Weed and Himyaritic Of Leaf Detection System With I.P

B.E. Dissertation

Submitted in partial fulfillment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering

(Electronics & Telecommunication Engineering)

by

Tanki Saim (13ET60)

Khan Aziz (13ET22)

Khan Junaid (14ET24)

Malik Mohammed Ajmal(14ET28)

Under the guidance of

Asst. Prof. Chaya.S Ravindra

Department of Electronics and Telecommunication Engineering

Anjuman-I-Islam's Kalsekar Technical Campus,

Sector 16, New Panvel, Navi Mumbai - 410206

(Affiliated to University of Mumbai)

Academic Year: 2018-19

Anjuman-I-Islam's

Kalsekar Technical Campus

Plot No. 2 3, Sector - 16, Near Thana Naka, Khandagao, New Panvel,

Navi Mumbai, Maharashtra - 410206

Certificate

This is to certify that, the dissertation titled

"Weed Himyaritic Of Leaf Detection System With I.P"

is a bonafide work done by

Tanki Saim (13ET60)

Khan Aziz (13ET22)

Khan Junaid (14ET24)

Malik Mohammed Ajmal (14ET28)

and is submitted in the partial fulfillment of the requirement for the degree of

Bachelor of Engineering

in

Electronics & Telecommunication Engineering

to the

University of Mumbai

Supervisor	**Examiner**
Head of Department	**Director**

Project Report Approval For Bachelor Of Engineering

This is to certify that the dissertation entitled **"Weed Himyaritic Of Leaf Detection System With I.P"** is a bonafide work done by **Tanki Saim**, **Khan Aziz**, **Khan Junaid** and **Malik Mohammed Ajmal** under the guidance of **Prof. Chaya.S Ravindra** This dissertation has been approved for the award of **Bachelor's Degree** in **Electronics & Telecommunication Engineering**, **University of Mumbai**.

Examiners:

____

Examiner

____

Supervisor

Declaration

We declare that this written submission represents ours ideas in ours own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in ours submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Tanki Saim(13ET60)

Khan Aziz (13ET22)

KhanJunaid(14ET24)

MalikAjmal(14ET28)

Date:April 12, 2019 .

Acknowledgement

We are highly grateful to the **PROF. AFZAL SHAIKH, HOD of Electronics & Telecommunication Department, Kalsekar Technical Campus (New Panvel)**, for providing this opportunity to carry out the project. We would like to express our gratitude to other faculty members of Electronics & Telecommunication Engineering Department for providing academic inputs, guidance and encouragement throughout this period. We would like to express our deep sense of gratitude and thank to **ASST. PROF. CHAYA S Ravindra**, for the wise council and able guidance it would have not been possible to carry out this project in this manner. Finally, we express our indebtedness to all who have directly or indirectly contributed to the successful completion of this project.

Tanki Saim (13ET60) ____________________

Khan Aziz (13ET22) ____________________

Khan Junaid (14ET24) ____________________

Malik Mohd Ajmal (14ET28) ______________

Abstract

India, the country where the main source of income is from agriculture. Farmers grow a variety of crops based on their requirement. Since the plants suffer from the disease, the production of crop decreases due to infections caused by several types of diseases on its leaf, fruit, and stem. Leaf diseases are mainly caused by bacteria, fungi, virus etc. Diseases are often difficult to control. Diagnosis of the disease should be done accurately and proper actions should be taken at the appropriate time. Image Processing is the trending technique in detection and classification of plant leaf disease. This work describes how to automatically detect leaf diseases. The given system will provide a fast, spontaneous, precise and very economical method in detecting and classifying leaf diseases. This paper is envisioned to assist in the detecting and classifying leaf diseases using Multiclass SVM classification technique. First, the affected region is discovered using segmentation by K-means clustering, then features (color and texture) are extracted. Lastly, classification technique is applied in detecting the type of leaf disease.

Keywords: *Image Processing, Leaf diseases detection, K-means clustering, feature extraction, Multiclass SVM Classification.*

Contents

Chapter 1

INTRODUCTION

1.1 Statement Of Project

The old method for detection of plant disease was very difficult, inconvenient and required a lot of expertise people to conduct experiment and detect the disease. This method was also time consuming and not so accurate. Image processing technique reduces human effort, requires less time and achieves higher rate of accuracy. This detects the plant disease at a very early stage and this helps the plant from getting destroyed. The productivity of the plants is increased which has a lot of benefits.

1.1.1 Project Architecture

The steps involved are

1) Training In training all the collected images are trained to the model and the 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

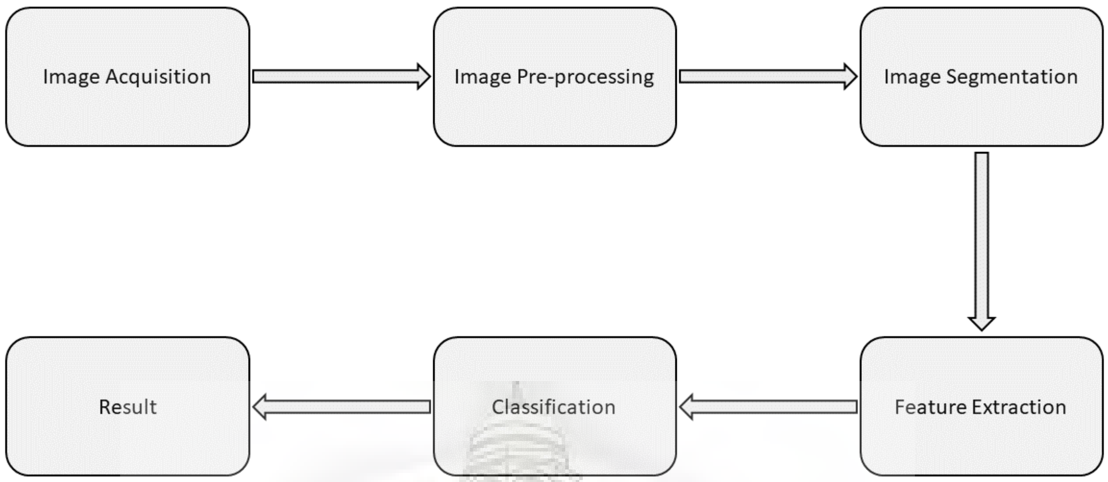


Figure 1.1: System Architecture

1) Image Acquisition
2) Image pre-processing
3) Image segmentation
4) Feature extraction
5) Classification

1.2 Objective and Goal:

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 a 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. Hence the disease needs to be identified at the early stages, recommending farmers to avoid the harm in the production of the crop to increase the yield. Plants suffer from diseases like Alternaria alternate (fungal), Anthracnose, Bacterial Blight (bacteria).

Plant disease will be basically identified by observing different patterns on the parts of the plant like leaf. The indications on the leaf are taken into consideration for detecting the disease. The disease can be categorized as bacterial, viral, fungal etc. The proposed work emphases on identifying and categorizing the disease like Alternaria Alternata, Anthracnose, Bacterial Blight, which are basically found on pomegranate, rice, soya bean, carrot, rose, watermelon, mango etc., using the technique called as image processing. It automatically detects leaf diseases. This system will provide a fast, spontaneous, precise and very economical method in detecting and classifying leaf diseases. The steps involved in disease detection are image acquisition, image pre processing, image segmentation, feature extraction, detection and classification. India is an agricultural country hence there is no less scope as far as utilization is concerned. The application will range from different fields. Some of them are as follows:

- Agriculture
- Nursery
- Biotechnology lab
- Forest department

Chapter 2

Literature Review

[1]The accurately detection and classification of the plant disease is very important for the successful cultivation for crops and this can be done using image processing. This paper discuss various techniques to segment the disease part of the plant. This paper also discuss some feature extraction and classification to extract the feature of infected leaf and classification of plnt disease. The usee of ANN methods for classification of disease in plant such as self organizing feature map, back propagation algorithm svm etc.

[2]In this review of image segmentation techniques, various image segmentation techniques are detailed described and compared. These all techniques are suitable for many medical image applications. These techniques can be used for object recognition and detection. In medical images these can be used to detect cancer and in satellite images these can be used to detect roads and bridges. Thus it is clear that various methods are suitable for various types of image applications. But from the study it is clear that no single method is sufficient for every image type and no all methods are suitable for a particular image type. Due to the need of image segmentation in many

applications, it has a challenging future.

[3]The machine learning methods bring this aspect to reality, by observing the database and helping the botanists in the diagnosis of diseases where a lot of precision is required. And one of the machine learning technique, SVM is used in this project for classification of leaf diseases. The accuracy results in an available range from mid-90 to top 90 percent. This can be bettered by increasing the database. However, the results obtained from real life images are very encouraging.

Chapter 3

Plant Leaf Diseases

A symptom of plant disease is a visible effect of disease on the plant. Symptoms may include a detectable change in color, shape or function of the plant as it responds to the pathogen. Learning about the major diseases that can affect your plants and how to treat those diseases can help to make your harvest healthy

Leaf spot is a common descriptive term applied to a number of diseases affecting the foliage of ornamentals and shade trees. The majority of leaf spots are caused by fungi, but some are caused by bacteria. Some insects also cause damage that appears like a leaf spot disease.

3.1 Alternaria alternata

Alternaria alternata is a fungus which has been recorded causing leaf spot and other diseases on over 380 host species of plant. It is an pathogen on numerous hosts causing leaf spots, rots and blights on many plant parts.

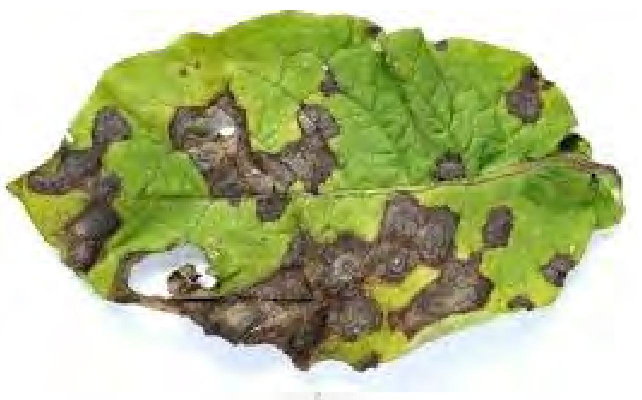


Figure 3.1: Alternaria Alternata

3.1.1 Hosts and symptoms:

Alternaria alternata has many different hosts depending on its forma species. This pathogen only infects certain cultivars of tomato plants and is often referred to as Alternaria stem canker of tomato. AALs main symptom is cankers in the stem. It resides in seeds and seedlings, and is often spread by spores as they become airborne and land on plants. It can also spread throughout other plants. Under severe infection, lesions enlarge and become coalesced causing blighting of the leaves. he symptoms on affected tomatoes started with yellowing and browning of the lower leaves, then began developing on the leaf tips and along the margins of the leaf petiole. This progression continued until the entire leaves were covered in diseased tissue and then fell off. The tomato fruit can also be infected as well, with brown cankers dotting them and making them inedible. Once the disease has spread to a certain point, little can be done to save the tomato plant.

3.1.2 Environment:

In order to survive, alternaria alternata needs a moist warm environment. It is often found in areas with humid climates, or where there has been significant rainfall. The fungus lives in seeds and seedlings, and is also spread by spores. This disease flourishes in dead plants that have been left in gardens over winter. Additionally, when dead infected debris is added to compost pile it can spread to other vegetables throughout the garden. There are no insect vectors for this disease. This means that using insecticides has no effect on the susceptibility of a tomato plants susceptibility to this pathogen. However, there are several cultural practices that can be done to suppress this fungal pathogen impact.

The disease first occurs in the hosts exposed leaves. Plants planted with rows in an east-west direction have more severe disease than do plants planted north-south. This implies that if one plants tomato plants in a north-south manner they will be less susceptible. Overall, AAL thrives in moist warm environments. There are no insect vectors, so applying insecticides have no effect on a plants resistance to this disease. Cultural practices for preventing this disease include planting tomatoes in a row north to south, monitoring plants heavily April through June, and using a drip irrigation system to keep as much plant tissue dry and free of favorable environments for this pathogen.

3.2 Anthracnose

Anthracnose is a general term used to describe diseases that result in a wide range of symptoms including leaf spots, blotches or distortion, defoliation, shoot blight, twig cankers and dieback on many different deciduous trees and shrubs. In most cases, *Anthracnose* does not cause permanent damage to established trees. However, consecutive years of defoliation can decrease the trees vigor, weakening the tree and thereby predisposing the plant to opportunistic pests that may further harm or damage the tree.

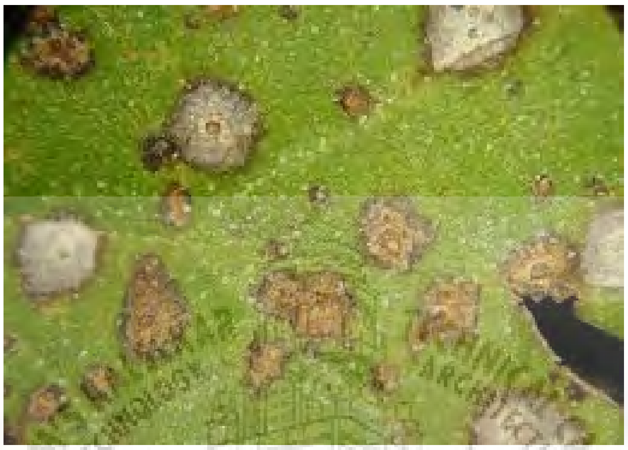


Figure 3.2: Anthracnose

3.2.1 Pathogen and susceptible plants:

A number of related fungi are responsible for *Anthracnose*. Most *Anthracnose* causing fungi are fairly host specific; they will infect many species of ash for example but will not infect maple or oak trees. Green ash, elm, sugar maple, Norway maple, white oak and black walnut are examples of frequently infected tree species in Minnesota.

3.2.2 Identification:

Tan to brown irregular shaped spots or blotches on leaves; often located close to leaf veins.

1. Infected leaves may be distorted, cupped or curled.
2. Severe infection can result in leaf drop in spring. A second growth of

leaves occurs by midsummer.
3. Infections on mature leaves are irregular tan spots, often associated with minor wounds like insect feeding. Leaf distortion is rarely seen in these infections.
4. Infections on green twigs can be small orange brown blisters to a brown band encircling the young twig resulting in shoot death. These infections are most common on young twigs of oak (Quercus spp.) and ironwood (Ostrya virginiana).
5. Disease is often most severe on the lower and inner branches of the tree but may progress up through the canopy.
6. In Minnesota, the disease is most common during cool, wet spring weather.

3.2.3 Biology:

Anthracnose fungi can over-winter in buds, twigs, fruit, fallen leaves or petioles depending on which hosts and pathogens are involved. The disease cycle begins in spring when spores are dispersed short distances by water or spread long distances by air to newly forming leaves. Spores are produced within new leaf infections several days to weeks after the initial infection and are further spread to new locations by splashing water.

The disease is most common in spring when new shoot and leaf growth are combined with temperatures ranging from 50-68F and spring rain. *Anthracnose* can also reoccur in the summer when cool, wet weather is paired with succulent leaf growth. For ash, maple and oak trees, young leaves and shoots are highly susceptible to infection from the *Anthracnose* fungi, but mature fully expanded leaves are largely resistant. Mature leaves of these trees only become infected through minor wounds like damage from insect pests. As a result, once the weather becomes dry and the leaves mature, dis-

ease growth will end, and the tree will replace lost leaves with a new flush of growth. In contrast, *Anthracnose* can continue to progress through summer months on trees like walnut and hornbeam.
Leaf spotting and leaf distortion have very little affect on the health of the tree. However if a tree is severely defoliated multiple years in a row, this can weaken the tree. In such cases, opportunistic pests like boring insects or canker causing fungi can attack the tree resulting in more significant damage.

3.2.4 Management:

If a tree is suffering from *Anthracnose*, reduce other stresses on the tree by maintaining them properly with adequate watering throughout the growing season and fertilizing only if determined necessary by a soil test.
1. Always plant healthy trees on the correct site for the species.
2. Species of certain trees may vary in susceptibility to *Anthracnose*. When possible choose the most resistant tree available.
3. White oak (Quercus alba) is highly susceptible, while red oak (Q. rubra) is relatively resistant.
4. Blue ash (Fraxinus quadrangulata) is highly resistant. Pumpkin (F. tomentosa) and American ash (F. americana) are less susceptible than green ash (F. pennsylvanica) and Chinese ash (F. chinensis).
5. Rock elm (Ulmus thomasii) is most resistant, Chinese elm (U. parvifolia) is intermediate, while American elm (U. americana) is susceptible.
6. Black walnut (Juglans nigra) is severely susceptible, while heartnut (J. ailanthifolia var. cordiformis) and Japanese walnut (J. ailanthifolia) are less susceptible.
7. Wet conditions promote disease so avoid or redirect sprinklers that wet the lower canopy of the tree.

8. Rake up and destroy fallen leaves before the first snowfall to eliminate locations where the fungus can survive to re-infect the plant the following growing season.
9. Prune the tree or shrub to remove infected twigs, increase light penetration, and improve air circulation throughout the canopy.

3.3 Cercospora Leaf Spot

Cercospora leaf spot can be caused by many different *Cercospora* fungal pathogen species depending on the plant type infected. For example, *Cercospora* beticola infects sugar beets whereas *Cercospora* rosicola infects rose plants. This is considered a foliar disease and can be especially devastating to sugar beet crops in North Dakota and Minnesota. This disease is sometimes misdiagnosed as black spot.

Infection will begin at the bottom of the plant and will work up toward leaves with new growth. This happens when the fungal spores germinate and enter through natural openings of leaves when conditions are optimal. If leaves do not have high moisture levels on their outside tissue for at least 11 hours, new spores are unlikely to enter the plant. While infection is most commonly found on leaves, sometimes stems, bracts, or fruit can also be affected. Once spores infect a plant, it will take 5 to 21 days for symptoms to appear. This fungus has the ability to overwinter in plant debris that remains in a field after harvest.

3.3.1 Identification:

As the name suggests, this disease causes spots on foliage. The circular blemishes typically have an average diameter of 3 millimeters that is between

brown and reddish- purple in color, bordering a gray center. However, if spots have just developed, the gray center will not yet be visible.
When this fungus experiences favorable conditions, it may progress through 4 or 5 cycles in one season. Each progressive cycle is often more severe than the previous one, with spots sometimes growing to have a diameter of 10 millimeters. As these spots kill the plant cells, leaves will begin to fall from the plant, causing defoliation. A plants yield will be reduced from the limited photosynthetic capacity and can also result in more juice impurities and less sucrose extraction for sugar beets.

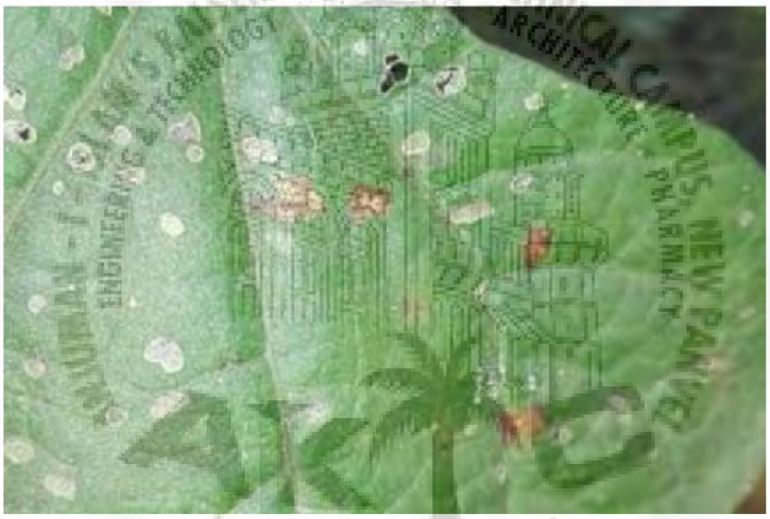


Figure 3.3: Cercospora

3.3.2 Habitat:

Cercospora leaf spot favors weather that is wet, warm, and humid; often most prevalent following canopy closure. During the day this fungus thrives when temperatures are between 80 and 90 degrees Fahrenheit and at night when temperatures exceed 60 degrees Fahrenheit. The disease will not develop well if temperatures reach 93 degrees or higher. Humidity levels are preferred between 90 and 100 percent. More susceptible conditions for *Cercospora* involve areas near waterways, plants close to other fields that were infected in the past, and those near shelter-belts.

3.3.3 Cultural Control:

Since infection grows worse with time, early control is essential for effective management of this disease. In order to do this, scout fields frequently to catch any infection early, especially in highly susceptible areas. Also, whenever possible keep

vulnerable plants away from areas that are most commonly affected by this disease. For example, plant sugar beets no closer than 100 yards from other previously infected areas. Practice fall tillage to bury infected plant debris that could be a possible overwintering site for *Cercospora* spores. Rotate crops, giving sugar beets a 2 year break in-between and replacing them with a crop type that is less susceptible. It is important to note that having satisfactory control is often dependent on using an integrated management system that includes the use of both cultural and chemical methods.

3.3.4 Chemical Control:

Fungicides should be applied once first symptoms appear. Most often, additional applications will be needed to continue effective control, especially when conditions are favorable for this fungus; however, if using Topsin with a protectant (e.g., Supertin), this application should only be used once per season. Other fungicides with the following active ingredients have also shown to be effective for the control of *Cercospora* : copper oxychloride, sulfur, maneb, mancozeb, chlorothalonil, propiconazole, and neem oil. Always be sure to carefully read the fungicides label for cautions and proper application before use. In order to avoid this fungi building a resistance to a certain chemical, ensure the same fungicide is not being used back-to- back. Additionally, if only one application type is being used annually, do not use the same type of application. High spray pressure is best, often 100 psi, and also high water volume. Using both of these methods can create more effective control. On average, fungicides can provide protection for 2 weeks. Also, treatment methods in place for powdery mildew and black spot have also helped in the management of *Cercospora* .

Chapter 4

Methodology

4.1 Image Acquisition

A digital image is a numeric representation, normally binary, of a two-dimensional image. Depending on whether the image resolution is fixed, it may be of vector or raster type. By itself, the term **"digital image"** usually refers to raster images or bitmapped images. Raster images have a finite set of digital values, called picture elements or pixels. The digital image contains a fixed number of rows and columns of pixels. Pixels are the smallest individual element in an image, holding antiquated values that represent the brightness of a given color at any specific point. Typically, the pixels are stored in computer memory as a raster image or raster map, a two-dimensional array of small integers. These values are often transmitted or stored in a compressed form.Raster images can be created by a variety of input devices and techniques, such as digital cameras, scanners, coordinate-measuring machines, seismographic profiling, airborne radar, and more. They can also be synthesized from arbitrary non-image data, such as mathematical functions or three-dimensional geometric models; the latter being a major sub-area of computer graphics. The field of digital image processing is the study of algorithms for their transformation. Image viewer software displays images. Web browsers can display standard internet image formats including GIF, JPEG, and PNG. Some can show SVG format which is a standard W3C format. In the past, when Internet was still slow, it was common to provide "preview" image that would load and appear on the web site before being replaced by the main image.

In Image processing, it is defined as the action of retrieving an image from some source, usually a hardware based source for processing. It is the first step in the work flow sequence because, without an image, no processing is possible. The image that is acquired is completely unprocessed.

Digital camera, scanner or similar devices are use to take image of leaf of different types, and then are use to identify affected area in the leaf. This image is in RGB form.

4.1.1 Camera:

You can choose between two main types of cameras analog and digital.Analog cameras are cameras that generate a video signal in analog format. The analog signal is digitized by an image acquisition device. The video signal is based on the television standard, making analog the most common standard for representing video signals.Digital cameras have several advantages over analog cameras. Analog video is more susceptible to noise during transmission than digital video. By digitizing at the camera level rather than at the image acquisition device, the signal-to-noise ratio is typically higher, resulting in better accuracy.

4.1.2 Scanner:

An image scanner is a device that optically scans images, printed text, handwriting or an object and converts it to a digital image.

Figure 4.1: Leaf Scanner

4.2 Image pre-processing

Image enhancement is basically improving the Interpret ability or perception of information in images for human viewers and providing better input for other automated image processing techniques. There exist many techniques that can enhance a digital image without spoiling it. The enhancement methods can broadly be divided in to the following two categories:

4.2.1 Spatial Domain Methods

In spatial domain techniques, we directly deal with the image pixels. The pixel values are manipulated to achieve desired enhancement. The value of a pixel with coordinates (x,y) in the enhanced image $\hat{F}$ is the result of performing some operation on the pixels in the neighbourhood of (x,y) in the input image.

4.2.2 Frequency Domain Methods

In frequency domain methods, the image is first transferred in to frequency domain. It means that, the Fourier Transform of the image is computed first. All the enhancement operations are performed on the Fourier transform of the image and then the Inverse Fourier transform is performed to get the resultant image. Image enhancement is applied in every field where images are tough to be understood and analyzed.

Contrast stretching

It is also calledas Normalization attempts to improve an image by stretching the range of intensity values it contains to make full use of possible values. Unlike histogram equalization, contrast stretching is restricted to a linear mapping of input to output values. The result is less dramatic, but tends to avoid the sometimes artificial appearance of equalized images.

The first step is to determine the limits over which image intensity values will be extended. These lower and upper limits will be called a and b, respectively (for standard 8-bit grayscale pictures, these limits are usually 0 and 255). Next, the histogram of the original image is examined to determine the value limits (lower = c, upper = d) in the unmodified picture. If the original range covers the full possible set of values, straightforward contrast stretching will achieve nothing, but even then sometimes most of the image data is contained within a restricted range; this restricted range can be stretched linearly, with original values which lie outside the range being set to the appropriate limit of the extended output range. Then for each pixel, the original value r is mapped to output value s using the function:

4.3 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.

4.3.1 K-Mean Clustering

K-mean 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.

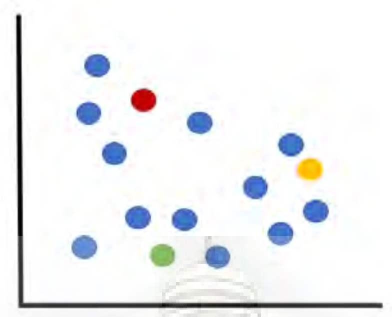


Figure 4.2: **Step 1.** Shows the initial randomized point and anumber of points

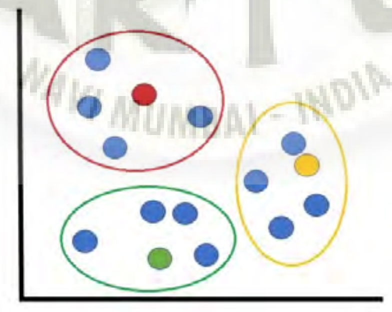


Figure 4.3: **Step 2.** Points are associated with the nearest initial randomized

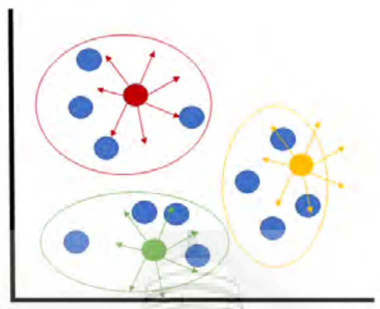


Figure 4.4: **Step 3.** Now the initial randomized points are moved to the center of their respective cluster(the centroids)

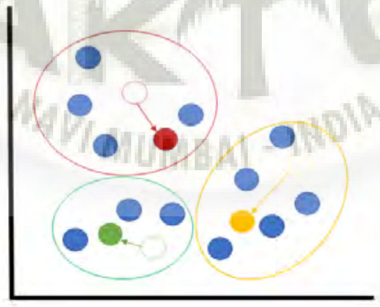


Figure 4.5: **Step 4.** step 23 are repeated until s suitable level of convergence has been reached

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=

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

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

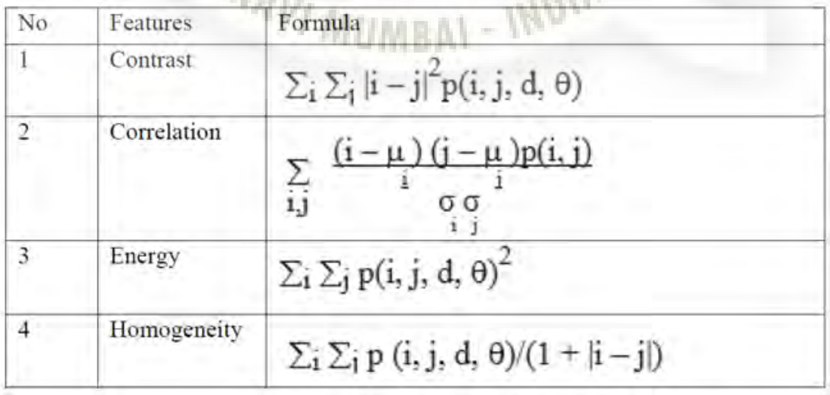
4. The data point which is not close to the cluster to which it belongs to should be then shifted to the nearby cluster.

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

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

4.3.2 Feature Extraction

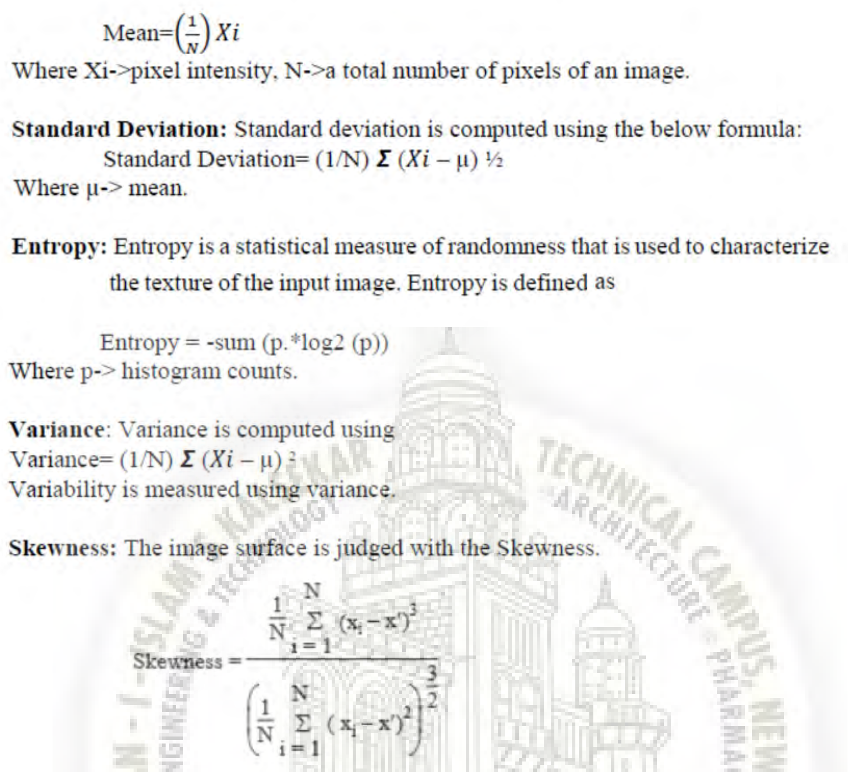
From the input images, the features are to be extracted. To do so instead of choosing the whole set of pixels we can choose only which are necessary and sufficient to describe the whole of the segment. The segmented image is first selected by manual interference. The affected area of the image can be 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.
Using the statistical MATLAB commands the other properties are found out. Those are Mean Standard Deviation, Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness, and IDM. Mean: Average or mean value of the array.



No	Features	Formula
1	Contrast	$\sum_i \sum_j \lvert i-j\rvert^2 p(i,j,d,\theta)$
2	Correlation	$\sum_{i,j} \frac{(i-\mu_i)(j-\mu_j)p(i,j)}{\sigma_i \sigma_j}$
3	Energy	$\sum_i \sum_j p(i,j,d,\theta)^2$
4	Homogeneity	$\sum_i \sum_j p(i,j,d,\theta)/(1+\lvert i-j\rvert)$

Figure 4.6: Formula

standard deviation is given by $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\overline{x})^2}$.



Mean= $\left(\frac{1}{N}\right) Xi$

Where Xi->pixel intensity, N->a total number of pixels of an image.

Standard Deviation: Standard deviation is computed using the below formula:

Standard Deviation= (1/N) $\boldsymbol{\Sigma}$ $(Xi - \mu)$ ½

Where μ-> mean.

Entropy: Entropy is a statistical measure of randomness that is used to characterize the texture of the input image. Entropy is defined as

Entropy = -sum (p.*log2 (p))

Where p-> histogram counts.

Variance: Variance is computed using

Variance= (1/N) $\boldsymbol{\Sigma}$ $(Xi - \mu)^2$

Variability is measured using variance.

Skewness: The image surface is judged with the Skewness.

$$\text{Skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - x')^3}{\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - x')^2\right)^{\frac{3}{2}}}$$

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.

4.3.3 Classification

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyper planes.

4.3.4 What is Support Vector Machine?

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in ndimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. 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.

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 greater than 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 (k1)/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 binary SVM for each of the class to isolate the members of one class from the other class.

4.3.5 How does it work?

Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

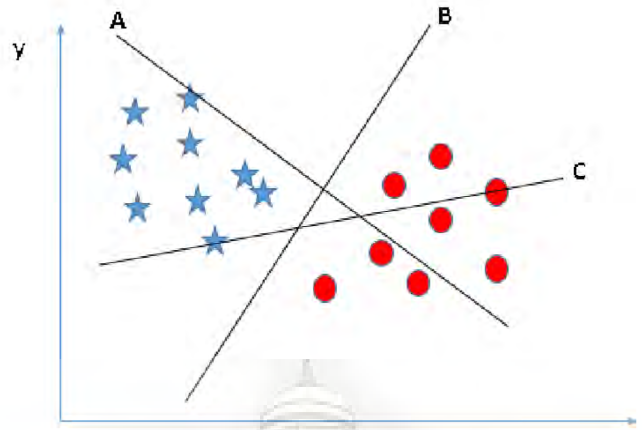


Figure 4.7:

You need to remember a thumb rule to identify the right hyper-plane: Select the hyper-plane which segregates the two classes better. In this scenario, hyper-plane B has excellently performed this job.

Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

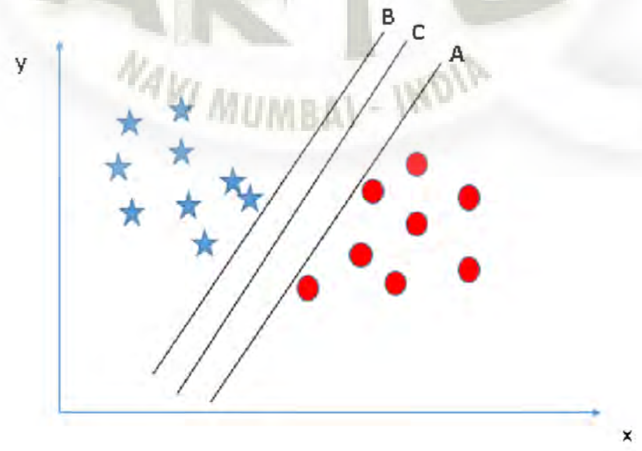


Figure 4.8:

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Above fig 3.8, you can see that the margin for hyper-

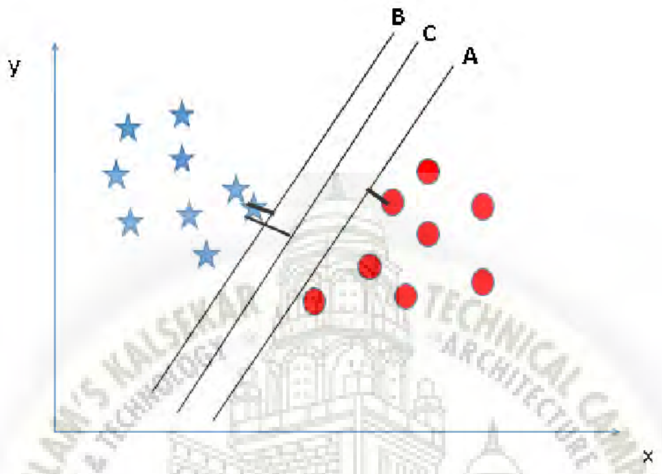


Figure 4.9:

plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyperplane having low margin then there is high chance of miss-classification.

Identify the right hyper-plane (Scenario-3): Hint: Use the rules as discussed in previous section to identify the right hyper-plane

Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.

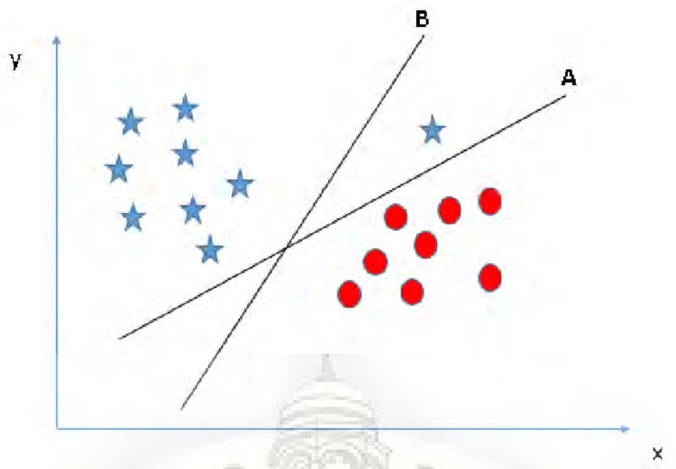


Figure 4.10:

Here fig 3.10, hyper-plane **B** has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A**.

Can we classify two classes (Scenario-4): Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.

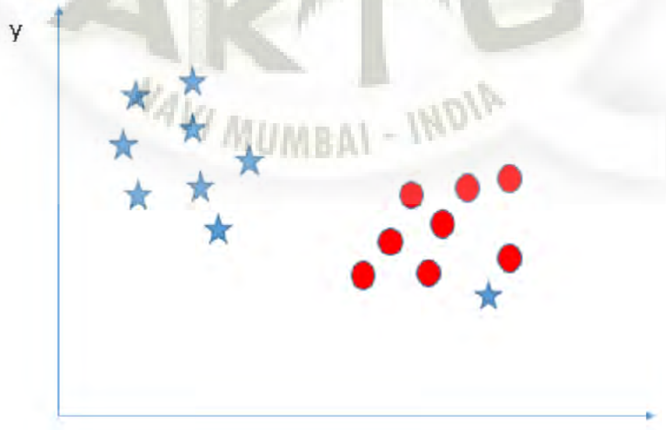


Figure 4.11:

As I have already mentioned, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

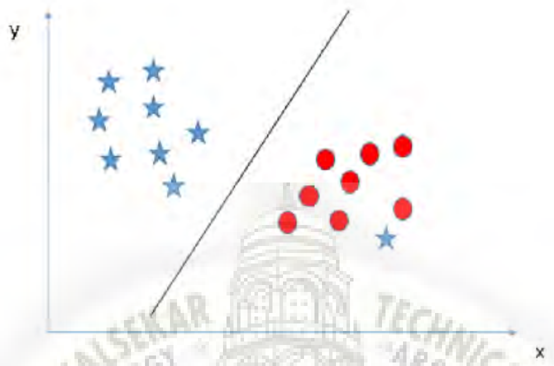


Figure 4.12:

Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we cant have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.

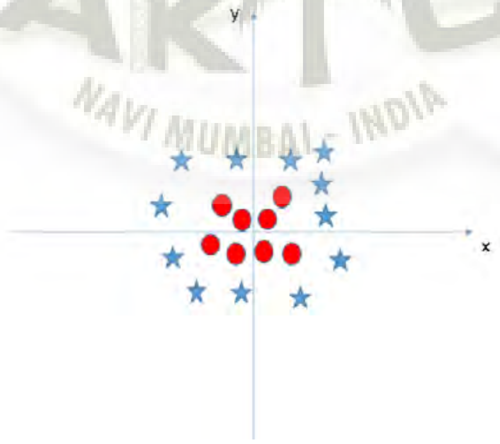


Figure 4.13:

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z = $x^2 + y^2$.

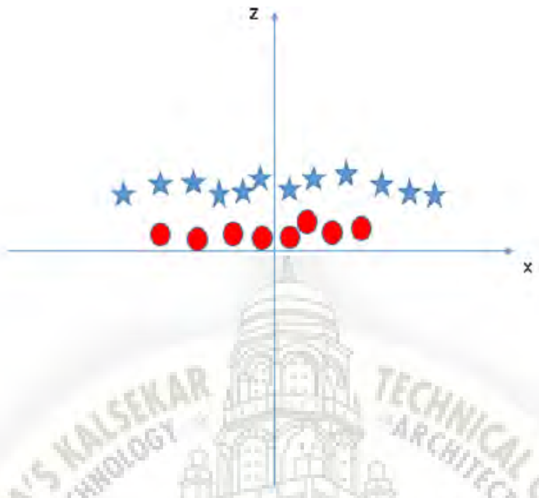


Figure 4.14:

In above plot, points to consider are: o All values for z would be positive always because z is the squared sum of both x and y o In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z. In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called **kernels**.

4.3.6 Advantages of SVM Classifier:

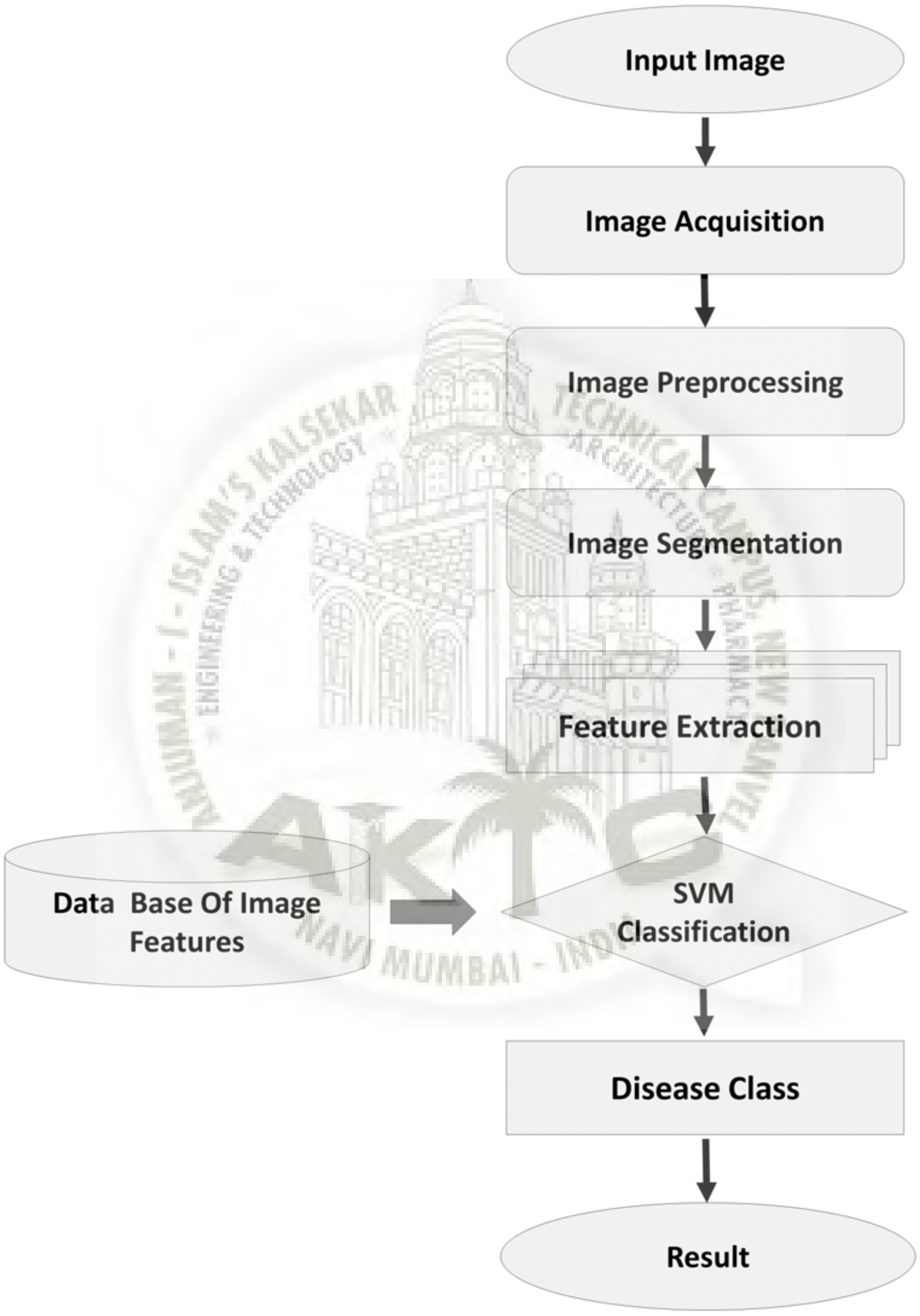
SVMs are effective when the number of features is quite large. It works effectively even if the number of features are greater than the number of samples.

Non-Linear data can also be classified using customized hyperplanes built by using kernel trick. It is a robust model to solve prediction problems since it maximizes margin.

4.3.7 Disadvantages of SVM Classifier:

The biggest limitation of Support Vector Machine is the choice of the kernel. The wrong choice of the kernel can lead to an increase in error percentage. With a greater number of samples, it starts giving poor performances. SVMs have good generalization performance but they can be extremely slow in the test phase. SVMs have high algorithmic complexity and extensive memory requirements due to the use of quadratic programming.

4.3.8 Work Flow of the Project



Chapter 5

MATLAB Graphical User Interface

The graphical user interface, is a type of user interface that allows users to interact with electronic devices through graphical icons. Using GUI we can perform any computations, communicate with any other GUIs, plot graps,create tables etc. MATLAB GUI contains several user interface tools like radio buttons,axes,check box,tables.sliders,list box,pannels..etc. Adding appropriate components we can create a GUI design for any application. GUI interface is an event driven programming. Flow of the application is based on events. Events may be click,double click,key press etc. Callback functions will be executed once an event occurs. We can edit the properties of each callback functions for making suitable response from the GUI as the user ineracts.
GUIDE generates two files for each GUI:
.fig file: it contains the layout of the GUI.
.m file: it contains the code that is needed to control GUI behavior.

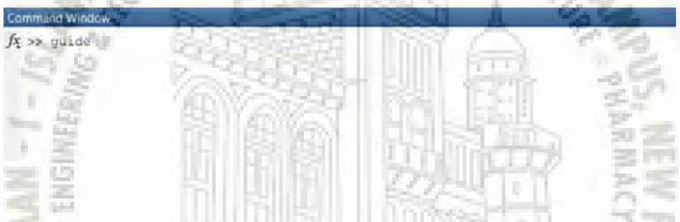
5.0.1 Why use GUI in MATLAB?

The main reason GUIs are used is because it makes things simple for the end-users of the program. If GUI's were not used, people would have to work from the command line interface, which can be extremely difficult and fustrating. Imagine if you had to input text commands to operate your web browser

5.0.2 How to creat basic GUI in matlab?

Initializing GUIDE (GUI Creator)

1. First, open up MATLAB. Go to the command window and type in guide



```
fx >> guide
```

Figure 5.1:

2. You should see the following screen appear. Choose the first option Blank GUI (Default).

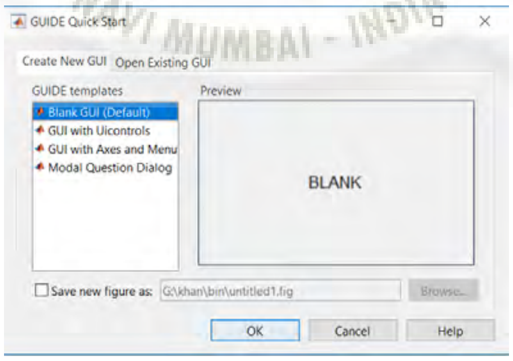


Figure 5.2:

3. You should now see the following screen (or something similar depending on what version of MATLAB you are using and what the predesignated settings are):

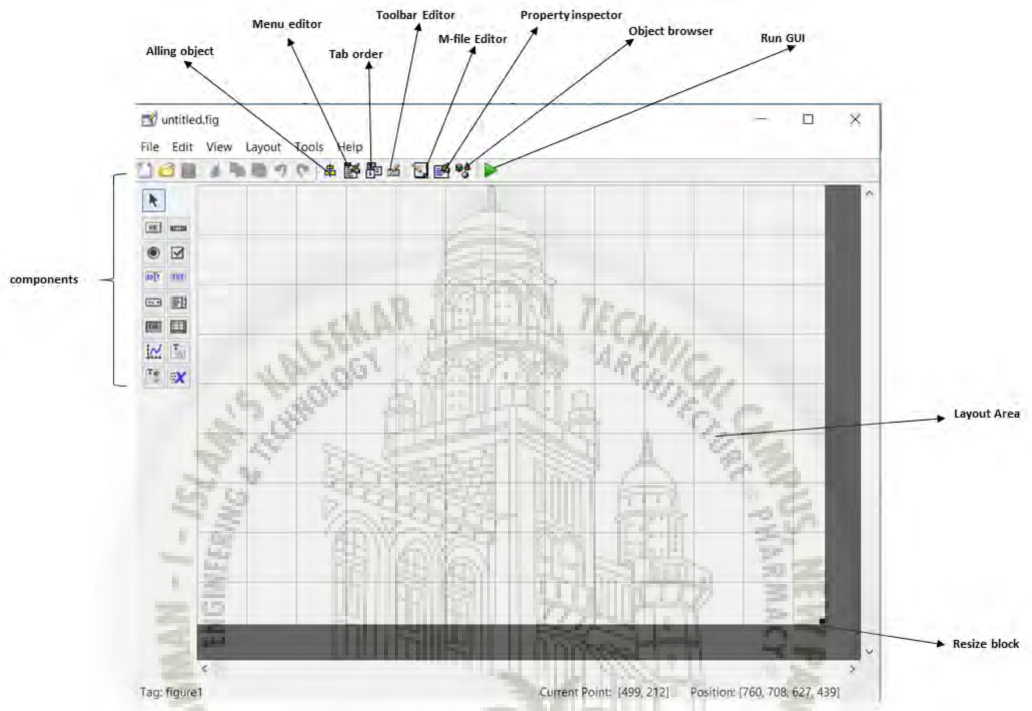


Figure 5.3:

4. Before adding components blindly, it is good to have a rough idea about how you want the graphical part of the GUI to look like so that itll be easier to lay it out. Below is a sample of what the finished GUI might look like.

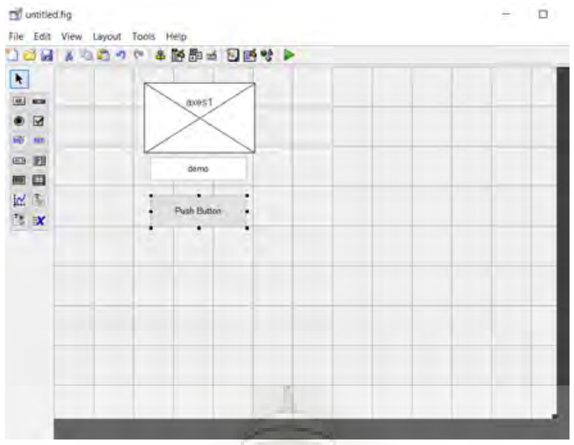


Figure 5.4:

5. after designing the GUI then create the push button function.

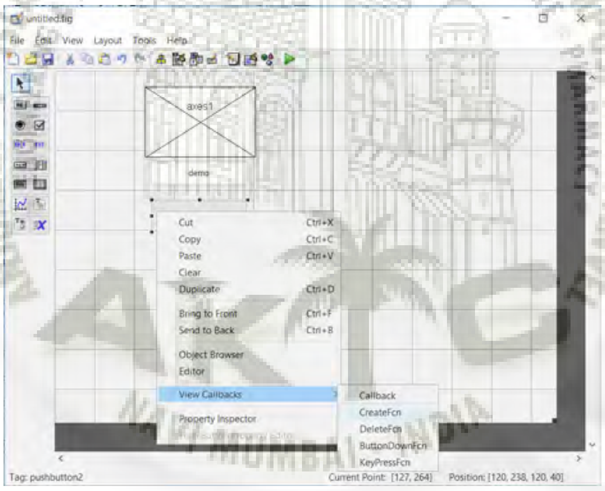
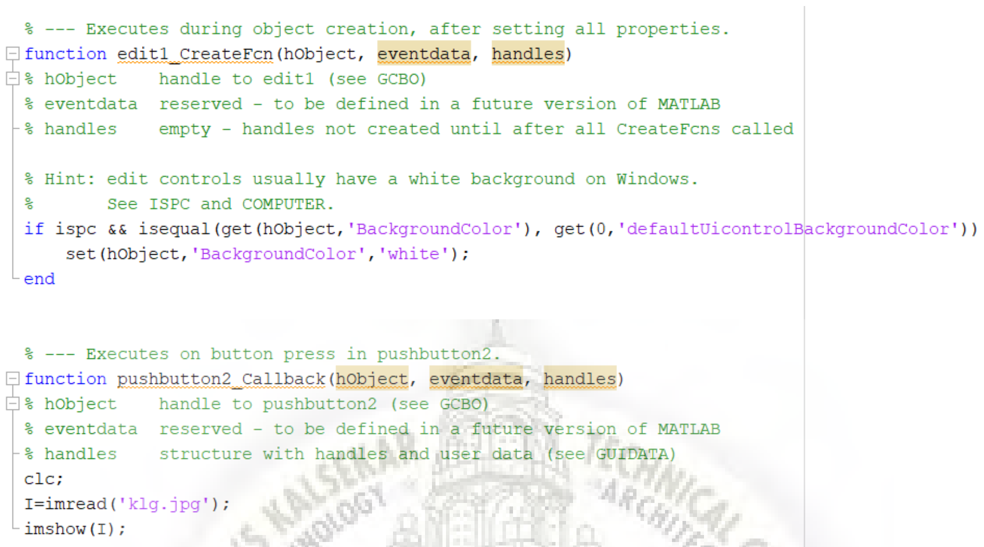


Figure 5.5:

6. write the MATLAB code in editor.



```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
I=imread('klg.jpg');
imshow(I);
```

Figure 5.6:

7. Run the program

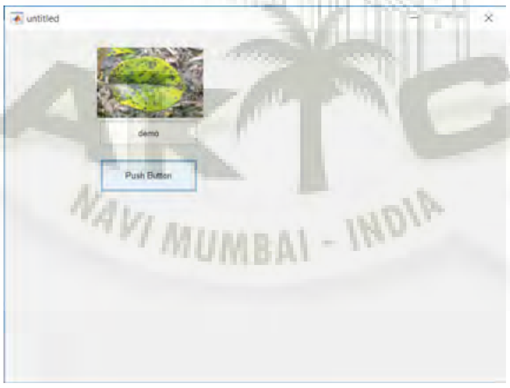


Figure 5.7:

MATLAB code for K-means Clustering Algorithm

```
% Project Title: Plant Leaf Disease Detection & Classification

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 07-Mar-2019 12:10:03

% 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','linear');
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 ');
    fprintf(arduino,'%s',char('1')); % send successfully matched to arduino
    disp('Serial Data Sent');
    fclose(arduino); % end communication with arduino
elseif result == 1
    R2 = 'Anthracnose';
    set(handles.edit2,'string',R2);
    set(handles.edit3,'string',Affect);
    helpdlg(' Anthracnose ');
    disp('Anthracnose');
    fprintf(arduino,'%s',char('2')); % send successfully matched to arduino
    disp('Serial Data Sent');
    fclose(arduino); % end communication with arduino
elseif result == 2
    R3 = 'Bacterial Blight';
    set(handles.edit2,'string',R3);
    set(handles.edit3,'string',Affect);
    helpdlg(' Bacterial Blight ');
    disp(' Bacterial Blight ');
    fprintf(arduino,'%s',char('3')); % send successfully matched to arduino
    disp('Serial Data Sent');
    fclose(arduino); % end communication with arduino
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');
    fprintf(arduino,'%s',char('4')); % send successfully matched to arduino
    disp('Serial Data Sent');
    fclose(arduino); % end communication with arduino
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


```

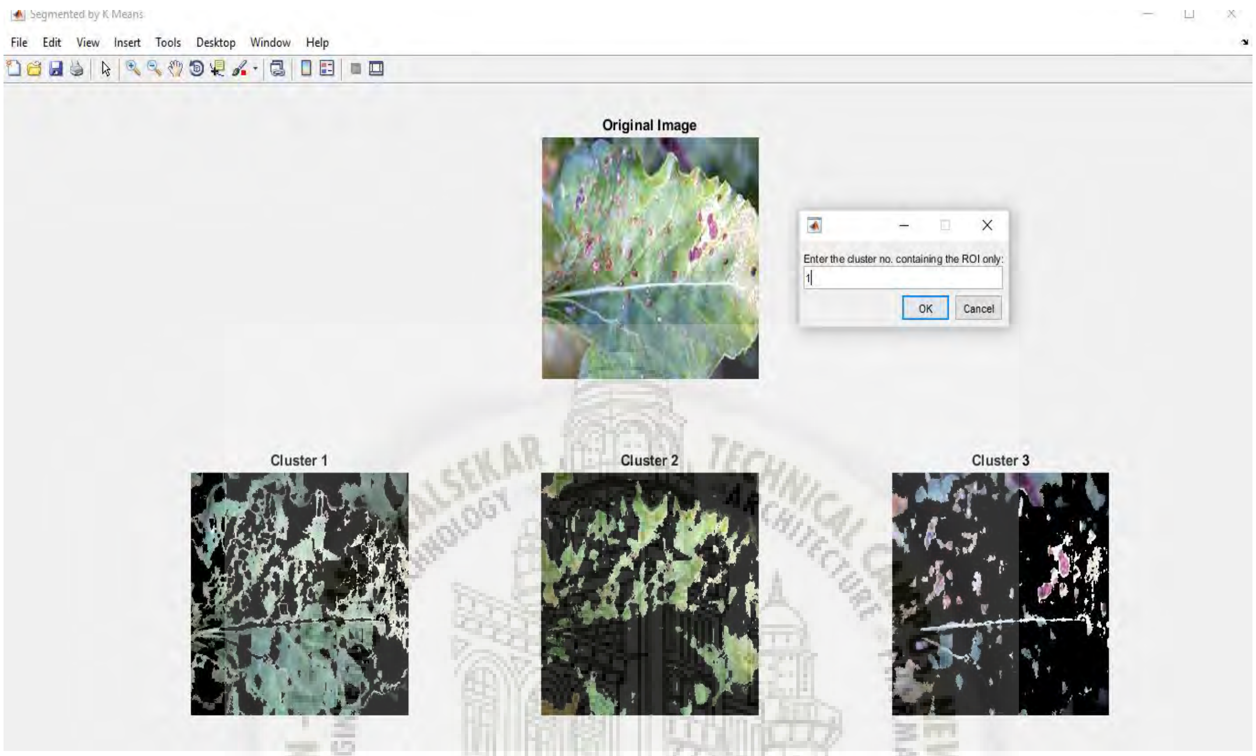
```

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


```

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
vid = videoinput('winvideo', 1);
set(vid, 'ReturnedColorSpace', 'RGB');
I = getsnapshot(vid);
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 pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
arduino=serial('COM2','BaudRate',9600); % create serial communication object 
on port COM4
fopen(arduino); % initiate arduino communication
disp('Serial Port Initialized');
```

Output of the program

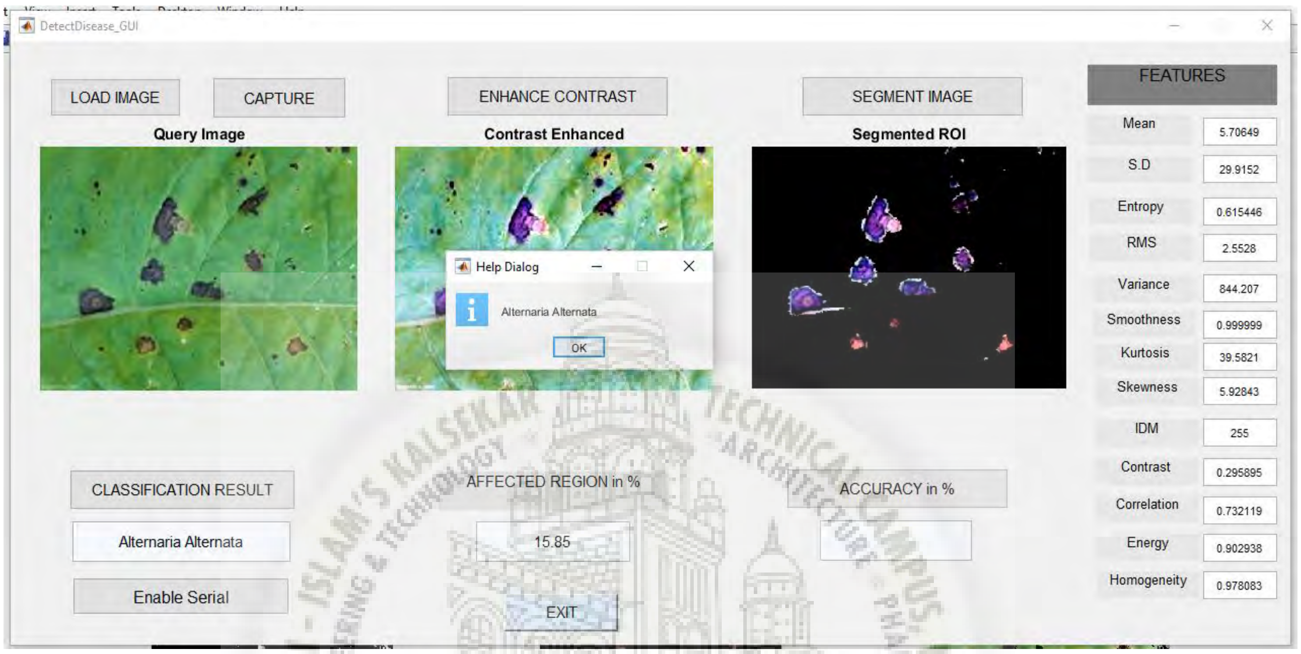


Enter the cluster no. containing the ROI only:
1

Figure 5.8: Plant leaf segmentation using k-means clusteirng

Result

Alternaria alternate



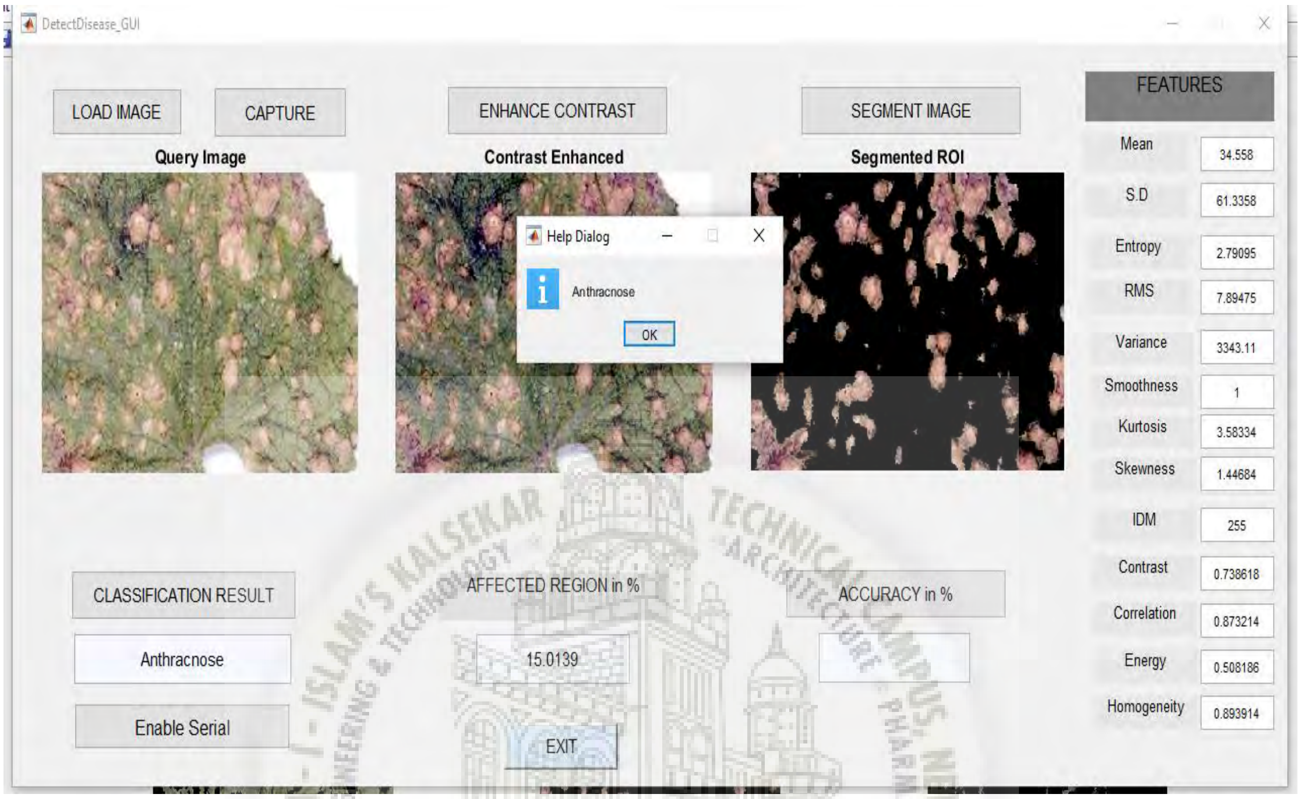
Alternaria Alternata

Classification result: Alternaria Alternata

Affected region in %: 15.85

FEATURES	
Mean	5.70649
S.D	29.9152
Entropy	0.615446
RMS	2.5528
Variance	844.207
Smoothness	0.999999
Kurtosis	39.5821
Skewness	5.92843
IDM	255
Contrast	0.295895
Correlation	0.732119
Energy	0.902938
Homogeneity	0.978083

Anthracnose



Anthracnose

AFFECTED REGION in %: 15.0139

ACCURACY in %:

CLASSIFICATION RESULT: Anthracnose

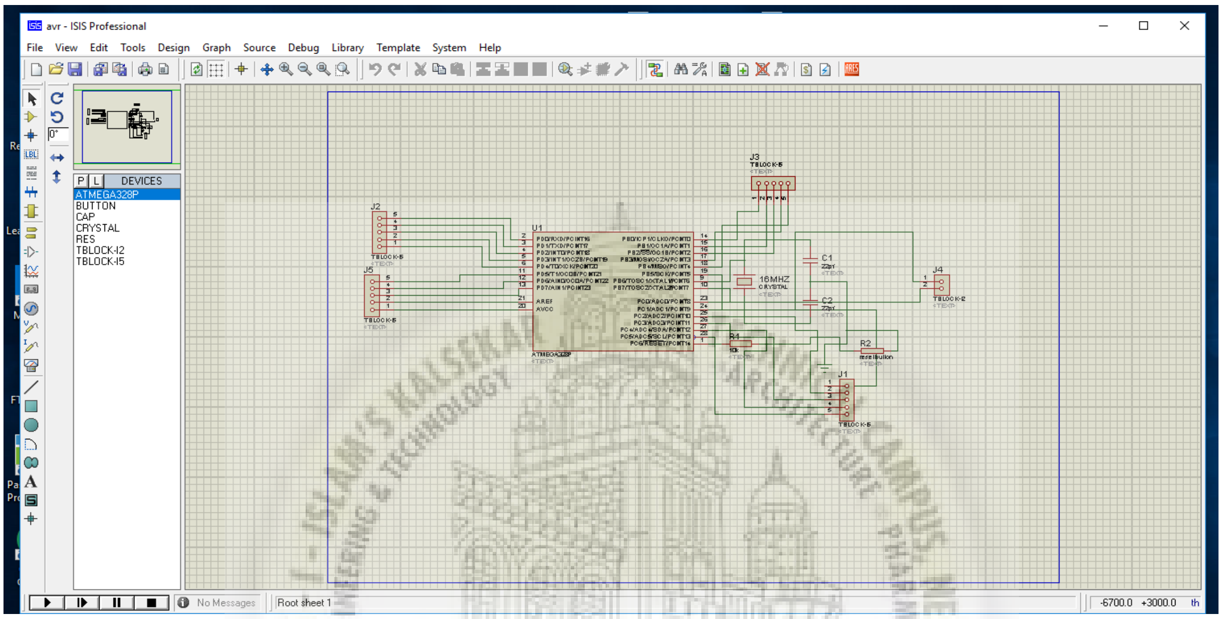
FEATURES	
Mean	34.558
S.D	61.3358
Entropy	2.79095
RMS	7.89475
Variance	3343.11
Smoothness	1
Kurtosis	3.58334
Skewness	1.44684
IDM	255
Contrast	0.738618
Correlation	0.873214
Energy	0.508186
Homogeneity	0.893914

Hardware Requirement

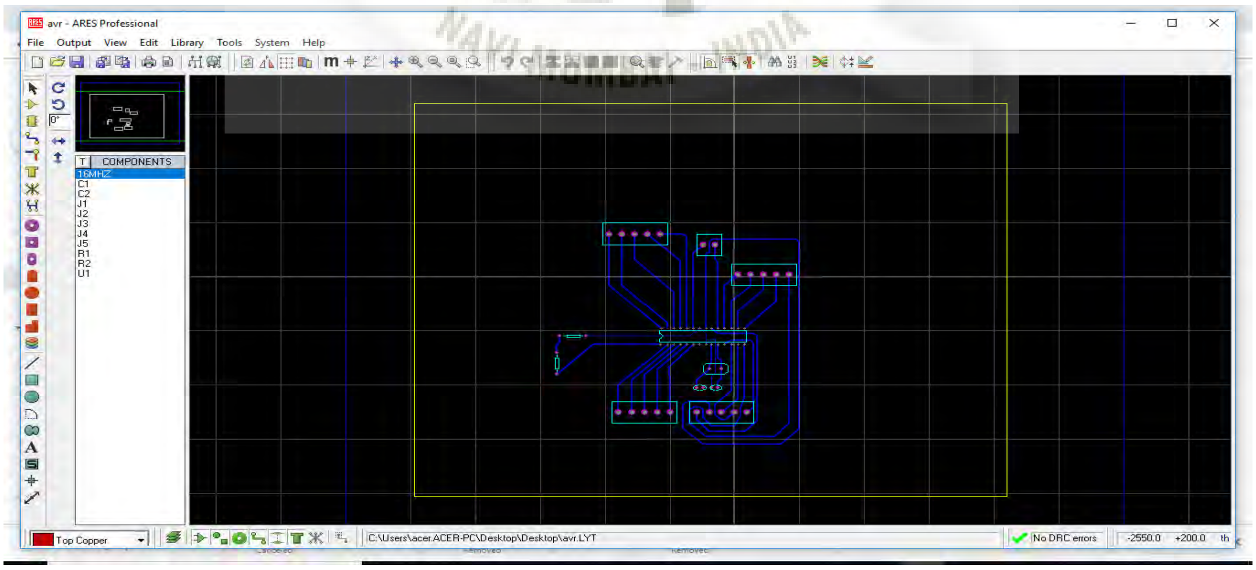
Digital camera , Laptop, Arduino board, Lcd, Motor diver

Designing of arduion board :-

In the first we have design it on Ares interface which will look like this



In the second step we well make sure all the component take less space so it can look Good and we will also make appropriate trace size through which an adducate amount Current can flow through it so that all component can work properly.



A liquid-crystal display (LCD)

Fig.1 A liquid-crystal display (LCD)

It is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly.

LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images which can be displayed or hidden, such as preset words, digits, and 7-segment displays as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.

GENERAL SPECIFICATION:

- Drive method: 1/16 duty cycle
- Display size: 16 character * 2 lines
- Character structure: 5*8 dots.
- Display data RAM: 80 characters (80*8 bits)
- Character generate ROM: 192 characters
- Character generate RAM: 8 characters (64*8 bits)
- Both display data and character generator RAMs can be read from MPU.
- Internal automatic reset circuit at power ON.
- Built in oscillator circuit.

PIN Configuration

JP1/JP14 Pins 1 – 8	Description	JP1/JP14 Pins 9 -16	Description
Pin1	Ground	Pin9	D2 (Not Used)
Pin2	VCC (+5)	Pin10	D3 (Not Used)
Pin3	Contrast	Pin11	D4
Pin4	Data/Command (R/S)	Pin12	D5
Pin5	Read/Write (W)	Pin13	D6
Pin6	Enable (E1)	Pin14	D7
Pin7	D0 (Not Used)	Pin15	VCC (LEDSV+)
Pin8	D1 (Not Used)	Pin16	Ground

L293D

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. Dual H-bridge *Motor Driver integrated circuit* (*IC*).

The l293d can drive small and quiet big motors as well, check the Voltage Specification at the end of this page for more info.

You can Buy L293D IC in any electronic shop very easily and it costs around 70 Rupees (INR) or around 1 $ Dollar (approx Cost) or even lesser cost. You can find the necessary pin diagram, working, a circuit diagram, Logic description and Project as you read through.

Fig.2 L239D Ic Model

Concept:-

It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor.

In a single L293D chip there are two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below is the pin diagram of a L293D motor controller.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If anyone of the either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It’s like a switch.

TIP: you can simply connect the pin16 VCC (5v) to pin 1 and pin 9 to make them high.

L293D Pin Diagram:-

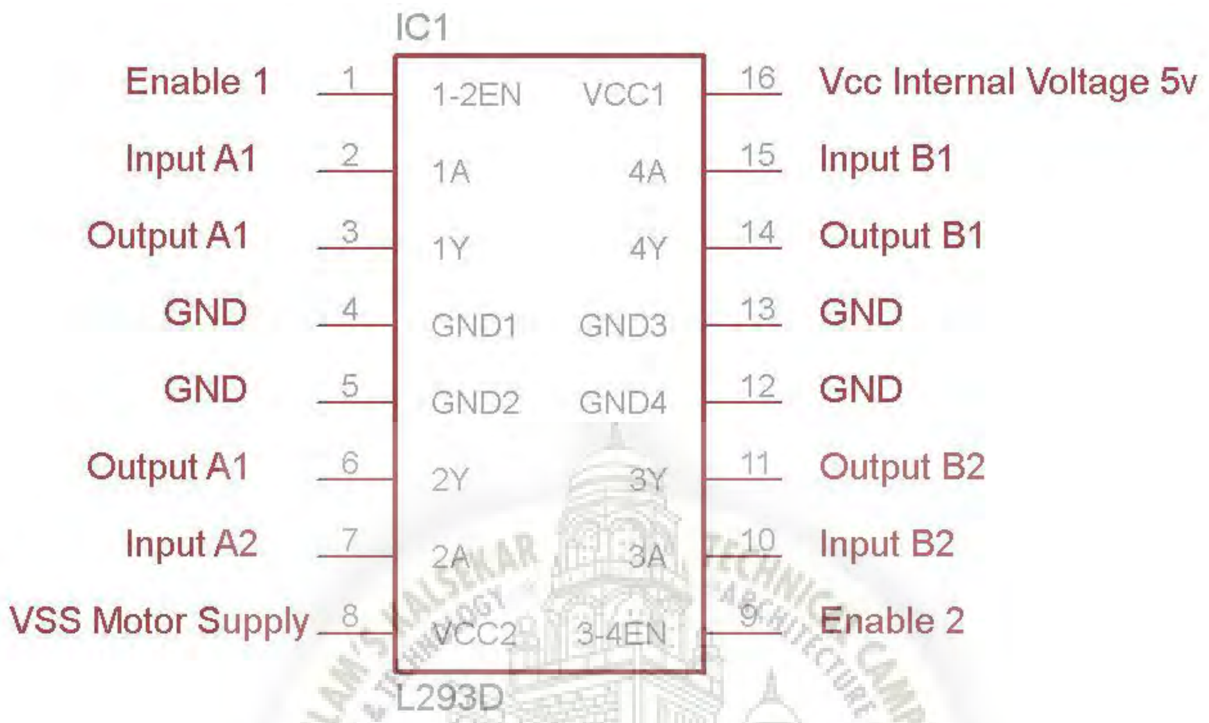


Fig .3 L293D Pin Diagram

Working of L293D:-

There are 4 input pins for l293d, pin 2,7 on the left and pin 15 ,10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.In simple you need to provide Logic 0 or 1 across the input pins for rotating the motor.

Final Hardware Design

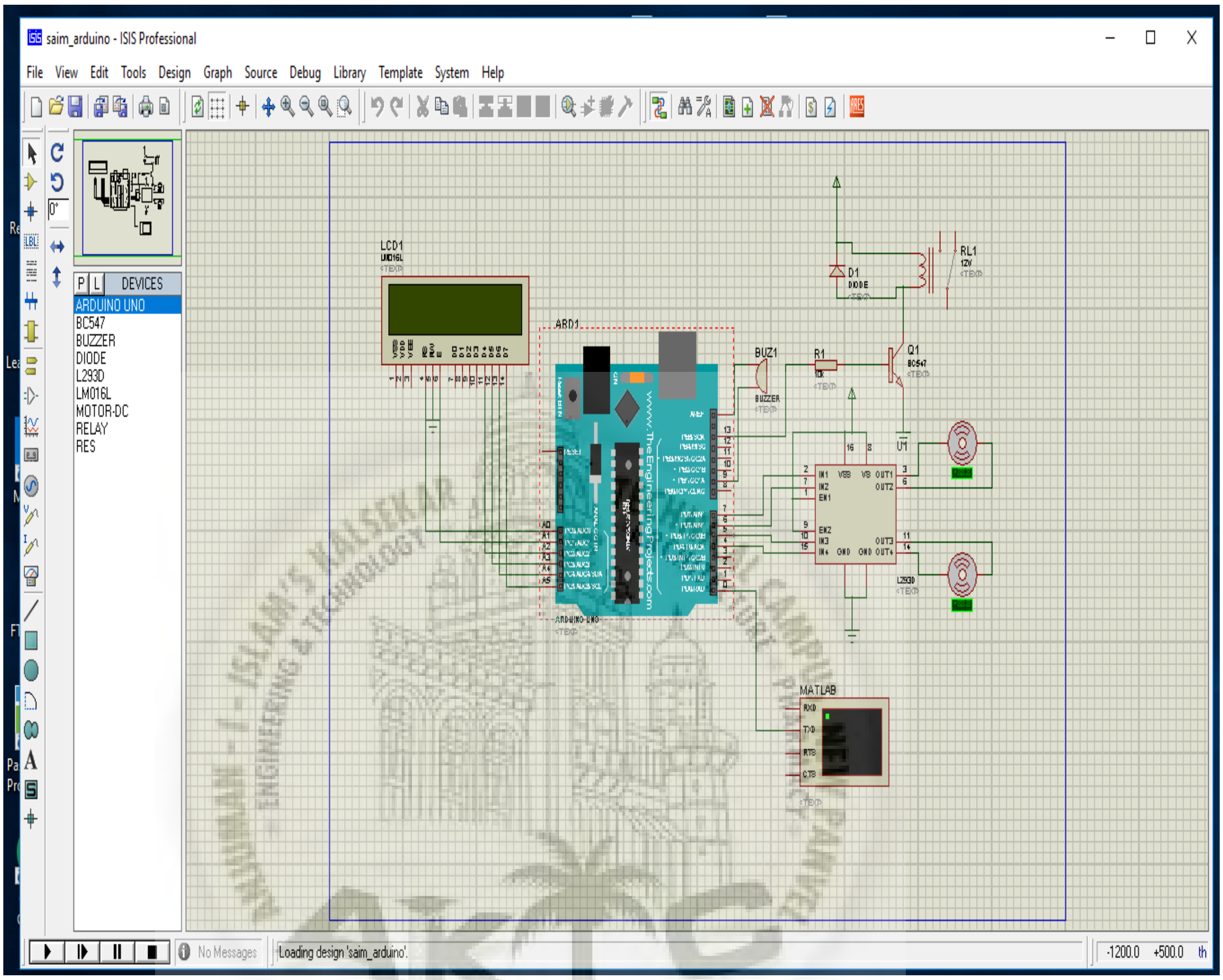


Fig.4 Final output Design

Project Requirements

Software Requirement

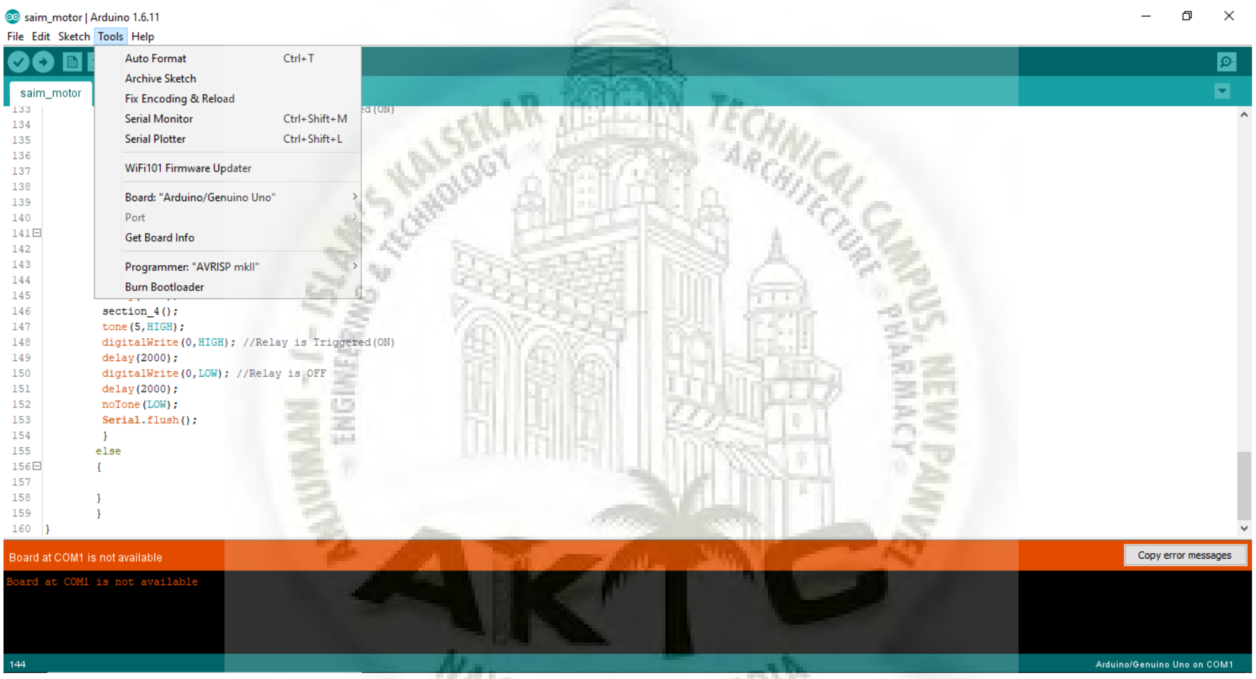
MATLAB Softwar3, Arduino IDE, Proteus for PCB designing

Code for arduino:-

```
#include <LiquidCrystal.h>
LiquidCrystallcd(A0, A1, A2, A3, A4, A5);

charindata;
int M1=4;
int M2=5;
int M3=6;
int M4=7;


void setup()
{
pinMode(M1,OUTPUT);
pinMode(M2,OUTPUT);
pinMode(M3,OUTPUT);
pinMode(M4,OUTPUT);
pinMode(0,HIGH);
pinMode(5,OUTPUT);
Serial.begin(9600);
lcd.begin(16, 2);
lcd.print("LEAF DETECTION");
lcd.setCursor(0,1);
lcd.print("System - IP");
delay(1000);
}

void section_1()
{
digitalWrite(M1,HIGH);
digitalWrite(M2,LOW);
delay(1000);
digitalWrite(M3,LOW);
digitalWrite(M4,HIGH);
delay(1000);
digitalWrite(M3,LOW);
digitalWrite(M4,LOW);
digitalWrite(M1,LOW);
digitalWrite(M2,LOW);
}
void section_2()
{
digitalWrite(M1,HIGH);
digitalWrite(M2,LOW);
delay(2000);
digitalWrite(M3,LOW);
digitalWrite(M4,HIGH);
delay(2000);
digitalWrite(M3,LOW);
digitalWrite(M4,LOW);
digitalWrite(M1,LOW);
digitalWrite(M2,LOW);
}

void section_3()
{
digitalWrite(M1,HIGH);
digitalWrite(M2,LOW);
delay(3000);
digitalWrite(M3,LOW);
digitalWrite(M4,HIGH);
delay(3000);
digitalWrite(M3,LOW);
digitalWrite(M4,LOW);
digitalWrite(M1,LOW);
digitalWrite(M2,LOW);
}

void section_4()
{
digitalWrite(M1,HIGH);
digitalWrite(M2,LOW);
delay(4000);
digitalWrite(M3,LOW);
digitalWrite(M4,HIGH);
delay(4000);
digitalWrite(M3,LOW);
digitalWrite(M4,LOW);
digitalWrite(M1,LOW);
digitalWrite(M2,LOW);
}

void loop()
{
lcd.clear();
lcd.print("Waiting...");
lcd.setCursor(0,1);
lcd.print("for Cmd / Data");
delay(300);
lcd.clear();
delay(300);
if(Serial.available()>0)
        {
indata=Serial.read();
if(indata=='1')
```

```
        {
lcd.clear();
Serial.println("Section 1");
lcd.print("Section 1");
delay(1000);
        section_1();
tone(5,HIGH);
digitalWrite(0,HIGH); //Relay is Triggered(ON)
delay(2000);
digitalWrite(0,LOW); //Relay is OFF
delay(2000);
noTone(LOW);
Serial.flush();
        }
else if(indata=='2')
        {
lcd.clear();
lcd.print("Section 2");
Serial.println("SXSSSSSSSSSSSSSSSSSSWASection 2");
delay(1000);
        section_2();
tone(5,HIGH);
digitalWrite(0,HIGH); //Relay is Triggered(ON)
delay(2000);
digitalWrite(0,LOW); //Relay is OFF
delay(2000);
noTone(LOW);
Serial.flush();
        }
else if(indata=='3')
        {
lcd.clear();
lcd.print("Section 3");
Serial.println("Section 3");
delay(1000);
        section_3();
tone(5,HIGH);
digitalWrite(0,HIGH); //Relay is Triggered(ON)
delay(2000);
digitalWrite(0,LOW); //Relay is OFF
delay(2000);
noTone(LOW);
Serial.flush();
        }
else if(indata=='4')
        {
lcd.clear();
lcd.print("Section 4");
Serial.println("Section 4");
delay(1000);
        section_4();
tone(5,HIGH);
digitalWrite(0,HIGH); //Relay is Triggered(ON)
delay(2000);
digitalWrite(0,LOW); //Relay is OFF
delay(2000);
noTone(LOW);
Serial.flush();
       }
else
      {

      }
      }
}
```

Result:-


saim_motor | Arduino 1.6.11
File Edit Sketch Tools Help
saim_motor
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Copy error messages
144
Arduino/Genuino Uno on COM1

```
  section_4();
  tone(5,HIGH);
  digitalWrite(0,HIGH); //Relay is Triggered(ON)
  delay(2000);
  digitalWrite(0,LOW); //Relay is OFF
  delay(2000);
  noTone(LOW);
  Serial.flush();
  }
 else
 {

 }
 }
}
```

```
Board at COM1 is not available
Board at COM1 is not available
```

After running this code at the end there is an hex file is generate using this hex code we can see that our code is running well by putting the location of the hex code into over proteus interface which we have make in proteus software.

Chapter 6

Conclusion

The world is moving more towards technology dependent era. Every day we keep hearing owes of farmers that even after using costly fertilizers the leaves were eaten away by various diseases.The machine learning methods bring this aspect to reality, by observing the database and helping the botanists in the diagnosis of diseases where a lot of precision is required. And one of the machine learning technique, SVM is used in this project for classification of leaf diseases. The accuracy results in an available range from mid-90 to top 90 percent. This can be bettered by increasing the database. However, the results obtained from real life images are very encouraging.The accurate detection and classification of the plant disease is very important for the successful cultivation of the crops, this can be done using digital image processing. This project reduces the time required for the detection of disease as well as the cost.The plant disease can be identified at an early stage.

References:

[1] plant disease detection using image processing Sachin.D.Khirade, A.B Patil, *2015 International conference on computing communication control and Automation*,978-4799-6892/15 31.100(c)2015 IEEE

[2] "Artificial Neural Network Based",Shivkumar Bagde, Swaranjali Patil, Snehal Patil, Poonam Patil, *International Journal of Computer Science and Mobile Computing*, IJCSMC, Vol. 4, Issue. 4, April 2015, pg.900 905

[3] "Plant Disease Deadimani, Classification of leaf disease based on Multiclass SVM classifier",Pooja Kulinavar, Vidya I. H *2017, www.IJARIIT.com tection,IJCSMC*, Vol. 4, Issue. 4, April 2015, pg.900 905

[4] "Image segmentation using K-means",elise.arnaud*UJF UFRIMA*,elise.arnaud@imag.fr

[5] D. Al Bashish, M. Braik, and S. Bani-Ahmad, A framework for detection and classification of plant leaf and stem diseases, in Signal and Image Processing (ICSIP), 2010 International Conference on, pp. 113118, IEEE, 2010.

[6] J. K. Patil and R. Kumar, Advances in image processing for detection of plant diseases, Journal of Advanced Bioinformatics Applications and Research, vol. 2, no. 2, pp. 135141, 2011.

[7] P. Revathi and M. Hemalatha, Classification of cotton leaf spot diseases using image processing edge detection techniques, in Emerging Trends in Science, Engineering and Technology (INCOSET), 2012 International Conference on, pp. 169173, IEEE, 2012.

[8] H. Zulkifli, A. Yeon, M. Shakaff, A. A. Abdul Halis, and S. Rohani, Feasibility study on plant chili disease detection using image processing techniques, 2012.