IMAGE PROCESSING PROJECT

Fall Semester 20-21

Submitted to:
JASMIN T. JOSE

Team members:
19BCE0488- Mansi Raturi
19BCE0588- Adithi K
19BCE2229- Joshita A

Classification of leaf disease based on Multiclass SVM classifier

Abstract: ***Plant disease detection is emerging field in India as agriculture is important sector in Economy and Social life. Earlier unscientific methods were in existence. Gradually with technical and scientific advancement, more reliable methods through lowest turnaround time are developed and proposed for early detection of plant disease. Such techniques are widely used and proved beneficial to farmers as detection of plant disease is possible with minimal time span and corrective actions are carried out at appropriate time. In this paper, we studied and evaluated existing techniques for detection of plant diseases to get clear outlook about the techniques and methodologies followed. The detection of plant disease is significantly based on type of family plants and same is carried out in two phases as segmentation and classification. Here, we have discussed existing segmentation method along with classifiers for detection of diseases in Monocot and Dicot family plant.***

Keywords: Classifier, Dicot Plant Disease, Feature Extraction, Monocot Plant Disease, Pre-Processing, Segmentation.

1. Introduction

Plant disease is one of the important factor which causes significant reduction in the quality and quantity of plant production. Detection and classification of plant diseases are important task to increase plant productivity and economic growth. Detection and classification are one of the interesting topics and much more discussed in engineering and IT fields.

There are various techniques emerged to detect the plant disease such as thresholding, region growing, clustering, watershed, etc. To detect plant disease the image should go through pre-processing, segmentation, feature extraction and classification processes. The pre-processing is an improvement process of image data to suppresses unwanted distortion or enhances some image features important for further processing [1].

The segmentation process is to partition an image into meaningful regions and it is vital process through which image features are extracted. There are various features of an image such as grey level, color, texture, shape, depth, motion, etc. Classification process is used to classify the given input data into number of classes and groups. It classifies the data based upon selected features [2].

Section 2 presents basic types of plant families. Section 3 introduces existing methods for detection of plant disease. Section 4 presents various techniques of segmentation and feature extraction. Section 5 presents classification.

LITERATURE SURVEY:

2. Basic Types of Plant Families

A. Monocot Family Plant:

Disease identification depends on the type of plant family. There are basically two types of plant Monocot family plant and Dicot family plant [1]. The Monocot family plant has different characteristics such as one seed leaf, leaf veins are straight and parallel, absence of wood. Examples of Monocot family plants are wheat, ginger, corn, rice, millet, lilies, banana, palm, sugarcane, onions, banana tree, bamboo, and grass, turmeric etc.

Turmeric plant has heterogeneous uses in most of the fields. It plays an important role in Indian lifestyle as it has tremendous use in different medicines along with daily food items. The turmeric plant diseases are discussed in detail in the following section.

a) Leaf Blotch:

It has small oval and rectangular or irregular brown spots appear on leaves and it will become dirty brown as shown in Figure 1. The disease is controlled by use of Mancozeb pesticides [3].

Figure 1: Leaf Blotch

b) Leaf Spot:

It causes greyish or whitish spots with brown boundary of different sizes which appear on the upper surface of leaves and the spots are greyish or whitish dark in the center. Due to leaf spot, leaves will get dry and died as shown in Figure 2. The disease is controlled by use of Zineb or Bordeaux pesticides [3].

Figure 2: Leaf Spot

B. Dicot Family Plant

Dicot family plant has characteristics such as two seed leaf, nested leaf veins and complex structured, woody as well as woodless. Examples of Dicot family plants are cotton, potatoes, tomatoes, beans, honeysuckle, roses, peppers, strawberry, coffee, etc. Cotton is preferred to make textile products and yarn products in India. Various precautions and pesticides are available to control the cotton Diseases [2] [4].The cotton plant diseases are discussed in detail in the following section.

a) Bacterial Blight

It is most dangerous disease in cotton plant which infects all the parts of plant leaf as shown in Figure 3. Because of bacterial disease 10% to 30% are losses in cotton production. This type of disease affects during the growth of cotton plant. And it causes seedling blight, black arm, boll rot and leaf spot. This spots turns into brown spots on plant leaf. Bacterial Blight can be controlled and various pesticides are available such as Pseudomonas fluorescens and Bacillus subtilis [4].

Figure 3: Bacterial Blight b) Fusarium Wilt

This is fungal disease as shown in Figure 4. It infects the plant at any growing stage. Fusarium disease causes the drooping of the older lower leaves, yellowing of the lower leaves, followed by stunting of the plant and death of the plant.

Figure 4: Fusarium Wilt c) Target Spot

This disease produces tan to brown color spot that have concentric rings like a bull's-eye as shown in Figure 5 Infected plant may look healthy from the top, so it is important to check lower leaves, where the first spot usually appears. It will start with only a few spots but after that, the disease will progress with more infection, and it does not take so much time spread on plant [4].

Fig

ure 5: Target Spot d) Leaf Curl

It is caused by fungal or virus and it can easily noticeable as shown in Figure 6. Because of leaf curl disease the growth of plant leaf will stop. It is an incurable disease [4].

F

igure 6: Leaf Curl e) Grey Mildew:

This disease found in middle aged or older plant and it looks like pale spot, irregular angular spots on leaf as shown in Figure 7. Usually this spots are 4-5 mm in diameter on plant leaf [4].

Figure 7: Grey Mildew

3. Existing Methods For Detection of Plant Diseases

Earlier papers are describing to detection of the plant leaves diseases using various approaches are discussed below,

In [1], they discussed about automatic detection and classification of diseases. Plant disease spots are different in color but not in intensity. Thus color transform of RGB image is used for better segmentation of disease spots. Median filter is used for image smoothing and Otsu method is used to calculate threshold values to detect the disease spot. It doesn't give accurate result for Dicot family plant.

P. Revathi and M. Hemalatha [4] investigated advance computing technology to assists the farmer in plant development process. This approach used mobile to capture infected cotton leaf images. RGB color feature segmentation is carried out to get disease spots. Edge detection technique is used for extraction of image features of spots to detect diseases. Neural network is used to categorize the diseases. The segmentation process is not suitable for Monocot family plant.

S. Dubey and R. Jalal [5] explored the concept of detection and classification of apple fruit diseases. The proposed approach is composed of three steps such as segmentation, feature extraction and classification. K-means clustering technique is used for the image segmentation. The features are extracted from the segmented image and images are classified based on a Multiclass Support Vector Machine (SVM). The proposed approach is specific to apple fruit diseases and cannot be extended to other fruit diseases.

In [6], the approach focused on Cercospora leaf spot detection in sugar beet using hybrid algorithms of template matching and support vector machine. The approach adopts three stages; first, a plant segmentation index of G-R is introduced to distinguish leaf parts from soil contained background for automatic selection of initial sub-templates. Second is robust template matching method adopted for continuous observation of disease development, foliar translation and dynamic object searching. Then, Support Vector Machine (SVM) is used to disease classification by a color features named two dimensional, xy color histogram. The segmentation process is not suitable for other Dicot family plant.

Yan, Han and Ming [7] proposed to select features of cotton disease leaf image by introducing fuzzy selection approach, fuzzy curves and fuzzy surfaces. The features which are extracted from fuzzy selection approach are used for diagnosing and identifying diseases. This approach removes the dependent features of image so as to reduce the number of features for classification.

Sannakki, Rajpurohit, Nargund and Kulkarni [8] proposed an approach to diagnose the disease using image processing and artificial intelligence techniques on images of grape plant leaf. The input image of grape leaf is complex at background. The thresholding is used to mask green pixels and image is processed to remove noise using anisotropic diffusion. Then, segmentation is done using K-means clustering technique. The diseased portion from segmented images is identified. The results were classified using back propagation neural network.

In [9], they investigated approach for automatic detection of chilies plant diseases. For that, CIELab color transformation model is used to extract color feature from infected image. Compare the color feature for detection of disease. There is no effective work done in feature extraction. But it could yield more result accuracy if appropriate work would have been done.

Next paper [10] discussed about the monitoring of grapes and apples plant diseases. It suggests a solution to farmers for healthy yield and productivity. K-means clustering is used for segmentation and artificial neural network is used for classification of features. Also back propagation concept is used for counting the weight of mango. Morphology, color and texture features are extracted for classification.

4. Techniques of Segmentation and Feature Extraction

The following section describes various segmentation and feature extraction techniques for detection of plant diseases.

A. Thresholding:

Thresholding technique is effective for images based on objects of contrast background. In thresholding technique, each pixel in images are separates into foreground (binary "1") and background (binary "0") object of classes based upon their correspondence in gray level intensity. The gray scale image is converted into binary image using Otsu method. The drawback of thresholding is not always straight forward and assigned pixels to a single class. It need not form coherent regions as the spatial locations of pixels and completely ignored [11].

B. Region Growing:

In this technique, if no edges are identified then a region or class is formed by examining the neighboring pixels. The process of region growing is repeated for each boundary pixel in the region under examination [12]. This technique started with considering a seed pixel value and comparing with local pixels values based on functions such as difference between neighboring pixels values. It grows in the direction where the intensity of pixel is low. The region growing technique is proceeding until no more pixels are added. The drawback of region growing method is that it consumes more power and time for computation [13].

C. Partition Clustering:

The segmentation based K-means technique is a partition clustering technique used to partition n number of observations into k clusters in which each observation belongs to the cluster with the nearest mean of the cluster. In this technique, k is the number of clusters in the segmented image and colors present in an image are used for the clustering. The main advantage of segmentation based Kmeans clustering technique is that it works on local information and global information of image. K- means clustering algorithm is easy to implement and fast, robust and flexible [12]. Particle Swarm Optimization (PSO) comes under evolutionary clustering algorithm. In segmentation process, PSO works on global information and local information of the image. But it does not work well with local information. Usually, PSO works with skew divergence method to extract features of the images. It is difficult to implement and can't work on the problems of noncoordinated system [14] [15].

D. Watershed:

In this technique, a topo-graphic relief can be seen by grey level image while its altitude in relief is considered from grey level of pixel. A water drop flows along a way to get local minimum on a topological relief. The watershed corresponds to relief that is a limited to the adjacent catchment basins of water drops. In segmentation, elevation information is interpreter from the length of

gradient. Markers and region merging are the important to conclude local minima of image gradient. Marker based watershed transformation is analyzed by the special marker positions which has determined automatically by morphological operator or user can explicitly define it. The drawback of watershed method is that it could not give accurate result at classification of diseases [12].

E. Edge Detection:

Edge Detection is a general technique that operates on an image and results in a line drawing of the image. The lines represent changes in values such as intersections of planes, cross sections of planes, textures, colors, and lines as well as differences in shading and textures. The main purpose of edge detection is to identify areas of an image where a large change in intensity occurs. In edge detection, canny edge detector is mostly used to detect wide range of edges of images. Canny detection technique has three steps i.e. first, it clears the noise in image and second it obtains gradients of intensity and direction. Last, it determines strong edges of image and also finds begin and end of edges by threshold value. This detection technique is better because, it reduces noise. Edge detection technique is used to extract the image features (color, shape, texture, etc) [13] [16] [17].

5. Classification

The feature extraction process extracts relevant information from the input image in order to reduce the amount of resources required to describe a set of data [7]. The input data transforms into the set of features is known as feature extraction. For feature extraction, feature detection and feature selection should be done. There are various types of features of images such as color, texture, shape, edge, depth, corners, blobs, ridges, morphology, etc. Color Coconcurrence Method (CMM) is a matrix defined over an image having a distribution of co-occurring values at a given offset [16]. Color Co-occurrence Matrix is a matrix in which features are taken into account to arrive at unique features which represent the image. CCM includes Gray Level Cooccurrence Matrices (GLCM), Gray Level Co-occurrence Histograms (GLCH) and Spatial Gray Dependence Matrix (SGDM).

Gray Level Co-occurrence Matrix is created from gray scale images and used to describe the shape feature [18]. The Gray Level Co-occurrence Matrix is based on the repeated occurrence of gray-level configuration in the texture. The spatial gray dependence matrix is used for texture analysis. A spatial gray dependence matrix is created based on hue, saturation and intensity [19]. Run Length Matrix (RLM) is another type of matrix. Same gray pixel values are the part of run and those gray values forms a two dimensional matrix. Example, RM- Q(x, y) x represents gray values and y represents run length [12].

Segmentation and feature extraction are techniques to extract relevant features from the plant leaf. To arrive at the decision and identification, the results from feature extraction are classified. On the basis of survey, we found various types of classifiers such as Artificial Neural Network (ANN) with 87% accuracy rate [19], Back Propagation Neural Network (BPN) with 87% accuracy rate [12][15], Fuzzy Logic with 88% accuracy rate[15] and Support Vector Machine (SVM) with 91% accuracy rate [2] [19].

Support Vector Machine performs classification by constructing an N-dimensional hyper plane that optimally separates the data into different parts. SVM models are closely related to neural networks. SVM evaluates more relevant information in a convenient way [19].

The proposed flow architecture from literature survey integrated with innovative idea is shown in Figure 8. The flow architecture have several steps i.e. first, the infected plant image goes through the image pre-processing phase for enhancing and removing noise from image. Second, the preprocessed image passed to the segmentation for partitioning into clusters. For segmentation, K-means clustering technique is fast, flexible and easy to implement than others. From segmented image extracts the features of image such as color feature, texture feature, shape feature. Color feature is used to simplify the object extraction and identification. For that various color models are used such as RGB, HSI, CMYK and CIELAB. The texture feature is a feature that classifies the segmented regions and also defines the characteristics of regions. Shape is a feature that interprets various facts of objects. Finally, classifier is used for classification and recognition of plant diseases. SVM is one of the best classifier which gives more accuracy than others.

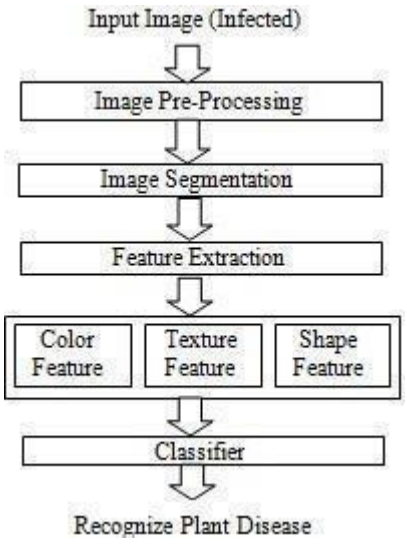


Figure 8: Flow of System Architecture

MERITS and DEMERITS :-

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.
In this study, the available images from the internet are also taken. The leaf images that are infected by Alternaria Alternata, Anthracnose, Bacterial Blight, Cercospora Leaf Spot and Healthy leaf are also included

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.

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.

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.

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

Future Scope :

SVM, though a binary classification technique, with a simple manipulation, can be used for a multiple class case. This provides more space not just to classify but to identify the diseases. Presently the system is semi-automatic. This can be completely automatic by choosing ROI based on criterion such as principal components analysis, or choosing the cluster with larger disease area etc. With the proper database, this method can be applied to more diseases. Example: liver diseases, skin cancer, breast cancer identification and classification etc.

6. Conclusion

The detection of plant disease is one of the important tasks. A plant disease reduces the production of agriculture. Every year the loss due to various diseases is challenging part in agriculture production. Although work is carried out till time on detection of diseases but proper segmentation of affected part based on type of plant family is still an open problem as a research area. The strategy used in existing system consists of an approach for either Monocot or Dicot family plants but not for both. Though most of the disease detection approaches are in existence but these

approaches can be developed further to provide more accuracy in Monocot and Dicot family plant disease detection. From the schemes discussed in the above section, K-means clustering method for segmentation is widely used by most of the researchers. For classification and feature extraction, GLCM along with SVM classifier were found to be better in performance in comparison to others. We have included a list of references sufficient to provide a more detailed understanding of the approaches described. We apologize to researchers whose important contributions may have been overlooked.

References

[1] Piyush Chaudhary, Anand K. Chaudhari, Dr. A. N. Cheeran and Sharda Godara, "Color Transform Based Approch For Disease Spot Detection on Plant Leaf", International Journal of Computer Science and Telecommunications, 2012, pp. 65- 70.

[2] Ms. Rupali S. Zambre, "Classification of Cotton Leaf Spot Disease Using Support Vector Machine", International Journal of Engineering Research and Applications, 2014, pp. 92-97.

[3] Kandiannan, Thankamani, Srinivasan, Rajeev, "Turmeric Plant", Indian Institute of Spices Research, Indian Council of Agricultural Research, Kerala, 2008, pp. 3-7.

[4] P. Revathi and M. Hemalatha, "Advance Computing Enrichment Evaluation of Cotton Leaf Spot Disease Detection Using Image Edge detection", IEEE, 2012.

[5] Shiv Ram Dubey, Anand Singh Jalal, "Detection and Classification of Apple Fruit Diseases using Complete
Local Binary Patterns", International Conference on Computer and Communication Technology IEEE, 2012, pp. 346-351. [6] Rong Zhou, Shunichi Kaneko, Fumio Tanaka, Miyuki
Kayamori, Motoshige Shimizu, "Early Detection and Continuous Quantization of Plant Disease Using
Template Matching and Support Vector Machine Algorithms", International Symposium on Computing and Networking IEEE, 2013, pp. 300-304.

[7] Yan-Cheng Zhang, Han-Ping Mao, Bo Hu, Ming-Xi Li, "Features Selection Of Cotton Disease Leaves Image Based On Fuzzy Feature Selection Techniques", International Conference on Wavelet Analysis and Pattern Recognition IEEE, 2007, pp. 124-129.

[8] Sanjeev S. Sannakki, Vijay S Rajpurohit, V. B. Nargund and Pallavi Kulkarni, "Diagnosis and Classification of Grape Leaf
Diseases using Neural Networks",
International Conference on Computing, Communications and Networking Technologies IEEE, 2013.

[9] Zulkifli Bin Husin, Abdul Hallis Bin Abdul Aziz,
"Feasibility Study on Plant Chili Disease Detection
Using Image Processing Techniques", International Conference on Intelligent Systems Modelling and Simulation IEEE, 2012, pp. 291-296.

[10] Monika Jhuria, Ashwani Kumar, Rushikesh Borse, "Image Processing for Smart Farming: Detection of Disease and Fruit Grading", International Conference on Image Information Processing IEEE, 2013, pp. 521-526.

[11] Tirthraj Das and Tanistha Nayak, "A Java Based Tool for Basic Image Processing and Edge Detection", Journal of Global Research in Computer Science, 2012, pp. 57-60.

[12] Jagadeesh Devdas Pujari, Rajesh Yakkundimath and

Abdulmunaf Byadgi, “Grading and Classification of Anthracnose Fungal Disease of Fruits based on Statistical Texture Features”, International Journal of Advanced Science and Technology, 2013, pp. 121-129.

[13] Huiqi Li and Opas Chutatape, “Automated Feature Extraction in Color Retinal Images by Model Based Approach”, IEEE Transactions on Biomedical Engineering, 2004, pp. 246-254.

[14] P. Revathi and M. Hemalatha, “Identification of Cotton Diseases Based on Cross Information Gain Deep Forward Neural Network Classifier with PSO Feature Selection”, International Journal of Engineering and Technology, 2014, pp. 4673-4642.

[15] P. Revathi, M. Hemalatha, “Cotton Leaf Spot Diseases Detection Utilizing Feature Selection with Skew Divergence Method”, International Journal of Scientific Engineering and Technology, 2014, pp. 22-30.

[16] Smita Naikwadi, Niket Amoda, “Advances in Image Processing for Detection of Plant Diseases”, International Journal of Application or innovation in Engineering and Management, 2013, pp. 168-175.

[17] Arati J. Vyavahare, “Canny Based DRLSE Algorithm for Segmentation”, International Journal of Computer Applications, International Journal of Computer Applications, 2014, pp. 1-5.

[18] Mr. Hrishikesh P. Kanjalkar, Prof. S Lokhande, “Feature Extraction of Leaf Disease”, International Journal of Advanced Research in Computer Engineering and Technology, 2014, pp. 153-155.

[19] Rajesh Yakkundimath, Jagdeesh D. Pujari and Abdulmunaf S. Byadgi, “Classification of Fungal Disease Symptoms affected on Cereals Using Color Texture Features”, International Journal of Signal Processing, Image Processing and Pattern Recognition, 2013, pp. 321- 330.

[20] Wenjiang Huang, Qingsong Guan, Juhua Luo, Jingcheng Zhang, Jinling Zhao, Dong Liang, Linsheng Huang, and Dongyan Zhang, “New Optimized Spectral Indices for Identifying and Monitoring Winter Wheat Diseases”, Journal Of Selected Topics in Applied Earth Observations And Remote Sensing IEEE, 2014, pp. 2516-2524.

MULTI SVM:-

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

```
itrind=size(test,1);
itrfin=[];
Cb=C;
Tb=T;
for tempind=1:itrind
   tst=test(tempind,:);
   C=Cb;
   T=Tb;
   u=unique(C);
   N=length(u);
   c4=[];
   c3=[];
   j=1;
   k=1;
   if(N>2)
      itr=1;
      classes=0;
      cond=max(C)-min(C);
      while((classes~=1)&&(itr<=length(u))&& size(C,2)>1 && cond>0)
      %This while loop is the multiclass SVM Trick
         c1=(C==u(itr));
         newClass=c1;
         %svmStruct = svmtrain(T,newClass,'kernel_function','rbf'); % I am using rbf kernel
function, you must change it also
         svmStruct = svmtrain(T,newClass);
         classes = svmclassify(svmStruct,tst);

         % This is the loop for Reduction of Training Set
         for i=1:size(newClass,2)
            if newClass(1,i)==0;
               c3(k,:)=T(i,:);
               k=k+1;
            end
         end
      T=c3;
      c3=[];
      k=1;

         % This is the loop for reduction of group
         for i=1:size(newClass,2)
            if newClass(1,i)==0;
               c4(1,j)=C(1,i);
               j=j+1;
            end
         end
      C=c4;
      c4=[];
```

```
      j=1;

      cond=max(C)-min(C); % Condition for avoiding group
                  %to contain similar type of values
                  %and the reduce them to process

        % 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);              % This logic is used to allow classification
val=Cb(valt==1);              % of multiple rows testing matrix
val=unique(val);
itrfin(tempind,:)=val;
end

end

% Give more suggestions for improving the program.
```

RGB TO HSI :-

```
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].

% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins
% Digital Image Processing Using MATLAB, Prentice-Hall, 2004
% $Revision: 1.5 $ $Date: 2018/10/18 13:44:59 $
```

```
% Extract the individual component images.
rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);

% 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));

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;

H(S == 0) = 0;

I = (r + g + b)/3;

% Combine all three results into an hsi image.
hsi = cat(3, H, S, I);
```

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];
```

DETECT DISEASE:-

```
% Project Title: Pomegranate Leaf Disease Detection

clc
close all
clear all

[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');

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

%% 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(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');
set(gcf, 'Position', get(0,'Screensize'));

% Feature Extraction
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))

% 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];
%%
% Load All The Features
load('Training_Data.mat')

% Put the test features into variable 'test'
test = feat_disease;
result = multisvm(Train_Feat,Train_Label,test);
%disp(result);

% Visualize Results
if result == 0
   helpdlg(' Alternaria Alternata ');
   disp(' Alternaria Alternata ');
elseif result == 1
   helpdlg(' Anthracnose ');
   disp('Anthracnose');
elseif result == 2
   helpdlg(' Bacterial Blight ');
   disp(' Bacterial Blight ');
elseif result == 3
   helpdlg(' Cercospora Leaf Spot ');
   disp('Cercospora Leaf Spot');
elseif result == 4
   helpdlg(' Healthy Leaf ');
   disp('Healthy Leaf ');
end

%% Evaluate 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 Linear Kernel with 500 iterations is: %g%%',Max_Accuracy)
```

OR DETECT DISEASE:-

```
% 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 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','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 ');
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 astext
%                      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 astext
%                      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 astext
%        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 astext
%        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 astext
%        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

ALTERNARIA ALTERNATA:-


OGA1440025


5358422


5368947


© Copyright APS Press


UGA1436046

ANTRACNOSE:-


© Joe Boggs
Walnut Anthracnose Spots on Leaflets


UGA5076064

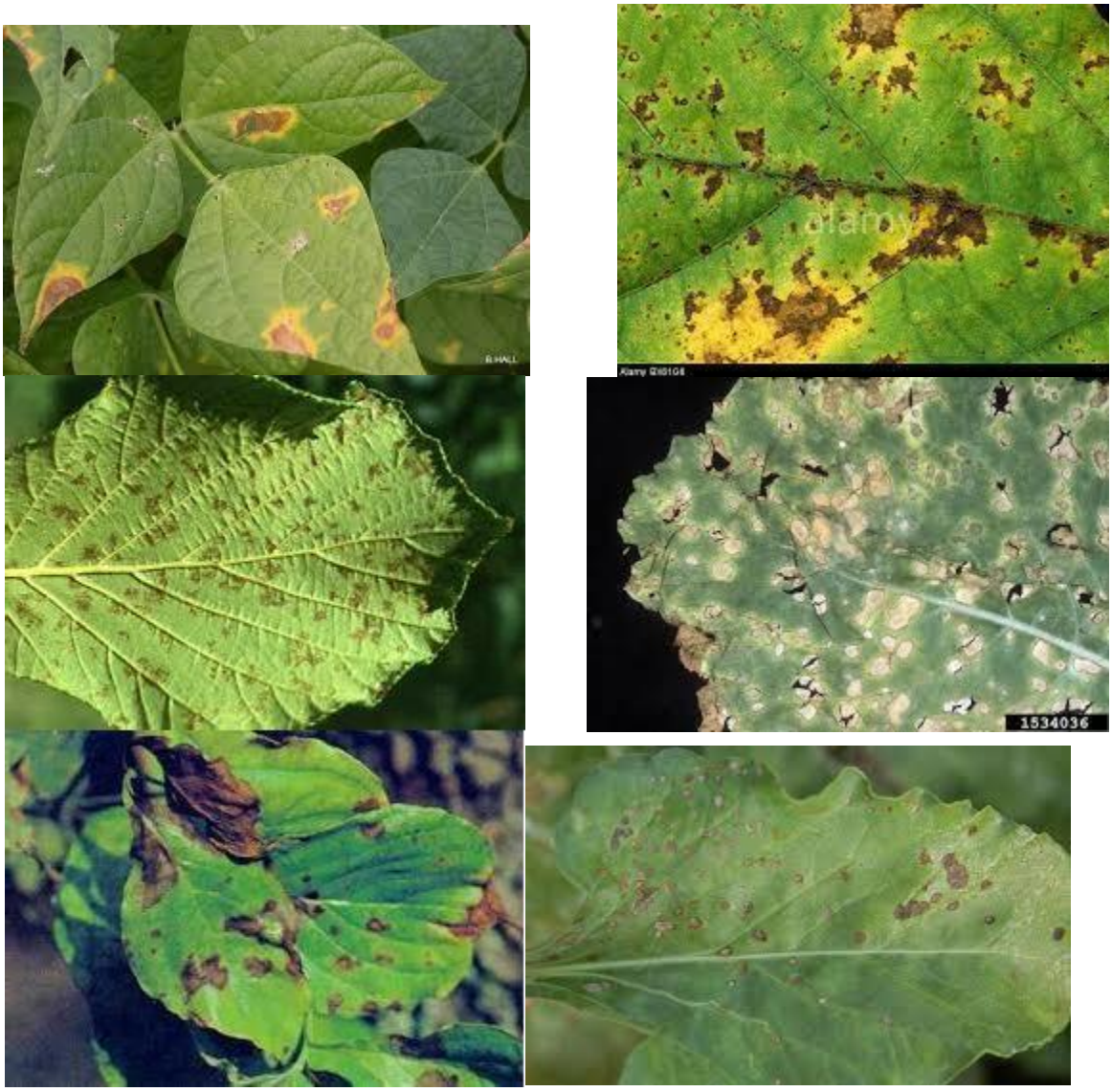

5405253


BF2061 (RM) (c) www.visualphotos.com


UGA1440024


S. Vann

BACTERIAL BLIGHT:-



CERCOSPORA LEAF SPOT:-

HEALTHY LEAVES:-