.0300) 2(0.8200) 3(0.1300) 4(1.0100) 5(1.0500)	10 10 10 10 9
49.	326038				0.9547 0.9879 0.9432	0.9457 0.9892 0.9354	0.9576 0.9896 0.9477	0.9451 0.9893 0.9350	0.9506 0.9875 0.9387	1(0.2700) 2(0.1500) 3(0.0200) 4(0.0600) 5(0.2800)	10 9 9 9 8
50.	311081				0.9296 0.9782 0.9094	0.9280 0.9279 0.9029	0.9744 0.9899 0.9646	0.9362 0.9847 0.9219	0.9319 0.9844 0.9174	1(0.1000) 2(0.0800) 3(0.0100) 4(0.0700) 5(0.0100)	10 10 10 10 10






















51.	307073 			0.8975 0.9732 0.8734	0.9514 0.9901 0.9420	0.9007 0.9731 0.8765	0.9754 0.9940 0.9695	0.8963 0.9778 0.8764	1(0.0500) 2(0.0100) 3(0.0600) 4(0.0500) 5(0.0100)	10 9 10 10 9
52.	304034 			0.9576 0.9860 0.9442	0.9575 0.9856 0.9437	0.9598 0.9888 0.9490	0.9526 0.9890 0.9422	0.9494 0.9851 0.9353	1(1.3800) 2(0.4100) 3(0.3000) 4(0.2100) 5(0.1300)	9 10 9 10 10
53.	302003 			0.8972 0.9732 0.8731	0.9102 0.9773 0.8895	0.8822 0.9752 0.8603	0.9072 0.9803 0.8894	0.9022 0.9764 0.8809	1(0.4900) 2(0.6200) 3(0.0200) 4(0.0800) 5(0.4700)	10 10 10 10 10
54.	291000 			0.9541 0.9891 0.9437	0.9409 0.9817 0.9236	0.9448 0.9860 0.9316	0.9419 0.9829 0.9258	0.9323 0.9819 0.9154	1(0.0600) 2(0.8600) 3(0.7000) 4(0.1000) 5(0.1300)	10 10 10 10 10
55.	309004 			0.9495 0.9833 0.9337	0.9515 0.9865 0.9386	0.9509 0.9872 0.9837	0.9296 0.9822 0.9131	0.94930 0.9834 0.9335	1(0.2500) 2(0.2000) 3(0.1300) 4(0.1700) 5(0.0100)	8 10 9 7 9
56.	285079 			0.9060 0.9785 0.8865	0.9023 0.9649 0.8706	0.9343 0.9812 0.9167	0.9405 0.9856 0.9270	0.9137 0.9746 0.8905	1(0.7200) 2(0.6200) 3(0.5400) 4(1.0100) 5(1.1500)	10 10 10 9 10
57.	285036 			0.8974 0.9813 0.8807	0.9296 0.9872 0.9178	0.9486 0.9898 0.9390	0.9118 0.9806 0.8941	0.9226 0.9827 0.9066	1(0.1000) 2(0.2800) 3(0.4800) 4(0.2000) 5(0.7400)	10 10 10 9 10
58.	271031 			0.9667 0.9955 0.9263	0.9670 0.9955 0.9627	0.9587 0.9942 0.9531	0.9671 0.9952 0.9625	0.9670 0.9962 0.9633	1(0.0300) 2(0.0400) 3(0.0400) 4(0.1900) 5(0.0300)	10 10 10 10 10

59.	268002				0.9473 0.9859 0.9339	0.9463 0.9881 0.9350	0.9463 0.9871 0.9341	0.9429 0.9875 0.9312	0.9445 0.9876 0.9327	1(0.0200) 2(0.0600) 3(0.0400) 4(0.0100) 5(0.0100)	10 10 10 9 9
60.	260058				0.9294 0.9872 0.9875	0.9326 0.9883 0.9217	0.9665 0.9447 0.9614	0.9490 0.9937 0.9429	0.9539 0.9928 0.9470	1(0.0100) 2(0.3400) 3(0.1000) 4(1.2800) 5(0.8700)	10 10 10 10 10
61.	247085				0.9439 0.9867 0.9313	0.9222 0.9834 0.9069	0.9484 0.9895 0.9384	0.9099 0.9811 0.8927	0.8946 0.9813 0.8779	1(0.0500) 2(0.1200) 3(0.5600) 4(0.1100) 5(0.0800)	10 10 10 10 10
62.	246016				0.9184 0.9860 0.9056	0.9317 0.9906 0.9229	0.9311 0.9887 0.9206	0.9345 0.9880 0.9233	0.9397 0.9905 0.9308	1(0.1900) 2(0.0900) 3(1.2100) 4(0.4600) 5(0.1500)	10 10 10 10 10
63.	242078				0.9469 0.9882 0.9357	0.8858 0.9700 0.8592	0.9213 0.9821 0.9048	0.9162 0.9796 0.8975	0.9022 0.9787 0.8830	1(0.5200) 2(0.4400) 3(1.2000) 4(1.4400) 5(0.3900)	10 10 10 10 10
64.	241004				0.9302 0.9895 0.9204	0.9493 0.9936 0.9432	0.9317 0.9897 0.9222	0.9311 0.9896 0.9214	0.9312 0.9892 0.9216	1(0.0200) 2(0.0400) 3(0.0100) 4(0.0600) 5(0.0800)	9 10 10 10 9
65.	232038				0.9323 0.9890 0.9220	0.9379 0.9903 0.9288	0.9213 0.9885 0.9107	0.9262 0.9853 0.9126	0.9262 0.9888 0.9158	1(0.0400) 2(0.1000) 3(0.0900) 4(0.9800) 5(0.9000)	10 10 9 10 9
66.	227092				0.9535 0.9925 0.9563	0.8725 0.9740 0.8498	0.9430 0.9897 0.9333	0.9510 0.9925 0.9439	0.9491 0.9925 0.9420	1(0.0600) 2(0.0200) 3(0.0700) 4(0.0700) 5(0.0700)	10 10 10 9 10










67.	225017 			0.9506 0.9881 0.9393	0.9439 0.9861 0.9308	0.9479 0.9874 0.9359	0.9396 0.9841 0.9246	0.9327 0.9838 0.9175	1(0.5000) 2(0.2500) 3(0.0300) 4(0.2200) 5(0.0700)	10 10 9 10 10
68.	209070 			0.9395 0.9812 0.9218	0.8865 0.9686 0.8587	0.8822 0.9682 0.8542	0.9195 0.9770 0.8983	0.9195 0.9709 0.8927	1(0.0800) 2(0.2100) 3(0.2600) 4(0.1800) 5(1.3700)	10 10 10 10 9
69.	208001 			0.8925 0.9831 0.8774	0.9274 0.9840 0.9126	0.8459 0.9634 0.8150	0.8822 0.9746 0.8598	0.9139 0.9800 0.8956	1(1.1300) 2(1.5000) 3(0.2700) 4(1.0900) 5(1.4800)	10 10 10 10 10
70.	196073 			0.9614 0.9899 0.9516	0.9626 0.9902 0.9532	0.9692 0.9917 0.9611	0.9689 0.9927 0.9618	0.9689 0.9919 0.9611	1(0.0400) 2(0.0600) 3(0.0300) 4(0.0900) 5(0.0700)	6 10 5 10 7
71.	188063 			0.9381 0.9874 0.9263	0.9523 0.9905 0.9432	0.9495 0.9881 0.9361	0.9433 0.9891 0.9330	0.9521 0.9898 0.9424	1(0.0900) 2(0.0300) 3(0.0800) 4(0.0700) 5(0.1100)	9 9 10 7 7
72.	187039 			0.9514 0.9852 0.9374	0.9420 0.9857 0.9285	0.9459 0.9848 0.9315	0.9504 0.9879 0.9389	0.9473 0.9851 0.9332	1(0.1100) 2(0.1200) 3(0.1100) 4(0.0400) 5(0.0300)	10 10 9 9 10
73.	187029 			0.9392 0.9850 0.9251	0.9439 0.9875 0.9321	0.9534 0.9890 0.9428	0.9401 0.9861 0.9271	0.9254 0.9123 0.9091	1(0.4900) 2(1.1100) 3(0.0900) 4(0.0600) 5(1.1000)	10 10 10 9 10
74.	183087 			0.9608 0.9894 0.9506	0.9212 0.9799 0.9026	0.9571 0.9901 0.9476	0.9490 0.9849 0.9346	0.9571 0.9887 0.9462	1(0.2100) 2(0.0800) 3(0.2200) 4(0.0700) 5(0.1800)	10 10 9 10 10

75.	183055				0.9545 0.9933 0.9482	0.8941 0.9827 0.8786	0.9555 0.9905 0.9465	0.9558 0.9925 0.9486	0.9449 0.9919 0.9373	1(0.0700) 2(0.0500) 3(0.0300) 4(0.1300) 5(0.0100)	9 9 8 10 8
76.	182053				0.9395 0.9876 0.9278	0.9419 0.9889 0.9314	0.8977 0.9783 0.8982	0.8946 0.9755 0.8726	0.9419 0.9892 0.9318	1(0.1500) 2(0.1000) 3(0.0500) 4(0.1400) 5(0.3400)	10 10 10 10 10
77.	181079				0.9342 0.9848 0.9200	0.9347 0.9852 0.9209	0.9341 0.9857 0.9208	0.9354 0.9846 0.9210	0.9354 0.9844 0.9268	1(0.3300) 2(0.1200) 3(0.0800) 4(0.1300) 5(0.0500)	10 10 9 9 10
78.	181091				0.9347 0.9845 0.9201	0.9392 0.9914 0.9275	0.9159 0.9844 0.9015	0.9285 0.9854 0.9149	0.9306 0.9891 0.9201	1(0.0300) 2(0.3200) 3(0.1400) 4(0.2500) 5(0.2200)	10 10 10 10 7










79.	181018				0.9419 0.9900 0.9325	0.9145 0.9825 0.8985	0.9116 0.9859 0.8987	0.9276 0.9874 0.9159	0.9183 0.9844 0.9039	1(0.1500) 2(0.1500) 3(0.0100) 4(0.0700) 5(0.0200)	10 10 10 9 7
79.	176039				0.9576 0.9927 0.9506	0.9708 0.9945 0.9655	0.9576 0.9931 0.9510	0.9576 0.9921 0.9500	0.9176 0.9722 0.8921	1(0.1100) 2(0.1900) 3(0.0700) 4(0.1600) 5(0.5900)	9 10 10 10 9
80.	175032				0.9197 0.9686 0.8909	0.9353 0.9576 0.9164	0.8778 0.9526 0.8362	0.9427 0.9767 0.9207	0.8325 0.9756 0.8463	1(0.4700) 2(0.1400) 3(0.6200) 4(0.1000) 5(0.0300)	10 10 9 9 9
81.	172032				0.9459 0.9920 0.9383	0.9325 0.9917 0.9247	0.9458 0.9935 0.9396	0.9490 0.9927 0.9420	0.9032 0.9881 0.8925	1(0.3900) 2(0.0200) 3(0.0200) 4(0.0100) 5(1.3500)	9 8 10 10 9










82.	167083				0.9586 0.9914 0.9503	0.9150 0.9772 0.8941	0.9578 0.9904 0.9487	0.9563 0.9897 0.9465	0.9589 0.9913 0.9506	1(0.1500) 2(0.9000) 3(0.1400) 4(0.0400) 5(0.00)	10 10 10 10 10
83.	166081				0.9396 0.9848 0.9253	0.9498 0.9495 0.9902	0.9174 0.9836 0.9023	0.9151 0.9806 0.8973	0.9257 0.9846 0.9115	1(0.0700) 2(1.2700) 3(1.3200) 4(0.6100) 5(0.9300)	10 10 10 10 10
84.	163014				0.9272 0.9889 0.9170	0.9291 0.9871 0.9170 0	0.9307 0.9856 0.9172	0.9274 0.9865 0.9149	0.9236 0.9839 0.9087	1(0.0900) 2(0.1400) 3(0.0400) 4(0.1000) 5(0.0500)	9 9 9 10 9
85.	161062				0.9484 0.9930 0.9418	0.9496 0.9927 0.9426	0.9503 0.9923 0.9430	0.9476 0.9933 0.9412	0.9449 0.9908 0.9362	1(0.0300) 2(0.1600) 3(0.0300) 4(0.0400) 5(0.0600)	9 10 10 9 10
86.	159091				0.9467 0.9876 0.9350	0.9574 0.9902 0.9479	0.9428 0.9859 0.9296	0.9428 0.9833 0.9271	0.9428 0.9856 0.9292	1(0.7500) 2(0.0100) 3(0.9500) 4(1.2200) 5(1.4800)	10 10 10 10 10
87.	159029				0.9287 0.9782 0.9085	0.9421 0.9821 0.9252	0.9433 0.9846 0.9288	0.9429 0.9840 0.9278	0.9429 0.9852 0.9289	1(01.0500) 2(0.8200) 3(01.3300) 4(0.9100) 5(0.4996)	10 10 10 10 10
88.	148089				0.9288 0.9748 0.9054	0.9034 0.9776 0.8831	0.9456 0.9819 0.9284	0.9357 0.9807 0.9716	0.9484 0.9847 0.9338	1(0.1400) 2(0.1400) 3(0.1900) 4(0.9600) 5(0.7000)	10 10 10 10 10




































89.	147091				0.9366 0.9878 0.9252	0.9280 0.9911 0.9197	0.9441 0.9913 0.9358	0.9280 0.9844 0.9135	0.9349 0.9878 0.9236	1(0.4300) 2(1.6000) 3(1.0200) 4(1.0300) 5(1.3600)	9 10 10 10 10
90.	147062				0.9391 0.9362 0.9234	0.9331 0.9792 0.9137	0.9165 0.9753 0.8938	0.9358 0.9837 0.9205	0.9567 0.9863 0.9436	1(0.1400) 2(0.6000) 3(0.4200) 4(0.9700) 5(0.9100)	10 10 10 10 10
91.	145086				0.9465 0.9902 0.9372	0.9446 0.9919 0.9369	0.9165 0.9842 0.9019	0.9266 0.9879 0.9153	0.9266 0.9870 0.9145	1(0.0400) 2(0.0700) 3(0.4300) 4(0.2600) 5(0.0600)	9 10 10 10 10



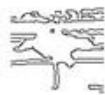









(ii) Result of Non Extensive Entropy shown below in table 4.3










NO. OF IMAGES	ORIGINAL IMAGE	CANNY	RESULT	PR,RE,FS 1	PR,RE,FS 2	PR,RE,FS 3	PR,RE,FS 4	PR,RE,FS 5	BW	CLUSTER
1.	25098 			0.9418 0.9875 0.9301	0.9404 0.9886 0.9297	0.9115 0.9783 0.8917	0.9046 0.9824 0.8887	0.9023 0.9799 0.8841	1(0.8500) 2(0.1800) 3(0.0700) 4(1.0000) 5(1.3300)	2 2 2 2 2
2.	8049 			0.8861 0.9813 0.8695	0.8750 0.9764 0.8544	0.9401 0.9916 0.9323	0.9221 0.9874 0.9105	0.9621 0.9939 0.9563	1(0.0200) 2(0.0400) 3(0.0300) 4(0.0300) 5(0.0200)	3 9 3 9 5
3.	15088 			0.9591 0.9919 0.9513	0.9597 0.9921 0.9521	0.9604 0.9918 0.9526	0.9620 0.9926 0.9548	0.9621 0.9914 0.9538	1(0.0500) 2(0.0400) 3(0.0100) 4(0.0300) 5(0.0100)	5 6 9 9 8

























4.	19021				0.9411 0.9865 0.9284	0.9422 0.9862 0.9292	0.9422 0.9853 0.9284	0.9211 0.9809 0.9035	0.9253 0.9808 0.9075	1(0.0200) 2(0.2800) 3(0.3500) 4(0.2800) 5(0.3500)	2 2 3 2 3
5.	22013				0.9555 0.9927 0.9485	0.9463 0.9911 0.9379	0.9493 0.9911 0.9408	0.9455 0.9908 0.9368	0.8698 0.9727 0.8461	1(0.6500) 2(0.6500) 3(1.0400) 4(1.3700) 5(0.1400)	6 6 2 2 4
6	23084				0.9372 0.9897 0.9276	0.9427 0.9906 0.9338	0.9500 0.9910 0.9415	0.9141 0.9852 0.9006	0.9403 0.9902 0.9311	1(1.2700) 2(1.1300) 3(0.6300) 4(1.2000) 5(1.0500)	4 4 2 2 2













7.	35008				0.9508 0.9925 0.9436	0.9024 0.9847 0.8886	0.90753 0.9837 0.8927	0.9488 0.9932 0.9423	0.8867 0.9773 0.8665	1(0.2700) 2(0.9200) 3(0.0400) 4(1.2400) 5(0.1600)	3 2 2 2 5
8.	35010				0.9009 0.9766 0.8799	0.9063 0.9805 0.8886	0.8855 0.9724 0.8611	0.8855 0.9775 0.8655	0.8867 0.9711 0.8611	1(1.2500) 2(1.1200) 3(0.1900) 4(0.8400) 5(0.6900)	2 2 2 2 3
9.	35058				0.9426 0.9896 0.9327	0.9346 0.9880 0.9235	0.9407 0.9890 0.9303	0.9436 0.9906 0.9347	0.9436 0.9901 0.9343	1(1.4800) 2(1.4800) 3(1.3200) 4(1.0200) 5(0.8400)	3 3 3 2 5
10.	35070				0.9353 0.9866 0.9228	0.9386 0.9883 0.9277	0.9354 0.9869 0.9231	0.9357 0.9867 0.9233	0.9353 0.9857 0.9219	1(0.5000) 2(0.5500) 3(0.4600) 4(0.7100) 5(0.0300)	8 5 10 2 4













11.	35091 			0.8948 0.9642 0.8672	0.8995 0.9742 0.8764	0.9134 0.9736 0.8893	0.9447 0.9846 0.9301	0.8900 0.9633 0.8573	1(1.1300) 2(1.1300) 3(1.2200) 4(1.1300) 5(0.4400)	8 5 6 6 3
12.	38092 			0.9236 0.9877 0.9122	0.9056 0.9874 0.8942	0.9249 0.9869 0.9128	0.9145 0.9851 0.9009	0.9119 0.9863 0.8994	1(1.2600) 2(.05000) 3(0.9900) 4(0.0700) 5(1.3900)	2 3 2 2 2
13.	41004 			0.9426 0.9920 0.9350	0.9332 0.9901 0.9240	0.9449 0.9926 0.9379	0.9418 0.9910 0.9333	0.9391 0.9917 0.9312	1(1.3900) 2(1.3800) 3(1.3800) 4(0.2500) 5(1.4300)	2 2 2 2 3
14.	41025 			0.9260 0.9809 0.9083	0.9472 0.9887 0.9365	0.9549 0.9901 0.9455	0.9255 0.9837 0.9104	0.9255 0.9807 0.9077	1(0.8600) 2(0.9400) 3(0.3900) 4(0.2200) 5(0.1200)	6 2 5 2 8
15.	41033 			0.9469 0.9916 0.9390	0.9430 0.9908 0.9344	0.9517 0.9928 0.9448	0.9430 0.9901 0.9337	0.9359 0.9901 0.9266	1(0.1300) 2(0.2100) 3(0.1500) 4(0.2000) 5(0.0400)	5 3 4 4 4
16.	42078 			0.9526 0.9952 0.9481	0.9510 0.9948 0.9461	0.9453 0.9943 0.9399	0.9520 0.9949 0.9472	0.9446 0.9943 0.9392	1(0.7400) 2(0.9000) 3(0.0500) 4(0.8700) 5(0.2100)	3 2 2 2 2
17.	43074 			0.9518 0.9928 0.9449	0.9534 0.9918 0.9455	0.9498 0.9927 0.9429	0.9499 0.9920 0.9423	0.9499 0.9911 0.9414	1(0.7200) 2(1.3700) 3(0.7500) 4(0.0800) 5(0.2400)	3 2 2 2 3

























18.	46076 			0.9439 0.9924 0.9367	0.9352 0.9916 0.9274	0.9333 0.9913 0.9251	0.9303 0.9931 0.9239	0.9406 0.9929 0.9339	1(0.8600) 2(1.3700) 3(1.1000) 4(0.4900) 5(1.1900)	9 4 2 2 2
19.	48055 			0.9451 0.9936 0.9390	0.9322 0.9921 0.9248	0.9505 0.9942 0.9449	0.9542 0.9945 0.9490	0.9220 0.9907 0.9134	1(0.4200) 2(0.1300) 3(1.2300) 4(1.0100) 5(1.3200)	2 2 2 3 2
20.	58060 			0.9574 0.9924 0.9502	0.9328 0.9887 0.9223	0.9555 0.9923 0.9481	0.9316 0.9884 0.9208	0.9048 0.9841 0.8904	1(0.1400) 2(0.0600) 3(0.0800) 4(0.9208) 5(0.0800)	2 2 2 2 2
21.	62096 			0.9471 0.9924 0.9399	0.9451 0.9925 0.9381	0.9318 0.9904 0.9229	0.9473 0.9935 0.9412	0.9471 0.9921 0.9397	1(0.6500) 2(0.0400) 3(0.2900) 4(0.6300) 5(0.2000)	2 3 2 2 2

























22.	66075 			0.9536 0.9948 0.9486	0.9554 0.9956 0.9512	0.9372 0.9934 0.9310	0.9550 0.9951 0.9503	0.9557 0.9940 0.9499	1(0.4200) 2(0.4900) 3(0.8200) 4(0.7200) 5(0.0700)	2 3 2 3 2
23.	76053 			0.9548 0.9917 0.9469	0.8931 0.9768 0.8724	0.9474 0.9889 0.9368	0.9507 0.9918 0.9428	0.9127 0.9842 0.8982	1(0.0100) 2(0.4700) 3(0.9500) 4(0.8200) 5(0.4700)	2 2 3 2 2
24.	78019 			0.9088 0.9818 0.8923	0.8906 0.9793 0.8772	0.9175 0.9849 0.9037	0.9044 0.9840 0.8899	0.9127 0.9827 0.8969	1(0.9900) 2(1.3500) 3(0.3300) 4(1.3900) 5(1.1200)	2 2 2 2 2

25.	86000 			0.9425 0.9908 0.9338	0.9267 0.9871 0.9148	0.9492 0.9915 0.9412	0.9382 0.9894 0.9283	0.8895 0.9813 0.8729	1(0.3000) 2(0.6800) 3(1.4600) 4(1.1700) 5(0.6900)	2 2 3 3 2
26.	89072 			0.9328 0.9852 0.9190	0.9198 0.9830 0.9042	0.9261 0.9834 0.9107	0.9382 0.9872 0.9262	0.9309 0.9871 0.9189	1(0.7000) 2(0.9100) 3(0.7400) 4(0.5900) 5(0.8900)	4 3 4 3 5
27.	92059 			0.9566 0.9892 0.9463	0.9472 0.9888 0.9365	0.9554 0.9901 0.9460	0.9414 0.9879 0.9301	0.9332 0.9854 0.9196	1(0.7300) 2(0.9100) 3(0.2400) 4(1.1400) 5(0.3400)	8 3 3 3 7
28.	97033 			0.9477 0.9920 0.9401	0.9160 0.9869 0.9040	0.9425 0.9923 0.9332	0.9159 0.9856 0.9027	0.9170 0.9857 0.9039	1(0.7500) 2(0.2500) 3(0.3800) 4(0.2000) 5(0.6400)	2 2 2 2 2
29.	100098 			0.8647 0.9702 0.8389	0.8738 0.9707 0.8483	0.9077 0.9829 0.8921	0.9645 0.9927 0.9574	0.9061 0.9769 0.8851	1(0.7500) 2(0.2700) 3(0.1100) 4(0.9600) 5(1.1200)	2 2 2 2 2
30.	106020 			0.9532 0.9881 0.9419	0.9543 0.9860 0.9409	0.9431 0.9855 0.9294	0.9432 0.9825 0.9267	0.9389 0.9849 0.9247	1(0.5600) 2(0.9543) 3(0.5600) 4(0.0600) 5(0.1000)	2 5 2 5 6
31.	108082 			0.9573 0.9913 0.9490	0.9512 0.9919 0.9435	0.9508 0.9918 0.9430	0.9501 0.9910 0.9416	0.9400 0.9884 0.9292	1(0.0700) 2(0.4400) 3(1.0600) 4(0.0400) 5(0.5800)	10 5 9 2 3
32.	112082 			0.9337 0.9905 0.9248	0.9409 0.9906 0.9321	0.9379 0.9890 0.9276	0.9394 0.9888 0.9289	0.9400 0.9909 0.9315	1(0.0700) 2(0.7400) 3(0.3300) 4(0.7500) 5(0.9200)	2 2 2 3 2

























33.	118020 			0.9421 0.9887 0.9317	0.9362 0.9889 0.9258	0.8924 0.9810 0.8755	0.9137 0.9838 0.8989	0.9145 0.9840 0.8999	1(1.4900) 2(0.4400) 3(0.4400) 4(0.8800)	2 2 2 2
34.	118035 			0.9422 0.9947 0.9373	0.9471 0.9949 0.9423	0.9067 0.9900 0.8976	0.9515 0.9951 0.9469	0.9356 0.9934 0.9294	1(0.2000) 2(0.5100) 3(0.9100) 4(1.0100) 5(0.4300)	2 4 6 3 2
35.	122048 			0.9356 0.9934 0.9294	0.9343 0.9915 0.9264	0.9420 0.9916 0.9340	0.9563 0.9936 0.9501	0.9275 0.9898 0.9181	1(0.4300) 2(0.1700) 3(0.0500) 4(1.1000) 5(0.9400)	2 2 2 2 2
36.	123074 			0.9476 0.9880 0.9362	0.9464 0.9882 0.9352	0.9491 0.9897 0.9393	0.9527 0.9891 0.9424	0.9722 0.9942 0.9665	1(0.0300) 2(0.1100) 3(0.4800) 4(1.1200)	2 2 2 3













37.	124084 			0.8906 0.9781 0.8711	0.9319 0.9858 0.9186	0.8974 0.9816 0.8808	0.9319 0.9863 0.9191	0.9353 0.9863 0.9224	1(0.9600) 2(1.0000) 3(0.0900) 4(0.8600) 5(1.0100)	2 2 2 2 2
38.	385028 			0.9208 0.9874 0.9092	0.9440 0.9920 0.9364	0.9451 0.9916 0.9372	0.9203 0.9874 0.9087	0.9318 0.9903 0.9228	1(1.2700) 2(0.9100) 3(0.4100) 4(1.4600) 5(0.8600)	6 2 6 2 2
39.	376043 			0.9494 0.9911 0.9410	0.9432 0.9903 0.9341	0.9215 0.9844 0.9071	0.9489 0.9913 0.9406	0.9318 0.9858 0.9186	1(0.3900) 2(0.0700) 3(1.1700) 4(0.0200) 5(0.0600)	3 2 2 2 2
40.	376001 			0.8916 0.9732 0.8677	0.8570 0.9608 0.8234	0.8727 0.9694 0.8460	0.8881 0.9730 0.8641	0.8830 0.9700 0.8565	1(0.8900) 2(1.4700) 3(0.3500) 4(0.5700) 5(0.7200)	2 3 2 2 2













41.	134008 			0.9603 0.9905 0.9511	0.9597 0.9909 0.9509	0.9602 0.9922 0.9527	0.9594 0.9919 0.9516	0.9559 0.9892 0.9455	1(0.2500) 2(0.9900) 3(1.0600) 4(1.4300) 5(1.3900)	2 2 2 6 6
42.	368016 			0.9465 0.9937 0.9406	0.9409 0.9904 0.9319	0.9485 0.9919 0.9408	0.9155 0.9854 0.9021	0.9252 0.9877 0.9137	1(1.3800) 2(1.0700) 3(0.5900) 4(0.8800) 5(1.3300)	2 2 2 2 3
43.	365025 			0.9402 0.9934 0.9339	0.9348 0.9912 0.9266	0.9198 0.9899 0.9105	0.9198 0.9901 0.9106	0.9397 0.9909 0.9312	1(1.0100) 2(0.3800) 3(0.0700) 4(0.4500) 5(0.3600)	3 2 2 2 2
44.	145014 			0.9416 0.9894 0.9316	0.8819 0.9782 0.8626	0.9429 0.9902 0.9306	0.9429 0.9896 0.9331	0.9261 0.9854 0.9126	1(0.9400) 2(0.3300) 3(0.9000) 4(0.2000) 5(1.0400)	2 2 2 2 2
45.	353013 			0.9421 0.9916 0.9342	0.9435 0.9905 0.9395	0.9368 0.9899 0.9272	0.9371 0.9901 0.9278	0.9193 0.9886 0.9088	1(0.3900) 2(1.1800) 3(0.4900) 4(0.2700) 5(0.0400)	2 8 2 2 2
46.	388016 			0.9010 0.9801 0.9831	0.9300 0.9834 0.9146	0.9321 0.9862 0.9192	0.9134 0.9817 0.8967	0.9126 0.9868 0.8752	1(0.0400) 2(0.2100) 3(0.0400) 4(0.1800) 5(1.5000)	9 7 8 7 2
47.	323016 			0.9478 0.9923 0.9405	0.9348 0.9906 0.9260	0.8958 0.9829 0.8805	0.9360 0.9910 0.9276	0.8986 0.9740 0.9006	1(1.4200) 2(0.6200) 3(0.5000) 4(0.7500) 5(0.3000)	2 2 2 2 3
48.	317080 			0.9345 0.9913 0.9263	0.9492 0.9932 0.9427	0.9401 0.9927 0.9332	0.9191 0.9897 0.9095	0.9090 0.9880 0.8981	1(0.0700) 2(0.2300) 3(0.5300) 4(0.2800) 5(0.1700)	9 2 2 2 2











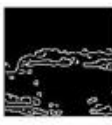

49.	326038 			0.9547 0.9897 0.9449	0.9457 0.9894 0.9356	0.9576 0.9908 0.9488	0.9451 0.9894 0.9350	0.9506 0.9891 0.9402	1(0.1300) 2(0.0100) 3(0.0100) 4(0.0400) 5(0.0300)	3 2 2 2 6
50.	311081 			0.9296 0.9810 0.9120	0.9280 0.9783 0.9079	0.9744 0.9922 0.9668	0.9362 0.9850 0.9222	0.9319 0.9843 0.9173	1(1.2300) 2(0.1100) 3(0.9400) 4(0.7100) 5(0.8200)	3 2 2 2 3
51.	307073 			0.8975 0.9757 0.8756	0.9514 0.9898 0.9417	0.9007 0.9745 0.8777	0.9754 0.9945 0.9700	0.8963 0.9772 0.8759	1(0.0600) 2(1.4900) 3(0.0800) 4(1.4900) 5(0.0200)	9 2 9 9 9
52.	304034 			0.9576 0.9878 0.9459	0.9575 0.9852 0.9433	0.9598 0.9889 0.9491	0.9526 0.9889 0.9421	0.9494 0.9866 0.9367	1(0.0200) 2(0.4800) 3(0.4100) 4(0.2700) 5(1.4800)	4 6 10 3 2
53.	302003 			0.8972 0.9801 0.8794	0.9102 0.9806 0.8926	0.8822 0.9799 0.8644	0.9072 0.9842 0.8929	0.9022 0.9821 0.8860	1(0.8200) 2(0.0700) 3(0.0300) 4(0.0800) 5(1.3500)	3 4 3 2 2
54.	291000 			0.9541 0.9904 0.9450	0.9409 0.9855 0.9272	0.9448 0.9879 0.9334	0.9419 0.9863 0.9290	0.9323 0.9847 0.9180	1(0.6900) 2(0.5400) 3(0.1100) 4(0.0300) 5(0.0300)	2 2 2 2 3
55.	309004 			0.9495 0.9856 0.9358	0.9515 0.9873 0.9394	0.9509 0.9876 0.9390	0.9296 0.9824 0.9132	0.9493 0.9855 0.9355	1(0.3600) 2(0.0800) 3(1.5000) 4(0.1500) 5(0.0200)	4 2 3 2 2
56.	285079 			0.9060 0.9786 0.8866	0.9023 0.9678 0.8733	0.9343 0.9833 0.9188	0.9405 0.9863 0.9276	0.9137 0.9777 0.8933	1(1.2000) 2(0.2900) 3(0.1000) 4(0.6800) 5(1.1500)	2 10 2 3 2



















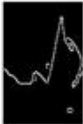













57.	285036 			0.8974 0.9855 0.8845	0.9296 0.9903 0.9206	0.9486 0.9936 0.9416	0.9118 0.9861 0.8992	0.9226 0.9887 0.9112	1(1.1700) 2(0.7200) 3(0.6400) 4(0.9500) 5(0.1300)	2 2 3 4 2
58.	271031 			0.9667 0.9973 0.9641	0.9670 0.9973 0.9645	0.9587 0.9966 0.9554	0.9671 0.9972 0.9644	0.9670 0.9977 0.9647	1(0.2100) 2(1.2500) 3(0.1300) 4(1.4900) 5(0.0600)	2 2 2 2 2
59.	268002 			0.9473 0.9881 0.9360	0.9463 0.9884 0.9354	0.9463 0.9879 0.9348	0.9429 0.9878 0.9314	0.9445 0.9878 0.9329	1(0.0300) 2(0.3400) 3(1.2100) 4(0.1300) 5(0.9600)	2 7 5 4 10
60.	260058 			0.9294 0.9909 0.9209	0.9326 0.9910 0.9243	0.9665 0.9962 0.9628	0.9490 0.9949 0.9441	0.9539 0.9947 0.9488	1(0.2700) 2(1.1700) 3(1.2100) 4(0.1300) 5(0.9600)	2 8 3 2 2
61.	247085 			0.9439 0.9912 0.9356	0.9222 0.9886 0.9117	0.9484 0.9926 0.9414	0.9009 0.9868 0.8979	0.8946 0.9862 0.8823	1(0.600) 2(1.1400) 3(1.2400) 4(0.5500) 5(1.2500)	2 2 2 2 4
62.	246016 			0.9184 0.9876 0.9071	0.9317 0.9916 0.9239	0.9311 0.9907 0.9224	0.9345 0.9903 0.9255	0.9397 0.9914 0.9316	1(0.4300) 2(1.0200) 3(1.4900) 4(0.1600) 5(0.9400)	2 3 3 3 2
63.	242078 			0.9469 0.9899 0.9373	0.8858 0.9756 0.8643	0.9213 0.9848 0.9072	0.9162 0.9810 0.8988	0.9022 0.9821 0.8860	1(1.2100) 2(0.1200) 3(1.0700) 4(0.2100) 5(1.2800)	2 2 2 6 2
64.	241004 			0.9302 0.9933 0.9239	0.9493 0.9960 0.9454	0.9317 0.9938 0.9259	0.9311 0.9930 0.9216	0.9312 0.9935 0.9257	1(0.5900) 2(1.1200) 3(0.2900) 4(0.7700) 5(0.7100)	2 3 2 2 2









65.	232038 			0.9323 0.9908 0.9237	0.9379 0.9916 0.9300	0.9213 0.9899 0.9120	0.9262 0.9882 0.9152	0.9262 0.9898 0.9168	1(1.1400) 2(1.2000) 3(0.0200) 4(1.4900) 5(1.1600)	3 4 3 5 3
66.	227092 			0.9353 0.9940 0.9477	0.8725 0.9804 0.8554	0.9430 0.9921 0.9355	0.9510 0.9939 0.9451	0.9491 0.9938 0.9432	1(0.6600) 2(1.2700) 3(1.1000) 4(0.2700) 5(0.3100)	2 2 2 3 2
67.	225017 			0.9506 0.9904 0.9415	0.9439 0.9889 0.9335	0.9479 0.9899 0.9383	0.9396 0.9873 0.9277	0.9327 0.9869 0.9204	1(0.3600) 2(0.5500) 3(0.4200) 4(1.2300) 5(0.0900)	2 2 2 2 2
68.	209070 			0.9395 0.9850 0.9253	0.8865 0.9730 0.8626	0.8822 0.9719 0.8574	0.9195 0.9808 0.9018	0.9195 0.9780 0.8992	1(0.1600) 2(0.6300) 3(1.1900) 4(0.1600) 5(0.5700)	2 2 2 2 2










69.	208001 			0.8925 0.9837 0.8779	0.9274 0.9869 0.9153	0.8459 0.9698 0.8204	0.8822 0.9787 0.8634	0.9139 0.9826 0.8980	1(0.8000) 2(0.8600) 3(0.6400) 4(0.1800) 5(1.4600)	2 2 2 2 3
70.	196073 			0.9614 0.9899 0.9517	0.9626 0.9903 0.9533	0.9692 0.9919 0.9613	0.9689 0.9924 0.9015	0.9689 0.9920 0.9611	1(0.0200) 2(0.0200) 3(0.0200) 4(0.0200) 5(0.0100)	2 2 2 2 2
71.	186063 			0.9381 0.9893 0.9281	0.9523 0.9920 0.9447	0.9495 0.9905 0.9404	0.9433 0.9905 0.9344	0.9521 0.9915 0.9440	1(0.4700) 2(1.4100) 3(0.0900) 4(0.1000) 5(0.0700)	2 2 2 2 2
72.	187039 			0.9514 0.9890 0.9410	0.9420 0.9883 0.9310	0.9459 0.9883 0.9348	0.9504 0.9901 0.9401	0.9473 0.9886 0.9365	1(0.000) 2(0.1200) 3(1.0700) 4(0.2100) 5(1.2800)	2 2 7 3 4

73.	187029 			0.9392 0.9893 0.9291	0.9439 0.9907 0.9351	0.9534 0.9920 0.9458	0.9401 0.9868 0.9305	0.9354 0.9871 0.9135	1(0.8700) 2(1.1000) 3(0.0900) 4(0.4800) 5(0.3200)	2 3 2 2 2
74.	183087 			0.9608 0.9916 0.9528	0.9212 0.9836 0.9060	0.9571 0.9916 0.9491	0.9490 0.9885 0.9381	0.9571 0.9910 0.9484	1(0.000) 2(0.1200) 3(1.0700) 4(0.2100) 5(1.2800)	2 2 2 2 2
75.	183055 			0.9545 0.9950 0.9504	0.8941 0.9831 0.8790	0.9555 0.9912 0.9471	0.9558 0.9941 0.9501	0.9449 0.9932 0.9385	1(1.1300) 2(0.8200) 3(0.1400) 4(0.3200) 5(1.0900)	2 2 3 2 2
76.	182053 			0.9395 0.9914 0.9314	0.9419 0.9917 0.9341	0.8977 0.9847 0.8840	0.8946 0.9833 0.8796	0.9419 0.9922 0.9345	1(0.0800) 2(0.3600) 3(0.0800) 4(0.4300) 5(1.4300)	2 2 2 2 3

77.	181079 			0.9342 0.9883 0.9233	0.9347 0.9873 0.9228	0.9341 0.9887 0.9236	0.9354 0.9866 0.9228	0.9354 0.9882 0.9243	1(0.1300) 2(0.1300) 3(0.0300) 4(0.1300) 5(0.1400)	2 5 2 4 2
78.	181091 			0.9347 0.9899 0.9252	0.9392 0.9874 0.9311	0.9159 0.9888 0.9056	0.9285 0.9899 0.9192	0.9306 0.9917 0.9229	1(0.6400) 2(0.0300) 3(0.1200) 4(0.3200) 5(0.0300)	2 2 2 2 2
79.	181018 			0.9419 0.9900 0.9325	0.9145 0.9825 0.8985	0.9116 0.9859 0.8987	0.9276 0.9874 0.9159	0.9183 0.9844 0.9039	1(0.0300) 2(0.2200) 3(0.1300) 4(0.0200) 5(0.0900)	6 10 9 10 8

80.	176039 			0.9576 0.9953 0.9531	0.9708 0.9966 0.9675	0.9576 0.9449 0.9527	0.9576 0.9944 0.9523	0.9176 0.9741 0.8938	1(0.9200) 2(0.2600) 3(0.1900) 4(0.0900) 5(0.5700)	3 2 2 2 3
81.	175032 			0.9197 0.9728 0.8947	0.9353 0.9801 0.9206	0.8778 0.9561 0.8393	0.9427 0.9802 0.9240	0.9562 0.9783 0.8974	1(0.4300) 2(0.4400) 3(0.8700) 4(1.3600) 5(0.1000)	3 7 6 2 2
82.	172032 			0.9459 0.9952 0.9414	0.9325 0.9946 0.9275	0.9458 0.9958 0.9418	0.9490 0.9955 0.9447	0.9032 0.9899 0.8940	1(0.1300) 2(1.0000) 3(0.7800) 4(0.4600) 5(0.8700)	4 2 3 3 3
83.	167083 			0.9586 0.9919 0.9508	0.9150 0.9821 0.8986	0.9578 0.9921 0.9502	0.9563 0.9907 0.9475	0.9589 0.9919 0.9511	1(0.0300) 2(0.8900) 3(0.2900) 4(0.3000) 5(1.4200)	3 3 2 2 6
84.	166081 			0.9396 0.9893 0.9295	0.9495 0.9924 0.9422	0.9174 0.9872 0.9057	0.9151 0.9857 0.9020	0.9257 0.9882 0.9148	1(1.2000) 2(0.3500) 3(0.8200) 4(0.6900) 5(1.1800)	2 4 2 2 2
85.	163014 			0.9272 0.9895 0.9175	0.9291 0.9888 0.9187	0.9307 0.9881 0.9196	0.9274 0.9884 0.9167	0.9236 0.9869 0.9115	1(0.9100) 2(0.2400) 3(1.4300) 4(0.9000) 5(0.9700)	2 2 2 2 2
86.	161062 			0.9484 0.9945 0.9432	0.9496 0.9945 0.9444	0.9503 0.9444 0.9430	0.9476 0.9945 0.9423	0.9449 0.9932 0.9385	1(0.7200) 2(0.4500) 3(0.7000) 4(0.4500) 5(0.8700)	3 4 3 2 2

87.	159091				0.9467 0.9896 0.9369	0.9574 0.9918 0.9495	0.9428 0.9884 0.9319	0.9428 0.9873 0.9308	0.9428 0.9884 0.9310	1(0.9000) 2(0.8800) 3(0.9800) 4(0.3000) 5(0.3900)	2 3 2 2 2
88.	159029				0.9287 0.9830 0.9129	0.9421 0.9863 0.9292	0.9433 0.9875 0.9315	0.9429 0.9872 0.9308	0.9429 0.9876 0.9313	1(0.6100) 2(0.0600) 3(0.7800) 4(0.3200) 5(0.8100)	3 4 2 2 2
89.	148089				0.9288 0.9823 0.9123	0.9034 0.9811 0.8863	0.9456 0.9870 0.9333	0.9357 0.9855 0.9221	0.9484 0.9885 0.9375	1(1.5000) 2(0.2400) 3(0.5200) 4(0.0900) 5(0.0400)	3 2 2 2 2

90.	147091				0.9366 0.9917 0.9289	0.9280 0.9928 0.9284	0.9441 0.9936 0.9380	0.9280 0.9897 0.9185	0.9249 0.9915 0.9270	1(0.1600) 2(0.6200) 3(0.0400) 4(0.0800) 5(0.6800)	2 2 2 2 2
91.	147062				0.9391 0.9864 0.9264	0.9331 0.9839 0.9181	0.9165 0.9872 0.9048	0.9358 0.9875 0.9241	0.9567 0.9896 0.9467	1(0.7400) 2(0.3200) 3(1.0600) 4(1.1100) 5(1.4200)	2 2 2 2 2
92.	145086				0.9465 0.9916 0.9386	0.9446 0.9922 0.9373	0.9165 0.9872 0.9048	0.9266 0.9897 0.9241	0.9266 0.9884 0.9158	1(0.3500) 2(0.4900) 3(1.1800) 4(0.5800) 5(0.4000)	9 2 3 2 2 2

In this ,we apply the code on the input image and obtain the result and after this we apply the canny method on the image and after that we take complement of the last obtain image .Now, we pass the image and compare the complemented image with available five ground truths by converting the complemented image and ground truth into one – directional image . After that we calculate the precision , recall and fscore value by using the respective formulae for the same .

$$Pre = t_p / (t_p + f_p);$$

$$Rec = t_p / (t_p + f_n);$$

$$fscore = 2(Pre * Rec) / (Pre + Rec);$$

Table 4.4

Mean F-score of 100 images for all Non-Extensive and Shannon Entropy .

Different Methods	Mean F-Score
Mean-Shift with bandwidth fixed to 0.8	0.8448
Automated Mean-Shift using Shannon Entropy	0.8550
Automated Mean-Shift using Non-Extensive Entropy	0.9222

By looking at the results obtained, we came to know that the Non-Extensive entropy is better than other . It could be because Non-Extensive Entropy is non additive Entropy .so, Non-Extensive Entropy can identify the correlated information content and reduces the number of overall clusters .

It happens because Shannon is additive and it add all the clusters as well but its result not get influenced by some value of clusters as it include the log in its formula .

$$H_1 = -\sum_{i=1}^n p_i \log p_i$$

In case of Non-Extensive Entrophy , it add up for all clusters but its result get affected by some value of clusters as it includes exponential in its formula .

$$H_2 = \sum_{i=1}^n -p_i e^{-p_i}$$

There are some cases which shows that the Automated MeanShift using Non-Extensive Entropy is better than other two methods .

In case of image 25058.jpg , when we apply the automated code it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 10 clusters and sums them up and calculate the entropy where as in other case , it visit only 2 clusters and sums them up and calculate the entropy . Results of Non Extensive Entropy are good in visualization than other . It is found that Non-Extensive Entropy (0.9408) is higher than Shannon Entropy(0.90100) due to which results of segmentation came better.

After applying the program on image 15088.jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 7 clusters and sums them up and calculate the entropy where as in other case , it visit only 8 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9592) is higher than Shannon Entropy(0.9529) and segmentation results are better.

For the image 22013.jpg , when we apply the program for Automated MeanShift using Shannon Entropy , it visit the all 10 clusters and sums them up and calculate the entropy where as in other case , it visit only 4 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9220) is higher than Shannon Entropy(0.95291) due to which results of segmentation came better.

When took the image 35008.jpg , and applied the concept and it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 10 clusters and sums them up and calculate the entropy where as in other case , it visit only 2 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9068) is higher than Shannon Entropy(0.9046) due to which results of segmentation came better.

In case of image 35010.jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 10 clusters and sums them up and calculate the entropy where as in other case , it visit only 2 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.87124) is higher than Shannon Entropy(0.86852) due to which results of segmentation came better.

In case of image 35058.jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 8 clusters and sums them up and calculate the entropy where as in other case , it visit only 3 clusters and sums them up and calculate the entropy for the same . It is found that Non-Extensive Entropy (0.9311) is higher than Shannon Entropy(0.9292) which satisfies our concept .

For image 118020 .jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 10 clusters and sums them up and calculate the entropy where as in other case , it visit only 2 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9063) is higher than Shannon Entropy(0.9036) due to which results of segmentation came better.

In case of image 118035.jpg , on passing it in the code of Automated MeanShift using Shannon Entropy , it visit the all 9 clusters and sums them up and calculate the entropy where as in other case , it visit only 5 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9307) is higher than Shannon Entropy(0.9283) supports the segmentation .

In case of image 124084.jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 9 clusters and sums them up and calculate the entropy where as in other case , it visit only 2 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9154) is higher than Shannon Entropy(0.9001) due to which results of segmentation came better.

In case of image 353013.jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 9 clusters and sums them up and calculate the entropy where as in other case , it visit only 3 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.9275) is higher than Shannon Entropy(0.9239) due to which results of segmentation came better.

In case of image 302003.jpg , it is found that in case of Automated MeanShift using Shannon Entropy , it visit the all 10 clusters and sums them up and calculate the entropy where as in other case , it visit only 3 clusters and sums them up and calculate the entropy . It is found that Non-Extensive Entropy (0.88306) is higher than Shannon Entropy(0.87864) due to which results of segmentation came better.



The results obtained by using Automated Meanshift on variable bandwidth of 0.01 to 1.5 and different results are obtained . In this we found the modes successfully .

To execute the code , we have used the matlab 2013 for making the code to run successfully .

The configuration of the system used is 4gb Ram , i5 processor .

# Conclusion

After implementing the code for the different value of bandwidth in different methods , we came to know that the Mean shift with bandwidth fixed to 0.8 shows the good results instead of other bandwidth and shows the good segmentation of image , whereas it doesnot show good results on other value of bandwidth . In this case , we used the brute force technique to obtain the correct value of bandwidth .i.e. 0.8 .

After that we executed the Auto-Segmentation using Shannon and Non-Extensive entropy we get the segmented image which better in visualization . In this , we took the image and apply the code and calculate the precision , recall and f-score for five different ground truth and came to know that it show the better value of f-score which proves that our results are better for wide range of bandwidth and clusters instead of fixed value of bandwidth of 0.8 . Non –Extensive visits the less number of cluster whereas Shannon entropy takes the more number of clusters .In this , Auto-Segmentation using Non-Extensive entropy shows better value of f-score and it could be because it is non-additive whereas Shannon is additive Entropy . Among the three methods applied in this , Auto-Segmentation using Non-Extensive Entropy shows the higher value of f-score which shows that the result is better than other methods .

As far Auto-Segmentation using Mean-Shift method has proved to be great tool for accessing information of importance . The result obtained provide us with a good value of Precision , Recall , F-score . This also compares the result obtained with algorithm with different available ground truth and shows that we improvise the Mean-Shift Algorithm to some extent .

# References

- [1] M.Hanmandlu S. Susan V.K. Madasu B.C. Lovel ,Fuzzy co-clustering of medical Images using Bacterial foraging ,IEEE,2008
- [2] Savita Agrawal , Deepak Kumar Xaxa, Survey on Image Segmentation Techniques and Color Models, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3025-3030
- [3] Jun Tang, A Color Image Segmentation algorithm Based on Region Growing, V6-636 2010 2nd International Conference on Computer Engineering and Technology
- [4] Michael E. Farmer, Member, IEEE, and Anil K. Jain, Fellow, IEEE, A Wrapper-Based Approach to Image Segmentation and Classification, IEEE transactions on image processing, vol. 14, no. 12, december 2005
- [5] Hui Zhang ,Quanyin Zhu , Xiang-feng Guan ,Image Segmentation Based on Sobel Operator and Maximum Entropy Algorithm, International Conference on Computer Science and Service System,2012
- [6] Juyong Zhang Jianmin Zheng Jianfei Cai, Nanyang Technological University, Singapore
- [7] Mei Yeen Choong, Image Segmentation via Normalised Cuts and Clustering Algorithm, IEEE International Conference on Control System, Computing and Engineering, 23 - 25 Nov. 2012, Penang, Malaysia,2012
- [8] Dorin Comaniciu, Peter Meer, Mean shift :A Robust Approach Toward Feature Space Analysis ,IEEE Transaction on Pattern Analysis and Machine Intelligence,May 2002
- [9] S.N. Jayaramamurthy and T.L. Huntsberger, Edge and region analysis using fuzzy sets,Proc. IEEE Workshop on Language Automation,pp. 205-9, 1985.
- [10] A. Tremeau, and N. Borel, "A region growing and merging algorithm to colour segmentation",Pattern Recognition, vol. 30, no. 7, pp. 11911203,1997.
- [11] Noha El-Zehiry, Prasanna Sahoo, Steve Xu, and Adel Almaghraby,“Graph cut optimization for the mumford shah model,” in Proc. of the 7th IASTED Int. Conf. on Visualization,Imaging, and Image Processing, VIIP, August 2007.
- [12]Dorin Comaniciu,Peter Meer, Mean Shift : A Robust Approach Toward Feature Space Analysis,IEEE transactions on Pattern Analysis and machine intelligence , vol.24,No.5,May 2002
- [13] Dorin Comaniciu, Visvanathan Ramesh , Peter Meer , [ieeexplore.ieee.org](http://ieeexplore.ieee.org) :The Variable Bandwidth Mean Shift and Data-Driven Scale Selection .
- [14] M. Hanmandlu, S. Susan, V.K. Madasu, B.C. Lovell “Fuzzy Co-Clustering of medical images using bacterial foraging “ in mage and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference; 12/2008

- [15] Dorin Comaniciu, Member, IEEE "An Algorithm for Data-Driven Bandwidth Selection " in IEEE transactions on pattern analysis and machine intelligence, vol. 25, no. 2, february 2003
- [16] Fast Mean Shift by Compact Density Representation by Daniel Freedman and Pavel Kisilev Hewlett-Packard Laboratories Haifa, Israel.
- [17] Seba Susan and Madasu Hanmandlu. "A non-extensive entropy feature and its application to texture classification." Neurocomputing 120 (2013): 214-225.
- [18] Drive Database can be downloaded from the given url  
[http://www.isi.uu.nl/Research/Databases/DRIVE/process\\_download.php](http://www.isi.uu.nl/Research/Databases/DRIVE/process_download.php)
- [19] Berkeley Database can be downloaded from the given url-  
<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html>
- [20] Y Cheng - Pattern Analysis and Machine Intelligence, IEEE , Mean shift, mode seeking, and clustering1995
- [21] D Comaniciu, P Meer - Pattern analysis & applications," Distribution free decomposition of multivariate data" ,1999
- [22] D Comaniciu - Pattern Analysis and Machine Intelligence, An algorithm for data-driven bandwidth selection IEEE
- [23] B Georgescu, I Shimshoni... - Computer Vision, Mean shift based clustering in high dimensions: A texture classification example 2003
- [24] Comaniciu, D.; Ramesh, V.; Meer, P." Kernel-based object tracking"  
 Pattern Analysis and Machine Intelligence, IEEE Transactions on
- [25] Barash, D.; Comaniciu, D." Meanshift clustering for DNA microarray analysis" Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE
- [26] Tada, K.; Takemura, H.; Mizoguchi, H." Robust tracking method by MeanShift using Spatiograms" SICE Annual Conference 2010, Proceedings of  
 Year: 2010
- [27] Taekyung Ryu; Ping Wang; Suk-Ho Lee "Image compression with meanshift based inverse colorization"
- [28] Yunji Zhao; Hailong Pei; Baoluo Liu," Meanshift algorithm based on kernel bandwidth adaptive adjust"
- [29] Kai Du; Yongfeng Ju; Yinli Jin; Gang Li; Yanyan Li; Shenglong Qian," Object tracking based on improved MeanShift and SIFT
- [30] Yonghong Long; Xiyu Xiao; Xiaohua Shu; Shenglan Chen," Vehicle Tracking Method Using Background Subtraction and MeanShift Algorithm"