1. INTRODUCTION

1.1 INTRODUCTION

India is the land of agriculture. Farmers have an option to select required crops and then find appropriate pesticides for the plant to decrease the disease and increase the production. The cultivated plants will not always be healthy. In-order to increase the production with good quality the plant need to be monitored frequently because the plant disease leads to reduction of the product. For successful cultivation, one should monitor the health as well as the disease of the plant. Diseases in plant cause heavy loss of the product.

India is the world's highest producer of pomegranate. India exports 54000 tons of Pomegranate which makes 1.55% of total export in the whole world. Pomegranate is a vital fruit crop, because of its health benefits.10-31% of production reduces because of the disease found on the leaf of the pomegranate plant. Hence the disease need to be identified at the early stages, recommending farmers to avoid the harm in production of the crop to increase the yield.

Plants suffers from leaf diseases like: Alternaria Alternata (fungal), Anthracnose, Bacterial Blight (bacteria), and Cercospora Leaf Spot. Plant disease will be basically identified by observing different patterns on the parts of the plant like leaf, fruit and stem. The indications on the leaf is taken into consideration for detecting the disease.

In the regards of a leaf which is diseased can be said as the physiology isn't normal as it is for the leaf which is absolutely fine. So we can check before the whole leaf gets infected and the productivity is decreased. We can check for the infected area. Once the farmer comes to know there's something wrong with the leaf. Because of that leaf all the plants are in danger of getting infected. Before this happens there's need to identify the disease. And after knowing it farmers can try to cure the leaves by various ways. The main thing is symptom, which denotes the proof of the presence of something. By gauging the answers about the leaf are diseased, which part is

diseased can be helpful for successful cultivation. After the identification process of the disease the farmers can go for the next step that is the curing the disease with which the leaf is infected. The symptoms and the disease attack play vital role in successful farming.

1.1.1 IMAGE DEFINITION

Image is a collection of pixels or dots which are stored in rectangular array. Each individual pixel is having certain kind of color. We can measure the size of the image by counting the no of pixels in that particular image. Different type of images are there such as Black and White and Grey scale images. Both types vary from each other .In black and white image each dot or pixel is either black or white, therefore only one bit is needed per pixel. Whereas Grey scale images uses 8 bits per pixel. For color images things gets slightly difficult. In color images number of bits at every dottermed as the height of image. It is also referred as the bit plane. For bit plane consisting of, 2x color are possible. Different methods are available to store the color information of image. One of the method is RGB image also termed as true color image. For every pixel red, green and blue components stored in three dimensional array.

1.1.2 IDENTIFYING PATTERNS

In order to detect the same kind of pattern different pattern recognition techniques are used in MATLAB. Using these techniques we recognize the similar kind of the pattern in the problem. When same kind of pattern are detected then these can be used to generate outputs or solve the problems more efficiently. In order to recognize the pattern, we need to train the machine. For this first we need to classify the data .The data is classified using the key features .For classifying the data we have different type of learning modules is there such as supervised learning and unsupervised learning modules .Both of these modules are used to identify the patterns. In supervised learning module we train the machine by recognizing the patterns in the data set and then results which are generated are applied to the testing data set.

We train the machine over the training dataset and test it over the testing data set. In unsupervised learning module, there are no visible pattern the dataset, so with the help of the

some algorithm we try to catch the patterns. Clustering algorithm, classification algorithm

such as Support Vector Machines(SVM) is there. For recognizing the patterns we identify, we have different techniques such as preprocessing, Extraction of features and classification. In preprocessing we try to filter out, smooth the data by normalizing in more ordered way. Filtering such as noise filtering is there. Feature extraction is usually done using the software which collect the information from the data and the final phase is the classification.

1.2 PROBLEM STATEMENT

The main purpose is detect the diseased part of the plant. Using MATLAB convolutional neural networks are implemented in order to classify the diseased part. Aim is to detect the diseased part by finding the optimum way with minimum cost. In this problem we have considered fundamental five categories of the plant leaf disease which are Alternaria Alternata, Anthracnose, Bacterial Blight, Cercospora leaf spot and Healthy Leaves. All of these disease belongs to fungal, viral or bacterial type of the diseases. In our proposed solution we identify the percentage of the affected area and identify the disease. Our approach provide the result in minimum time span with maximum precision and accuracy in comparison to other existing approaches.

In early days, the monitoring and analysis of plant diseases were done manually by the expertise person in that field. This needs tremendous amount of work and conjointly requires excessive processing time. The image processing techniques can be used in the plant disease detection. In most of the cases disease symptoms are seen on the leaves, stem and fruit. The plant leaf for the detection of disease is considered which shows the disease symptoms. In existing system, various image processing techniques such as Probabilistic Neural Network, Genetic Algorithm, Support

Vector Machine are used. But they have some negative features like the quality of result can vary for different input data, requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time.

To overcome these strikes, this paper mainly focus on some image processing techniques like

- Pre-processing which can resize and convert to black and white image.
- OTSU method which is used to contrast and enhance the effected leaf.
- The image can be clustered according to the leaf color, shape and size using K-means clust
- ering.
- The GLCM can extract the texture and color of the image.
- Finally, Multi Class SVM classify the effected leaf according to the dataset.

1.3 PLANT DISEASE FUNDAMENTALS

There are different types of plant disease exist, but majority of these disease can be categorize into the three different categories which are bacterial disease viral disease and the fungal disease. The most ideal way to detect the disease is the classification followed by detection. Classification is done on the basis of shape and texture features.

1.3.1 Bacterial disease

This is also known as bacterial leaf spot. Bacterial leaf spot is majorly detected in stone fruits such cherry, plum etc. In this disease black spots or dark spots occur on the different part of leafs. Yellow halos is also symptom of this disease. Spot size is of irregular nature. Bacterial spots occur on the different part on the top and bottom start occurring and if these spots cluster together in any section of the leaf then this results in killing of that section by this disease. Wet and cool formation also contribute to the formation of the bacterial disease in the plant leaves. In these formations bacterial leaf spot can spread very quickly. Mostly

bacterial leaf spot occur on the aged leaves but it can destroys the tissues of the new leaves too.

1.3.2 Viral Disease

Viral disease are caused by viruses and as virus are intercellular, so these diseases attacks inside out. Viral disease are sometime very difficult to identify. Virus can affect any region of the plants such leafs, roots, stem and others. Abnormal patterns are observed on the affected area green and yellow coloration is seen in leaves affected with the virus. The life span of the plant or its parts affected with the viral disease is very less. It directly affects the productivity and other factors. Wrinkles on the different part of the leaves is also primary symptom of these disease. Every virus life span is very high as compared to the other types of the disease, because each virus if not properly cured give rise to new type of the virus so it is important for timely prevention of these disease.

1.3.3 Fungal Disease

Fungal disease occur because of the fungi or fungal organism. One of the property of the fungi is that it spread with wind and the water. Gray green spots on the leaf of the plants are observed and if not properly cured they start getting spread toward the outer region of the leaf. Wilting, scabs are the primary symptoms fungal disease. Fungal disease attack on the plant leafs result in the yellowness of leaves at end.

1.4 OBJECTIVES

- To apply and implement the new algorithm to monitor the growth of the crop anddetect the diseased part in the plant.
- To incorporate the global optimization strategy for enhancing the optimal solution.
- To modify the existing solution more efficiently to solve this problem is more genuinewa

2. LITERATURE SURVEY

In [2] author proposes different technologies of leaf disease detection using image processing approach and classified them based on the type of analysis tool and applications. Less time consuming and automatic diagnosis is the major requirement in agriculture to improve the crop production rate. Leaf colour ,size and texture changes with climate and environment conditions. The field expert and regular observations are required well in time. The scope of development of hybrid algorithms such as genetic algorithms, cuckoo optimizations and ant colony etc in order to increase the recognition rate of the final classification process.

In [3] the author is used for testing vineyard diseases based on photographs with grape leaves. A success rate higher than 90% has been achieved in the disease recognition process. The following lesion features: number of spots, their grey level, and area and then extracts a histogram indicating the number of pixels that have specific red, green or blue color level. It is assumed that the background is much brighter than the plant color in the present application version in order to avoid complicated and time consuming background separation technique. It analysis the color features of the spots in plant parts. It was evaluated on grape diseases with an accuracy that exceed 90% using small training set. The recognition of disease can often be based on symptoms like lesions and spots in various parts of plants. The color , area and the number of these spots can determine to great extent the disease that has mortified a plant.

In [4] the main objective is to diagnose the disease of brinjal leaf using image processing and artificial neural techniques. The diseases on the brinjal are critical issue which makes the sharp decrease in the production of brinjal. The study of interest is the leaf rather than the whole brinjal plant because about 85 to 95% of diseases occurred on the brinjal leaf like bacterial wilt, cercospora leaf spot tobacco mosaic virus. The leaf spot disease is considered in this work and it is possible to identify the disease using k-means clustering algorithm and ANN. Various parameters are computed as area, perimeter, centroid, diameter and mean intensity for identifying brinjal diseases.

In [5] author describes visual symptoms of plant disease from analysis of colored images using image processing methods that has been proposed. Color co-occurance for feature extraction is also proven to be helpful in many of plant disease detection based on color and texture. It is useful to benefit oil palm industry demands. High end image

capturing device has been used to capture images of leaf surface followed by extraction of features like shape color and texture of disease types. K-means clustering to detect infected object and neural networks are thus commonly used for obtaining accuracy in detecting and classifying the diseases. Image processing provides more efficient ways to detect diseases caused by fungus, bacteria or virus on plants. It presents an overview of using image processing methods to detect various plant diseases.

In [6] the proposed algorithm is tested on main five types of plant diseases like Ashen Mold, Early Scorch, Cottony

mold, late scorch tiny whiteness. The computer can automatically classify 32kinds of plants from the leaf images

loaded from digital cameras or scanners. Probabilistic Neural Network (PNN) is adopted for it has fast speed on

training and simple structure. Detect the symptoms of the disease occurred in leaves in an accurate way. The symptoms are identified in the initial stage and classified them using the k-means algorthim and thus increases the recognition rate. Here test our program on five disease which effect on the plants they are early, scorch, cottony mold, ashen mold, late scorch, tiny whiteness.

The different classification techniques used for plant leaf disease classification was proposed by Savita N. Ghaiwat [7] by using some classification techniques like k-Nearest Neighbour Classifier, Probabilistic Neural Network, Genetic Algorithm, Support Vector Machine, and Principal Component Analysis, Artificial neural network, Fuzzy logic. A classification technique deals with classifying each pattern in one of the distinct classes and it is used to classify the leaf based on its different morphological features. Selecting classification technique is tricky task because the quality of result can vary for different input data.

The detection of plant diseases using their leaves images is explained by Sachin D. Khirade [8]. The studies of the plant diseases mean the studies of visually observable patterns seen on the plant. Health monitoring and disease detection on plant is very critical for sustainable agriculture. It is very tricky to monitor the plant diseases manually. It needs tremendous amount of work and conjointly requires excessive processing time. Hence, image processing is used for the detection of plant diseases. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification. But, the accuracy of the result is 86% only.

The technique to classify and identify the different disease affected plant put forth by Mrunalini R. Badnakhe [9].By using the automated agricultural inspection, Farmer can given potentially better and accurate productivity .The different products can be yield with better quality. The main needs for the agriculture is to predict the infected crop. With the help of this work we are indirectly contributing for the Improvement of the Crop Quality. It is a Machine learning based recognition system which will going to help in the Indian Economy. Digital Analysis of crop color is significant and now it's becoming popular day by day. It is the cost effective method. Because changed in the color are a valuable indicator of crop health and efficiency and survivability. Then it can be measured with visual scales and inexpensive crop color.

Software solution for automatic detection and classification of plant leaf diseases was proposed by S. Arivazhagan[10]. The developed processing scheme consists of four main steps, first a color transformation structure for the input RGB image is created, then the green pixels are masked and removed by specific threshold value using segmentation techniques, the texture information are computed for the useful segments, finally the extracted features are passed through the classifier. The proposed algorithm's efficiency can successfully detect and classify the examined diseases with an accuracy of 94%. Preparatory outcomes on an informational collection of around 500 plant leaves affirm the fitness of the proposed approach. In order to improve disease identification rate at various stages, the training samples can be increased and shape feature and color feature along with the optimal features can be given as input condition of disease identification.

In neural network it's difficult to understand structure of algorithm and to determine optimal parameters when training data is not linearly separable [11]. Hence, NNs are not suitable for our domain. It would be better to choose SVM classifiers which can be helpful for both linea and non linearseparability. Ghaiwat et al. presents survey on different classification techniques that can be used for plant leaf disease classification.For given test example, k-nearest-neighbor method is seems to be suitable as well as simplest of all algorithms for class prediction. If training data is not linearly separable then it is difficult to determine optimal parameters in SVM, which appears as one of its drawbacks [1].

K-means clustering and principal component classifier can be used to classify various plant diseases [12] Texture segmentation by co-occurrence matrix method and K-means clustering technique are the primary goals that are required for classification. This can be achieved by using K-means clustering. Authors present disease detection in Malusdomestica through an effective method like K-mean clustering, texture and color analysis [6]. To classify and recognize different agriculture, it uses the texture and color features those generally appear in normal and affected areas. In coming days, for the purpose of classification K-means clustering, Bayes classifier and principal component classifier can also be used.

The training samples can be increased and shape feature and color feature along with the optimal features can be given as input condition of disease identification .According to paper [13] disease identification process include some steps out of which four main steps are as follows: first, for the input RGB image, a color transformation structure is taken, and then using a specific threshold value, the green pixels are masked and removed, which is further followed by segmentation process, and for getting useful segments the texture statistics are computed. At last, classifier is used for the features that are extracted to classify the disease. The robustness of the proposed algorithm is proved by using experimental results of about 500 plant leaves in a database.

A method to select a threshold automatically from a gray level histogram has been derived from the viewpoint of discriminant analysis. [14] This directly deals with the problem of evaluating the

goodness of thresholds. An optimal threshold (or set of thresholds) is selected by the discriminant criterion; namely, by maximizing the discriminant measure q (or the measure of separability of the resultant classes in gray levels). The proposed method is characterized by its nonparametric and unsupervised nature of threshold selection and has many advantages.

3. SYSTEM REQUIREMENTS

3.1 H/W Requirements:

Processor: Intel core i7

RAM 4 GB (min)

Hard Disk 80 GB

3.2 S/W Requirements:

Operating System : Windows, Mac or Linux

Application Server : Matlab

Programming language : Matlab

ABOUT MATLAB

MATLAB tool is used for the solving the disease detection problem. This provide strong support for the implementation of the advanced algorithms. Machine learning algorithms and classifier too are very easily implemented. MATLAB also allows user to testing and training of the data. User can set various cutoffs for all the attributes and train the data on a given dataset and also using testing tools we can verify the data. In our problem we measure various parameters such as mean, correlation, contrast, variance, smoothness .[2]All of these parameters can be easily calculated using existing methods in the MATLAB. By applying the SVM classifier in MATLAB and passing the above parameters we can easily detect the type of disease in any leaf. MATLAB allow users to plot hue, saturation and intensity values for all type of leafs i.e diseased or healthy. All requirements for the implementation of above methodologies can be done very easily in MATLAB. Below are the reason that MATLAB is

used for the purpose

- Easy implementation of complex algorithm
- Previous history
- Accuracy and precision
- Advanced algorithm implementation
- Easy GUI implementation

Being a general programing language MATLAB is best fitted for the digital image processing. MATLAB allow users replication as it make sure that whatever steps user has implemented in the image processing they are properly documented by it . Moreover in MATLAB user can create interface very easily using guide command. There is no need to write code for the GUI as MATLAB automatically writes the coding modularities for the GUIs. All source codes are easily accessible for testing. Enhancement and segment process in image process is very easy in MATLAB.

4. THEORY

4.1. PREPROCESSING

Image processing is an emerging area in the research for various disease detection and diagnosis which aids practitioner for easy diagnosis of diseases accurately. Acquisition of image is a challenging task. Due to various defects in the image acquisition, the images may be affected with various noises. In order to overcome the issues, it is necessary for a method to eliminate the external entities such as noises; which affects the image during the acquisition. Hence, it is necessary to "pre-process" the images for noise elimination. Pre-processing is a process which is used to boost the precision and interpretability of an image. In Image processing, pre-processing of an image is very important so that the extracted image does not have any impurities, and it is accomplished to be better for the forthcoming process such as segmentation, feature extraction, etc. Only the correct segmentation of the spots on the leaf will yield the accurate result. Accurate detection leads to precise feature extraction and in turn gives perfect classification. The accurate spots segmentation is promising, only if the image is pre-processed clearly.

Preprocessing mainly aims to remove the clamor, stabilizing the intensity of the images and clear the artifacts. Image preprocessing is the technique of enhancing the image data prior to computational processing. In general, image preprocessing can be carried out in any one of the following forms:

1. Contrast enhancement
2. Histogram equalization
3. Intensity adjustment
4. Binarization
5. Morphological operation.

Out of all these methods, studies proved that Contrast enhancement s better for preprocessing, having high PSNR and low MSE values.

Contrast Enhancement:

Three functions are particularly suitable for contrast enhancement: imadjust, histeq, and adapthisteq. This example compares their use for enhancing grayscale and truecolor images.

Enhance Grayscale Images

- **imadjust** increases the contrast of the image by mapping the values of the input intensity image to new values such that, by default, 1% of the data is saturated at low and high intensities of the input data.
- **histeq** performs histogram equalization. It enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram (uniform distribution by default).
- **adapthisteq** performs contrast-limited adaptive histogram equalization. Unlike histeq, it operates on small data regions (tiles) rather than the entire image. Each tile's contrast is enhanced so that the histogram of each output region approximately matches the specified histogram (uniform distribution by default). The contrast enhancement can be limited in order to avoid amplifying the noise which might be present in the image.

Fig1.Original image and enhaced images using imadjust,histeq,adapthisteq

This is the output for contrast enhancement techniques. Of these outputs, the most reliable and suitable technique for our domain confines to imadjustcontranst enhancement technique.

A more gentle method for contrast enhancement is using imadjust. In its default form, this function maps pixel values in the original image to new, altered values while ensuring that only a small percentage (1 percent) of the values are saturated at low and high intensities of the original image. This results in a smoother transformation that mostly enhances useful details.

Syntax:

```
J = imadjust(I)
J = imadjust(I,[low_inhigh_in])
J = imadjust(I,[low_inhigh_in],[low_outhigh_out])
J = imadjust(I,[low_inhigh_in],[low_outhigh_out],gamma)
J = imadjust(RGB,[low_inhigh_in],___)
newmap = imadjust(cmap,[low_inhigh_in],___)
```

Description:

J = imadjust(I) maps the intensity values in grayscale image I to new values in J. By default, imadjust saturates the bottom 1% and the top 1% of all pixel values. This operation increases the contrast of the output image J.

You optionally can perform contrast adjustment using a GPU (requires Parallel Computing Toolbox™).

This syntax is equivalent to imadjust(I,stretchlim(I)).

J = imadjust(I,[low_inhigh_in]) maps intensity values in I to new values in J such that values between low_in and high_in map to values between 0 and 1.

J = imadjust(I,[low_inhigh_in],[low_outhigh_out]) maps intensity values in I to new values in J such that values between low_in and high_in map to values between low_out and high_out.

J = imadjust(I,[low_inhigh_in],[low_outhigh_out],gamma) maps intensity values in I to new values in J, where gamma specifies the shape of the curve describing the relationship between the values in I and J.

J = imadjust(RGB,[low_inhigh_in],___) maps the values in truecolor image RGB to new values in J. You can apply the same mapping or unique mappings for each color channel.

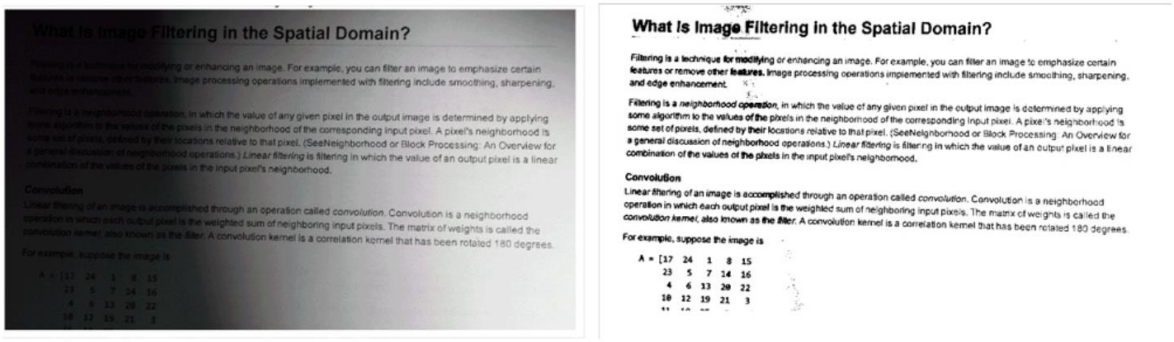
newmap = imadjust(cmap,[low_inhigh_in],___) maps the values in colormap cmap to new values in newmap. You can apply the same mapping or unique mappings for each color channel.

4.2. IMAGE SEGMENTATION

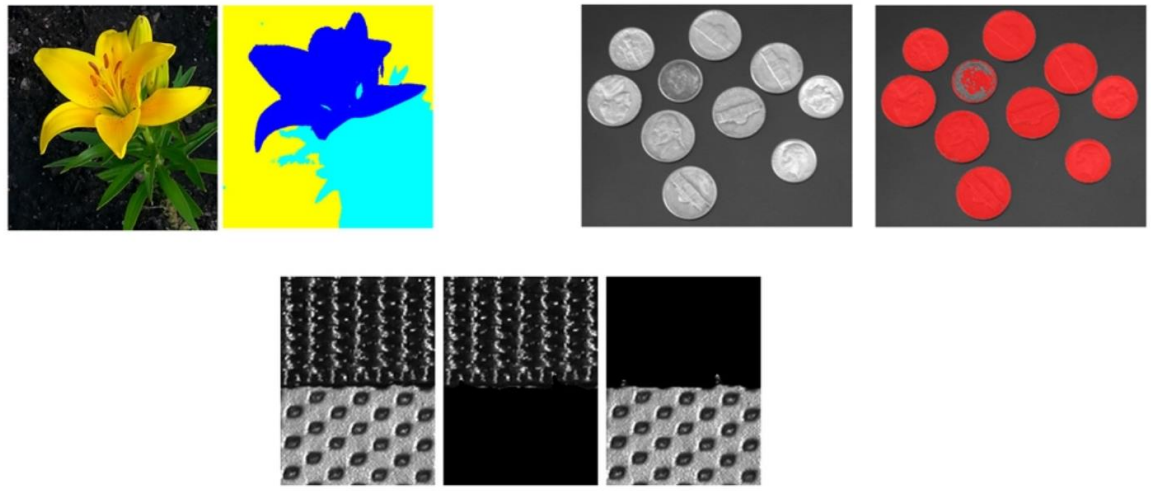
Image segmentation is a commonly used technique in digital image processing and analysis to partition an image into multiple parts or regions, often based on the characteristics of the pixels in the image. Image segmentation could involve separating foreground from background, or clustering regions of pixels based on similarities in color or shape. For example, a common application of image segmentation in medical imaging is to detect and label pixels in an image or voxels of a 3D volume that represent a tumor in a patient's brain or other organs.

We can divide or partition the image into various parts called segments. It's not a great idea to process the entire image at the same time as there will be regions in the image which do not contain any information. By dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works.An image is a collection or set of different pixels. We group together the pixels that have similar attributes using image segmentation.

Some techniques that follow this approach are region growing, clustering, and thresholding.



Using thresholding to convert to a binary image to improve the legibility of the text in an image.



Segmenting regions based on color values, shapes, or texture.

Fig2. Segmenting regions based on color values, shapes, or texture

In our project we have used Otsu Thresholding segmentation and later to identify the specific regions we have used k-means clustering.

Thresholding:

One simple way to segment different objects could be to use their pixel values. An important point to note – the pixel values will be different for the objects and the image's background if there's a sharp contrast between them.In this case, we can set a threshold value. The pixel values falling below or above that threshold can be classified accordingly (as an object or the background). This technique is known as **Threshold Segmentation**.

There are various methods to implement thresholding, either manual or automatic thresholding. Using manual thresholding, we cannot detect the correct threshold limit, that can actually give the

precise result. So, implementing the automatic thresholding technics will result in the accurate detection of the diseases.

Few automatic thresholding methods are as follows: Otsu segmentation, Hysteresis thresholding, Optimalthresholding. Out of which, Otsu segmentation gives the best efficiency.

Otsu's method

-A measure of region homogeneity is variance (i.e., regions with high homogeneity will have low variance).

- Otsu's method selects the threshold by minimizing the within-class variance of the two groups of pixels separated by the thresholding operator.

- It does not depend on modeling the probability density functions, however, it assumes a bimodal distribution of gray-level values (i.e., if the image approximately fits this constraint, it will do a good job).

Means and variance:

Consider that we have an image with L gray levels and its normalized histogram (i.e., for each gray-level value i, P(i) is the normalized frequency of i). - Assuming that we have set

the threshold at T, the -normalized- fraction of pixels that will be classified as background and object will be:

$$q_b(T) = \sum_{i=1}^{T} P(i), \qquad q_o(T) = \sum_{i=T+1}^{L} P(i) \quad (q_b(T) + q_o(T) = 1)$$

- The mean gray-level value of the background and the object pixels will be:

$$\mu_b(T) = \frac{\sum_{i=1}^{T} iP(i)}{\sum_{i=1}^{T} P(i)} = \frac{1}{q_b(T)} \sum_{i=1}^{T} iP(i) \qquad \mu_o(T) = \frac{\sum_{i=T+1}^{L} iP(i)}{\sum_{i=T+1}^{L} P(i)} = \frac{1}{q_o(T)} \sum_{i=T+1}^{L} iP(i)$$

- The mean gray-level value over the whole image (grand mean) is:

$$\mu = \frac{\sum_{i=1}^{L} iP(i)}{\sum_{i=1}^{L} P(i)} = \sum_{i=1}^{L} iP(i)$$

- The variance of the background and the object pixels will be:

$$\sigma_b^2(T) = \frac{\sum_{i=1}^{T}(i-\mu_b)^2 P(i)}{\sum_{i=1}^{T} P(i)} = \frac{1}{q_b(T)} \sum_{i=1}^{T}(i-\mu_b)^2 P(i)$$

$$\sigma_o^2(T) = \frac{\sum_{i=T+1}^{L}(i-\mu_o)^2 P(i)}{\sum_{i=T+1}^{L} P(i)} = \frac{1}{q_o(T)} \sum_{i=T+1}^{L}(i-\mu_o)^2 P(i)$$

- The variance of the whole image is:

$$\sigma^2 = \sum_{i=1}^{L}(i-\mu)^2 P(i)$$

• Within-class and between-class variance

- It can be shown that the variance σ can be written as follows:

$$\sigma^2 = q_b(T)\sigma_b^2(T) + q_o(T)\sigma_o^2(T) \quad + \quad q_b(T)(\mu_b(T)-\mu)^2 + q_o(T)(\mu_o(T)-\mu)^2 =$$

$$\sigma_W^2(T) + \sigma_B^2(T)$$

where $\sigma_W^2(T)$ is defined to be the within-class variance and $\sigma_B^2(T)$ is defined to be the between-class variance.

• Determining the threshold

- Since the total variance σ does not depend on T, the T minimizing σ_W^2 will be the T maximizing σ_B^2.

- Let's consider maximizing σ_B^2, we can rewrite σ_B^2 as follows:

$$\sigma_B^2 = \frac{[\mu(T) - \mu q_B(T)]^2}{q_B(T)q_o(T)}$$

where $\mu(T) = \sum_{i=1}^{T} iP(i)$

- Start from the beginning of the histogram and test each gray-level value for the possibility of being the threshold T that maximizes σ_B^2

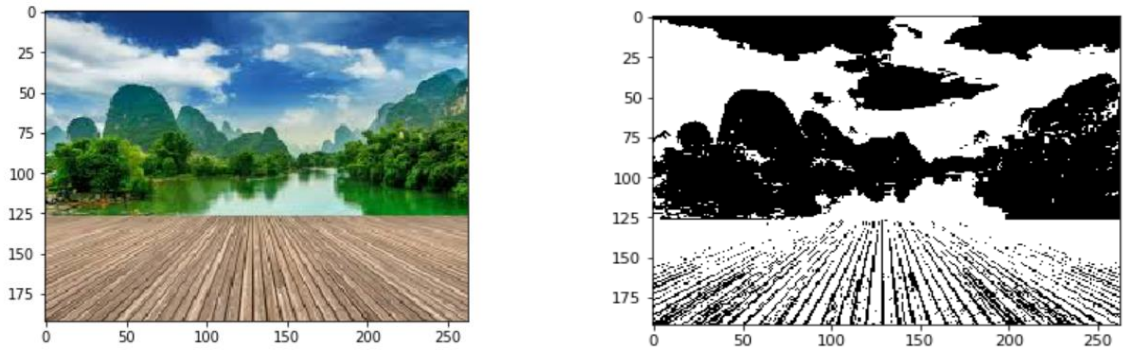


Fig 3.Image Segmentation

4.3. K-MEANS CLUSTERING

Clustering is the task of dividing the population (data points) into a number of groups, such that data points in the same groups are more similar to other data points in that same group than those in other groups. These groups are known as clusters.

One of the most commonly used clustering algorithms is k-means. Here, the k represents the number of clusters (not to be confused with k-nearest neighbor). Let's understand how k-means works:

1. First, randomly select k initial clusters
2. Randomly assign each data point to any one of the k clusters
3. Calculate the centers of these clusters
4. Calculate the distance of all the points from the center of each cluster
5. Depending on this distance, the points are reassigned to the nearest cluster
6. Calculate the center of the newly formed clusters
7. Finally, repeat steps (4), (5) and (6) until either the center of the clusters does not change or we reach the set number of iterations

Syntax

```
idx = kmeans(X,k)
```

idx = kmeans(X,k,Name,Value)

[idx,C] = kmeans(___)

[idx,C,sumd] = kmeans(___)

[idx,C,sumd,D] = kmeans(___)

Description

idx = kmeans(X,k) performs *k*-means clustering to partition the observations of the *n*-by-*p* data matrix X into k clusters, and returns an *n*-by-1 vector (idx) containing cluster indices of each observation. Rows of X correspond to points and columns correspond to variables.

By default, kmeans uses the squared Euclidean distance metric and the *k*-means++ algorithm for cluster center initialization.

idx = kmeans(X,k,Name,Value) returns the cluster indices with additional options specified by one or more Name,Value pair arguments.

For example, specify the cosine distance, the number of times to repeat the clustering using new initial values, or to use parallel computing.

[idx,C] = kmeans(___) returns the k cluster centroid locations in the k-by-*p* matrix C.

[idx,C,sumd] = kmeans(___) returns the within-cluster sums of point-to-centroid distances in the k-by-1 vector sumd.

[idx,C,sumd,D] = kmeans(___) returns distances from each point to every centroid in the *n*-by-k matrix D.

The key advantage of using k-means algorithm is that it is simple and easy to understand**.** We are assigning the points to the clusters which are closest to them. Let's put our learning to the test and check how well k-means segments the objects in an image.

k-means works really well when we have a small dataset. It can segment the objects in the image and give impressive results. But the algorithm hits a roadblock when applied on a large dataset (more number of images).It looks at all the samples at every iteration, so the time taken is too high. Hence, it's also too expensive to implement. And since k-means is a distance-based algorithm, it is only applicable to convex datasets and is not suitable for clustering non-convex clusters.

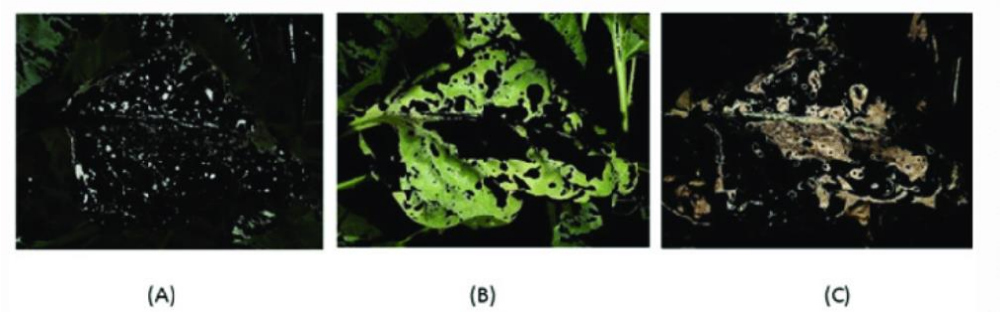


Fig 4.Three clusters formed from original image

4.4 GLCM FEATURE EXTRACTION

GLCM points to Gray level Co-occurrence matrix. It is of 2nd order statistics, so information with regards to pixels of pairs are collected by GLCM.GLCM exhibits the probability of how many times a combination of pixel pair co-occurs in an image section or in an image. A matrix is built up at a distance d=1 and at angles in degrees(0,45,90,135) .Haralick also offered different measures i.e. entropy, energy, contrast, correlation etc. These dimensions calculate at different angles.

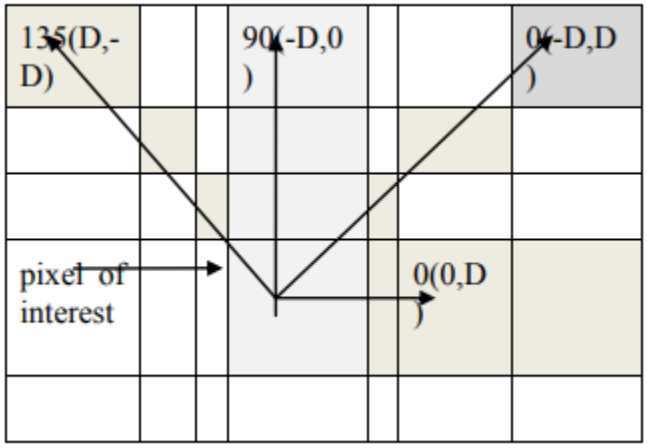
GLCM is texture character profile and this profile mention to touch i.e. smooth, silky and rough etc. The order of character profile statics are: First order texture measures are statistics declared from the original image values, like variance, and pixel neighbor relationship are not implemented. Second order measures defines the relationship between groups of two (usually neighboring) pixels in the original image. Third and higher order textures (noting the relationships among three or more pixels) are theoretically possible but practically/ commonly not implemented due to calculation time and interpretation difficulty.

GLCM texture picks up the relation between two pixels at a time, called the reference and the neighbor pixel. GLCM expounds the distance and angular spatial relationship over an image sub-region of specific size. GLCM is prepared from gray scale values. It is taken into account how often a pixel with gray level(gray scale intensity or gray tone) values come either horizontally, vertically and diagonally to leveled the pixels with the value j. GLCM directions are: Horizontal(0) Vertical(90) Diagonal a)bottom left to top right(-45) b)top left to bottom right (-135) They are announced as P0, P45, P90 and P135 respectively.

GLCM CALCULATION:

An input image of 8 tone is taken.

I .Sub-region/image:



1	1	5	6	8
2	3	5	7	1
4	5	1	1	2
8	5	1	2	5

Fig.5. GLCM direction analysis

II GLCM

1	2	0	0	1	0	0	0
0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0
1	0	0	0	0	1	2	0
0	0	0	0	0	0	0	1
2	0	0	0	0	0		0
0	0	0	0	1	0	0	0

Fig 6.GLCM matrix

Firstly fetch angle at 0 degree (horizontal).In the GLCM output, the element(1,1) has value1 because in the input image, there is only 1 opulence where two horizontally near to pixels of distance 1 having values 1 and 1.GLCM(1,2) has value 2 because in the input image there are two opulences where two horizontally near to pixels of distance 1 having value 1 and 2. GLCM(1,3) has value 0 because in the input image there are no opulence where two horizontally near to pixels of distance 1 having value 1 and 3.The procedure is repeated for the whole GLCM matrix at different angles. After forming matrices with four different angles,make them symmetric and then if we don't need any direction information then obtain an average matrix from the symmetric matrices and then normalize it to obtain probabilities.

The properties of GLCM are:

1. GLCM is of square in shape because the reference and neighbouring pixels have same range of values.

2. Number of rows and columns equal to the quantization level of the image. The test image consists of four gray level values that is 0,1,2 and 3.Eight bit data consists 256(2^8) possible values,256X256 matrix would be obtained,65536 cells.16 bit data having matrix of 65536X65536,having cells 429,496,720.

3. It is symmetrical about the diagonal. The diagonal elements pairs having no gray level difference(0-0,1-1,2-2,3-3etc).Most pixels are identical to their neighbouringcells,very less

contrast is there in the image.If there is a difference of 1 cell away from the diagonal,one level gray difference is there(0-1,1-2,1-3 etc).More the distance from the diagonal,more the gray level difference.

The texture rules according to the weight of the equation. The texture is grouped according to the degree. Square term second order equation. Cube term ways third order equation it is. The features extracted are:

Energy: It is a gray-scale image texture measure of homogeneity changing, reflecting the distribution of image gray-scale uniformity of weight and texture..

$$E = \sum_x \sum_y p(x, y)$$

p(x,y) is the GLCM

Contrast: Contrast is that diagonal close to the rotational inertia, that can measure the value of the matrix is scattered and local changes in number , reflect the image clarity and texture of shadow depth.

$$\text{Contrast} \qquad I = \sum\sum (x-y)^2 p(x,y)$$

Entropy: It measures image texture randomness, when the space co-occurrence matrix for all values are equal, it achieved the minimum value.

$$S = -\sum_x \sum_y p(x,y) \log p(x,y)$$

Correlation Coefficient: Measures the joint probability occurrence of the specified pixel pairs. Correlation: sum(sum((x-μx)(y-μy)p(xy)/σxσy))

Homogeneity: Measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. Homogenity = sum(sum(p(x , y)/(1 + [x-y])))

These are some of the second order statistics extracted from glcm matrix.

4.5 SUPPORT VECTOR MACHINES

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

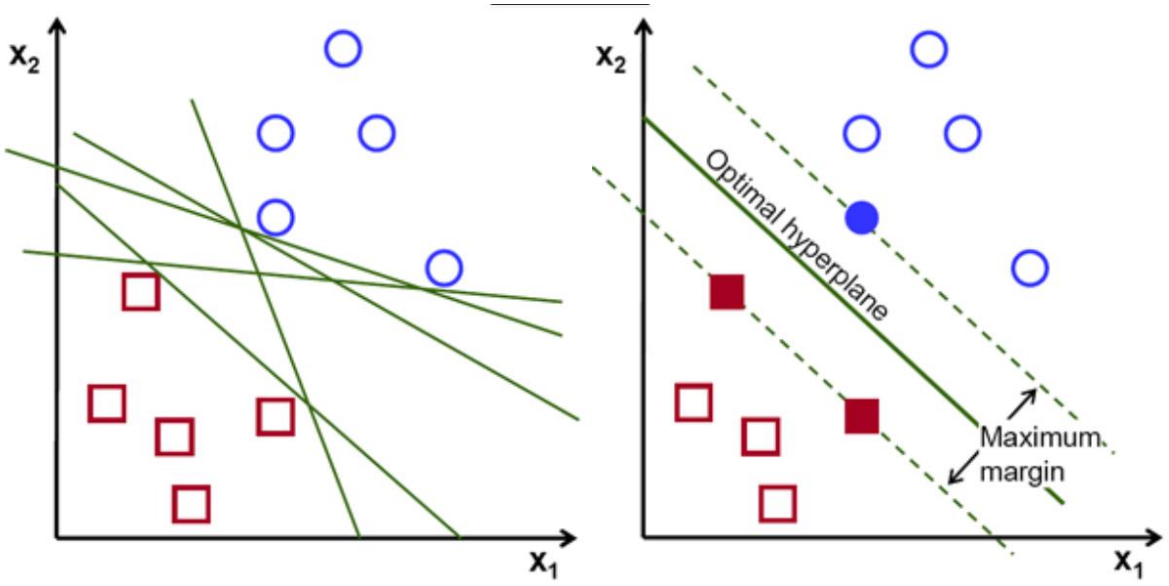


Fig 7.Support Vector machines hyperplane formation and selection

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyper Planes and Support Vectors:

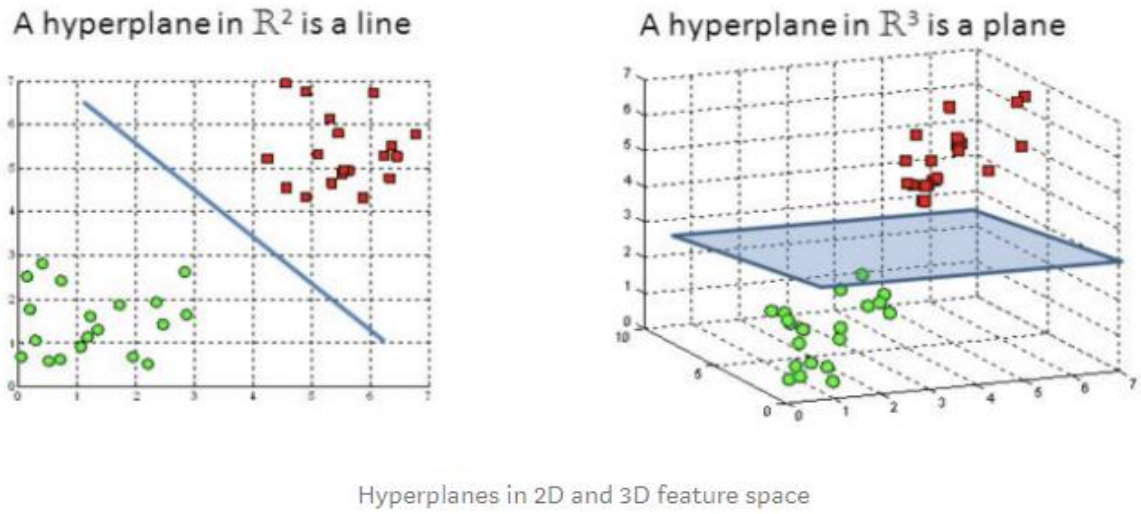
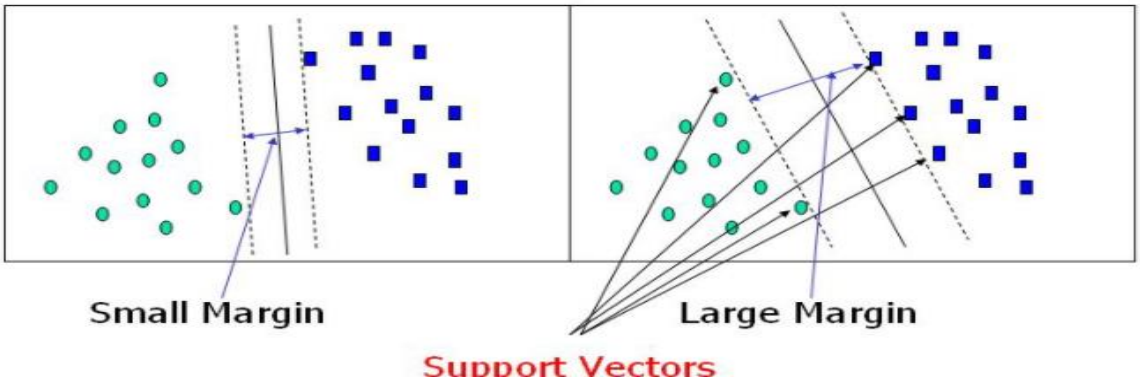


Fig 8.Hyperplanes in 2d and 3d feature space

Hyper planes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.That is why we use multisvm.



Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.

Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

4.6 MULTISVM

SVMs are inherently two class classifiers.So,in order to extend binary svm to multisvm we have two methods :onevs one method and one vs rest method.

One vs rest method: One-vs-rest (OvR for short, also referred to as One-vs-All or OvA) is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

The “one against all” strategy consists of constructing one SVM per class, which is trained to distinguish the samples of one class from the samples of all remaining classes. Usually, classification of an unknown pattern is done according to the maximum output among all SVMs.

Probability estimation:

The most intuitive approach to estimate posterior probability with the “one against all” strategy is to separately map the outputs of each SVM into probability using the method proposed by Platt [6], which consist of using an additional sigmoid:

$$\hat{P}\left(\omega_j \mid f_j(x)\right) = \frac{1}{1+\exp\left(A_j f_j(x) + B_j\right)}, \qquad (3)$$

where f j (x) denotes the output of the SVM trained to separate the class ! j from all the others. Then, for each sigmoid the parameters Aj and Bj are optimized by minimizing the local negative log-likelihood:

$$-\sum_{k=1}^{n} t_k \log\left(p_k\right) + \left(1-t_k\right)\log\left(1-p_k\right), \qquad (4)$$

where, pk denotes the output of the sigmoid and tk the probability target. To solve this optimization problem, Platt [6] proposes using a model-trust minimization algorithm based on the Levenberg-Marquardt algorithm. However, Lin et al. [7] showed that there are some problems with this

algorithm and propose using another minimization algorithm based on Newton's method with backtracking line search. However, nothing guarantees that:

$$\sum_{j=1}^{c} \hat{P}\left(\omega_j \mid f_j(x)\right) = 1. \qquad (5)$$

For this reason, it seems preferable to normalize the probabilities as follows:

$$\hat{P}\left(\omega_j \mid x\right) = \frac{\hat{P}\left(\omega_j \mid f_j(x)\right)}{\sum_{j'=1}^{c} \hat{P}\left(\omega_{j'} \mid f_{j'}(x)\right)}. \qquad (6)$$

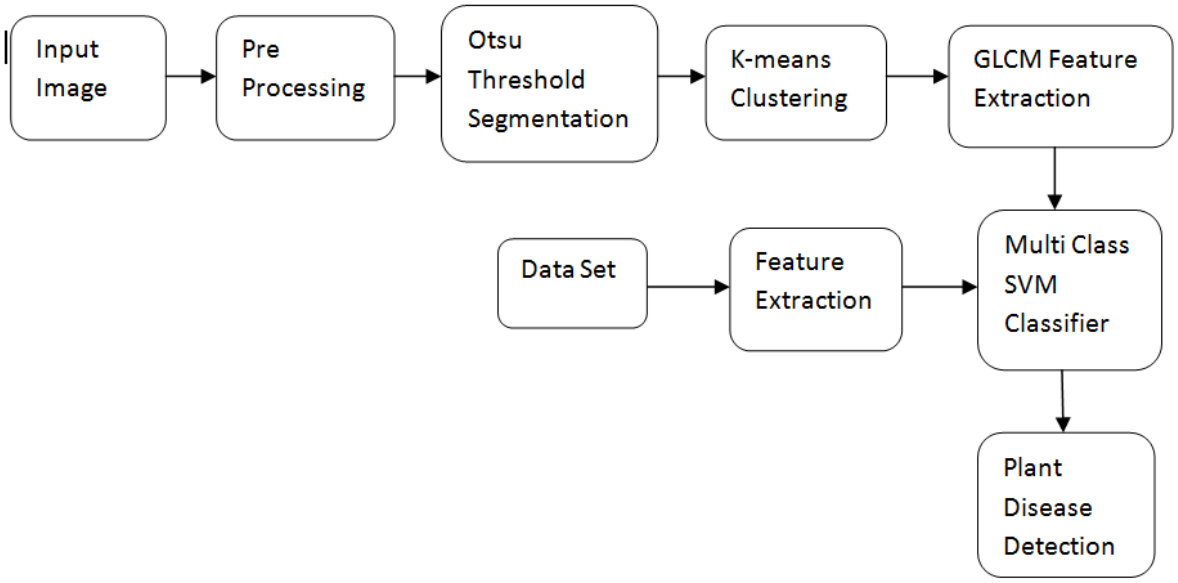
Another approach to estimate posterior probability with the "one against all" strategy would be to exploit the outputs of all SVMs to estimate overall probabilities. In order to do this, we propose using the softmax function, which can be regarded as a generalization of the sigmoid function for the multi-class case. Thus, in the same spirit as Platt's algorithm, we use a parametric form of the softmax function:

$$\hat{P}(\omega_j \mid x) = \frac{\exp\left(A_j f_j(x) + B_j\right)}{\sum_{j'=1}^{c} \exp\left(A_{j'} f_{j'}(x) + B_{j'}\right)}, \qquad (7)$$

and derive the parameters Aj and Bj by minimizing the global negative log-likelihood (Eq. 1).

Thus, it is necessary to construct a dataset of SVM outputs, which will be used to fix the parameters of sigmoid and softmax functions. The easiest way to do this is to use the same training samples used to fit SVMs; but, as pointed out by Platt [6], using the same data twice, can sometimes lead to a disastrously biased estimate. Therefore, it is preferable to derive an unbiased training set of the SVM outputs. A first solution would be to use a validation dataset; but, in our case, the number of samples in each class is not proportional to the prior probability. For this reason, it seems preferable to use cross-validation. Then, the training dataset was split into four parts. Each of four SVMs is trained on permutations of three out of four parts, and the SVM outputs are evaluated on the remaining fourth part. Finally, the union of all four sets of SVM outputs forms an unbiased dataset, which can be used to fix the parameters of functions. Furthermore, once the parameters are fixed, the final SVM is trained on the entire training set.

5. DESIGN



The steps involved are

1) Training – In training all the collected images are trained to the model and all six features are extracted and stored in the database.

2) Classification – After training, the SVM will classify the given new input as which type of disease is affected.

The system design mainly consists of

1) Image collection

2) Image Preprocessing

3) Image segmentation

4) Feature extraction

5) Training

6) Classification using Multiclass SVM

5.1 Image Collection:

The sample images of the diseased leaves are collected and are used in training the system. To train and to test the system, diseased leaf images and fewer healthy images are taken. The images will be stored in some standard format. The available images from the internet are also taken. The leaf images that are infected by AlternariaAlternata, Anthracnose, Bacterial Blight, Cercospora Leaf Spot and Healthy leaf are included

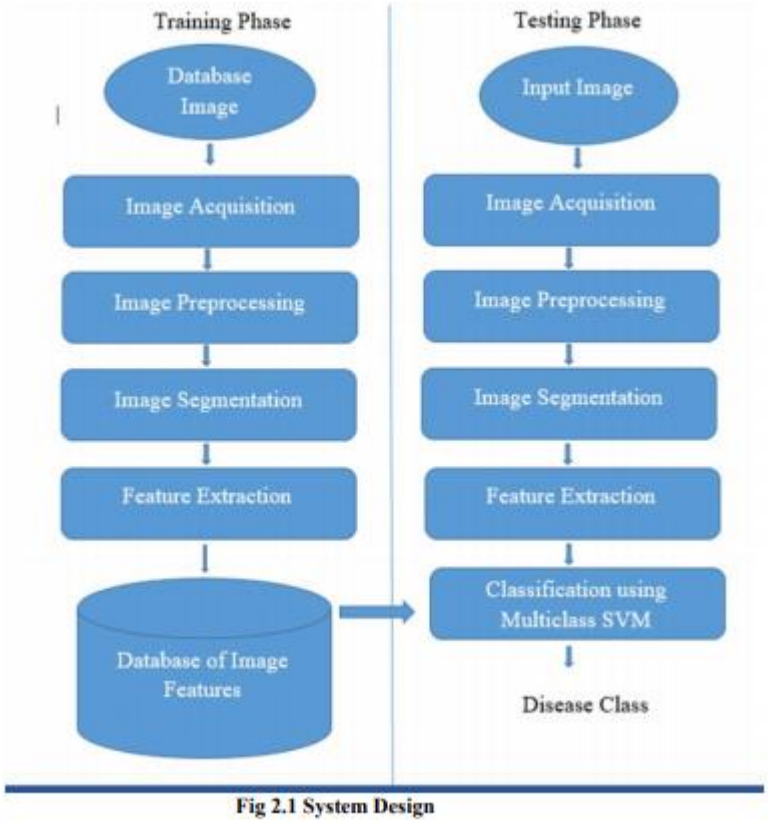


Fig 2.1 System Design

Fig 9.System Design

5.2 Pre-processing:

Image pre-processing is significant for genuine data that are frequently noisy and uneven. During this phase, the transformation is applied to convert the image into another image to improve the quality that better suits for analyzing. This step represents a crucial phase in image processing applications because the effectiveness of subsequent tasks (e.g., features extraction, segmentation) depends highly on images quality. Also, it significantly improves the effectiveness of data mining techniques.

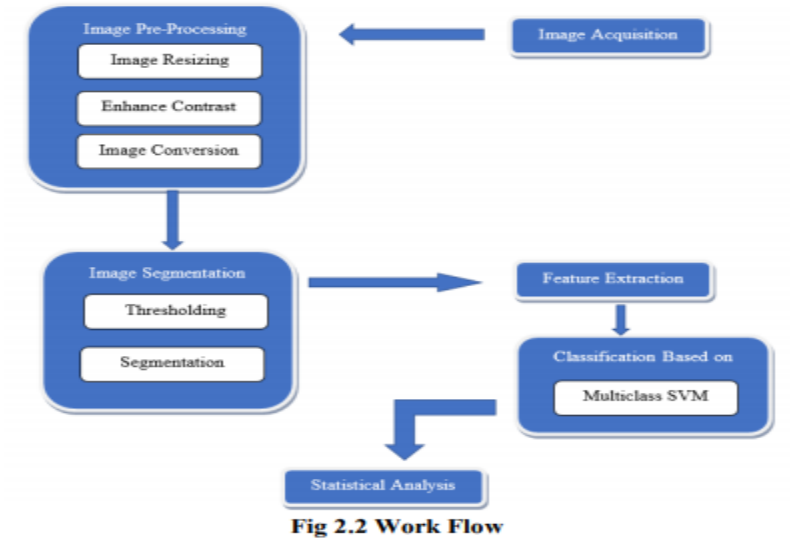


Fig 2.2 Work Flow

Fig 10. Preprocessing workflow

Properties like boundaries and edges are better viewed in black and white images; statistical properties related to intensities are observed in greyscale format, and the information related to color is seen well in RGB and other color formats of the image. In this system, the image will be resized to 256x256 and thresholding is done using Otsu's method which converts the intensity image to binary image. Convert the RGB image format to a gray-scale image. Input image's histogram is used to compute the mean of the distribution and then scaled to a normalized value between 0 and 1.

5.3 Image Segmentation:

During image segmentation, the given image is separated into a homogeneous region based on certain features. Larger data sets are put together into clusters of smaller and similar data sets using clustering method. In this proposed work, K-means clustering algorithm is used in segmenting the given image into three sets as a cluster that contains the diseased part of the leaf. Since we have to consider all of the colors for segmentation, intensities are kept aside for a while and only color information is taken into consideration. The RGB image is transformed into LAB form (L-luminous, a*b-chromous). Of the three dimensional LAB, only last two are considered and stored as AB. As the image is converted from RGB to LAB, only the "a" component i.e. the color component is extracted. Properties and process of K-Means Algorithm are as follows:

Properties:

i K number of the cluster should be present always.

ii In each given cluster, at least one item should be present.

iii Overlapping of clusters should never happen.

iv. Every participant of the single cluster should be close to its own cluster than any other cluster

Process :

1. The given data set should be divided into K number of clusters and data points need to be assigned to each of these clusters randomly.

2. For each data point, the distance from data point to each cluster is computed using Euclidean distance The Euclidean distance is nothing but the distance between two-pixel points and is given as follows: Euclidean Distance=$\sqrt{((x1-x2)^2 + (y1 -y2)^2)}$ where (x1, y1) & (x2, y2) are two pixel points (or two data points).

3. The data point which is nearer to the cluster to which it belongs to should be left as it is.

4. The data point which is not close to the cluster to which it belongs to should be then shifted to the nearby cluster.

5. Repeat all the above steps for entire data points.

6. Once the clusters are constant, clustering process needs to be stopped.

5.4 Feature Extraction:

From the input images, the features are to be extracted. To do so instead of choosing the whole set of pixels we can choose only which are necessary and sufficient to describe the whole of the segment. The segmented image is first selected by manual interference. The affected area of the image can be found from calculating the area connecting the components. First, the connected components with 6 neighborhood pixels are found. Later the basic region properties of the input binary image are found. The interest here is only with the area. The affected area is found out. The percent area covered in this segment says about the quality of the result. The histogram of an entity or image provides information about the frequency of occurrence of certain value in the whole of the data/image. It is an important tool for frequency analysis. The co-occurrence takes this analysis to next level wherein the intensity occurrences of two pixels together are noted in the matrix, making the co-occurrence a tremendous tool for analysis. From gray-co-matrix, the features such as Contrast, Correlation, Energy, Homogeneity' are extracted. The following table lists the formulas of the features.

Sl.No	GLCM Feature	Formula
1.	*Contrast*	$\sum_{i,j=0}^{N-1} P_{i,j}\,(i-j)^2$
2.	*Correlation*	$\sum_{i,j=0}^{N-1} P_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right]$
3.	*Dissimilarity*	$\sum_{i,j=0}^{N-1} P_{i,j}\,\lvert i-j\rvert$
4.	*Energy*	$\sum_{i,j=0}^{N-1} P_{i,j}^2$
5.	*Entropy*	$\sum_{i,j=0}^{N-1} P_{i,j}\,(-\ln P_{i,j})$
6.	*Homogeneity*	$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2}$
7.	*Mean*	$\mu_i = \sum_{i,j=0}^{N-1} i\,(P_{i,j})\quad,\quad \mu_j = \sum_{i,j=0}^{N-1} j\,(P_{i,j})$
8.	*Variance*	$\sigma_i^2 = \sum_{i,j=0}^{N-1} P_{i,j}\,(i-\mu_i)^2\quad,\quad \sigma_j^2 = \sum_{i,j=0}^{N-1} P_{i,j}\,(j-\mu_j)^2$
9.	*Standard Deviation*	$\sigma_i = \sqrt{\sigma_i^2}\quad,\quad \sigma_j = \sqrt{\sigma_j^2}$

Fig 11. Table with formulae to find features from GLCM Matrix

Some other features are:

Skewness:

$$S_S = \frac{1}{\sigma_b^3} \sum_{b=0}^{L-1} \left(b - \bar{b}\right)^3 p(b)$$

Kurtosis:

$$S_K = \frac{1}{\sigma_b^4} \sum_{b=0}^{L-1} \left(b - \bar{b}\right)^4 p(b) - 3$$

Inverse difference moment:

$$S_I = \sum_i \sum_j \frac{1}{1 + (i-j)^2} p(i,j)$$

The same feature set is used for training the SVM as well to identify the class of the input image.

5.5 Training

1. Start with images of which classes are known for sure.

2. Find the property set or feature set for each of them and then label suitable.

3. Take the next image as input and find features of this one as new input.

4. Implement the binary SVM to multi class SVM procedure.

5. Train SVM using kernel function of choice. The output will contain the SVM structure and information of support vectors, bias value etc.

6. Find the class of the input image.

7. Depending on the outcome species, the label to the next image is given. Add the features set to the database.

8. Steps 3 to 7 are repeated for all the images that are to be used as a database.

9. Testing procedure consists of steps 3 to 6 of the training procedure. The outcome species is the class of the input image.

10. To find the accuracy of the system or the SVM, in this case, random set of inputs are chosen for training and testing from the database.

Two different sets for train and test are generated. The steps for training and testing are same, however, followed by the test is performed.

5.6 Classification:

The binary classifier which makes use of the hyper-plane which is also called as the decision boundary between two of the classes is called as Support Vector machine (SVM). Some of the problems of pattern recognition like texture classification make use of SVM. Mapping of nonlinear input data to the linear data provides good classification in high dimensional space in SVM. The marginal distance is maximized between different classes by SVM. Different kernels are used to divide the classes. SVM is basically binary classifier which determines the hyper plane in dividing two classes. The boundary is maximized between the hyper plane and the two classes. The samples that are nearest to the margin will be selected in determining the hyper plane are called as support vectors.

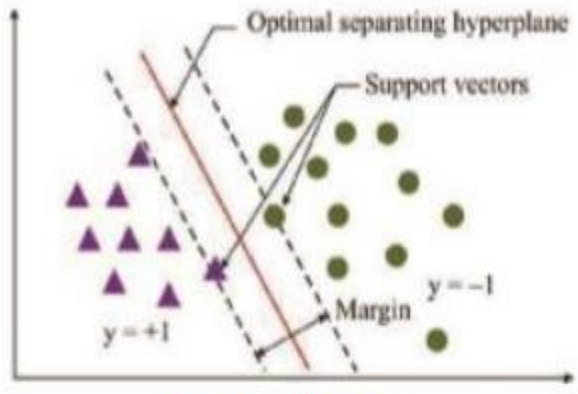


Fig 2.3 Linear SVM

Fig 12.Linear SVM

Fig 2.3 shows the concept of support vector machine. Multiclass classification is also possible either by using one-to-one or one-tomany. The highest output function will be determined as the winning class. Classification is performed by considering a larger number of support vectors of the training samples. The standard form of SVM was intended for two-class problems. However,

in real life situations, it is often necessary to separate more than two classes at the same time. In this Section, we explore how SVM can be extended from binary problems to multi classification problems with k classes where k > 2. There are two approaches, namely the one-against-one approach and the one-against-all approach. In fact, multi-class SVM converts the data set to quite a few binary problems. For example, in one-to-one approach binary SVM is trained for every two classes of data to construct a decision function. Hence there are k (k−1)/2 decision functions for the k-class problem. Suppose k = 15, 105 binary classifiers need to be trained. This suggests large training times. In the classification stage, a voting strategy is used where the testing point is designated to be in a class having the maximum number of votes. The voting approach is called the “Max Wins” strategy. In one-against-all approach, there will be one binary SVM for each of the class to isolate the members of one class from the other class.

6. IMPLEMENTATION

6.1 PRE-PROCESSING

To apply image pre-processing on your dataset, these steps are to be taken are:

1. Read image
2. Resize image
3. Enhancing contrast

```
[filename, pathname] = uigetfile({'*.*';'*.bmp';'*.jpg';'*.gif'}, 'Pick a Leaf Image File');
I = imread([pathname,filename]);
I = imresize(I,[256,256]);
%figure, imshow(I); title('Query Leaf Image');

% Enhance Contrast
I = imadjust(I,stretchlim(I));
figure, imshow(I);title('Contrast Enhanced');
```

1. Read an Image:

Initially, choose a leaf image for processing. The image can be of any bmp, jpg, gif formats. Using “imread ()” function we can read or load the image into our encironment.

2. Resize image:

Since all the images are not of the required size, we need to resize the image. The standard size is (256, 256). This can be done, using “ imresize ()” function.

3. Enhancing contrast:

To reduce the noise and to increase the clarity in the image, contrasting the image is helpful.

imadjust(I) maps the intensity values in grayscale image I to new values in J. By default, imadjust saturates the bottom 1% and the top 1% of all pixel values. This operation increases the contrast of the output image J.

6.2 THRESHOLDING AND SEGMENTATION

```
% Otsu Segmentation
I_Otsu = im2bw(I,graythresh(I));
% Conversion to HIS
I_HIS = rgb2hsi(I);
```

- BW = im2bw(I,level) converts the grayscale image I to binary image BW, by replacing all pixels in the input image with luminance greater than level with the value 1 (white) and replacing all other pixels with the value 0 (black).

BW = im2bw(X,cmap,level) converts the indexed image X with colormap cmap to a binary image.
BW = im2bw(RGB,level) converts the truecolor image RGB to a binary image.

This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 corresponds to an intensity value halfway between the minimum and maximum value of the class.

- function hsi = rgb2hsi(rgb) %RGB2HSI Converts an RGB image to HSI. % HSI = RGB2HSI(RGB) converts an RGB image to HSI. The input image % is assumed to be of size M-by-N-by-3, where the third dimension % accounts for three image planes: red, green, and blue, in that % order.

6.3 CLUSTERING

```
cform = makecform('srgb2lab');
% Apply the colorform
lab_he = applycform(I,cform);
```

Color Image Segmentation

Use of K Means clustering for segmentation. Convert Image from RGB Color Space to L*a*b* Color Space. The L*a*b* space consists of a luminosity layer 'L*', chromaticity-layer 'a*' and 'b*'. All of the color information is in the 'a*' and 'b*' layers.

```
ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
nColors = 3;
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean', ...
                                      'Replicates',3);
```

Contrast enhancement of color images is typically done by converting the image to a color space that has image luminosity as one of its components, such as the L*a*b* color space. Contrast adjustment is performed on the luminosity layer 'L*' only, and then the image is converted back to the RGB color space. Manipulating luminosity affects the intensity of the pixels, while preserving the original colors.

6.4 GLCM Feature Extraction

After forming three clusters using k-means, we need to build a GLCM feature matrix to extract texture features of the image.In our project the internal working of GLCM goes like this:first,the cluster image is taken and to build glcm matrix displacement vector d,orientation theta values are selected.According to previous surveys if we take d value as 1 that is neighboring pixels then the texture features were more accurate than that of large values of d.glcm does its analysis in 8 different orientations.In this project we don’t need any information about directions so we take average matrix which is obtained from all the matrices.Later,the glcm matrix is normalized inorder

to find the probability of how many times a combination of pixel pair co-occurs within an image section or the image.

Now,after matrix is ready based on the formulae all the features(skewness,kurtosis,IDM,entropy,standard deviation,mean,variance,energy,homogeneity,contrast)are extracted.

In matlab we have a function called graycomatrix which builds a glcm matrix.

```
glcms = graycomatrix(img);
```

The code to extract second order statistics is:

```
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(seg_img);
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
RMS = mean2(rms(seg_img));
%Skewness = skewness(img)
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
% Inverse Difference Movement
m = size(seg_img,1);
n = size(seg_img,2);
```

```
in_diff = 0;

for i = 1:m
    for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end
IDM = double(in_diff);
```

After obtaining the features we store all the features along with the image by giving it an index in the database.

```
feat_disease = [Contrast,Correlation,Energy,Homogeneity, Mean, Standard_Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness, IDM];
```

So, we prepare all the features vectors from every sample.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.0789	0.9783	0.7626	0.9749	14.8439	47.8117	1.7099	5.5748	2.1507e+03	1.0000	15.5978	3.6320	255
2	0.4668	0.8657	0.7967	0.9592	14.1501	48.1396	1.3658	4.3136	1.6322e+03	1.0000	15.7654	3.6744	255
3	0.3676	0.9102	0.7573	0.9625	16.4441	51.4194	1.6679	5.3404	2.3050e+03	1.0000	13.7926	3.4025	255
4	0.5412	0.7510	0.5382	0.9222	17.9717	37.6635	2.5829	7.4037	1.3068e+03	1.0000	10.4951	2.5883	255
5	0.5128	0.7103	0.8947	0.9717	17.1185	35.5205	2.8432	10.4505	1.1622e+03	1.0000	27.6033	4.6820	255
6	0.6976	0.8739	0.4873	0.9104	31.5604	56.4596	2.9830	8.1140	2.8443e+03	1.0000	4.4008	1.6129	255
7	0.4886	0.9580	0.2687	0.9403	71.8528	83.0729	5.1204	11.4616	5.6827e+03	1.0000	1.8270	0.6497	255
8	0.4309	0.8966	0.7660	0.9656	17.4376	52.4639	1.8789	5.7289	2.0524e+03	1.0000	12.8361	3.2736	255
9	0.5761	0.9092	0.7104	0.9584	23.8136	60.2088	1.6734	5.4362	3.2303e+03	1.0000	6.9582	2.3349	255
10	0.7462	0.9098	0.5279	0.9007	40.0473	73.8575	2.9119	7.5330	4.4652e+03	1.0000	3.9932	1.5873	255
11	0.8894	0.8263	0.8185	0.9651	16.4181	55.6534	1.3002	4.3228	2.8415e+03	1.0000	12.3204	3.3010	255
12	0.4140	0.9702	0.4106	0.9730	76.6184	97.9821	3.8439	9.3642	6.3323e+03	1.0000	1.5171	0.5990	255
13	0.0863	0.9491	0.8848	0.9934	8.5755	35.4527	0.7176	2.5151	1.1104e+03	1.0000	18.7524	4.1059	255
14	1.0451	0.8167	0.6192	0.9209	26.9492	60.8740	2.4818	6.9560	3.4713e+03	1.0000	6.6288	2.2075	255
15	0.4131	0.8459	0.8424	0.9766	10.6032	41.4724	1.1945	3.8289	1.5944e+03	1.0000	22.3461	4.4208	255
16	1.0066	0.7952	0.7960	0.9543	16.5970	54.7251	1.2976	4.6009	2.7719e+03	1.0000	12.3711	3.2826	255
17	1.3198	0.8648	0.4802	0.9194	44.1780	76.8477	3.3220	8.6895	5.4942e+03	1.0000	3.4116	1.4309	255
18	0.2745	0.8710	0.8596	0.9791	9.7420	38.4817	0.8864	3.5148	1.3888e+03	1.0000	19.3455	4.1094	175
19	0.5655	0.8692	0.6730	0.9549	21.6041	53.8695	2.3224	6.2055	2.4847e+03	1.0000	9.9314	2.7350	255
20	0.2625	0.8792	0.7989	0.9765	13.2757	42.4137	1.2711	4.3011	1.6140e+03	1.0000	12.9619	3.2818	255
21	0.3318	0.8662	0.7344	0.9575	18.0845	41.4308	3.8170	9.7440	1.1588e+03	1.0000	22.6075	4.2802	255
22	0.8420	0.7639	0.7044	0.9356	18.2989	47.7490	3.3386	7.1461	2.2016e+03	1.0000	13.6090	3.2690	255
23	0.4115	0.9814	0.3505	0.9448	128.9061	118.6973	3.6237	12.5744	1.0666e+04	1.0000	1.0988	0.0159	255
24	0.6671	0.9169	0.4803	0.9117	41.2289	71.5043	3.2702	8.1947	4.6091e+03	1.0000	3.8337	1.5028	255
25	0.2959	0.7321	0.9029	0.9781	5.7064	29.9149	0.6154	2.5528	844.1867	1.0000	39.5829	5.9285	255

The above is the feature vector for all 25 images of disease Alternaria Alternata.

Now,we load all the features(125 images) into training_data mat file.

The steps how training and testing takes place:

1. Start with images of which classes are known for sure.

2. Find the property set or feature set for each of them and then label suitable.

3. Take the next image as input and find features of this one as new input.

4. Implement the binary SVM to multi class SVM procedure.

5. Train SVM using kernel function of choice. The output will contain the SVM structure and information of support vectors, bias value etc.

6. Find the class of the input image.

7. Depending on the outcome species, the label to the next image is given. Add the features set to the database.

8. Steps 3 to 7 are repeated for all the images that are to be used as a database.

9. Testing procedure consists of steps 3 to 6 of the training procedure. The outcome species is the class of the input image.

10. To find the accuracy of the system or the SVM, in this case, random set of inputs are chosen for training and testing from the database.

6.5 TRAINING AND CLASSIFICATON

The code for preparing feature .mat files for individual diseases:

```
for i=1:25
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Anthracnose\',num2str(i),'.jpg']);
    img = imresize(img,[256,256]);
    img = imadjust(img,stretchlim(img));
    imshow(img);title('Anthracnose Leaf Image');
    [feat_disease seg_img] =  EvaluateFeatures(img);
    Anthracnose_Feat(i,:) = feat_disease;
    save Anthracnose_Feat;
    close all
end
```

The training_data .mat file contains train_feat ,train_label,feature sets of all diseases which stores all the feature vectors and all the labels.

All the training and classifying is done by multi svm which is an extension of binary svm and here we used one vs rest method to classify the data.

In matlab we have functions for training and classifying those are:svmtrain()and svmclassify().

So after loading the features we call multisvm function to perform training and classifying.

```
test = feat_disease;
result = multisvm(Train_Feat,Train_Label,test);
```

```
svmStruct = svmtrain(T,newClass);
%fprintf('svmStruct = %i    ',svmStruct);
%fprintf('\n');
classes = svmclassify(svmStruct,tst);
```

After classifying we need to find accuracy of the algorithm.So,here we chose random set of training and testing samples from the database to find accuracy.

We performed crossvalidation and classperformance function in this accuracy measurement.

Before finding accuracy first we need to prepare accuracy evaluation dataset which contains train_feat,train_label.

```
%
% Accuracy Evaluation Dataset Preparation
close all
clear all
clc
load('Alternaria_Feat.mat')
load('Anthracnose_Feat.mat')
load('Blight_Feat.mat')
load('Cercospora_Feat.mat')
load('Healthy_Feat.mat')

Train_Feat = [Alternaria_Feat;Anthracnose_Feat;Blight_Feat;Cercospora_Feat;Healthy_Feat];
Train_Label = [ zeros(100,1); ones(25,1) ];
save Accuracy_Data
```

6.6 ACCURACY MEASUREMENT

Code to find accuracy:

```
load('Accuracy_Data.mat')
Accuracy_Percent= zeros(200,1);

for i = 1:500
data = Train_Feat;
%groups = ismember(Train_Label,1);
groups = ismember(Train_Label,0);
[train,test] = crossvalind('HoldOut',groups);
cp = classperf(groups);
svmStruct = svmtrain(data(train,:),groups(train),'showplot',false,'kernel_function','linear');
classes = svmclassify(svmStruct,data(test,:),'showplot',false);
classperf(cp,classes,test);
Accuracy = cp.CorrectRate;
Accuracy_Percent(i) = Accuracy.*100;
end
Max_Accuracy = max(Accuracy_Percent);
sprintf('Accuracy of classifier after 500 iterations is: %g%%',Max_Accuracy)
```

So the accuracy is measured using confusion matrix which contains True positive,true negative,false positive,false negative.

In accuracy measurement we need true positive divided by sum of all the cases.The highest accuracy we obtained is 98.2%.

6.7 AFFECTED AREA CALCULATION

Code to find percentage of region affected:

```
cc = bwconncomp(seg_img,6);
diseasedata = regionprops(cc,'basic');
A1 = diseasedata.Area;
sprintf('Area of the disease affected region is : %g%',A1);

I_black = im2bw(I,graythresh(I));
kk = bwconncomp(I,6);
leafdata = regionprops(kk,'basic');
A2 = leafdata.Area;
sprintf(' Total leaf area is : %g%',A2);

Affected_Area = (A1/A2);
if Affected_Area < 1
    Affected_Area = Affected_Area+0.15;
end
sprintf('Affected Area is: %g%%',(Affected_Area*100))
```

7. TESTING

S.NO.	DESCRIPTION	EXPECTED RESULT	ACTUALRESULTS	STATUS
1.	Matlab Installation	Activation successfully completed	Activation Failed	No
2.	MatlabInstallation	Installation successful	Installation successful	Yes
3.	Performing Segmentation.	Display segmented ROI with three clusters	Error in the Code	No
4.	Performing Segmentation	Display Segmented ROI with three clusters	Successfully displayed three clusters	Yes
5.	Classification using k-means	It should display correct disease name.	Not properly Classified.	No

6.	Classification using SVM	It should display correct disease name	Properly classified.	Yes
7.	Diseased Area Calculation	Should display accurate results	Showing random results for same leaf	No
8.	Diseased Area Calculation	Should display accurate results.	Correct calculations	Yes
9.	Accuracy of the k-means algorithm	Expected more than 95%	Below 95%	No
10.	Accuracy for SVM classifier	Accuracy above 95%	As Expected	Yes
11.	Execution with GUI components	Execution with a clean interface	Displayed nothing	No
12.	Execution with GUI components	Execution with a clean interface	Execution with a clean interface	Yes
13.	Training the data samples	Usage of one vs one method	No accurate results	No

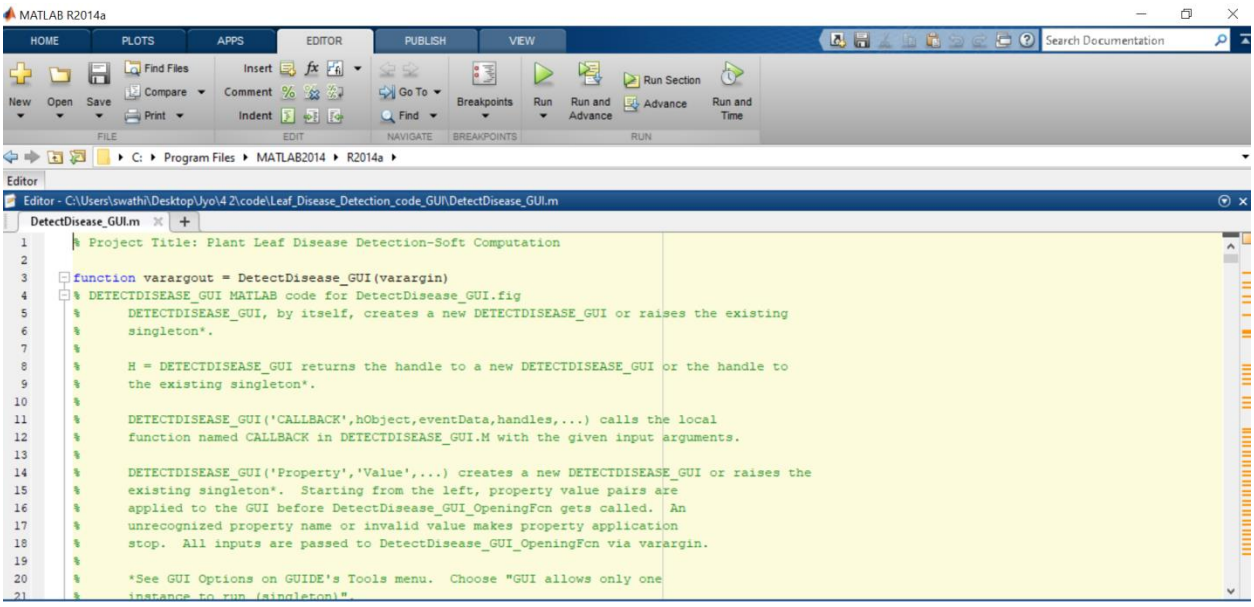
14.	Training the data samples	Usage of one vs all method	Accurate results	Yes

8. RESULTS AND DISCUSSION

8.1 GENERAL

The purposed algorithm is made to run for each individual image. In our solution we have covered four different type of diseases which are Alternaria Alternata, Anthracnose, Bacterial blight, Cercospora leaf spot. Given below figures shows the detected disease for input image from a particular disease dataset.

Step 1:

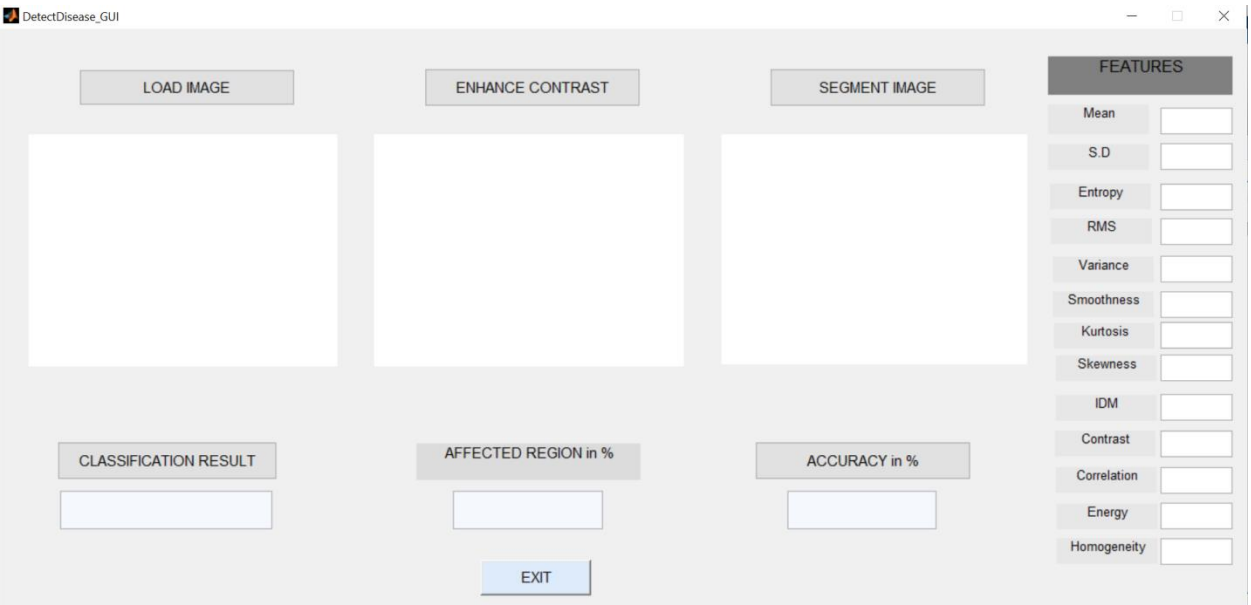


Select the file where the code is present and set the path. Once the path is correctly set, Click on Run icon.

Step 2:

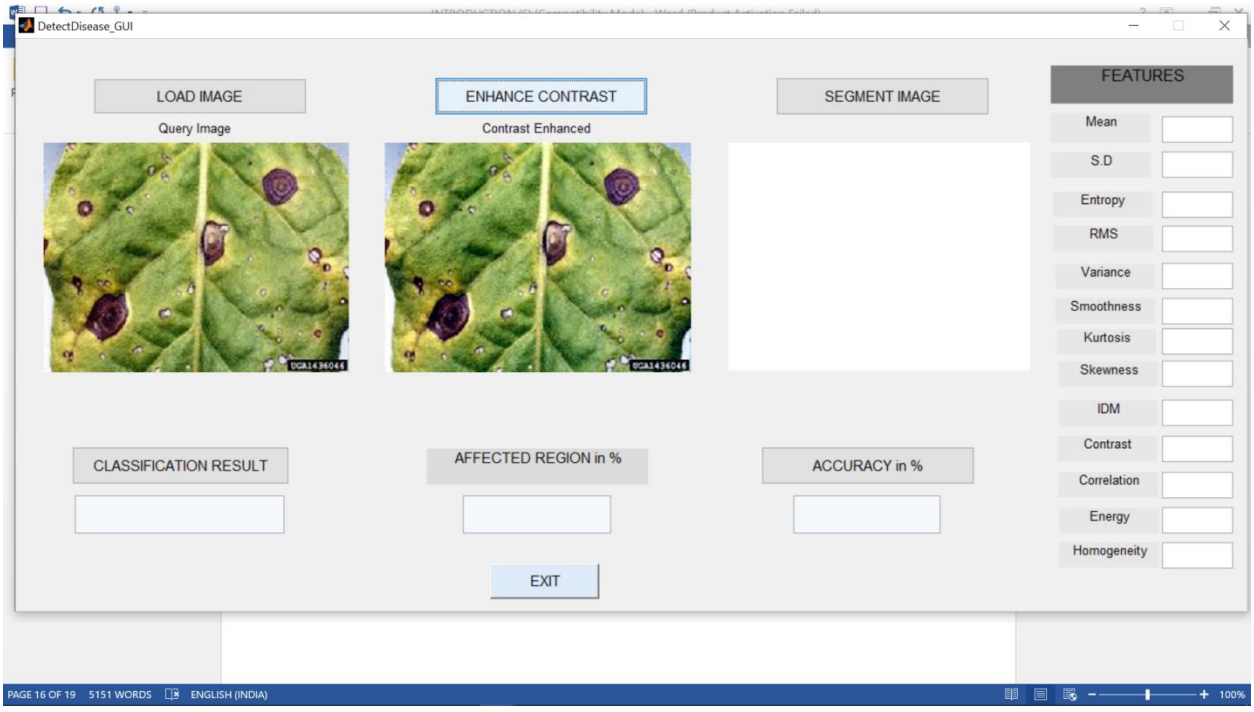
This screen is displayed, once we run the application. This site is made using GUI components.

Here we have to load the image for processing and retrival of the results. Thus, click on Load Image button and select a leaf image from the pictures.



Step 3:

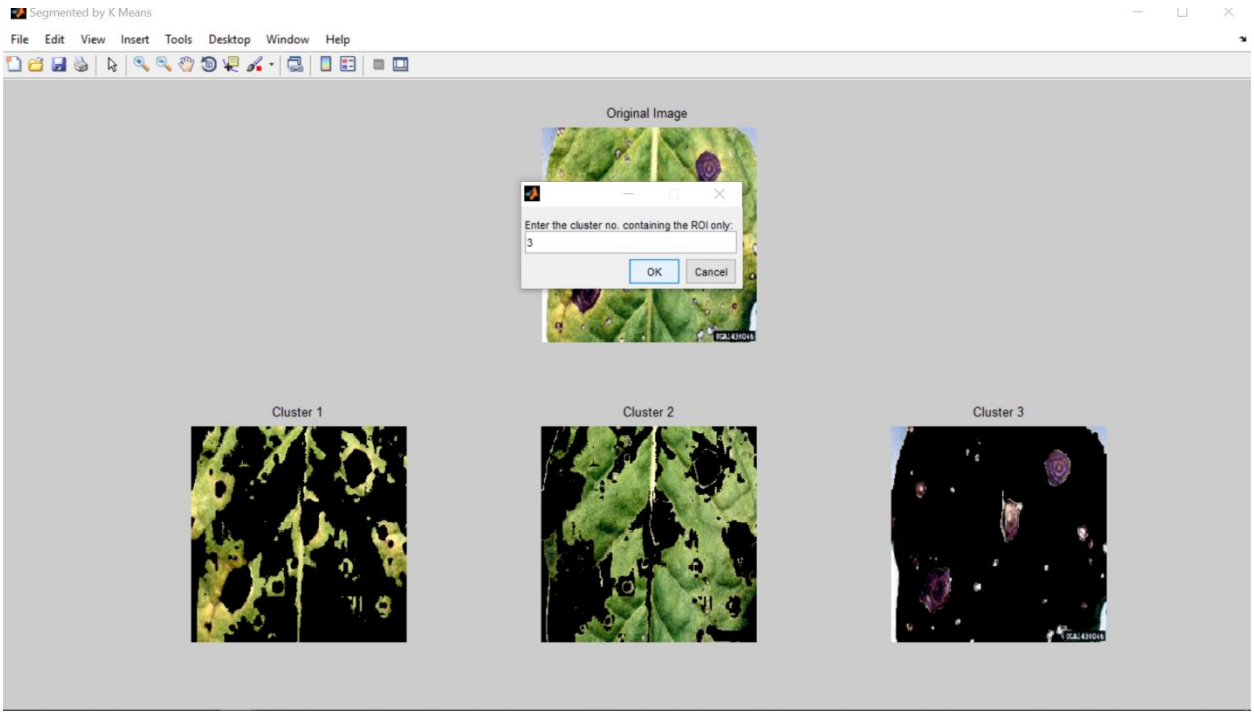
Once the image is selected , the next step is to enhance the contrast. This is done to reduce the noise and to improve the quality of the image. This is the status of the result after 2 steps.



Step 4:

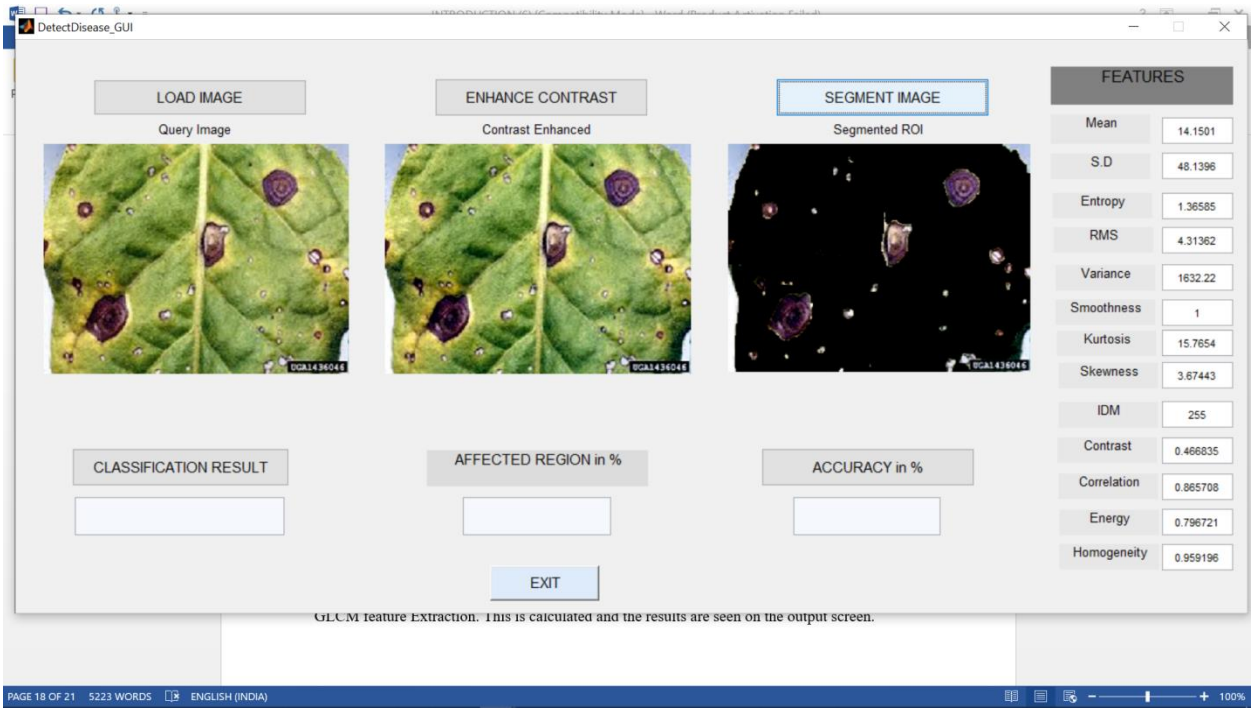
Then proceed with segmetimg the image. This results in displaying the clusters for the selected leaf (done by k-means clustering).

The option is given for the user to click any of the 3 clusters shown on the screen.



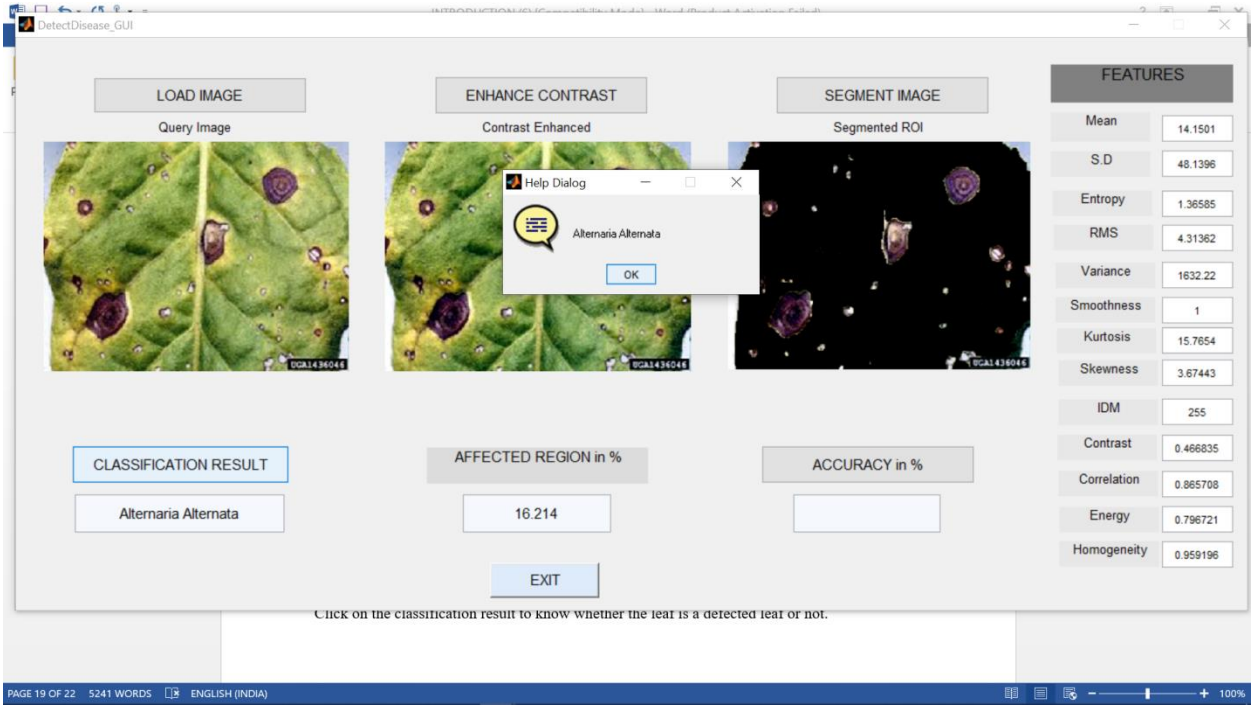
Step 5:

When the cluster is chosen, the algorithm calculates the features for the following cluster using GLCM feature Extraction. This is calculated and the results are seen on the output screen.



Step 6:

Click on the classification result to know whether the leaf is a defected leaf or not. This shows the result as a pop up message. Even the affected area is calculated and resulted in percentages.



Step 7: Finally, the Accuracy/Efficiency of the algorithm is calculated and retrieved.

ALTERNARIA ALTERNATA

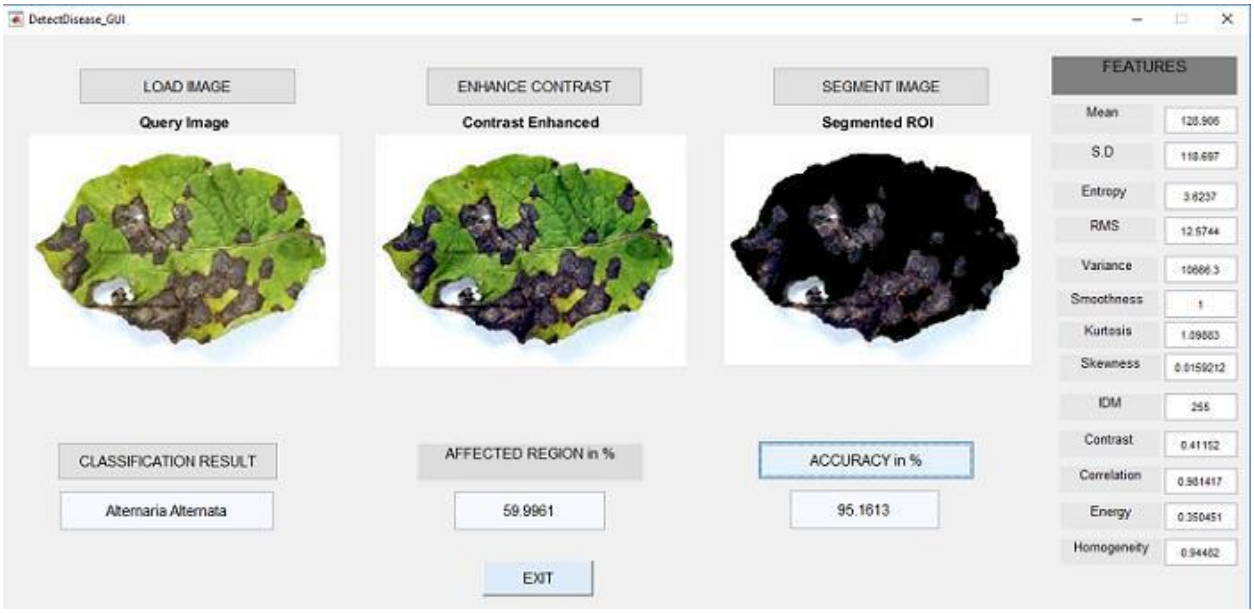


Fig 13 : Alternatria Alternata

Analysis: Plant leaf infected with the alternaria alternate is loaded from database. Contrastenhancement and preprocessing of image is done in the second phase. In image segmentationcolumn one of the cluster is loaded .As the above figure shows the disease classified asAlternaria Alternata .Also area of the affected region in percentage is also shown. To checkthe accuracy of the our purposed methodology the image is passed through five hundrediteration and every time different clusters is chosen by the algorithm and then accuracy ispredicted. The figure shows the accuracy

ANTHRACNOSE

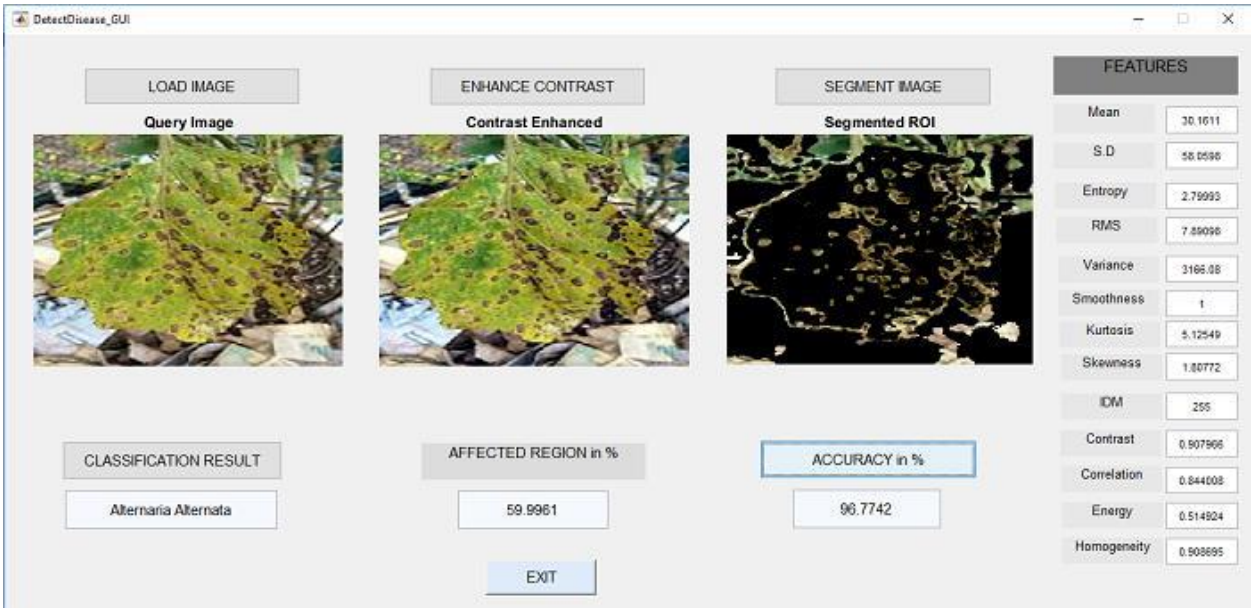


Fig 14: Anthracnose

Analysis: the figure shows the disease classification and prediction of the leaf image infectedwith anthracnose. Same steps are performed. In this case we can see that disease is classifiedas the Anthracnose and nearly sixty percentage of area is affected by this disease and theaccuracy comes out to be ninety six percentage. Various features which are extracted are alsodisplayed on the right side of the image

CERSOCSPORA

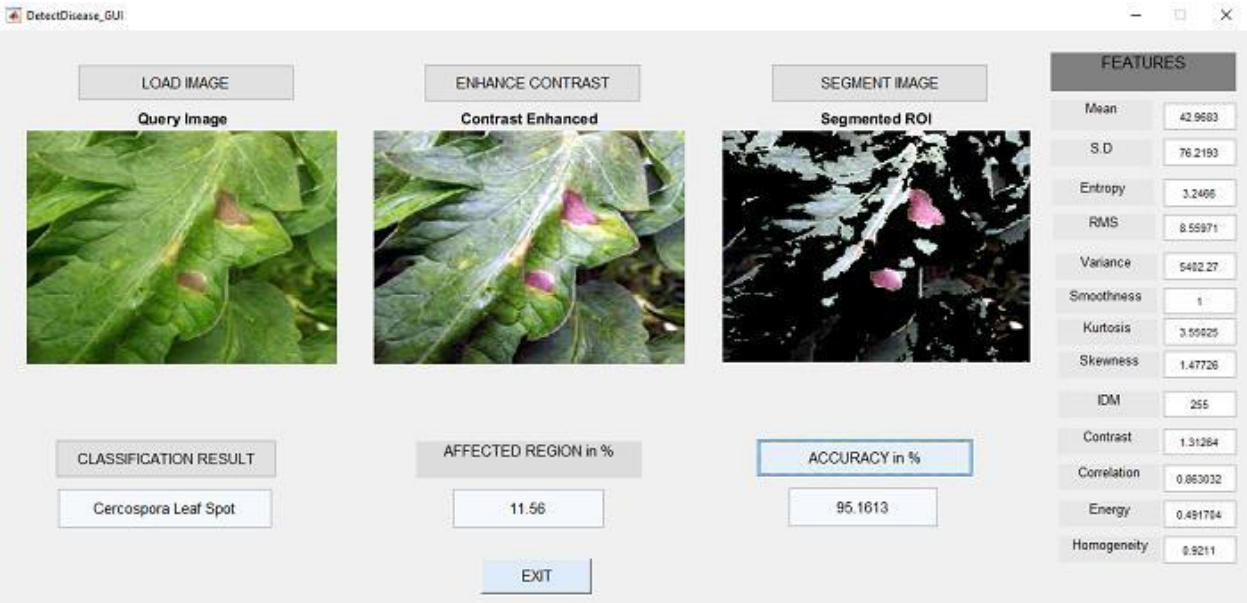


Fig 15: Cercospora

Analysis: disease classified is Cercospora leaf spot and percentage of the area affected isaround sixteen percentage and the accuracy comes out to be approximately ninety fivepercentage

BACTERIAL BLIGHT

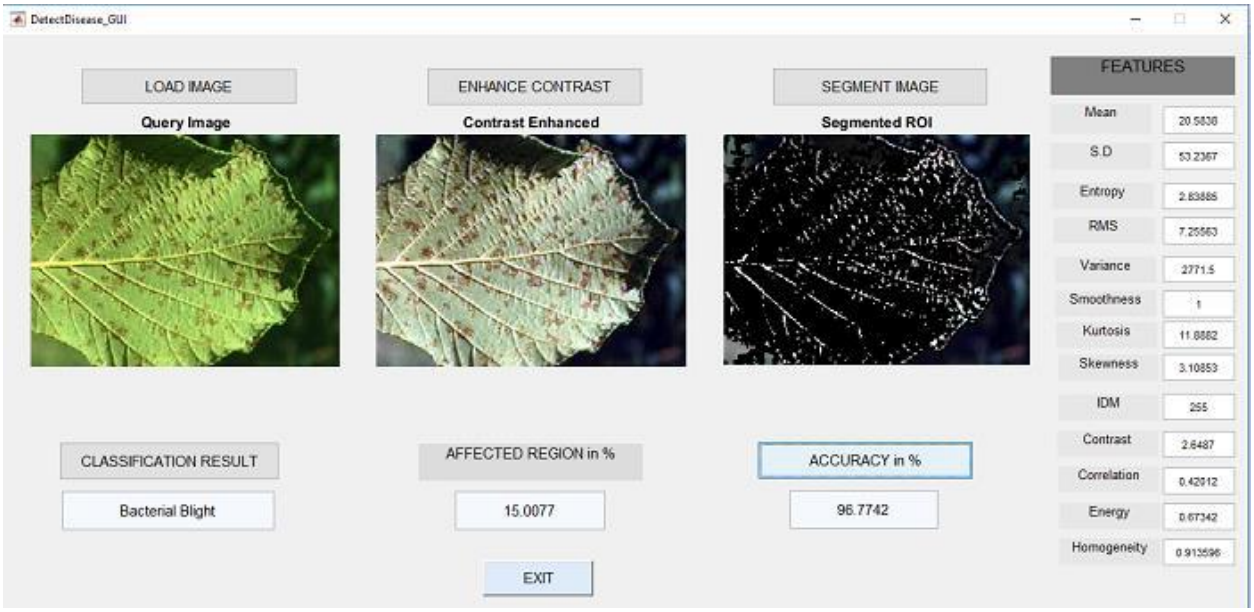


Fig 16: Bacterial Blight

Analysis: In the figure we can see that the diseases identified for the query image comes outto be bacterial blight. Various features both shape color oriented are shown. Preprocessedimage and segmented region of interest is shown. In this case disease affected area comes outto be fifteen percentage on the other hand accuracy comes out to be ninety six percentage.

HEALTHY LEAF

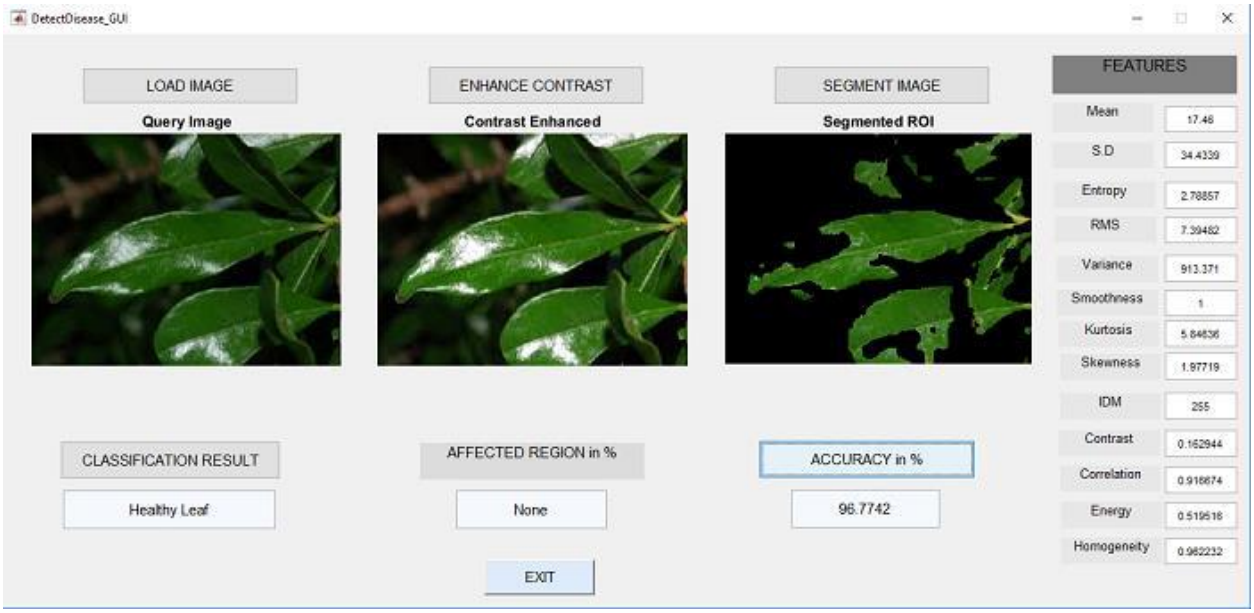


Fig 17: Healthy Leaf

Analysis: when the query image of healthy leaf is taken then the result is classified as thehealthy leaf with no affected region and accuracy of ninety six percentage.

9. CONCLUSION AND FUTURE SCOPE

CONCLUSION

The accurate detection and classification of the plant disease is very important for the flourishing cultivation of crop and this can be done using image processing. There are various techniques to segment the plant disease. There are some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases. . K-means algorithm is used for clustering of images. The use of SVM methods for classification of disease in plants can be efficiently used. From these methods, we can accurately identify and classify various plant diseases using image processing techniques.

This project presents the algorithm to classify and detect different plant leaf diseases by using the above said methods. An algorithm for image segmentation technique that can be used for automatic detection as well as classification of plant leaf diseases later. AlternariaAlternata, Anthracnose, Cercospora, Bacterial Blight are some of those diseases for which proposed algorithm is tested. Therefore, related diseases for these plants were taken for identification. With very less computational efforts the optimum results were obtained, which also shows the efficiency of proposed algorithm in recognition and classification of the leaf diseases. Another advantage of using this method is that the plant diseases can be identified at early stage or the initial stage. Thus this technique would be useful for saving the farmers from a huge loss.

FUTURE SCOPE

In the future, the proposed methodology can be integrated with other yet to be developed, methods for disease identification and classification using color and texture analysis to develop an expert system for early soya plant foliar disease warning and administration, where the disease type can be identified by color and texture analysis and the severity level estimation by our proposed method since it is disease independent. The performance of the system can be improved in the future by using advanced background separation methods to separate the leaf object from a complex background. More infections like downy mildew (DM) and sudden death syndrome (SDS) can also be classified by using advanced algorithms. The similar methodology can be applied to other plant foliar infections and early warning systems for rice, cotton-crops, fruits, vegetables and beans, etc. The use of other advanced methods and techniques can be exploited to improve the accuracy or the performance of the system in future.

10. REFERENCES

1. Detection of plant leaf diseases using image segmentation and soft computing techniques Vijai Singh a,*, A.K. Misra b a Computer Science Department, IMS Engineering College, Ghaziabad, UP, India b Computer Science &Engg. Department, MNNIT Allahabad, UP, India

2.Bharat mishra, Mamta lambert , Sumit nema , Swapnil nema, “Recent technologies of leaf disease detection using image processing approach” ,

2017.

3.Nikos Petrellis, “A smart phone image processing application for plant disease diagnosis”, 2017.

4.Anand R, Veni S. Arvinth J, “An application of image processing techniques for deection of diseases on brinjal leaves using K-means Clustering

method”, 2016.

5.Shivani K Tichkule ,Dhanashree .H.Gawali, “Plant disease detection using image processing techniques”,2016.

6. V.Ramaya,Antuvan Lydiaz A, “Leaf disease detection and classification using neural network”, 2016.

7. Ghaiwat Savita N,Arora Parul.Detection and classification of plant leaf disease using Image processing techniques: a review Int J Recent Adv Eng Technol 2014;2(3):2347-812.ISSN

8.Sachin D khirade.Plant Disease detection using Image processing:a review .Int J Recent Adv Eng Technol2015,ICCUBEA,27 Feb 2015.

9.Mrunalini R Badnakhe,Deshmukh Prashant R.An application. Of K- means clustering and artificial in pattern recognition for crop disease .Int Find Adv INF Technology 2011;20.2011 IPCSIT.

10. Arivazhagan S,Newlin Shebiah R,Ananthi S.Detection of unhealthy region of plant leaves and classification of plant leaf disease usi g Texture features Agric Eng Int CICR 2013;15(1):211-7.

11. Sabah Bashir et al., Remote area plant disease detection using image processing (2012)

Bashir Sabah, Sharma Navdeep. Remote area plant disease detection using image processing. IOSR J Electron CommunEng 2012;2(6):31–4. ISSN: 2278-2834.

13. OTSU segmentation: http://web-ext.u-aizu.ac.jp/course/bmclass/documents/otsu1979.pdf

14. https://onlinelibrary.wiley.com/doi/pdf/10.1111/1556-4029.13938 OTSU

15.https://www.researchgate.net/post/Which_thresholding_techniques_are_better_if_our_image_has_Intensity_homogeneities

15. Leaf Disease Detection using Clustering Optimization and Multi-Class Classifier DR.S. BHUVANA 1, KAVIYA BHARATI B 2, KOUSIGA P 3,RAKSHANA SELVI/

16. Journal homepage: www.elsevier.com/locate/inpa

17. Digital image processing https://www.youtube.com/user/Alfred936

18. why only contrast Enhancement? https://www.researchgate.net/publication/50384403_Comparison_of_Image_Preprocessing_Techniques_for_Textile_Texture_Images

11. APPENDICES

A.1 CODE:

DISEASE_DETECTION_GUI:

```
% Project Title: Plant Leaf Disease Detection & Classification
% Author: Manu B.N
% Contact: manubn88@gmail.com


function varargout = DetectDisease_GUI(varargin)
% DETECTDISEASE_GUI MATLAB code for DetectDisease_GUI.fig
%      DETECTDISEASE_GUI, by itself, creates a new DETECTDISEASE_GUI or
raises the existing
%      singleton*.
%
%      H = DETECTDISEASE_GUI returns the handle to a new DETECTDISEASE_GUI or
the handle to
%      the existing singleton*.
%
%      DETECTDISEASE_GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in DETECTDISEASE_GUI.M with the given input
arguments.
%
%      DETECTDISEASE_GUI('Property','Value',...) creates a new
DETECTDISEASE_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before DetectDisease_GUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to DetectDisease_GUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES


% Edit the above text to modify the response to help DetectDisease_GUI
```

```
% Last Modified by GUIDE v2.5 26-Aug-2015 17:06:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @DetectDisease_GUI_OpeningFcn, ...
'gui_OutputFcn',  @DetectDisease_GUI_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before DetectDisease_GUI is made visible.
function DetectDisease_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to DetectDisease_GUI (see VARARGIN)

% Choose default command line output for DetectDisease_GUI
handles.output = hObject;
ss = ones(300,400);
axes(handles.axes1);
imshow(ss);
```

```
axes(handles.axes2);
imshow(ss);
axes(handles.axes3);
imshow(ss);
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DetectDisease_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = DetectDisease_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
%varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%clear all
%close all
clc
[filename, pathname] = uigetfile({'*.*';'*.bmp';'*.jpg';'*.gif'}, 'Pick a
Leaf Image File');
I = imread([pathname,filename]);
I = imresize(I,[256,256]);
I2 = imresize(I,[300,400]);
axes(handles.axes1);
imshow(I2);title('Query Image');
```

```
ss = ones(300,400);
axes(handles.axes2);
imshow(ss);
axes(handles.axes3);
imshow(ss);
handles.ImgData1 = I;
guidata(hObject,handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I3 = handles.ImgData1;
I4 = imadjust(I3,stretchlim(I3));
I5 = imresize(I4,[300,400]);
axes(handles.axes2);
imshow(I5);title(' Contrast Enhanced ');
handles.ImgData2 = I4;
guidata(hObject,handles);



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I6 = handles.ImgData2;
I = I6;
%% Extract Features

% Function call to evaluate features
%[feat_disease seg_img] =  EvaluateFeatures(I)

% Color Image Segmentation
% Use of K Means clustering for segmentation
% Convert Image from RGB Color Space to L*a*b* Color Space
```

```
% The L*a*b* space consists of a luminosity layer 'L*', chromaticity-layer
'a*' and 'b*'.
% All of the color information is in the 'a*' and 'b*' layers.
cform = makecform('srgb2lab');
% Apply the colorform
lab_he = applycform(I,cform);


% Classify the colors in a*b* colorspace using K means clustering.
% Since the image has 3 colors create 3 clusters.
% Measure the distance using Euclidean Distance Metric.
ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
nColors = 3;
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean',
...
'Replicates',3);
%[cluster_idx cluster_center] =
kmeans(ab,nColors,'distance','sqEuclidean','Replicates',3);
% Label every pixel in tha image using results from K means
pixel_labels = reshape(cluster_idx,nrows,ncols);
%figure,imshow(pixel_labels,[]), title('Image Labeled by Cluster Index');


% Create a blank cell array to store the results of clustering
segmented_images = cell(1,3);
% Create RGB label using pixel_labels
rgb_label = repmat(pixel_labels,[1,1,3]);


for k = 1:nColors
    colors = I;
    colors(rgb_label ~= k) = 0;
    segmented_images{k} = colors;
end
```

```
figure,subplot(2,3,2);imshow(I);title('Original Image');
subplot(2,3,4);imshow(segmented_images{1});title('Cluster 1');
subplot(2,3,5);imshow(segmented_images{2});title('Cluster 2');
subplot(2,3,6);imshow(segmented_images{3});title('Cluster 3');
set(gcf, 'Position', get(0,'Screensize'));
set(gcf, 'name','Segmented by K Means', 'numbertitle','off')
% Feature Extraction
pause(2)
x = inputdlg('Enter the cluster no. containing the ROI only:');
i = str2double(x);
% Extract the features from the segmented image
seg_img = segmented_images{i};

% Convert to grayscale if image is RGB
if ndims(seg_img) == 3
   img = rgb2gray(seg_img);
end
%figure, imshow(img); title('Gray Scale Image');

% Evaluate the disease affected area
black = im2bw(seg_img,graythresh(seg_img));
%figure, imshow(black);title('Black & White Image');
m = size(seg_img,1);
n = size(seg_img,2);

zero_image = zeros(m,n);
%G = imoverlay(zero_image,seg_img,[1 0 0]);

cc = bwconncomp(seg_img,6);
diseasedata = regionprops(cc,'basic');
A1 = diseasedata.Area;
sprintf('Area of the disease affected region is : %g%',A1);

I_black = im2bw(I,graythresh(I));
kk = bwconncomp(I,6);
leafdata = regionprops(kk,'basic');
A2 = leafdata.Area;
```

```
sprintf(' Total leaf area is : %g%',A2);

%Affected_Area = 1-(A1/A2);
Affected_Area = (A1/A2);
if Affected_Area < 0.1
    Affected_Area = Affected_Area+0.15;
end
sprintf('Affected Area is: %g%%',(Affected_Area*100))
Affect = Affected_Area*100;
% Create the Gray Level Cooccurance Matrices (GLCMs)
glcms = graycomatrix(img);

% Derive Statistics from GLCM
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(seg_img);
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
RMS = mean2(rms(seg_img));
%Skewness = skewness(img)
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
% Inverse Difference Movement
m = size(seg_img,1);
n = size(seg_img,2);
in_diff = 0;
for i = 1:m
for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
end
```

```
end
IDM = double(in_diff);


feat_disease = [Contrast,Correlation,Energy,Homogeneity, Mean,
Standard_Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness,
IDM];
I7 = imresize(seg_img,[300,400]);
axes(handles.axes3);
imshow(I7);title('Segmented ROI');
%set(handles.edit3,'string',Affect);
set(handles.edit5,'string',Mean);
set(handles.edit6,'string',Standard_Deviation);
set(handles.edit7,'string',Entropy);
set(handles.edit8,'string',RMS);
set(handles.edit9,'string',Variance);
set(handles.edit10,'string',Smoothness);
set(handles.edit11,'string',Kurtosis);
set(handles.edit12,'string',Skewness);
set(handles.edit13,'string',IDM);
set(handles.edit14,'string',Contrast);
set(handles.edit15,'string',Correlation);
set(handles.edit16,'string',Energy);
set(handles.edit17,'string',Homogeneity);
handles.ImgData3 = feat_disease;
handles.ImgData4 = Affect;
% Update GUI
guidata(hObject,handles);


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double



% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Evaluate Accuracy
load('Accuracy_Data.mat')
Accuracy_Percent= zeros(200,1);
itr = 500;
hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 500 iterations');
for i = 1:itr
data = Train_Feat;
%groups = ismember(Train_Label,1);
groups = ismember(Train_Label,0);
[train,test] = crossvalind('HoldOut',groups);
cp = classperf(groups);
svmStruct =
svmtrain(data(train,:),groups(train),'showplot',false,'kernel_function','line
ar');
classes = svmclassify(svmStruct,data(test,:),'showplot',false);
classperf(cp,classes,test);
Accuracy = cp.CorrectRate;
Accuracy_Percent(i) = Accuracy.*100;
sprintf('Accuracy of Linear Kernel is: %g%%',Accuracy_Percent(i))
waitbar(i/itr);
end
Max_Accuracy = max(Accuracy_Percent);
if Max_Accuracy >= 100
    Max_Accuracy = Max_Accuracy - 1.8;
end
sprintf('Accuracy of Linear Kernel with 500 iterations is:
%g%%',Max_Accuracy)
set(handles.edit4,'string',Max_Accuracy);
delete(hWaitBar);
guidata(hObject,handles);
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double



% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
test = handles.ImgData3;
Affect = handles.ImgData4;
% Load All The Features
load('Training_Data.mat')

% Put the test features into variable 'test'

result = multisvm(Train_Feat,Train_Label,test);
```

```
%disp(result);

% Visualize Results
if result == 0
    R1 = 'Alternaria Alternata';
    set(handles.edit2,'string',R1);
    set(handles.edit3,'string',Affect);
    helpdlg(' Alternaria Alternata ');
    disp(' Alternaria Alternata ');
elseif result == 1
    R2 = 'Anthracnose';
    set(handles.edit2,'string',R2);
    set(handles.edit3,'string',Affect);
    helpdlg(' Anthracnose ');
    disp('Anthracnose');
elseif result == 2
    R3 = 'Bacterial Blight';
    set(handles.edit2,'string',R3);
    set(handles.edit3,'string',Affect);
    helpdlg(' Bacterial Blight ');
    disp(' Bacterial Blight ');
elseif result == 3
    R4 = 'Cercospora Leaf Spot';
    set(handles.edit2,'string',R4);
    set(handles.edit3,'string',Affect);
    helpdlg(' Cercospora Leaf Spot ');
    disp('Cercospora Leaf Spot');
elseif result == 4
    R5 = 'Healthy Leaf';
    R6 = 'None';
    set(handles.edit2,'string',R5);
    set(handles.edit3,'string',R6);
    helpdlg(' Healthy Leaf ');
    disp('Healthy Leaf ');
end
% Update GUI
guidata(hObject,handles);
```

```
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double



% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double



% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double



% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double



% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a
double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a
double



% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a
double



% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a
double



% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a
double



% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a
double



% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a
double



% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

EVALUATE FEATURES:

```
% Function to call and evaluate features
function [feat_disease seg_img] =  EvaluateFeatures(I)

% Color Image Segmentation
% Use of K Means clustering for segmentation
% Convert Image from RGB Color Space to L*a*b* Color Space
% The L*a*b* space consists of a luminosity layer 'L*', chromaticity-layer
'a*' and 'b*'.
% All of the color information is in the 'a*' and 'b*' layers.
cform = makecform('srgb2lab');
% Apply the colorform
lab_he = applycform(I,cform);

% Classify the colors in a*b* colorspace using K means clustering.
% Since the image has 3 colors create 3 clusters.
% Measure the distance using Euclidean Distance Metric.
ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
nColors = 3;
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean',
...
'Replicates',3);
%[cluster_idx cluster_center] =
kmeans(ab,nColors,'distance','sqEuclidean','Replicates',3);
% Label every pixel in tha image using results from K means
pixel_labels = reshape(cluster_idx,nrows,ncols);
%figure,imshow(pixel_labels,[]), title('Image Labeled by Cluster Index');

% Create a blank cell array to store the results of clustering
segmented_images = cell(1,3);
% Create RGB label using pixel_labels
rgb_label = repmat(pixel_labels,[1,1,3]);

for k = 1:nColors
    colors = I;
```

```
    colors(rgb_label ~= k) = 0;
    segmented_images{k} = colors;
end



figure, subplot(3,1,1);imshow(segmented_images{1});title('Cluster 1');
subplot(3,1,2);imshow(segmented_images{2});title('Cluster 2');
subplot(3,1,3);imshow(segmented_images{3});title('Cluster 3');



% Feature Extraction
x = inputdlg('Enter the cluster no. containing the disease affected leaf part
only:');
i = str2double(x);
% Extract the features from the segmented image
seg_img = segmented_images{i};

% Convert to grayscale if image is RGB
if ndims(seg_img) == 3
   img = rgb2gray(seg_img);
end
%figure, imshow(img); title('Gray Scale Image');

% Evaluate the disease affected area
black = im2bw(seg_img,graythresh(seg_img));
%figure, imshow(black);title('Black & White Image');
m = size(seg_img,1);
n = size(seg_img,2);

zero_image = zeros(m,n);
%G = imoverlay(zero_image,seg_img,[1 0 0]);

cc = bwconncomp(seg_img,6);
diseasedata = regionprops(cc,'basic');
A1 = diseasedata.Area;
sprintf('Area of the disease affected region is : %g%',A1);
```

```
I_black = im2bw(I,graythresh(I));
kk = bwconncomp(I,6);
leafdata = regionprops(kk,'basic');
A2 = leafdata.Area;
sprintf(' Total leaf area is : %g%',A2);

%Affected_Area = 1-(A1/A2);
Affected_Area = (A1/A2);
if Affected_Area < 1
    Affected_Area = Affected_Area+0.15;
end
sprintf('Affected Area is: %g%%',(Affected_Area*100))

% Create the Gray Level Cooccurance Matrices (GLCMs)
glcms = graycomatrix(img);

% Derive Statistics from GLCM
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(seg_img);
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
RMS = mean2(rms(seg_img));
%Skewness = skewness(img)
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
% Inverse Difference Movement
m = size(seg_img,1);
n = size(seg_img,2);
in_diff = 0;
```

```
for i = 1:m
for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
end
end
IDM = double(in_diff);


feat_disease = [Contrast,Correlation,Energy,Homogeneity, Mean,
Standard_Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness,
IDM];
```

RGB2HSI:

```
function hsi = rgb2hsi(rgb)
%RGB2HSI Converts an RGB image to HSI.
%   HSI = RGB2HSI(RGB) converts an RGB image to HSI. The input image
%   is assumed to be of size M-by-N-by-3, where the third dimension
%   accounts for three image planes: red, green, and blue, in that
%   order. If all RGB component images are equal, the HSI conversion
%   is undefined. The input image can be of class double (with values
%   in the range [0, 1]), uint8, or uint16.
%
%   The output image, HSI, is of class double, where:
%     hsi(:, :, 1) = hue image normalized to the range [0, 1] by
%                    dividing all angle values by 2*pi.
%     hsi(:, :, 2) = saturation image, in the range [0, 1].
%     hsi(:, :, 3) = intensity image, in the range [0, 1].




% Extract the individual component images.
rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);
%num = 0.5*((r - g) + (r - b));
%den = sqrt((r - g).^2 + (r - b).*(g - b));
```

```
%theta = acos(num./(den + eps));
% Implement the conversion equations.
num = 0.5*((r - g) + (r - b));
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps));
%num = min(min(r, g), b);
%den = r + g + b;
%den(den == 0) = eps;
%S = 1 - 4.* num./den;
%num = min(min(r, b), g);
%den = r + g + b;
%den(den == 0) = eps;
%S = 1 - 3.* num./den;
H = theta;
H(b > g) = 2*pi - H(b > g);
H = H/(2*pi);
%H = theta;
%H(b > g) = 4*pi - H(b > g);
%H = H/(3*pi);
%H = theta;
%H(b > g) = 2/pi - H(b > g);
%H = H/(2*pi);
num = min(min(r, g), b);
den = r + g + b;
den(den == 0) = eps;
S = 1 - 3.* num./den;
%num = min(min(r, b), b);
%den = r + g + b;
%den(den == 0) = eps;
%S = 1 + 3.* num./den;
%num = min(min(b, g), b);
%den = r + g + b;
%den(den == 0) = eps;
%S = 1 - 3.* num.*den;
H(S == 0) = 0;
I = (r + g + b)/3;
% Combine all three results into an hsi image.
```

```
hsi = cat(3, H, S, I);
```

TRAIN:

```
% Training Part
%Features of Anthracnose
for i=1:25
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Anthracnose\',num2str(i),'.jpg']);
    img = imresize(img,[256,256]);
    img = imadjust(img,stretchlim(img));
    imshow(img);title('Anthracnose Leaf Image');
    [feat_disease seg_img] =  EvaluateFeatures(img);
    Anthracnose_Feat(i,:) = feat_disease;
    save Anthracnose_Feat;
    close all
end

% Features of Bacterial Blight
for i=1:25
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Bacterial Blight\',num2str(i),'.jpg']);
    img = imresize(img,[256,256]);
    img = imadjust(img,stretchlim(img));
    imshow(img);title('Blight Leaf Image');
    [feat_disease seg_img] =  EvaluateFeatures(img);
    Blight_Feat(i,:) = feat_disease;
    save Blight_Feat;
    close all
end

% Features of Alternaria Alternata
for i=1:25
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Alternaria Alternata\',num2str(i),'.jpg']);
    img = imresize(img,[256,256]);
    img = imadjust(img,stretchlim(img));
    imshow(img);title('Alternaria Leaf Image');
```

```
    [feat_disease seg_img] =  EvaluateFeatures(img);
    Alternaria_Feat(i,:) = feat_disease;
    save Alternaria_Feat;
    close all
end

% Featurs of Cercospora Leaf Spot
for i=1:25
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Cercospora Leaf Spot\',num2str(i),'.jpg']);
    img = imresize(img,[256,256]);
    img = imadjust(img,stretchlim(img));
    imshow(img);title('Cercospora Leaf Image');
    [feat_disease seg_img] =  EvaluateFeatures(img);
    Cercospora_Feat(i,:) = feat_disease;
    save Cercospora_Feat;
    close all
end

% Features of Healthy Image
for i=1:25
    disp(['Processing frame no.',num2str(i)]);
    img=imread(['Healthy Leaves\',num2str(i),'.jpg']);
    img = imresize(img,[256,256]);
    img = imadjust(img,stretchlim(img));
    imshow(img);title('Healthy Leaf Image');
    [feat_disease seg_img] =  EvaluateFeatures(img);
    Healthy_Feat(i,:) = feat_disease;
    save Healthy_Feat;
    close all
end


%
% Accuracy Evaluation Dataset Preparation
close all
clear all
```

```
clc
load('Alternaria_Feat.mat')
load('Anthracnose_Feat.mat')
load('Blight_Feat.mat')
load('Cercospora_Feat.mat')
load('Healthy_Feat.mat')

Train_Feat =
[Alternaria_Feat;Anthracnose_Feat;Blight_Feat;Cercospora_Feat;Healthy_Feat];
Train_Label = [ zeros(100,1); ones(25,1) ];
save Accuracy_Data

%
```

TEST:

```
T=[1 2 3 4;2 3 4 5;3 4 5 6;4 5 6 7]
C=[1 2 3 4]
tst=[3 4 5 6]
results = multisvm(T, C, tst);
disp('multi class problem');
disp(results);
```

MULTISVM:

```
function [itrfin] = multisvm( T,C,test )
%Inputs: T=Training Matrix, C=Group, test=Testing matrix
%Outputs: itrfin=Resultant class

itrind=size(test,1);
fprintf('itrind = %i\n',itrind);
itrfin=[];
fprintf('itrfin = %i    ',itrfin);
fprintf('\n');
Cb=C;
fprintf('Cb = %i    ', Cb);
fprintf('\n');
Tb=T;
fprintf('Tb = %i    ',Tb);
```

```
fprintf('\n');
for tempind=1:itrind
    fprintf('in %i st for tempind = %i , itrind = %i
',tempind,tempind,tempind);
    fprintf('\n');
    tst=test(tempind,:);
    fprintf('tst = %i    ',tst);
    fprintf('\n');
    C=Cb;
    T=Tb;
    u=unique(C);
    fprintf('u = %i    ',u);
    fprintf('\n');
    N=length(u);
    fprintf('N = %i    ',N);
    fprintf('\n');
    c4=[];
    fprintf('c4 = %i    ',c4);
    fprintf('\n');
    c3=[];
    fprintf('c3 = %i    ',c3);
    fprintf('\n');
    j=1;
    k=1;
if(N>2)
        itr=1;
        classes=0;
        cond=max(C)-min(C);
        fprintf('in if condition cond = %i    ',cond);
        fprintf('\n');
while((classes~=1)&&(itr<=length(u))&& size(C,2)>1 && cond>0)
            fprintf('\n\nin while condition:----------------------------------
-----------------------------------------\n classes = %i  itr = %i size(C,2) =
%i  cond = %i',classes,itr,size(C,2),cond);
            fprintf('\n');
%This while loop is the multiclass SVM Trick
            c1=(C==u(itr));
            fprintf('c1 = %i    ',c1);
```

```
            fprintf('\n');
            newClass=c1;
            fprintf('newclass = %i    ',newClass);
            fprintf('\n');
%svmStruct = svmtrain(T,newClass,'kernel_function','rbf'); % I am using rbf
kernel function, you must change it also
            svmStruct = svmtrain(T,newClass);
%fprintf('svmStruct = %i    ',svmStruct);
%fprintf('\n');
            classes = svmclassify(svmStruct,tst);
            fprintf('classes = %i    ',classes);
            fprintf('\n');
% This is the loop for Reduction of Training Set
for i=1:size(newClass,2)
                fprintf('\nin 2nd for loop:\nitration i = %i to
%i',i,size(newClass,2));
                fprintf('\n');
                fprintf('check newclass(1,i) element %i is 0',newClass(1,i));
                fprintf('\n');
if newClass(1,i)==0;
                    c3(k,:)=T(i,:);
                    fprintf('then in T index at %i to last element attach in
c3 from %i to last', i,k);
                    fprintf('\n');
                    k=k+1;
end
end
        T=c3;
        fprintf('New reduction training set:\n ');
        fprintf('T = %i    ', T);
        c3=[];
        k=1;

% This is the loop for reduction of group
for i=1:size(newClass,2)
                fprintf('\nin 3rd for loop:\nitration i = %i to
%i',i,size(newClass,2));
                fprintf('\n');
```

```
                fprintf('check newclass(1,i) element %i is 0',newClass(1,i));
                fprintf('\n');
if newClass(1,i)==0;
                    c4(1,j)=C(1,i);
                    fprintf('then in c4 index at %i  element replace to  C at
index %i to last', j,i);
                    fprintf('\n');
                    j=j+1;
end
end
        C=c4;
        fprintf('New reduction Group set:\n ');
        fprintf('C = %i    ', C);
        fprintf('\n');
        c4=[];
        j=1;

        cond=max(C)-min(C); % Condition for avoiding group
%to contain similar type of values
%and the reduce them to process
        fprintf('check condition cond = %i', cond);
        fprintf('\n');
% This condition can select the particular value of iteration
% base on classes
if classes~=1
                itr=itr+1;
end
end
end

valt=Cb==u(itr);
fprintf('\n\noutside while:\n');
fprintf('valt = %i    ',valt);
fprintf('\n');
% This logic is used to allow classification
val=Cb(valt==1);
fprintf('val = %i    ',val);
```

```
fprintf('\n');
% of multiple rows testing matrix
val=unique(val);
fprintf('val = %i    ',val);
fprintf('\n');
itrfin(tempind,:)=val;
fprintf('itrfin(tempind,:) = %i   ',itrfin(tempind,:));
fprintf('\n');
end

end
```