University of Nottingham

Final year project

Leaf Disease Identification

Author:
Yao Zhang

Supervisor:
Professor Tony Pridmore

November 5, 2014

Abstract

Plant diseases causes many significant damages and looses in crops around the world. Appropriate measures on disease identification should be introduced to prevent the problems and minimise the losses. Technical approaches using machine learning and computer vision are actively researched to achieve intelligence farming by early detection on plant disease. An mobile application is obviously desirable to aid the farmers or garden enthusiasts in diagnosing what sorts of diseases a plant has. Although some similar applications exist, most of them achieve the function by submitting the image to a team of plant pathologists or expert garden advisors to get possible identification results and some advise. This dissertation presents the research, design and implementation of an mobile application which can automatically identify the plants diseases based on its leaf appearance with some computer vision and machine learning technique. Many experiments and evaluations on different segmentations, feature extractions and classification methods were done to find the most effective approach. The target group of the user is those who request a free and quick diagnosis on common leaf disease at any time of the day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

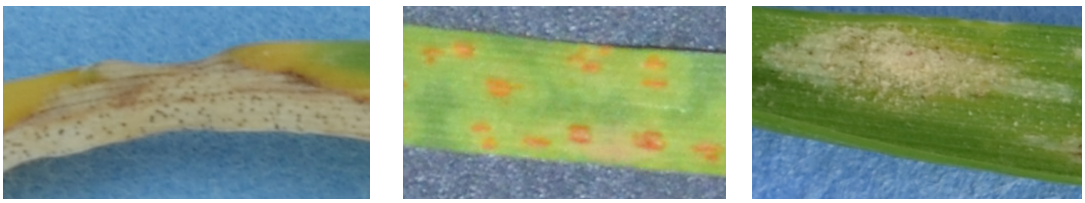
1.1 Background and Motivation

To increase the food quality and productivity, research in agriculture increasingly gained attention recently. Accurate diagnosing crops disease is one of the most difficult task to perform as it is influenced by many parameters such as nutrition, environment, climate and so on. With the development of the science and technology, more and more scientific tools based on machine vision for achieving intelligent farming becomes active computing research areas. Besides crops production, garden enthusiasts often confuse about the symptom of their plants. Student taking plant pathology are often asked by their friends about what is wrong with their plants and how to get rid of the problem. Although curing the plant is too late at that time, it is still very useful to make other plants prevent the same disease in the future. A convenient tool is really desirable to help the plants enthusiasts diagnose the disease as quickly as possible when the problem just appear. A smartphone app is an apparently convenient approach for an untrained person to learn about what kinds of disease a plant suffer. Although there have been some similar applications such as 'Garden Compass Plant / Disease Identifier' (iTunes Store) for iPhone or 'A&L Plant Disease Diagnosis'(Google Play) for Android, most of them need to take a photo and submit it to a team of plant experts who will identify the symptom. However the user have to take lots of time to wait for the result and pay some money for the application. If there is a wonderful and completed mobile application based on computer vision and machine learning technique and analyse the picture by the application itself to get the diagnose results, it will be much quicker and cheaper so that the user can freely use the application at any time of the day with a quick result. There is an excellent and mature iPhone app called Leadsnap which uses visual recognition to help user identify plant species from photographs of their leaves (Kumar *et al.* , 2012). Leadsnap was developed by researchers from Columbia University, the University of Maryland, and the Smithsonian Institution. This application is slightly resembles to the project's aim, but there is still not a complete app in identifying the leafs disease which is much more difficult than species identification. There is also a commercial and expensive software called Assess which can easily and quickly quantify the disease symptom on plants, but it can not identify the category of disease exactly. Moreover Access is hardware-independent, requiring a computer system, which is inconvenient for the user to use it. Thus it can be seen that a implementation completely successful application for leaf disease detec-

tion is extremely useful for the garden enthusiasts as well as agricultural production.

1.2 Description of the work

This study describes a mobile computer vision system to classify an plant image by the disease spot category that it contains from coloured image analysis. Three kinds of common leaf diseases Powdery Mildew, Septoria and Wheat Rust were used to be analysed as an sample shown in Figure 1.1. The plant disease images were provided by my supervisor Tony Pridmore. The provided images has different complex background and light intensity. At first the first challenge is how to identify and segment the leaf from the different complex backgrounds, some approaches were tried and tested. And then the disease area need to be partitioned out from the segmented leaf. At last, Expected Maximisation (EM) segmentation were used to segment the leaf from the background and Otsu is for the disease spot segmentation from the leaves. After segmentation, what features should be the most appropriate to be extracted as discriminators when the disease does not follow a well defined shape or colour domain pattern. Additionally the light intensity is another tricky problem to choose the most proper features for differentiating these three diseases. Hue Colour Intensity histogram and texture features were extracted to discriminate the disease spot in my application. What classifier is the most appropriate for the disease spot identification is another important study in this project. Support Vector Machine (SVM), Artificial Neuro Newark (ANN) and Random Forest (RF) were implemented and evaluated to compare the pros and cons of them and which one is the best one for this situation. The features on the whole leaf or only disease spot point are also compared and evaluated to decide which one can be the best discriminator. At last, as planned in project proposal, an android application was implemented to send leaf image to the server, and the server will recognise the disease category. An additional computer based program was also implemented to allow the user use computer in window to do the same function.



(a) Powdery Mildew (b) Septoria (c) Wheat Rust

Figure 1.1: The tree kinds of leaf disease included in this study

1.3 Organisation of the Report

The dissertation consists of six chapters representing the work of the project. The first part of the report contains the background information and motivation to do the project. It is necessary study on whether a the project is worthy to be done.

It also concisely describes the overall work included in the project. Related work on the previous research related to the project and the basic knowledge required to read the dissertation are discussed in the second part. The reports reader should be the generous computer science student, however the content of the report includes too much specialised knowledge in machine vision and statistics. Therefore it is necessary and helpful to talk about some basic knowledge about my project I focused on. The third part illustrates how this system was designed by some design proposals and the firth part mainly discusses the implementation of the project including various algorithm designs and the implementation process. The fifth part mainly concerns the evaluation of the project. It not only talks about the generalization of the segmentation method for other categories of leaves, but also includes the comparison of these different implemented pattern regonition method with 10-fold cross validation. The final part summarises what I have done and what I will done in the future if there are some future works. Personal reflection is a also included in this part.

Chapter 2

Related Work and Background Knowledge Research

2.1 Literature Review

Medical image processing for diagnosing disease becomes popular these years, because it can not only give a rapid and reliable result to access patients condition, but also reduce the cost significantly in comparison to the traditional diagnosis (Dhawan, 1990). In the past decades, image processing and pattern recognition for plant disease detection has also been an active research area. The overwhelming majority of the technique is similar. First some images are taken from the environment of study by camera. And useful features are extracted from the meaningful part segmented from the original image by some image processing technique. Afterward various classifiers should be applied as a determinant to classify the images into different categories. The segmentation and feature extraction technique of image processing technique and classification learning method are extremely vital to get a correct result. Shapes, colour properties and texture based classifier are commonly used approach in the past researches.

Texture is a kind of very useful feature to discriminate the species of disease. The fist time to use colour texture as the feature to analyse was reported by Shearer *et al.* (1990) developed an algorithm to recognise cucumber anthracnose using infrared reflection features and optical filter features. However the recognition accuracy was not high enough because the colour texture information was not fully utilised. Pydipati *et al.* (2006) reported a study to identify citrus disease using a texture based hue, saturation and intensity (HSI) feature combined with statistical algorithms, which achieves over 95% classification accuracy rate. Another method successfully recognising rice brown spot and rice blast via colour texture analysis, was developed by Sanyal & Patel (2008).

Segmentation is another important task which can affect the whole leaf disease identification. However the usual segmentation method is too slow to implement an interactive mobile application. Mobile systems aiding to identify the plant species based on leaf photograph recognition were proposed by Agarwal *et al.* (2006); Belhumeur *et al.* (2008). The systems use shape of the leaf as the discriminator to classify the species and the photograph requires to be taken in a white background.

A white background is a somewhat controlled condition, while the leaf pictures in this project are in an complex environment with many coins, branches and different colour background. Leaf segmentation in the unconstrained background is much more difficult. Some automatic segmentation on leaf images have been developed by some researcher, however none of them is completely successful with limited result. A k-means segmentation clustering the pixels into leaf and background was developed by Casanova *et al.* (2012). Yanikoglu *et al.* (2012) also experimented an approach by assuming the leaf in the centre of the image, so that the leaf colour can be attained by the most clustered pixels in the centre. With the leaf colour known, it is easy to segment the leaf image by a simple watershed method according to the distance to the sample leaf colour. However this method is not applied to many leafs which randomly distributed in the image. Neto *et al.* (2006) also develop a method to segment the leaf in unconstrained condition. At the begin of their method, candidate pixels are selected out based on the colour index and make a fragmentation on the candidate pixels. Final decision was made by sieving the convex shape on the fragments. This approach works but cannot segment all the leafs out sometimes.

2.2 Expectation Maximazation Algorithm in the Mixture Model of Gaussians

The segmentation part of the project implemented the algorithm of the Expectation Maximisation (EM) in Gaussian Mixture Model. The EM algorithm is an iterative algorithm for fitting the probability distribution models via maximum likelihood (ML) estimation on the incomplete data which are the observations. In this section, we will focus on a brief description of the EM algorithm. This section is my understanding of the EM algorithm with the reference book of Prince & Prince (2012).

2.2.1 Maximum Likelihood Estimation

Suppose we are given an observation as a data set $\{x_i\}_i^I$, this process can be viewed as a learning process which learns the parameters $\boldsymbol{\theta}$ of the probability model. The likelihood of the $\boldsymbol{\theta}$ given $\{x_i\}_i^I$ names $\ell(\boldsymbol{\theta}|x_1..x_I)$, which equals the probability of $\{x_i\}_i^I$ given $\boldsymbol{\theta}$.

$$\ell(\theta|x_1..x_I) = Pr(x_1..x_I|\theta) \tag{2.1}$$

To calculate the likelihood $\ell(\boldsymbol{\theta}|x_i)$ at a single point x_i is to evaluate the probability density function (PDF) $Pr(x|\boldsymbol{\theta})$ at x_i. Suppose each data point x_i independently drawn from the probability distribution model, the likelihood of a data set $\{x_i\}_i^I$ equals the product of the likelihood of the each point x_i in the data set.

$$\begin{aligned}\ell(\boldsymbol{\theta}|x_1..x_I) &= Pr(x_1..x_I|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{I} Pr(x_i|\boldsymbol{\theta})\end{aligned} \tag{2.2}$$

As the maximum likelihood's name suggests, the aim of the method is to find a set of parameters $\hat{\boldsymbol{\theta}}$ which makes the imcomplete observation data set $\{x_i\}_i^I$ are most likely. Thus the ML can estimate the $\hat{\boldsymbol{\theta}}$ like that:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \left[\ell(\boldsymbol{\theta}|x_1..x_I)\right] \\ &= \arg\max_{\boldsymbol{\theta}} \left[\prod_{i=1}^{I} Pr(x_i|\boldsymbol{\theta})\right]\end{aligned} \tag{2.3}$$

where $\arg\max_{\boldsymbol{\theta}}[f(\boldsymbol{\theta})]$ returns a $\boldsymbol{\theta}$ which maximizes $f(\boldsymbol{\theta})$.
In principle, the maximization can be attained by calculating the derivative of Equation 2.2 with respect to $\boldsymbol{\theta}$, making the result equals zero and then solving the function. However if the probability distribution are complex such as normal distribution, we will find it is difficult to solve the messy resulting equation. To simplify this, we shall take the logarithm of the expression instead of solving the derivative directly. Since the logarithm is a monotone increasing function, it will not affect the position of the maximum after we do the logarithm transformation. Logarithm transformation can transform the equation from product of the likelihood of each data into a sum of them. Hence the maximum likelihood of the distribution parameters now can be calculated as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{I} \log\left[Pr(x_i|\boldsymbol{\theta})\right]\right] \tag{2.4}$$

2.2.2 Latent Variable

To model an complex density function it is helpful to introduce a concept called latent variable or hidden variable w, which can not be observed directly. For now, we only consider the descrete latent variable. A final density function $Pr(x)$ over random variable x, can be viewed as the marginalisation of a joint distribution $Pr(x, w)$ between x and w:

$$\begin{aligned}Pr(x) &= \sum_w Pr(x, w) \\ Pr(x|\boldsymbol{\theta}) &= \sum_w Pr(x, w|\boldsymbol{\theta})\end{aligned} \tag{2.5}$$

2.2.3 Mixture of Gaussian and EM algorithm

For this part we will discuss how to use EM algorithm to estimate the densities of the mixture of Gaussians. As the same as we discussed in the ML part, observation data set $\{x_i\}_i^I$ are given. The data in Mixture of Gaussians can be described as a weighted sum of N normal distributions.

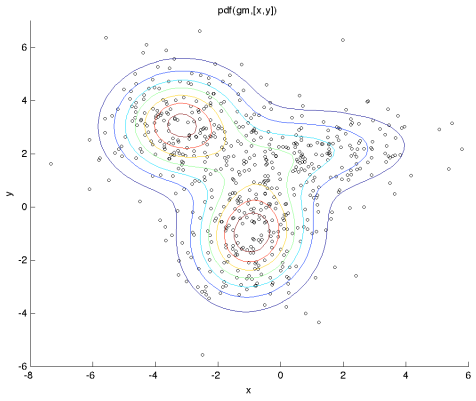
$$Pr(x|\boldsymbol{\theta}) = \sum_{j=1}^{J} \lambda_j Norm_x[\mu_j, \Sigma_j] \tag{2.6}$$

where $\mu_{1...J}$ and $\Sigma_{1..J}$ represents the means and covariances of these J normal distributions. $\lambda_{1..j}$ are positive weghits which sums to one. The complex density model combines some single Gaussian distributions together(Figure 2.1). To get the parameters $\boldsymbol{\theta} = \{\mu_j, \Sigma_j, \lambda_j\}$, we can simply use the maximum likelihood estimation, hence:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{I} \log\left[Pr(x_i|\boldsymbol{\theta})\right]\right] \\ &= \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{I} \log\left[\sum_{j=1}^{J} \lambda_j Norm_x[\mu_j, \Sigma_j]\right]\right]\end{aligned} \tag{2.7}$$

However when we try to solve the resulting equation, we will find that it is impossible to solve that in a closed form. Because the summation part inside the logarithm makes the solving process very difficult. Instead of solving that directly, we could use latent variable w_i to label each point. $w \in \{1...J\}$ means we can use w to

Figure 2.1: A 2D mixture of Gaussian model consist of three weighted mix three constituent with different means and covariance. Every point in this model has three weights to be classified to the corresponding normal distribution. The sum of the weights equals one.



'lable' a point belongs to which Gaussian distribution. So the distribution of w is a categorical distribution over $\{\lambda_j\}_1^J$, say $Pr(\boldsymbol{w}|\boldsymbol{\theta}) = Cat_w[\boldsymbol{\lambda}]$ Hence we can get that:$Pr(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{\theta}) = Norm_x[\mu_j, \Sigma_j]$

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{I} \log\left[\sum_{w} Pr(x_i, w_i|\boldsymbol{\theta})\right]\right] \\ &= \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{I} \log\left[\sum_{w=1}^{J} Pr(x_i|w_i = j, \boldsymbol{\theta}) Pr(w_i = j|\boldsymbol{\theta})\right]\right]\end{aligned} \tag{2.8}$$

After taking the derivative of the likelihood, we will find it is easy to solve the equation if we know $\boldsymbol{w}$. However, $\boldsymbol{w}$ is unknown. We can use EM algorithm to solve this problem. First step we should estimate w for each point. Second step we can use the estimated w to calculate the parameter. Then iterate these two steps until the cost function convergence. We initialize the parameter randomly to then do the iteration between E and M steps

Repeat until convergence: {
(E-step) For each i, j, set
$$r_{ij} := Pr(w_i = j|x_i, \boldsymbol{\theta})$$

(M-step) Estimation the parameter based on Maximum likelihood:
Solving μ_j, Σ_j
}
In the E-step, we use the current parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\lambda}\}$ to calculate the posterior probability distribution $Pr(w_i = j|x_i, \boldsymbol{\theta})$ for each latent variable given the overvation data set $\{x_i\}_1^I$.

$$\begin{aligned} Pr(w_i = j|x_i, \boldsymbol{\theta}) &= \frac{Pr(x_i|w_i = j, \boldsymbol{\theta})Pr(w_i = j|\boldsymbol{\theta})}{Pr(x_i|\boldsymbol{\theta})} \\ &= \frac{Pr(x_i|w_i = j, \boldsymbol{\theta})Pr(w_i = j|\boldsymbol{\theta})}{\sum_{k=1}^{J} Pr(x_i|w_i = k, \boldsymbol{\theta})Pr(w_i = k, \boldsymbol{\theta})} \\ &= \frac{\lambda_j Norm_{x_i}[\mu_j, \Sigma_j]}{\sum_{k=1}^{J} \lambda_k Norm_{x_i}[\mu_k, \Sigma_k]} \end{aligned} \tag{2.9}$$

In the M-step, we use the estimated parameter to maximize the likelihood of the parameters, then we can get the result rules below to update $\boldsymbol{\theta}$:

$$\begin{aligned} \lambda_j &= \frac{\sum_{i=1}^{I} r_{ij}}{\sum_{k=1}^{J}\sum_{i=1}^{I} r_{ik}} \\ \mu_j &= \frac{\sum_{i=1}^{I} r_{ij} x_i}{\sum_{i=1}^{I} r_{ij}} \\ \Sigma_j &= \frac{\sum_{i=1}^{I} r_{ij}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{I} r_{ij}} \end{aligned} \tag{2.10}$$

2.3 Otsu algorithm

Otsu is a segmentation algorithm by thresholding through the grey level histograms. The thresholding method gets the low point between two histogram peak by assuming that the background and foreground pixels are randomly drawn from two different Gaussian distributions with different mean values and variances. If there are two groups of pixels, a threshold was tried through the intensity histogram to get a value which can maximise the variance between these two classes. It is easy to get total mean μ_t, background mean μ_b and foreground mean μ_f. we also need to get the proportion of two class pixels, w_0 and w_1 where $w_1 = 1 - w_0$. Then we can calculate the between class variance by the following formula below:

$$\sigma = w_0 * (\mu - \mu_0)^2 + w_1 * (\mu - \mu_1)^2; \tag{2.11}$$

Then calculated the between class variance for every threshold through the histogram and make the variance maxmimum. And the corresponding threshold is the final threshold which should be used to segment the image. The method described by Parker (2010).

2.4 Decision Trees

It is very intuitive and useful to classify a set of training data through a sequence of questions, where the question depends on the previous question answer. This kinds of sequence can be classified by a directed decision tree, which is very effective for the non-metric data. The trees root is conventionally located at the top, and branches links the root to the successive node. The successive node will be continues linked until a leaf or terminal has been reached. The leaf node corresponds to the category of the classification and each intermediate node has a property test (eg. is $feature_1 > 30$?) to split the data into different sub-nodes based on the one kind of difference of their features. The property test should split the data into exhaustive and mutually exclusive partitions. A task of a training process is to build such a directed decision tree with a set of labeled data and choose a property test for each node. The classification should start at the top root of tree and test the corresponding features at each node along the branch until arriving at a leaf. We also need to consider how many outcomes should be split into for each node, when the tree should stop splitting and how to declare a leaf category when the node is impure. If the tree becomes too large, whether should we prune the tree? Such a kind of tree is called Classification and Regression Trees (CART). For this project, we only consider the binary decision tree which means every node only have two branches, because of its simplicity and expressive ability.

Firstly stopping criteria will be talked first, which means we should decided to further split the tree by selecting a property test for the end nodes or stop the splitting and accept the result. In the ideal case, the samples in each node belong to the same category which are usually called pure, we can definitely stop. But how does it do in other cases? For the decision tree algorithm used in the project, the tree will stop when the depth is more than 50. Next lets move on how to select a property test to split the data. In order to save the search time and improve the efficiency, we should build a simple decision tree with as few node as possible. To this end, we can seek a property test to split the data into descendent with as pure as possible. However, purity is much more difficult to calculate than impurity. Hence unsteadily, we will maximise the impurity reduction for each node to seek the property test. There are many methods can represent impurity, for this project, we only consider the entropy impurity which is one of the most popular measure. The formula is shown below(Duda *et al.* , 2012).

$$i(N) = -\sum_j P(w_j) log_2 P(w_j); \tag{2.12}$$

where $P(w_j)$ is the proportion of the j^{th} category data in node N. If all the data belong to the same category, the entropy of impurity equals zero. The entropy is greatest when the amount of different categories of data in the same node are equal. In order to build a simple and compact tree, we should decrease the local entropy as much as possible, although there is no guarantee that it would lead to a global optimisation. We calculated impurity reduction called Gain for each node and then

maximise it. As we only consider binary decision tree, the formula of Gain are shown below(Duda *et al.* , 2012):

$$Gain = \Delta i(N) \tag{2.13}$$
$$= i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \tag{2.14}$$

where N_L and N_R represent the left and right descendant of the node N, and P_L represents the left proportion of the samples on the leaf decedent to the samples on node N.

2.5 Artificial Nueral Network

The Linear models of the classification has useful analytical and computational ability to train and classify a set data by combining some fixed basis functions. However the computational ability will be limited if the dimension of the data is too large. It is necessary to adapt the fixed function based on data. Artificial Neural Network(ANN) fix the number of basis functions, but adapt the parameters of the basis functions during the training, which is the most successful pattern recognition model with using this type of method. For this section, only a simple introduction on the Feed-forward Network is presented due to the limitation of word amount.

The logistic regression consists of a linear combination of fixed non-linear fixed functions with a set of fixed input data and parameters as formate below(Bishop *et al.* , 2006):

$$y(\mathbf{x},\mathbf{w}) = f\left(sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right) \tag{2.15}$$

where f is a non-linear activation function and ϕ is a nonlinear basis function whose parameters can be adjusted along with the adjustment of coefficient w_j. The Neural Network use the similar form of the logistic regression, where basis function can be viewed as non-linear function of a linear combination of the input data. The basis function usually called activation is shown below in (2.16) for the j^{th} neuro, which represent M linear combinations to the input from x_1 to x_D.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + {w_{j0}}^{(1)} \tag{2.16}$$

where j means it is the j^{th} cell and the subscript (1) means it is the first layer, so $w_{ji}^{(1)}$ means the coefficient between the i^{th} input and cell j in the first layer and $w_{j0}^{(1)}$ means a coefficient between the bias and cell j in the first layer and $w_{j0}^{(1)}$. The

network diagram of this sample is shown in Figure 2.2, which is intuitive. Each activation is transformed by a differentiable, non-linear function h as following(Bishop *et al.* , 2006):

$$z_j = h(a_j) \tag{2.17}$$

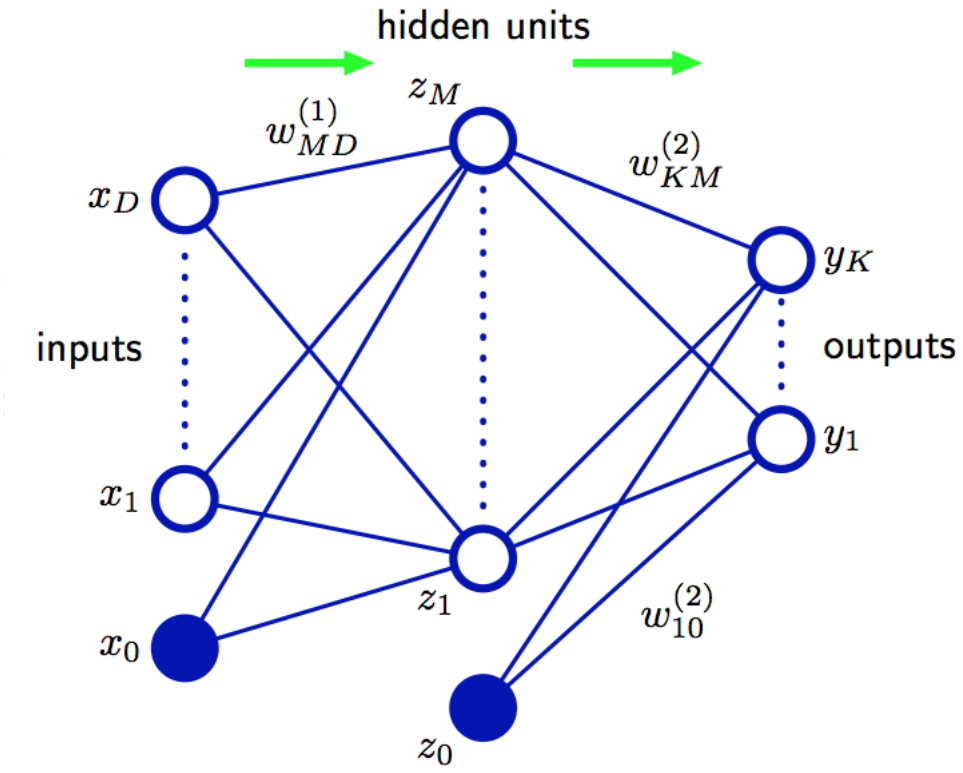


Figure 2.2: Artificial Neural Network with leaf features as inputs and disease category as outputs

The ouput of hidden layer cell z_j corresponds to an input of the output layer. The activation function h are usually the sigmoid or *tanh*. There are linear combinations of hidden units to the output units in the output layer. For output layer, the activation function is identity which means the $y_k = a_k$ while solving regression problem. If the problem is binary or multiple classification, the activation functions are two different kinds of function which denoted as σ, and the overall function shows below in

$$y_k(x, w) = \sigma\left(\sum_{j=0}^{M} wkj^{(2)} h\left(\sum_{i=0}^{D} w^1_{ji} x_i\right)\right) \tag{2.18}$$

Many gradient descendent of error function can be used for training even it is non-linear. Back-propagation can be used to calculate the derivative of the error function efficiently. Figure 2.3 shows the neural network of our project.

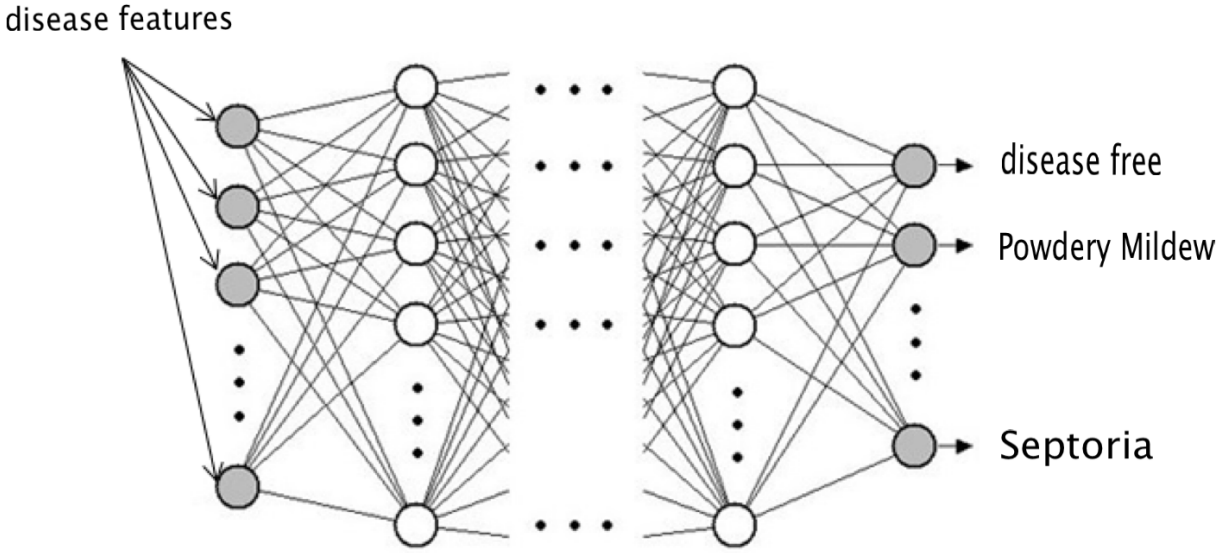


Figure 2.3: Artificial Neural Network with leaf features as inputs and disease category as outputs

2.6 Support Vector Machine

All the pattern recognition methods introduced above are parametric which means these method learn parameters in the training process and then discard the input after training. The classification will be purely based on the parameters given a set of new data. In this section, a memory based learning method called Support Vector Machine(SVM) will be introduced and used in the training of my project which use all the training data for each prediction with using kernel methods. Because of the complexity of the SVM with lots of mathematic formulas, it is have to use a big chapter to introduce it. Hence in this section just a simple description will be described. SVM train the machine by processing the data into a higher dimension and then separating the data categories with a hyperplane. With a non-linear mapping from original feature dimension to a sufficiently higher dimension, data from two categories can be always separated. A simple diagram in Figure 2.4 show a illustration of two dimension data, and the nearest training patterns marked as solid points are called support vectors. The optimal hyperplane separate the back data and red data into two region which represents two different categories. SVM use the maximum margin, which means the distance from the hyperplane to the support vectors are always maximised.

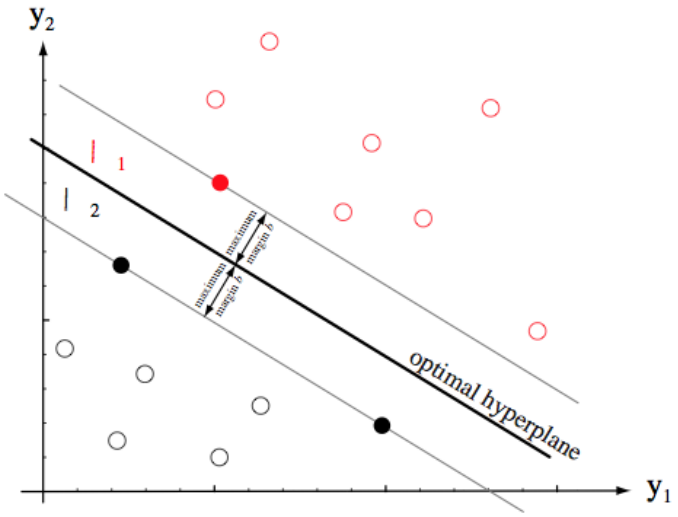


Figure 2.4: Artificial Neural Network with leaf features as inputs and disease category as outputs

2.7 Random Forest

Chapter 3

System Design

This chapter focus on the design of the project including system architecture design, graphical user interface (GUI) design, and main development component design. The system architecture will describe how the application works interacted with users. The GUI draft would be shown to introduce the initial design at the start of the project. The main development component design will describe the design of the development plan for each main component.

3.1 Architecture Design

In order to ease the implementation, verification and documentation, the system could be divided into three different components based on its functionalities. Figure 3.1 below shows the architecture of the program. The first part is the android application as a client which provide a user interface to interact with the functions of the application. The user can choose to take a picture or select a image form the gallery of the mobile phone as the leaf image to be diagnosed. Once the use decide which leaf picture he/she wants to use, the android application will send the image to the second part. The second part is the server implemented with java servlet which aims to communicate between client and main functions. The servlet will open a socket on a port for each connection and receive input image and call the main function on the third part, finally sent the result image back to the client. The third part is the main component of the application which is used to recognise the category of the disease and implemented by Matlab, and it is also the most difficult and central program. The diagnosis program design would be roughly introduced in the next section. After the recognition, the result image would be saved in the server for retuning to the client.

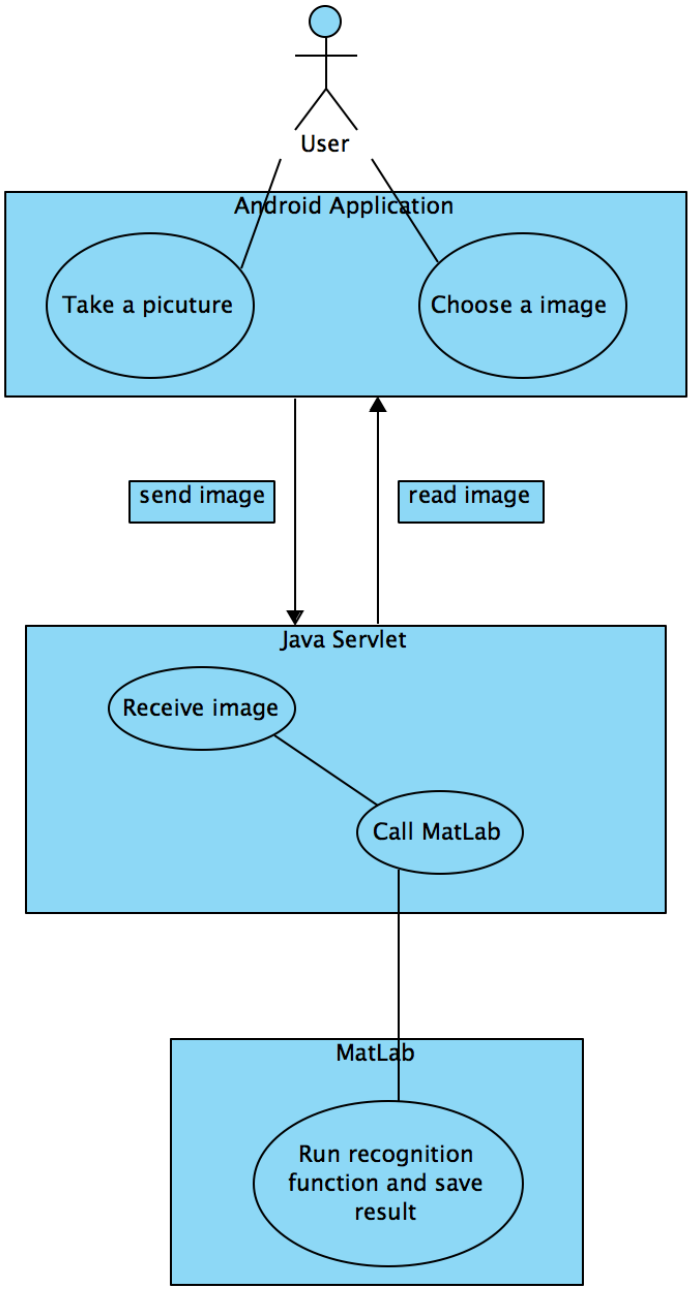


Figure 3.1: User Interaction Flow Diagram

3.2 User Interface Design

Figure 3.3 shows the graphical user interface draft of the application in android mobile phone. As the difficulty of the project focused on the implementation of diagnosis algorithm, the function buttons and provided operation are limited. There are two functional buttons at the bottom of the screen which allows the user to choose wether they want to use an existing leaf image in the gallery or take a new photo of leaf. Above the buttons, user can see the diagnosis result in this area which contains category labels for every leaf. The design of the logo of the application is shown in Figure 3.3 below.

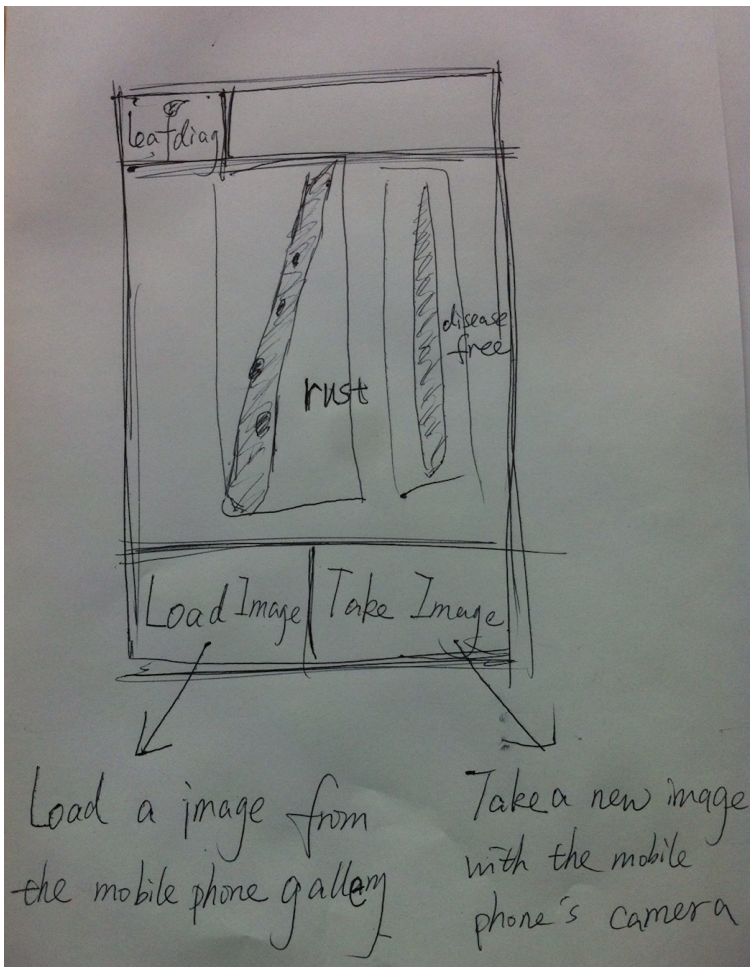


Figure 3.2: Graphical User Interface Draft

Figure 3.3: Logo of the application

3.3 Design for Main Component

This section will focus on the design of the main component development plan on the server side. As the recognition program programmed with Matlab is the central problem solving part, it is necessary to introduce how the program designed to be followed. The Figure 3.4 and Figure 3.5 shows the process of the development of the main component.

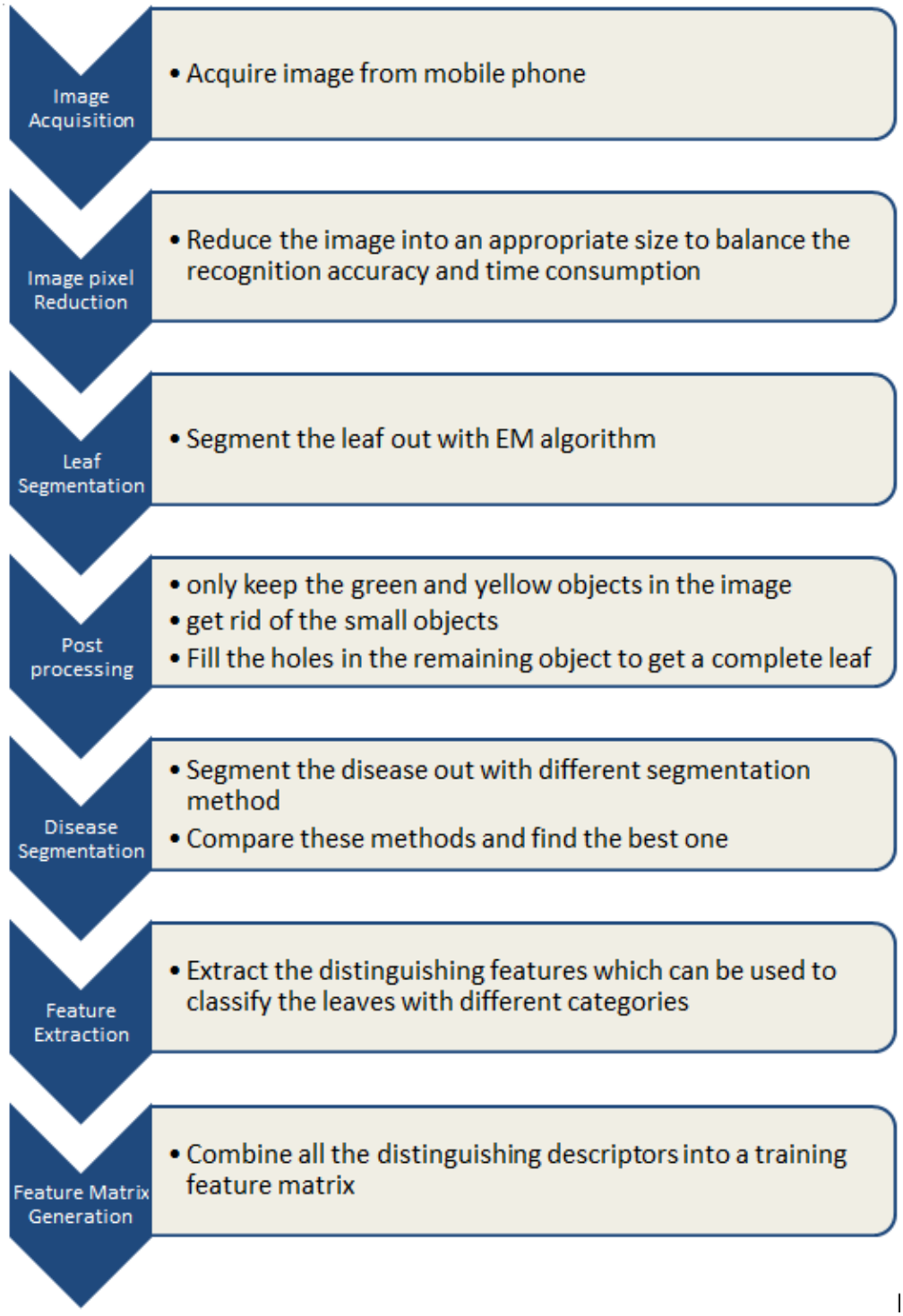


Figure 3.4: Central Component Deveolpment Process 1

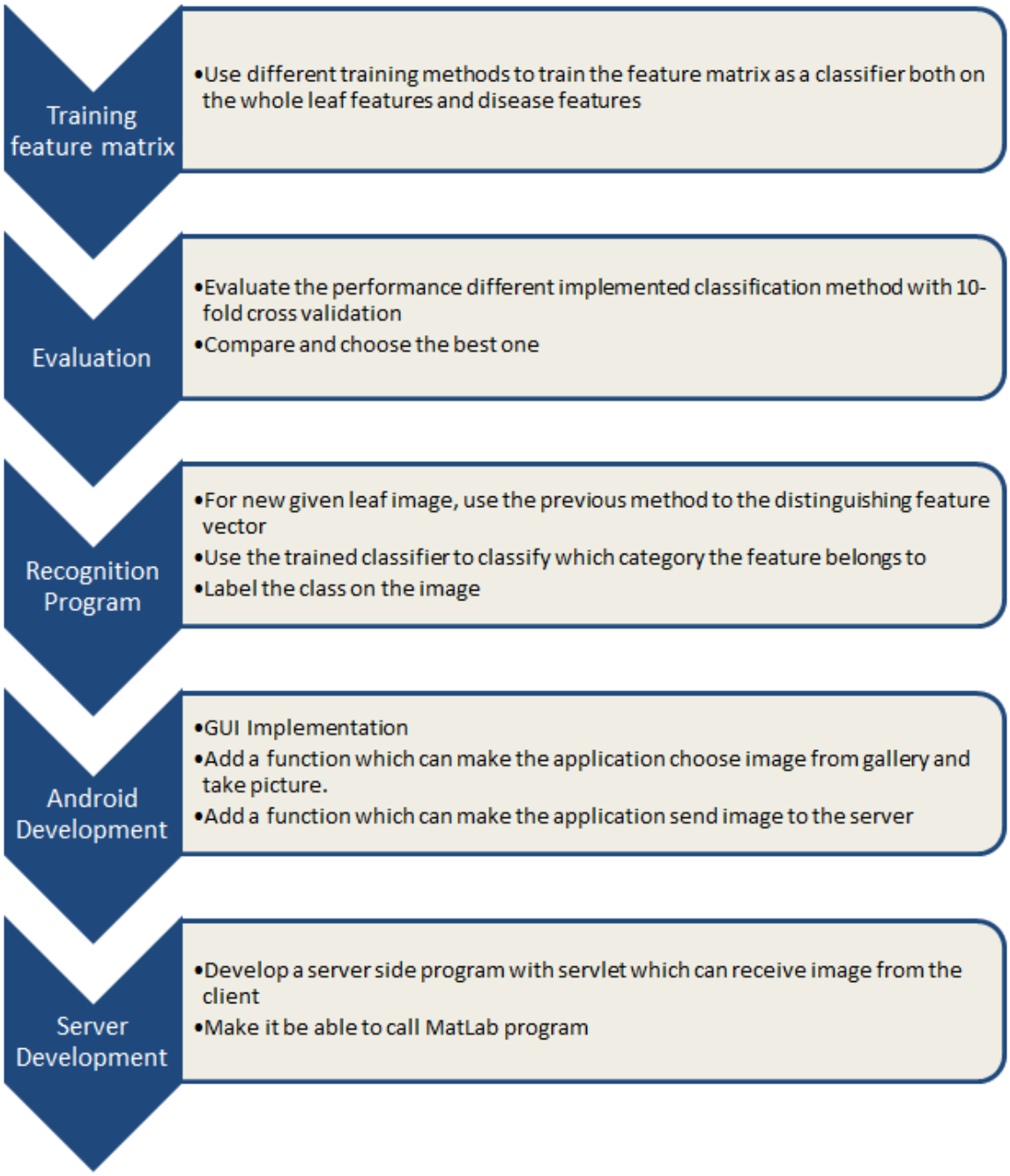


Figure 3.5: Central Component Deveolpment Process 2

Chapter 4

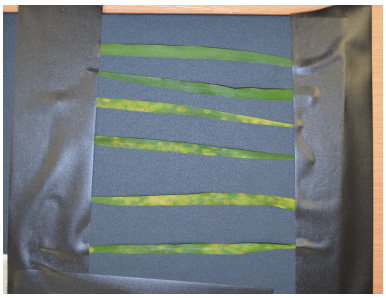
System Implementation

4.1 Segmentation

Segmentation is one of the most difficult part in this project. The provided image is very large and the background is complex with some coins and branches (examples shown in Fig 2), so it is not easy to segment the disease spot out in an interactive application for a short time. In addition the aim of the project is to produce an mobile application, thus the segmentation has to be automatic rather than by human manipulation.

(a) Natural leaf

(b) Powdery Mildew disease



(c) Wheat Rust disease

(d) Septoria disease

Figure 4.1: The tree kinds of leaf disease included in this study

Image Segmentation approach can be generally split into two categories, which are edge detection or pixel classification. Both of them have different kinds of advantages and disadvantages. For the edge detection some unclosed regions may

appear in the segmented image, and pixel classification can bring some isolated classifications. Isolated pixels can be easily wiped off by certain filters, while unclosed region usually have an huge negative impact on the object determination. Additionally, for a complex background, some inaccurate object may be segmented out. Pixel classifications are usually unsupervised clustering of the pixels. There are some well-known clustering algorithm. One of them is the k-means algorithm which is based on the recursion of two operations to classify a set of data into k categories. The algorithm assigns each point to the nearest mean to be labeled, and then calculate the means for labeled points of each categories. This two operations iterate again and again until the error function converges. However the first k means are randomly initialised, which could cause different results for the same data set. Therefore if the segmented elements number is relatively large, slightly different segmented results may appear. For the leaf segmentation we think Expectation Maximisation (EM) segmentation in Gaussian Mixture Model should be suitable, which is a colour segmentation method based on Maximum Likelihood (ML) estimation for the pixel clustering. The EM algorithm in Gaussian Mixture Model is explained in the Chapter 2, so for this part we just care about the EM segmentation and its implementation.

4.1.1 Leaf segmentation

EM segmentation

The EM colour segmentation segments the image into different images based on the iterative EM algorithm. An RGB image consists of many pixels, and every pixels consists of three elements which represents for red, green and blue respectively. It can also be viewed as red, green and blue three planes combined together. IF a RGB image has I pixels which can be expressed as $y_i = (y_i^r, y_i^g, y_i^b)(i = 1, .., I)$, where y_i^x represents a scalar value in the X plane. To build a mixture of Gassuain densities model, we can assume that each pixel y_i is drawn from the a mixture of three-variate (R,G,B). Gaussian densities model, Suppose we want to segment the image into J parts, the number of Gaussian densities in the model should be J and each pixel has a specific weight λ_j to a Gaussian.

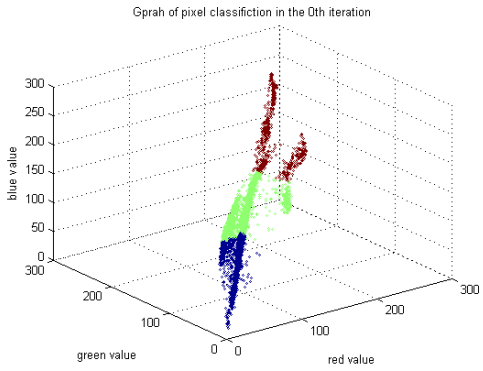
$$Pr(x|\boldsymbol{\theta}) = \sum_{j=1}^{J} \lambda_j Norm_x[\mu_j, \Sigma_j] \tag{4.1}$$

According to the Expectation Maximisation algorithm introduced in Chapter 2, an algorithm was implemented to estimate the prior possibility of the mixture of Gaussians given a set of pixel data, and then each pixel can be classified into corresponding category by possibility comparison. To visualise the process, the function was tested by given a leaf picture (Figure 4.2) and 3 categories, and the classification result during each iteration were plotted. The size of the given picture shown in Figure 4.2 is 49283264, it means that the program has to calculate 49283264 times for each iteration. Since the quality of the given pixel is too high, we have to resize the image into smaller one for further classification. For a quick and clear plot, the image was resized into the one width 100;
As the data shown in Figure 4.3, the classification becomes more and more accurate

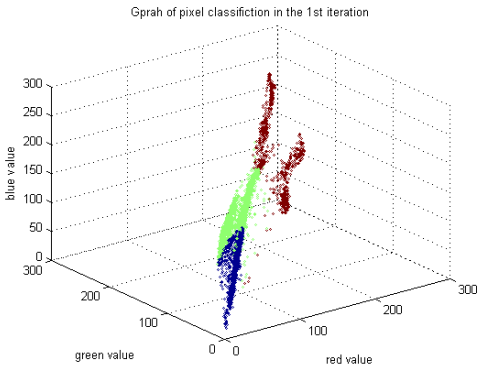
(a)

Figure 4.2: Test picture

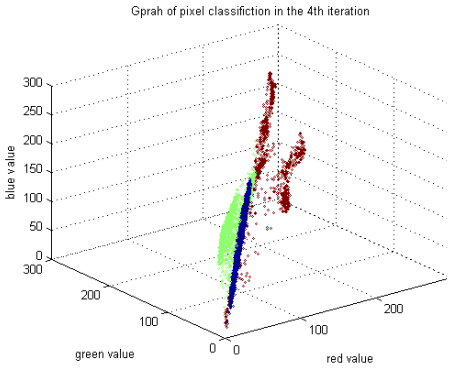
with the increasing EM iterations. In order to save the segmentation time, stopping criteria should be introduced. If the error rate is less than a certain value (0.1), the iteration will stop. The final parameter estimates of the means and the proportions of the clusters are shown in Table 4.1



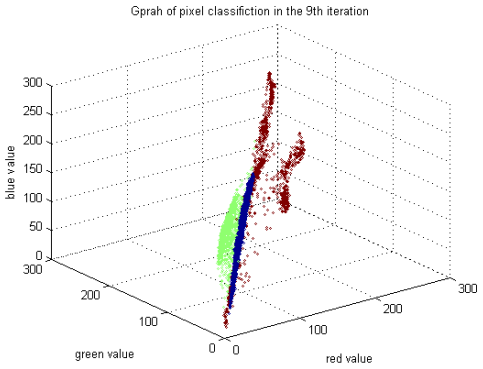
(a) 0^{th} iteration



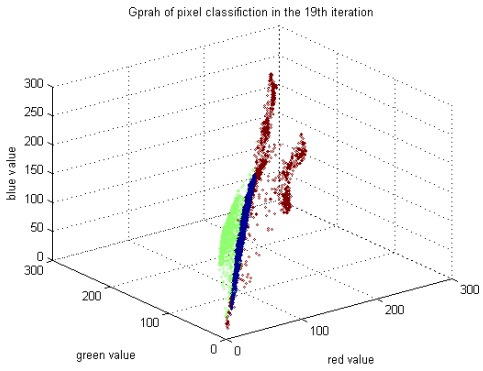
(b) 1^{st} iteration



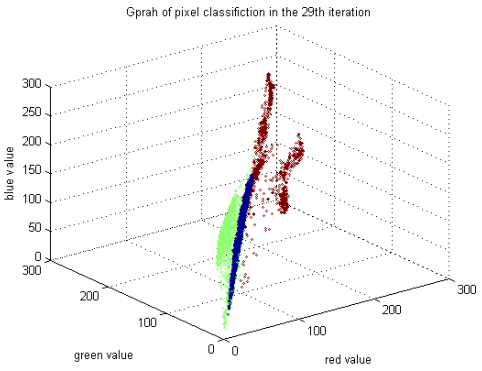
(c) 4^{th} iteration



(d) 9^{th} iteration



(e) 19^{th} iteration



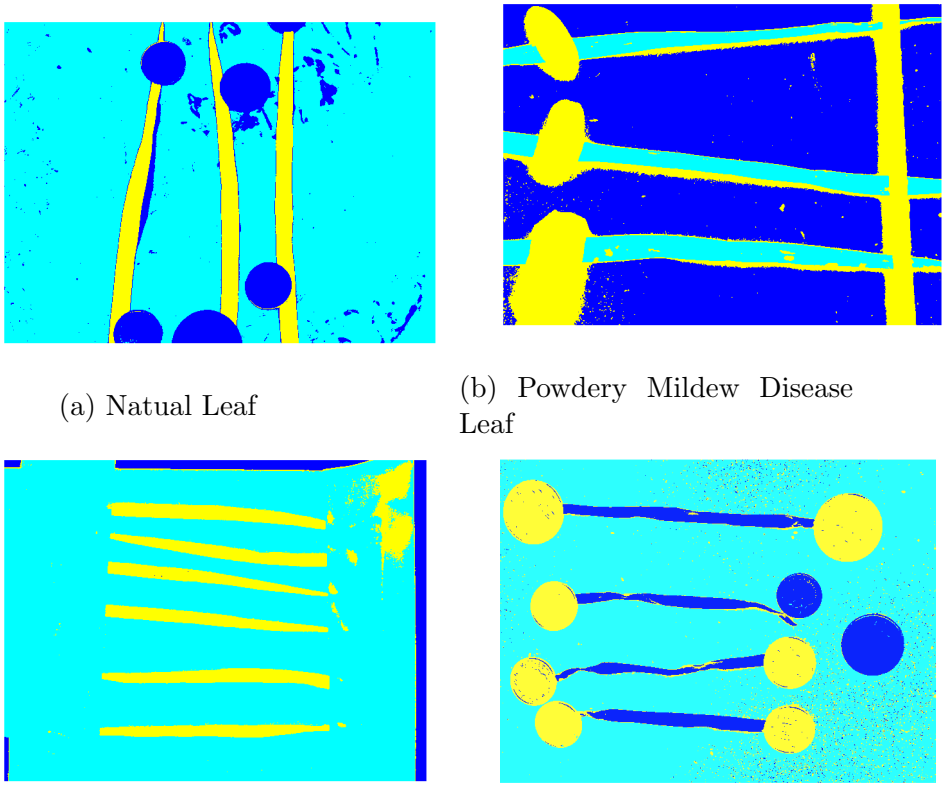
(f) 29^{th} iteration

Figure 4.3: This Figure visualize the changes of clustering for each pixel during the EM process, which was implemented with MatLab. The red, green points reprepresentent the pixel data clustered into three kinds of categories

Case	Cluster#1	Cluster#2	Cluster#3
R	105.6604	105.8797	182.5012
G	107.3948	134.7149	167.3033
B	110.1285	91.4143	156.2868
Proportion	0.8158	0.0895	0.0947

Table 4.1: Final parameter estimates of the means and the proportions for the Test picture.

In terms of the real implementation on EM segmentation, I find that the image with width 500 should be appropriate for a good balance on the time spending and result accuracy. resizing the image, the image processed by the function and the sample segmentation result shown in Figure 4.4. Normally, the number of segment partitioned into should be given by the user. All the example image was partitioned into three segments and each colour corresponds to one of them. From the the segmented image below, it can be seen that the leaves are segmented out as a part of one segment.



(a) Natual Leaf

(b) Powdery Mildew Disease Leaf

(c) Wheat Rust Disease Leaf

(d) Septoria Disease Leaf

Figure 4.4: The tree kinds of leaf disease included in this study

After that, we need to partition the colour image into some coloured segments with the segmentation labels for each pixel. Figure 4.5 shows an example of segmented colour image.

(a) Origin image

(b) Segment 1

(c) Segment 2

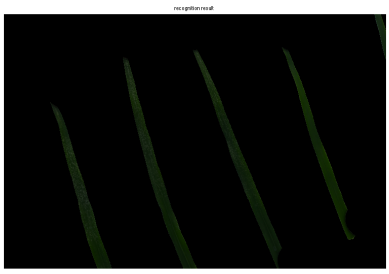
(d) Segment 3

Figure 4.5: Leaf Segmentation Example. (a) is the origin image, and (b),(c),(d) are segmented color images from (a) with EM algorithm. The segmentation result for this example is good.

Post processing

Although the EM segmentation can perform well for most images, there are still some serious problems for some particular images. Therefore some post processing technique can be taken for a better segmentation result. Look at the Figure 4.6, (a) is one of segmentation result images, however, some coins and blue edges can still not be removed. For the some defects in the image, we can use colour threshold to get rid of some defects. Thus a keeping green and yellow method was added to post process the image by color threshold. Picture (b) is the result of the method, which looks good.

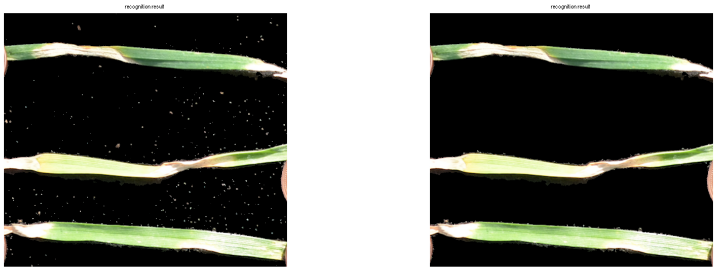
(a) Before processing



(b) After processing

Figure 4.6: Processing segmented image with a function to get rid of some irrelevant color

Next we want to get and separate each leaf out of the image, so we use connected neighbourhood method to label each connected pixels as an object. However as shown in Figure 4.7 (a), there are lots of small pixel which are isolated which will be still labelled. To improve the efficiency and save the execution time, the small connected component in a binary image would be removed that have fewer than 50 pixels. The results of the operation are shown in Figure 4.7 (b).



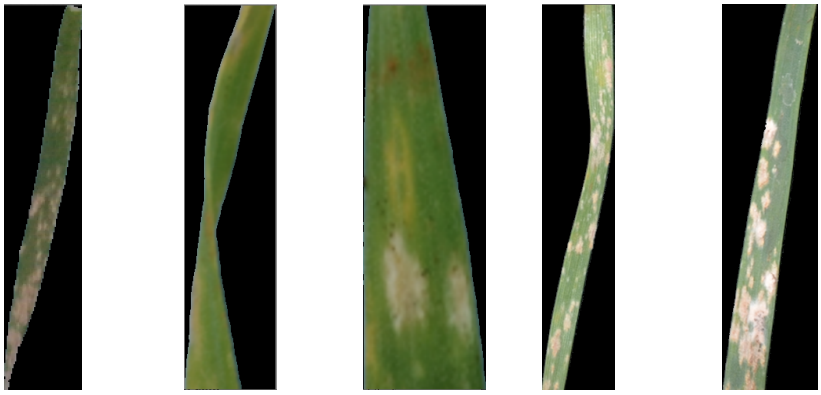
(a) Before processing (b) After processing

Figure 4.7: Processing segmented image with a function to get rid of some irrelevant color

For each label connected component, our goal is to sieve the leaf out. First we transfer it into RGB colour image and then the sum of the RGB value of each pixel were calculated, which denoted S. Via experiments, it was found that the ratio of the sum of red value by green value should be less than 1.1 and less than 0.855 for the ratio by blue by green. After that the program calculate the maximum and minimum boundary of the object and then extract it out. Some segmented leaf examples are shown in Figure 4.8.

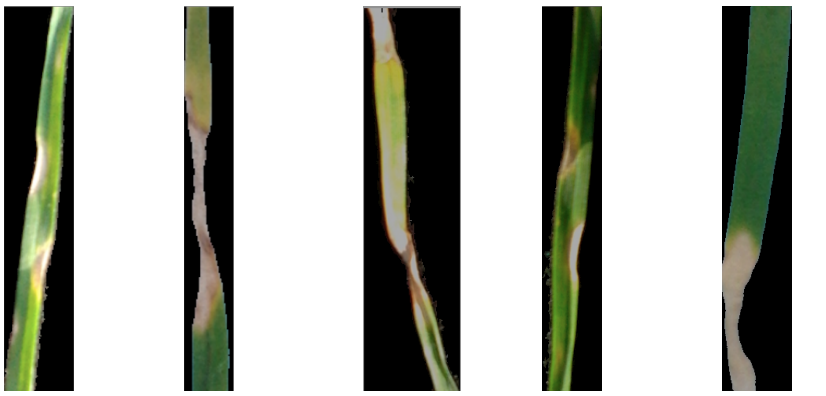
(a) (b) (c) (d) (e)

Figure 4.8: Disease free leaf segmented result



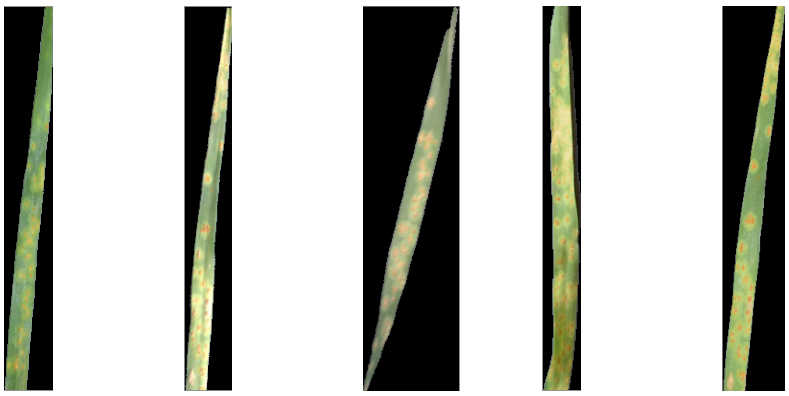
(a) (b) (c) (d) (e)

Figure 4.9: Powdery Mildew disease leaf segmented result



(a) (b) (c) (d) (e)

Figure 4.10: Septoria disease leaf segmented result

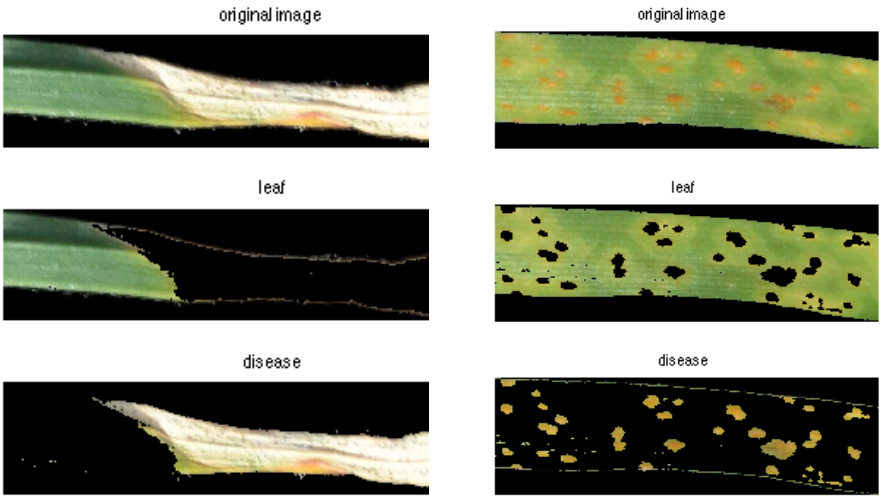


(a) (b) (c) (d) (e)

Figure 4.11: Wheat Rust disease leaf segmented result

4.1.2 Disease Segmentation

The aims of segmentation is for getting useful and discriminate features to train a classifier. Both of leaf features and disease features were used to do experiment to compare which classification result is better. Obviously disease features should be extracted from the disease segments, thus we would talk about diseases segmentation from pure leaf image in this section. As we discussed in the last section, a EM segmentation algorithm has been implemented, which can be used to segment the leaf disease. In some ideal cases, EM works very well for disease segmentation. However some fatal problems still exists as an automatic disease recognition smart phone application. In terms of the disease segmentation, the EM segmentation function was modified a little by partitioning the leaf into two elements which are leaf and disease without considering the black background. For a given clear image, some segmentation results shown in Figure 4.9. From the result, it can easily be seen that the segmentation is extremely good.

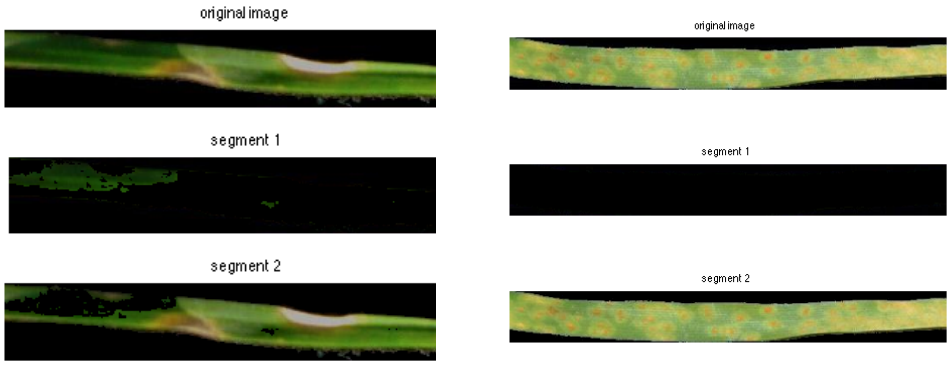


(a) Septoria (b) Wheat rust

Figure 4.12: Successful examples by EM segmentation

However there are two kinds of fatal limitations which make EM algorithm for disease segmentation impractical within a smart phone application. One problem is that the segmentation time is too long to get a result from an interactive application. Another one is that the original leaf image must be very clear and contain no defects. If not, the segmentation result would be definitely wrong. For some unclear image, the colour of the disease is similar to the leaf which means the distance between each pixel is very short, so the EM algorithm may not cluster the disease very well. Some fail EM disease segmentation examples shown in Figure 4.10.

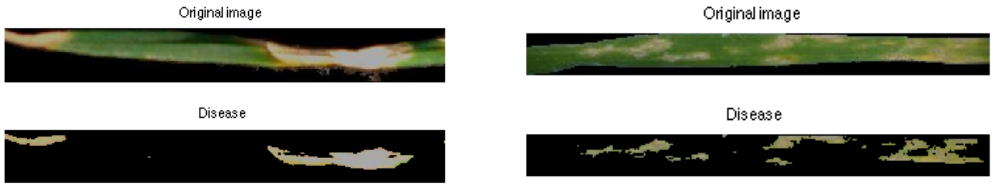
Thus an simpler and quicker segmentation method Otsus segmentation was used to segment the disease out based on the threshold which maximise the ratio of between-class variances to the overall variance. The algorithm was introduced in the Chapter Two. By using this threshold segmentation method, it is very quick to get an



(a) Septoria (b) Wheat rust

Figure 4.13: Failure examples by EM segmentation

relatively better result for almost all leaf pictures with generalised features although the result is not outstandingly accurate. Some segmentation result examples are shown in Figure 4.11. The average time of disease segmentation by Otsu's method is around 0.4 second.



(a) Septoria (b) Powdery Mildew

Figure 4.14: Disease examples segmented by Otsu

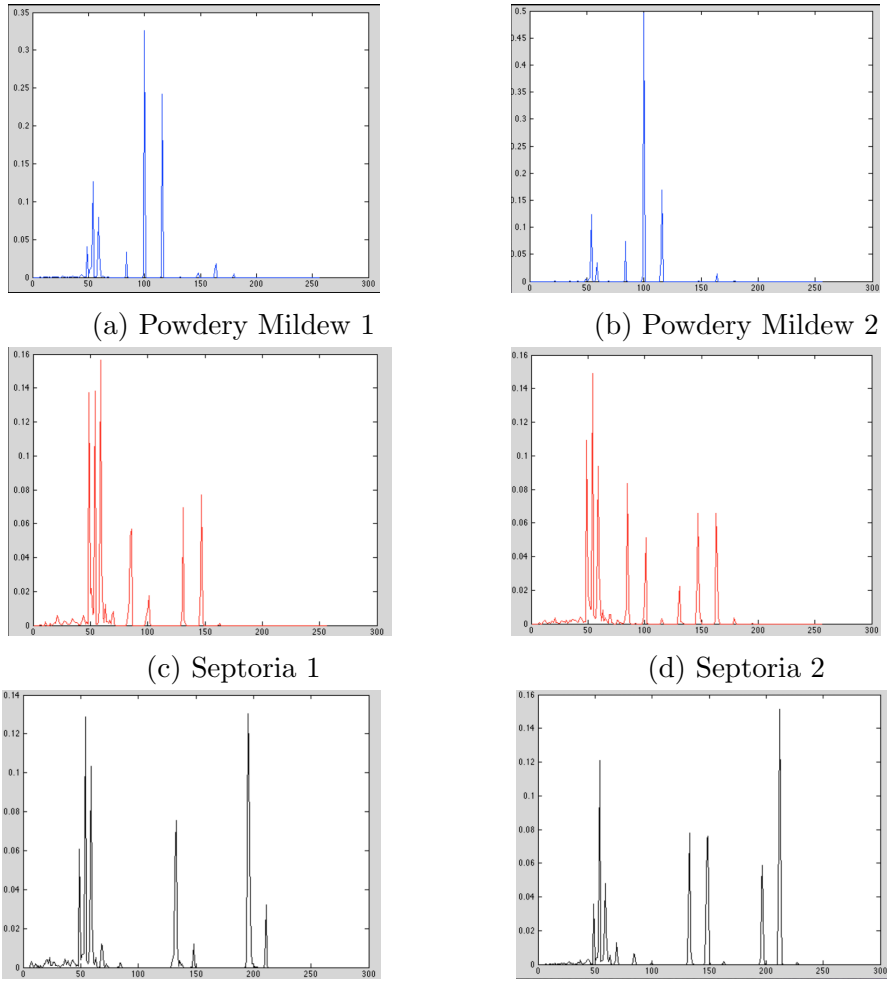
4.2 Feature Extraction

In order to identify the leaf disease, we should choose some appropriate features as a descriptor to distinguish different kinds of leaf disease. Inappropriate or excessive features would lead the classification to overfitting and long search time. Hence it is critically important to choose a good descriptor from the various diseases. Feature extraction is a kind of dimension reduction which can effectively represent the interesting part as an compact feature vector. After lots of experiments, it was found that the combination of colour histogram and Tamura Texture can be a good feature descriptor. We will take about that in the following two sections.

4.2.1 Color histogram

Colour is often the most expressive among all the visual features. Different disease has different colour distribution, and the same disease should have similar colour.

Thus we choose colour histogram as a descriptor to discriminate the leaf diseases. We choose use HSV (Hue, Saturation and Value) colour space to calculate the histogram. The program was implemented with the method introduced by Manjunath B.S. et al (2001). As the figures shown in Figure 4.11, they are the results of the colour histograms of same empales. It can be easily found that, the colour histogram of the leaves with the same disease are similar, but different diseases are totally different. Hence HSV colour histogram should be an appropriate descriptor.



(a) Powdery Mildew 1

(b) Powdery Mildew 2

(c) Septoria 1

(d) Septoria 2

(e) Wheat Rust 1

(f) Wheat Rust 2

Figure 4.15: This figure shows the colour histogram of some examples for the tree kinds of disease. (a) and (b) in blue line are the colour histograms of two different leaves with the Powdery Mildew disease. Similarly (c) and (d) in red line are the histograms of septoria, and (e) and (f) in black line of the Wheat rust.

4.2.2 Tamura's Texture Features

Texture feature is another commonly used and very important feature in many image analysis and computer vision applications. Image texture represents the spatial arrangement of intensities and colour in the image. Characterising the textures falls into two approaches which are Statistical and Structural. For this project, one statistical method call Tamuras texture (Tamura et al, 1977) was used as discriminate to classify the diseases. Tamura Features are designed based on the texture of humans perception according to psychological study. Tree properties of Temura texture were implemented in this project which are Coarseness, Contrast and Directionality.

Finally we store the colour histogram and Tamura texture features in an $1 * M$ vector for each segmented image, where M is the number of features. Then we put the features vectors together in a $N * M$ matrix, where M represents a segmented disease images features and N represents the total number of image examples. There is also a $N * 1$ vector which represents the labels of the corresponding disease examples. These labels are numbered from 1 to 4, which represents disease free leaf, Powdery Mildew leaf, Septoria leaf, Wheat Rust leaf respectively.

4.3 Learning and Classification

In order to ease the implementation, verification and documentation, the system could be divided into three different components based on its functionalities. The first part is the android application as a client which provide a user interface to interact with the functions of the application. The user can choose to take a picture or select a image form the gallery of the mobile phone as the leaf image to be diagnosed. Once the use decide which leaf picture he/she wants to use, the android application will send the image to the second part. The second part is the server implemented with java servlet which aims to communicate between client and main functions. The servlet will open a socket on a port for each connection and receive input image and call the main function on the third part, finally sent the result image back to the client. The third part is the main component of the application which is used to recognise the category of the disease and implemented by Matlab, and it is also the most difficult and central program. The diagnosis program design would be roughly introduced in the next section. After the recognition, the result image would be saved in the server for retuning to the client.

4.3.1 Dimension Reduction

Before the training the data, it should be better to pre-process the training data by reducing the dimension of the feature vector to increase the search speed and efficiency. Correlation feature selection(CFS) is a central problem to identify which features should be chosen to construct a classification model for a particular task through a correlation based method. A good feature subset can enhance the model interpretability, save training times and improve the generalisation by reducing overfitting. The assumption is that a good feature set should be highly correlated with the class and uncorrelated with other classes. CFS is an algorithm that eval-

uate the given heuristic merit function with the Pearson correlation measure and a search strategy. The merit function are the fomula given below in .

$$M_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{4.2}$$

where M_s is the heuristic merit of the feature subset which contains k features, $\overline{r_{cf}}$ means that the average Pearsons correlation coefficient between class with feature and $overlindr_{ff}$ means that the average correlation of feature to features. The higher the merit function, the better the feature subset is. The Pearsons correlation function can be represented as the following formula:

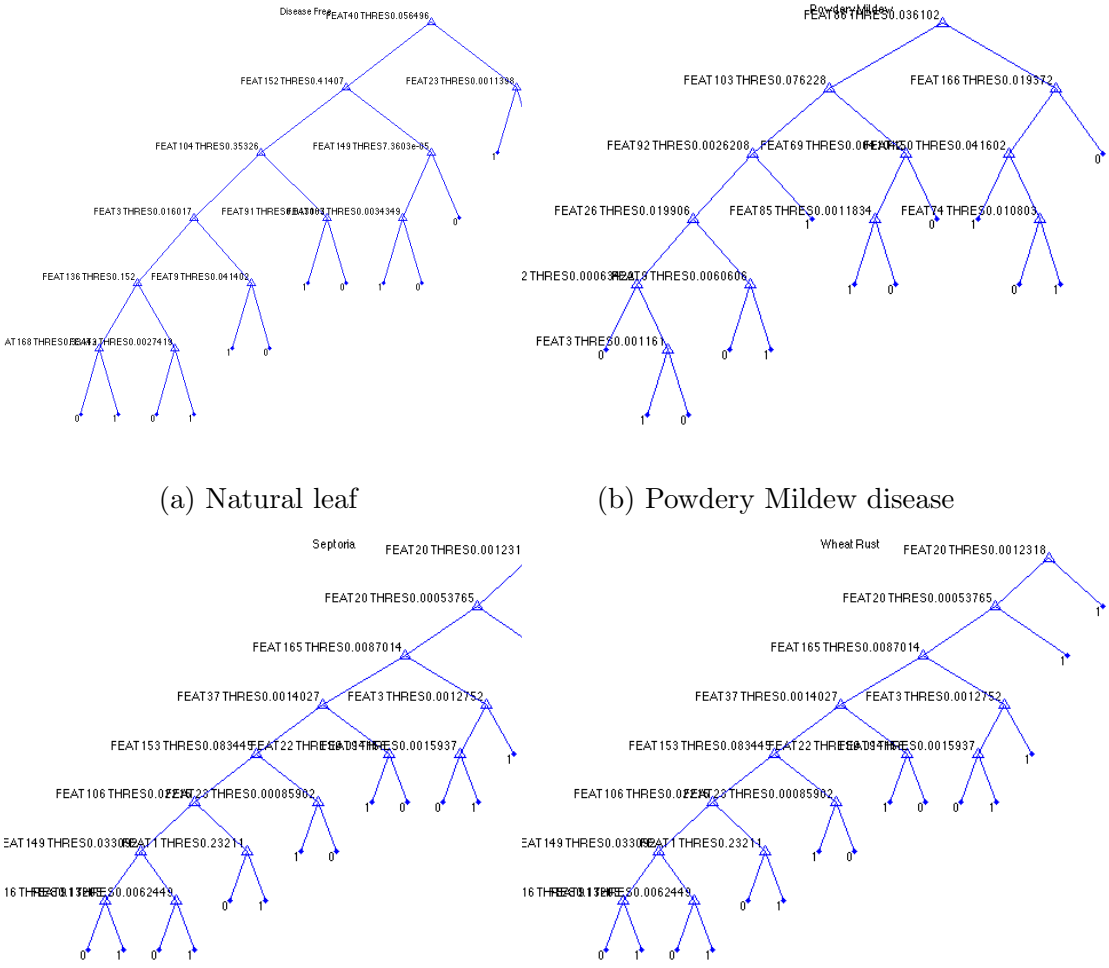
$$|r_{xy}| = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{4.3}$$

For the implementation of the dimension reduction in the project, we use the forward-selection manner to select the useful feature subsets. First we create a Pearson correlation matrix between each two features and classes and an empty feature index set. Then we consistently add a new feature from the given feature sets to make the value of the new feature subsets merit function greatest until the merit function converge which means that it can not be smaller by adding a new feature. The Pearson correlation coefficient linearly measures the correlation between two features M and N, returning a value between $+1$ and -1, where $+1$ means totally postive correlation, 0 means there is no correlation and totally negative represented by -1 . Basically a Pearson correlation draws a line of best fit through the data, and the coefficient indicates the distance of the data point to the line. The Pearson correlation coefficient can be calculated by a ratio of the covariance of M and N and the production of the standard deviation of M and N.

4.3.2 Decision Tree

At first the 'decisionTree_learning(X,label)' function was created to initialise the tree structure and then call the 'createTree(T,X,label) function to do the recursive learning for the subtrees. At the start of the createTree function, stopping criteria was performed first. If all the input label have the same value, then the same value (0 or 1) will be marked as the trees class and meanwhile returning the tree. Suppose the matrix data x has n rows of leaf disease data and m columns of features. Then the program find a maximum and Minimum value for each leaf data feature. And then generate a vector of 10 linearly equally spaced points as a set of threshold between each maximum and minimum value of each feature. Try splitting the data x into two matrixes by each threshold in the vectors and then calculate the Gain every time until the iteration stops. The threshold leads to the best gain should be the decision attribute. After that, divide the leaf disease data into two matrixes by the best attribute. And then call createTree function for each of the two divided leaf disease data to get two kid trees. 'learnDisease(x,y)' is a function to learn the six leaf disease decision trees for a given set of leaf disease datas and labels and returns a list of six trees. decideTree(x,T)' is a function to decide whether an disease feature data belongs to an leaf disease by the leaf diseases decision tree. diseaseDecision(x,trees)'

is a function to decide which disease of a row disease features belongs to, given a set of disease decision trees. The function try to classify by each binary decision tree and then get a list of binary results. First find the indexes of the positive results. If the positive index list is empty, then return a random result between those four leaf diseases, else random get one from the positive index list and translate it to the leaf disease and return. The binary decision trees built on the four kinds of leaf disease are shown below in Figure 4.12.



(a) Natural leaf

(b) Powdery Mildew disease

(c) Septoria disease

(d) Wheat Rust disease

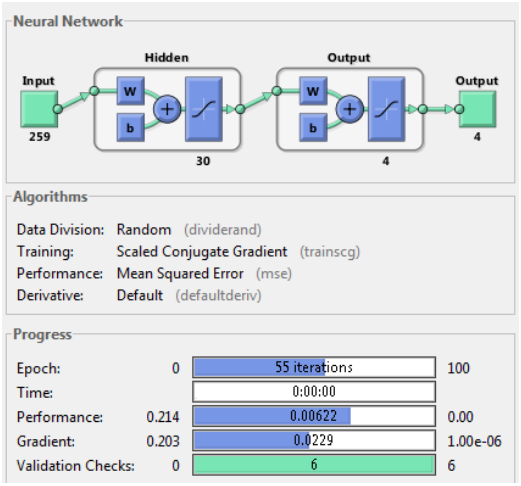
Figure 4.16: The decision trees of these four categories

4.3.3 ANN

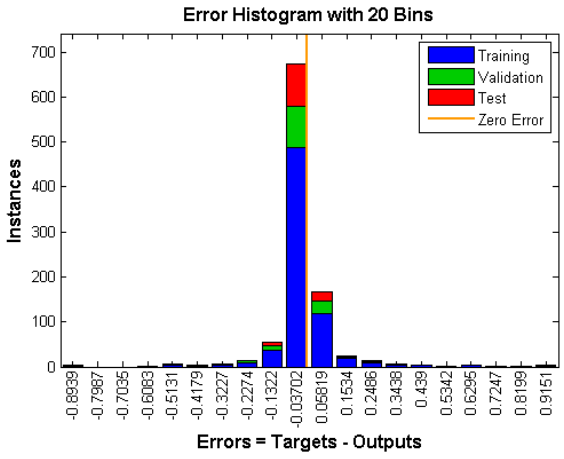
In neural networks, in order to fit the training set best, gradient descent is used to tune network parameters by back-propagation. Back-propagation over multilayer networks is sure to converge toward some local minimum error between the training example values and the output values. However, it is not guaranteed that the algorithm will find the global minimum error. The number of hidden unit layers defines exactly the expressive power of the network, and also the complexity of the decision boundary.

Multiple experiments were produced in order to find an appropriate model for reducing the error, and keeping a performance graph in which the way training data is processed is as accurate as possible. The structure is formed with input data, hidden layers and 4 outputs. We implement a ANNdata function to transform the data into the corresponding input format which can be accepted by the ANN tools, we have used one hidden layer of sigmoid units containing 20 hidden units (neurons) with a number of epochs of 100.

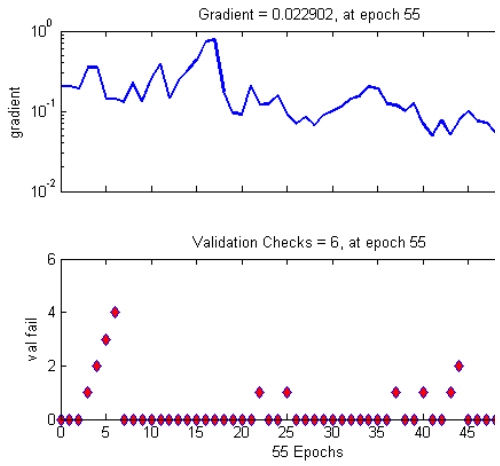
Figure 4.17 shows the some diagrams about the information of the ANN training. Subfigure (a) shows the details information of our ANN classifier, from which it can be seen that the structure of the Network. The training are stopped when the epoch equals 55 in this case, and the performance (error rate) are 0.214. The training stop when the validation error rate start to increase consfinuesely for 6 epochs to get the minimum validation error which are shown in subfigure (d). From Subfigure (c), we can see the information about at which epochs the validation error rate increased. Subfigure (b) shows the distribution of the instances for each error value, from which, it is clear to see that most of instances have a error around -0.037.



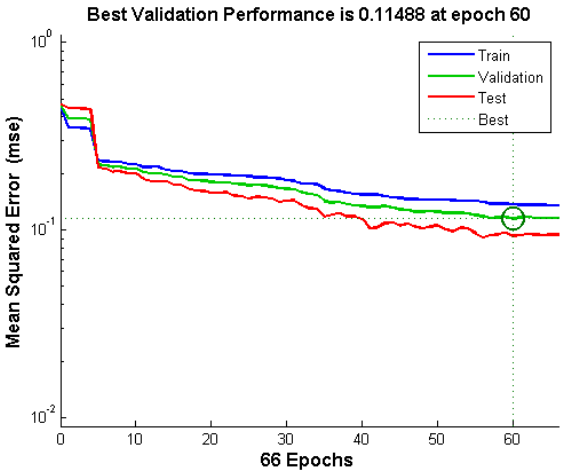
(a) ANN details information



(b) Error Histogram



(c) Validation diagram



(d) Performance diagram

Figure 4.17: Some diagrams show the information of the ANN trainning

The figure below shows the receiver operating characteristic(ROC) which represent the relationship between the true positive and false positive for the four classes in this the ANN classifier training process. It consists the training ROC, validation ROC, test ROC and the sum of them.

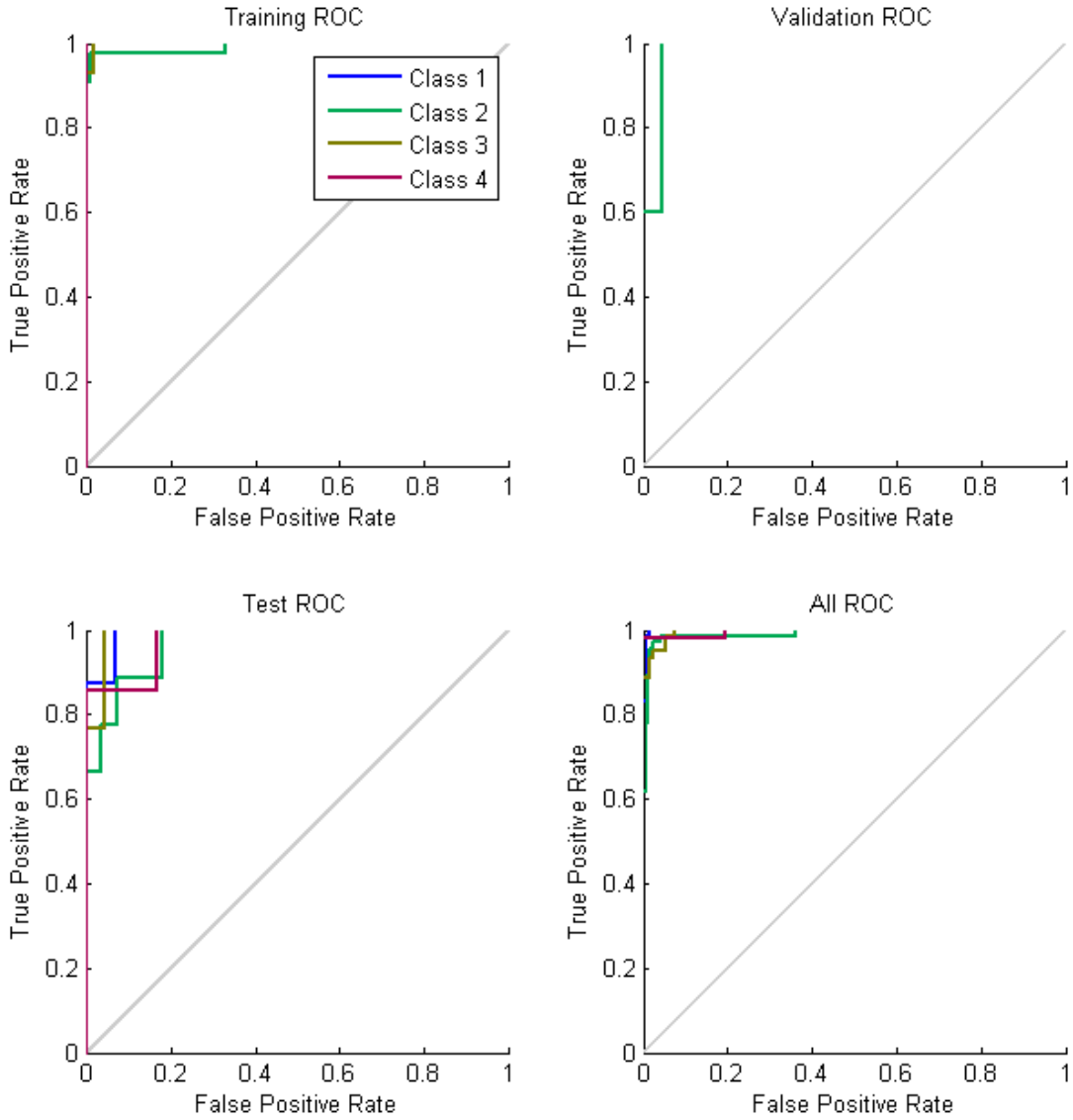


Figure 4.18: Receiver Operating Characteristic

Figure 4.19 shows the confusion matrix of the ANN training process in terms of training, validation, test and the sum of them. As for what is confusion matrix, it will be introduced at the Evaluation chapter. The test confusion matrix shows that there is a little confusion between second class and third class.

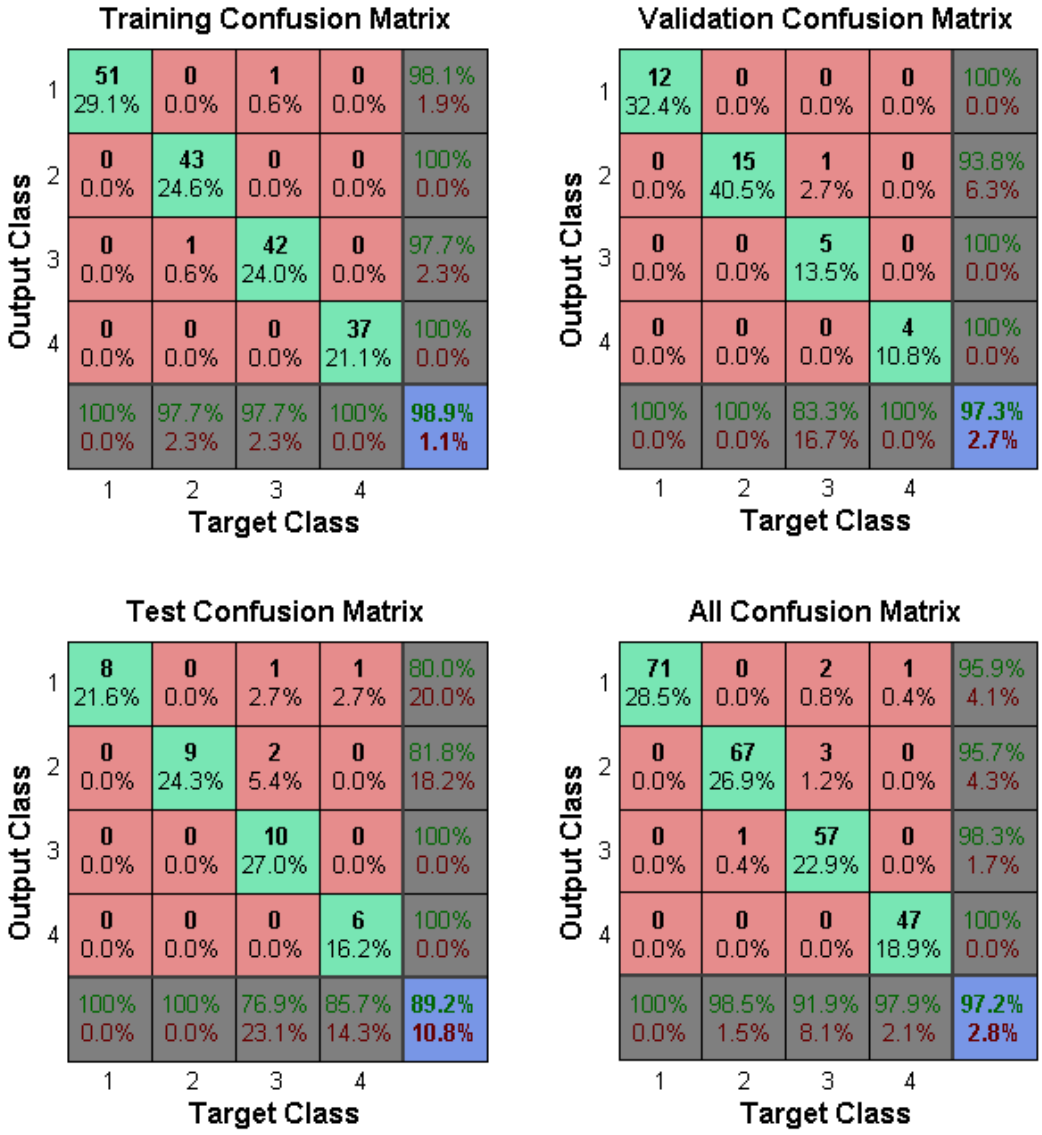


Figure 4.19: Confustion Matrices

Generalization and Overfitting

One of the most important problems in an training algorithm is overfitting. As the training times increase, the error on the training set will be very small, while the error in validation and test set may become larger. It means that the classifier may loss the generalisation for the new data. One of the built-in functions to avoid overfilling is early stopping. The ann tools in MatLab divided the data into training, test and validation sets. The training set is used to update the bias and weights and compute the gradient of the network. The validation subset is used to monitor the training. At start of the training, the validation error will decrease. After some iteration, the validation may increase which would lead to overfit. Once the validation error starts to increase, the training process will be stop, and the minimum error will be returned. However, each network in the MatLab will start from different initial weights and bias, which may lead to different training, test and validation set. The test set is a very good method to improve generalisation, but the data in the test set of one network may be in the training set of another network. To avoid that, we implement an algorithm to avoid that. We divide the origin data into two sets, one part (2/3) is for designing the network and another part (1/3) is for testing the result to ensure that is an completely independent test set. For each parameter setting, the network will be trained 10 times, and for each time the performance will be recorded. Then the average mean square error will be calculated. For each topology and parameter setting, the average performance will be used to compare the efficiency and accuracy of the network.

4.3.4 Support Vector Machine

In terms of the implementation of the support vector machine, the MatLab built-in function svmtrain was used to train the classifier for each two classes. Kernel functions are provided by the svmtrain including linear kernel, quadratic kernel, polynomial kernel, Gaussian Radial Basis Function kernel with a default scaling factor, sigma, of 1 and Multilayer Perceptron kernel. All the kernel functions were used to train and classify with testing on the feature matrix to find which one is better. At last, the linear kernel get the best performance result which would be shown in the Evaluation Chapter below.

Multiple Class classification

Pair wise classification was used by the support vector machines. In terms of training process, each two of the leaf disease will create a SVM struct. For this project, we know that there are four leaf disease classes, so we should build (4-1)*4/2 = 6 SVM struct. When we test a leaf disease data, we shall evaluate the 6 SVM and then classify according to which of the classes gets the highest numbers of vote. A vote means a classifier put a pattern into the class.

4.3.5 Random Forest

The Random Forest classifier was implemented via using the TreeBagger function in the MatLab statistic toolbox. The TreeBagger function can create ensemble of bagged decision trees. The parameters were adjust by the given features. An ensemble of 50 trees was created to train the features. Figure 4.20 shows that the out-of-bag classification error decreases with the increase of the number of grown trees. When the number of grown trees is less than 25, the classification decrease remarkably, whereas the decrease becomes flatting. The final error nearly equals 0.05 as the number of grown trees increasing to the 50, which is a quite satisfactory result.

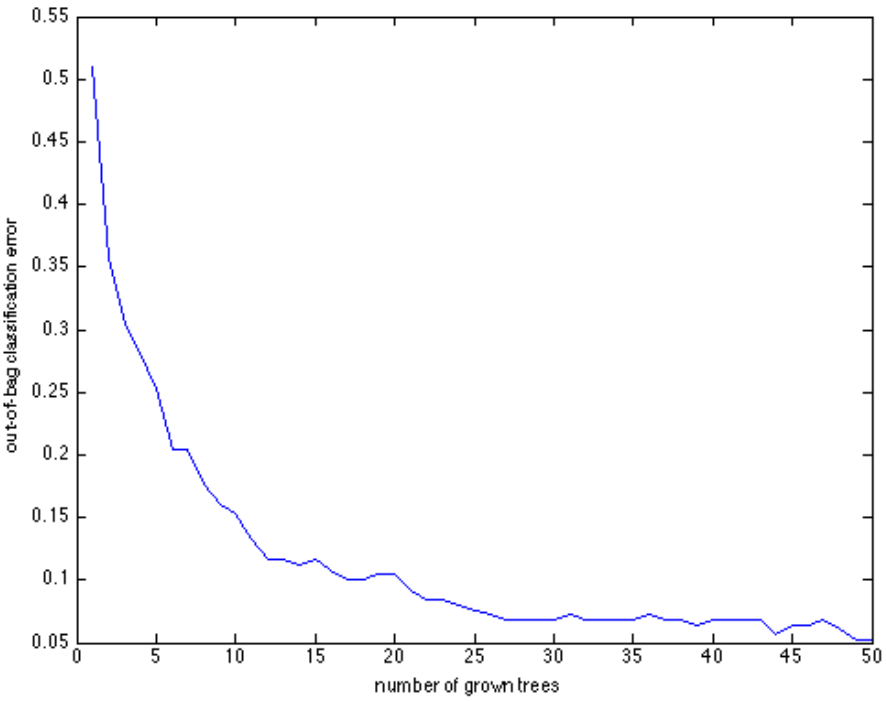
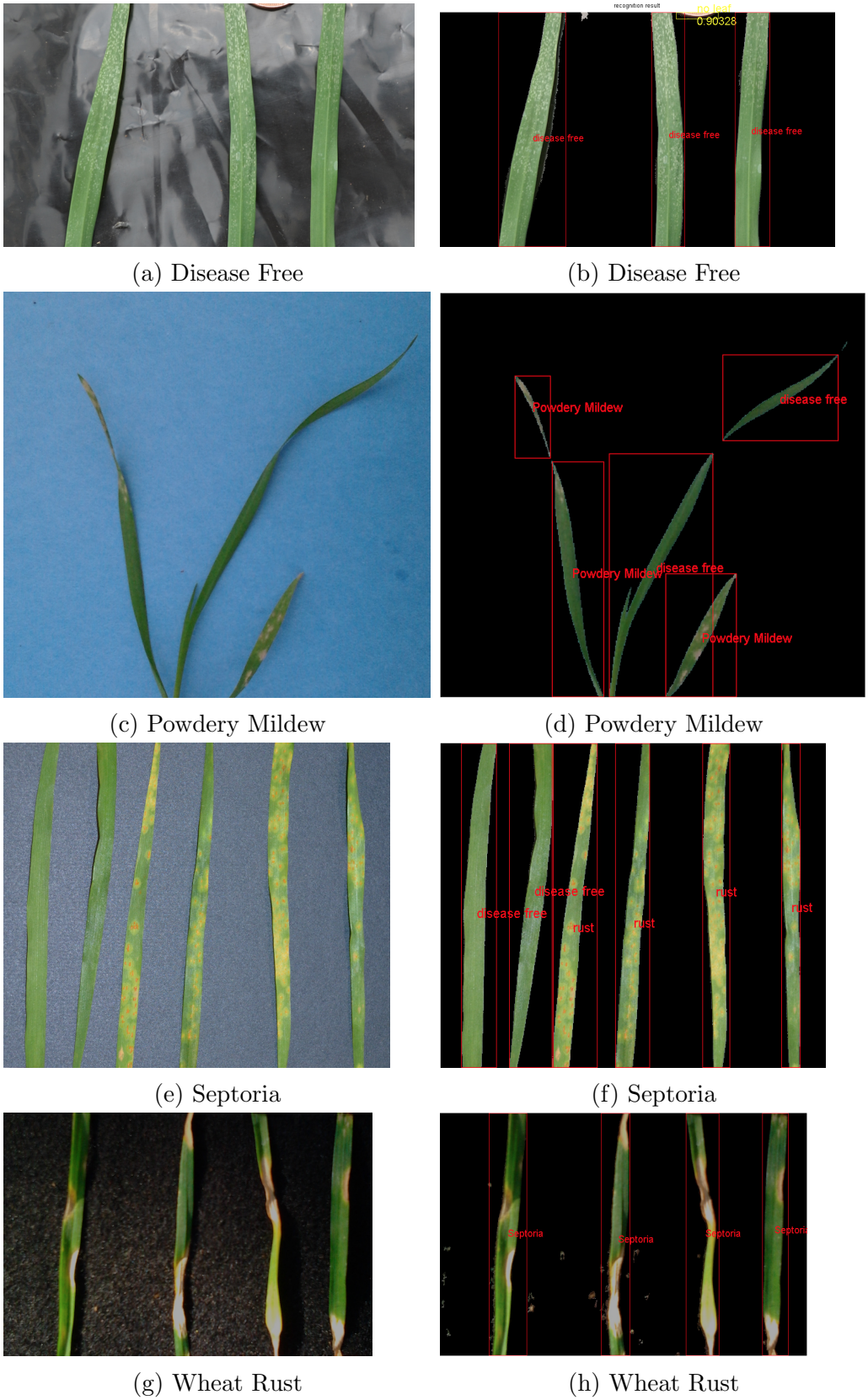


Figure 4.20: Confusion matrix Decision Trees on leaf

4.3.6 Final Recognition

After the implementation of the classifier, it is not difficult to build a recognition program to classify which category every leaf belongs to. Firstly segmentation function should be used to segment the meaningful area out and then feature extraction function was called to get the distinguished features from the segment storing in a vector. Successively the feature vector should be put into the classifier to predict its category. Finally, the predicted label will be drawn on the leaf. The Figure 4.21 shows some sample results of the leaves over this four categories.



(a) Disease Free (b) Disease Free

(c) Powdery Mildew (d) Powdery Mildew

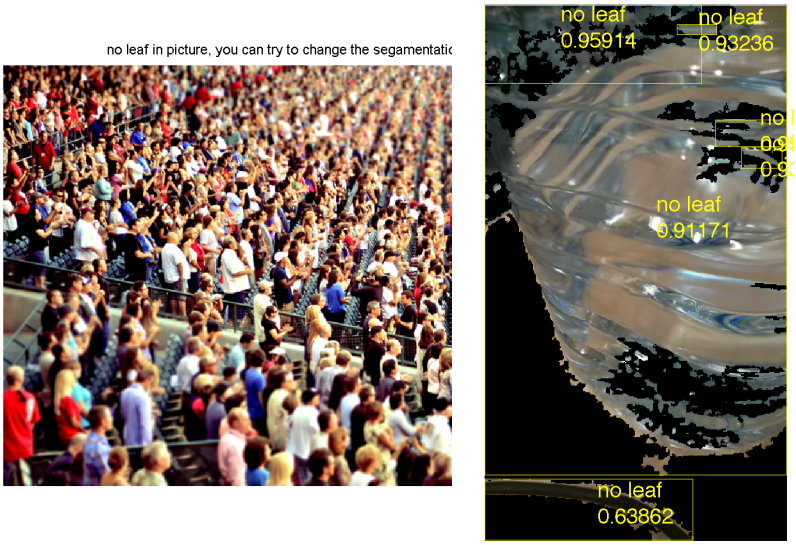
(e) Septoria (f) Septoria

(g) Wheat Rust (h) Wheat Rust

Figure 4.21: Some sample recognition results over the leaves in these four categories

Sometimes wrong pictures for example without a big enough leaf in the image may be entered into the program, it is necessary to handle this situation at that time.

Figure 4.22 shows the two kinds of catch methods used in this program for this kind of exception. After the EM segmentation, if none of the segmented partition has a overwhelming green proportion, the result as sample 1 will be returned with showing 'no leaf in the picture' at the top of the image. If the segmented part is eligible but the pixels of the objects in the partition don't satisfy certain leaf condition, this kinds of object will be marked as no leaf in yellow as the sample 2 shown in Figure 4.22 (b).



(a) Sample 1 (b) Sample 2

Figure 4.22: Recognition results for two non-leaf images

4.4 Android Development

For now, the central program has been finished, however we still need a user interface to interact with the mobile phone users. A simple android application has been implemented which is shown in Figure 4.22 below. The program can choose to select a image from the image from the mobile phone gallery by clicking the button 'Load Image' or take a new leaf photograph. Then the image will be automatically sent to the server programmed with java servlet in Netbeans, and the server will call the Matlab function by using the 'matlabcontrol' library.

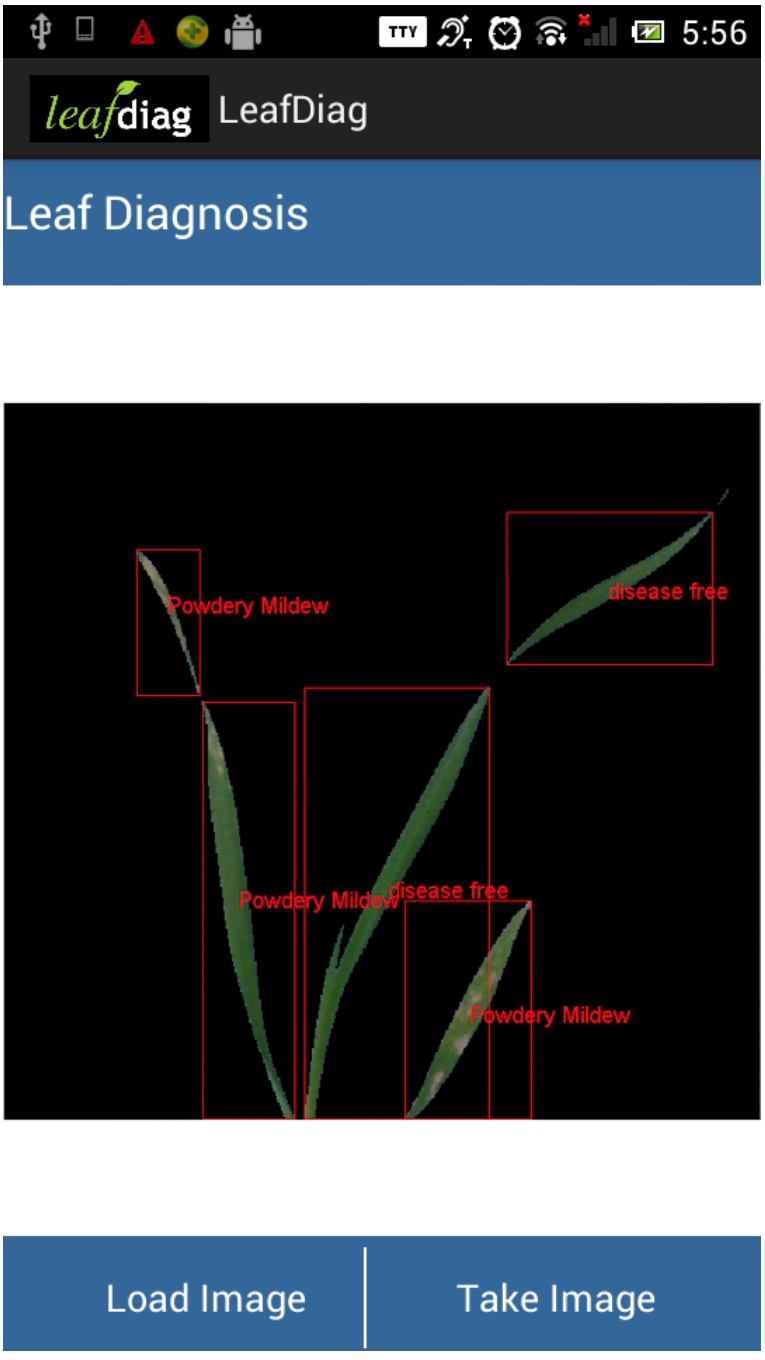


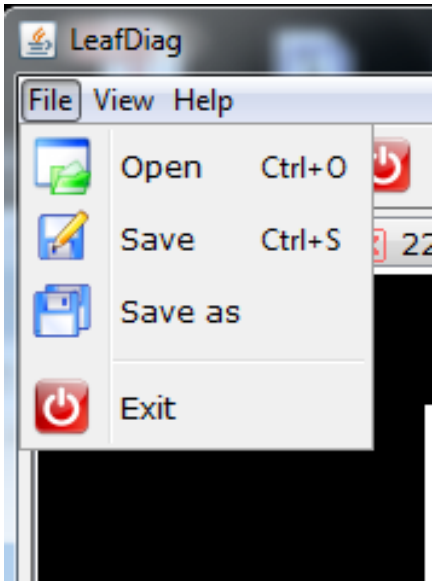
Figure 4.23: The Application Graphical User Interface

4.5 Additional Computer Based Software Development

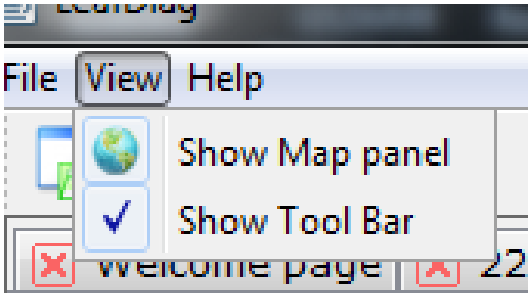
To extend the usability of the program, an additional computer based software was developed to aid the computer user to operate the function more comfortably. User can choose image from the disk on the computer and do some operations on the result image which will be introduced below. Figure 4.24 and 4.25 show the graphical user interface of the software. After the user opened the software, welcome page will be shown as Figure 4.24.

Figure 4.24: The Application Graphical User Interface

The menu bar is on the top of the software shown in Figure 4.25. If the user click the file button in the menu, the list in the Figure 4.25 (a) will appear. Then the user can choose to open, save or exit the software. If the user click the view button, two options are provided which make the user to show or hide the panel and tool bar by their preference as shown in Figure 4.25 (b).



(a) File Button



(b) View Button

Figure 4.25: Menus Bar

The toolbar is also available shown in Figure 4.26 which is very convenient to the user and contain all the main functions. If user dont want to show the toolbar, he can make it invisible by the setting of view preference.

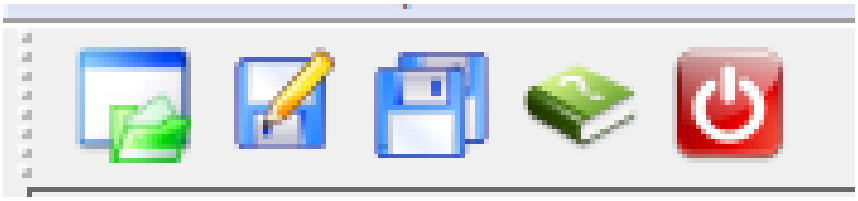


Figure 4.26: The Application Graphical User Interface

At the start of using this software, user can click the button in the menu, or tool bar or type Ctrl + O to open a image form the computer. Once a image is chosen, the Matlab program will start to diagnose what disease the leaf in the image has. And the result will be shown in a new tab panel with leaving the previous ones as Figure 4.27. User can also save or close any tab panel of them at any time.

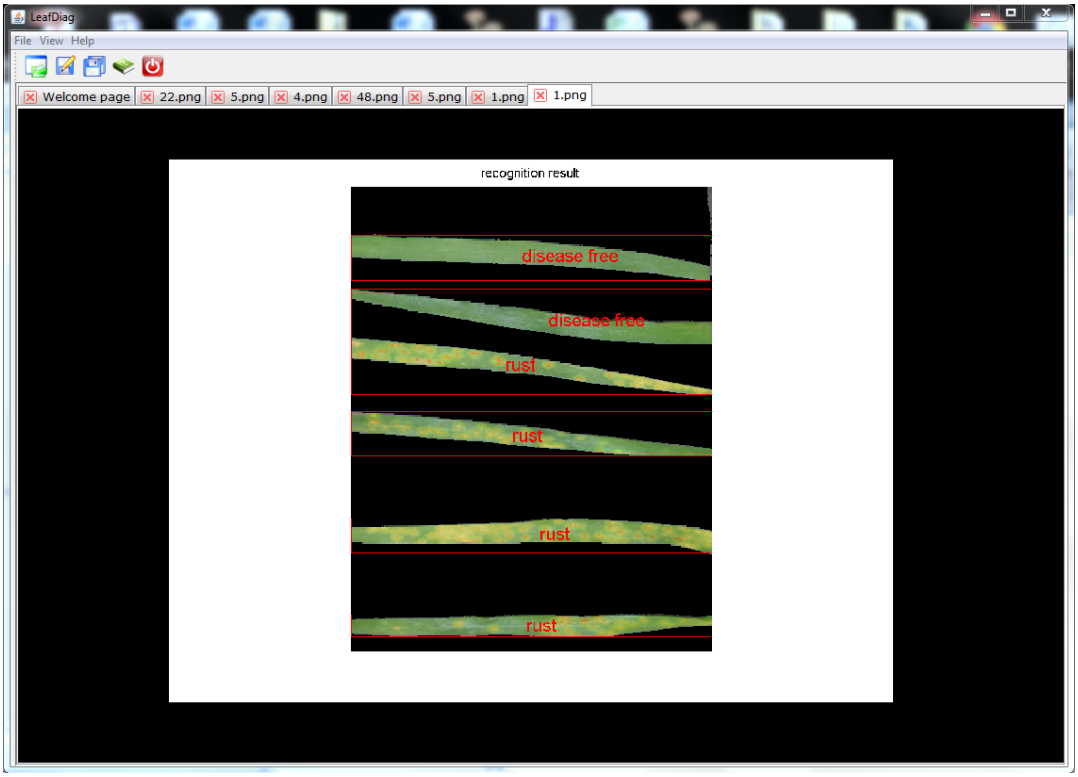


Figure 4.27: The Application Graphical User Interface

The tabbed pane can also allow the user to drag the tabs to in any order if they like as shown in Figure 4.28.

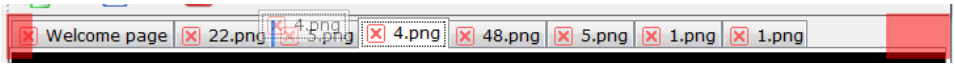


Figure 4.28: The Application Graphical User Interface

The user can use mouse to click and drag the result image to any position they want, and can even use mouse whole to zoom in and out the image to the bigger or smaller size. When user zoom in or zoom out the map, four red rectangle arcs will show around the zoom area and the zoom centre will be around the mouse position which is shown in Figure 4.29.

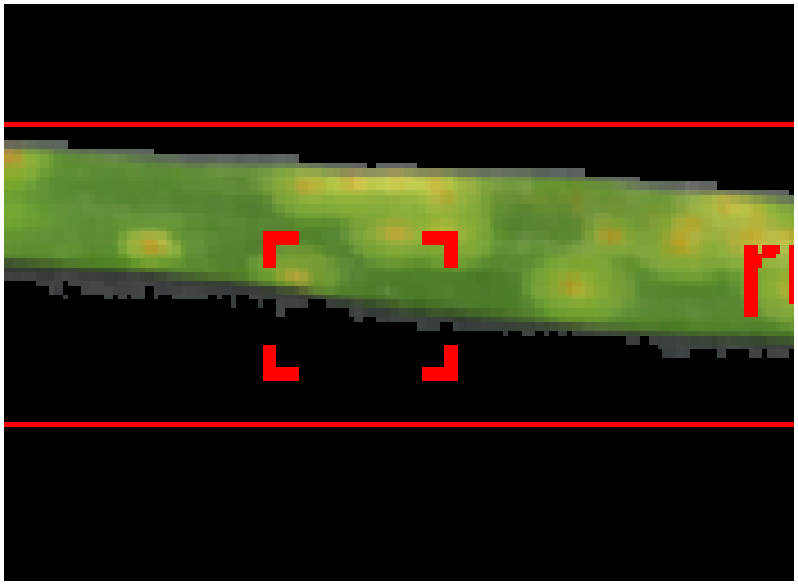


Figure 4.29: The Application Graphical User Interface

If the user click the right button of the mouse, a pop menu will occur to allow the user to zoom or centre the image in another manner as shown in Figure 4.30.

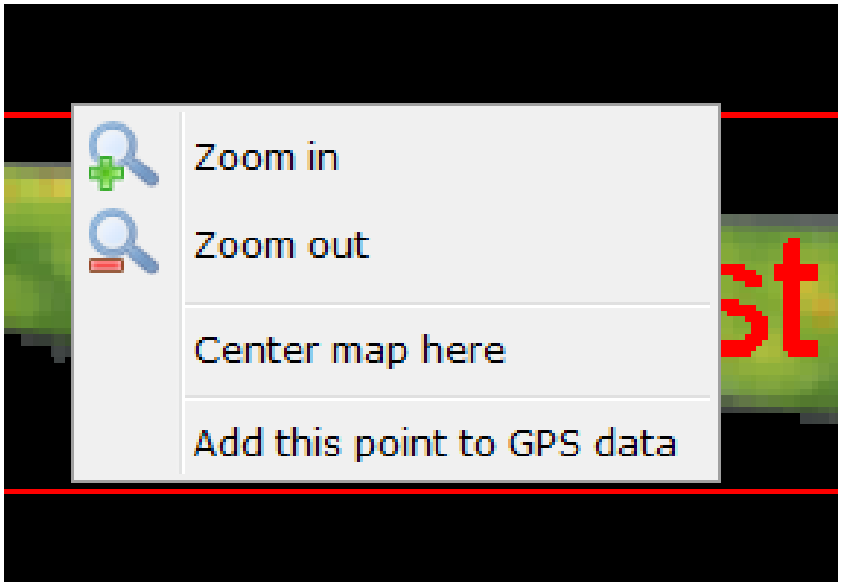


Figure 4.30: The Application Graphical User Interface

Chapter 5

Evaluation and Testing

In previous chapters, we have introduced the implementation of the project including segmentation, feature extraction, classification methods and software development. In this chapter, we will focus on the testing of the generalisation of the segmentation method on leaves in other categories. Next we will talk about the evaluation and comparison of the recognition accuracies with different machine learning methods both on the whole leaf and disease spot to find which one is the best for out application. After that we will examine whether we have achieved all the functions presented in the first chapter and whether the program will work properly in the normal or erroneous conditions. Finally user case evaluation will be presented to test user friendly.

5.1 Segmentation Test

To determine the success of the segmentation method on general leaves, it is vitally important to test the segmentation program on other kinds of leaves. All the segmentations presented above are operated on only one kind of leaf with different diseases. If the program cannot work well on other kinds of disease, the application will lose the generalisation property. To this end, five other categories of leaves were chosen to be tested by the EM segmentation program to show the generalisation. Four of five categories leaf examples are shown in the Figure 5.1 which are all in a relatively complex background. After the segmentation we got the segmented leaves which are shown in Figure 5.2, 5.3, 5.4 and 5.5. From the segmentation images, it can be obviously seen that the results is extremely good with no defect. The result images are complete and no mixed with other objects.

Figure 5.1: Leaf segmentation results of Category 4

Figure 5.2: Leaf segmentation results of Category 4

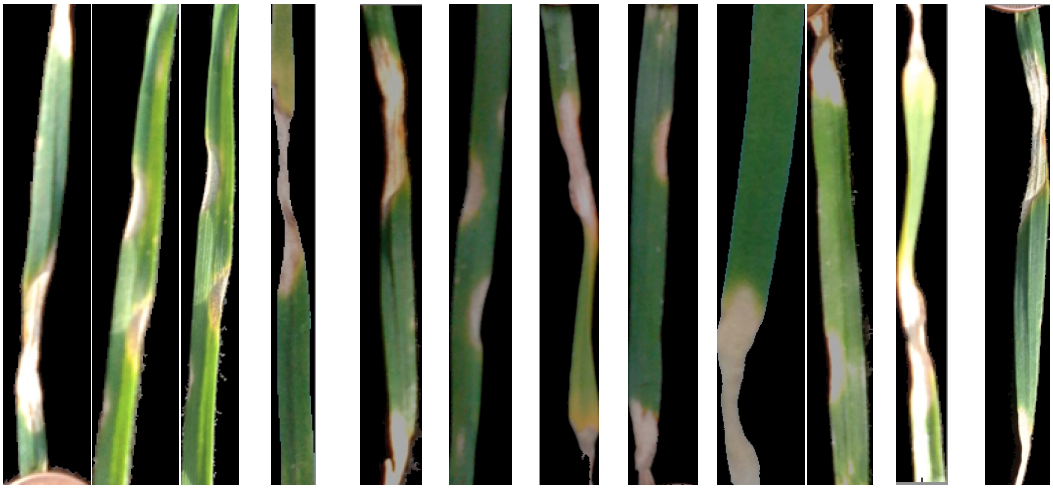


Figure 5.3: Leaf segmentation results of Category 4

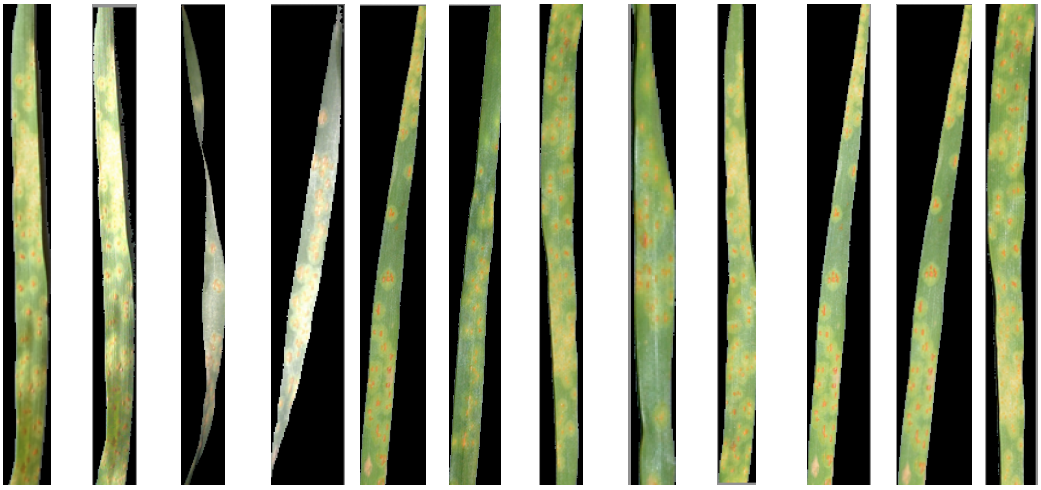


Figure 5.4: Leaf segmentation results of Category 4

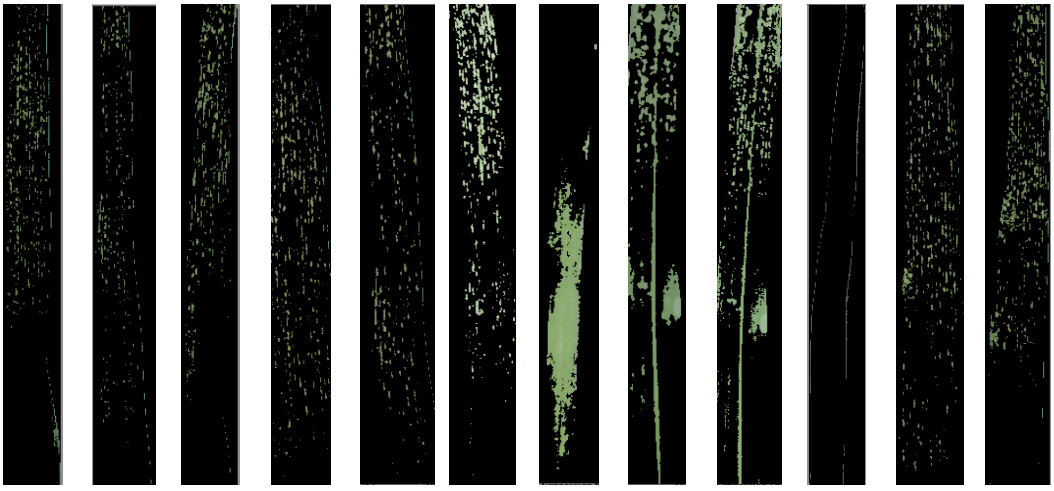


Figure 5.5: Leaf segmentation results of Category 4

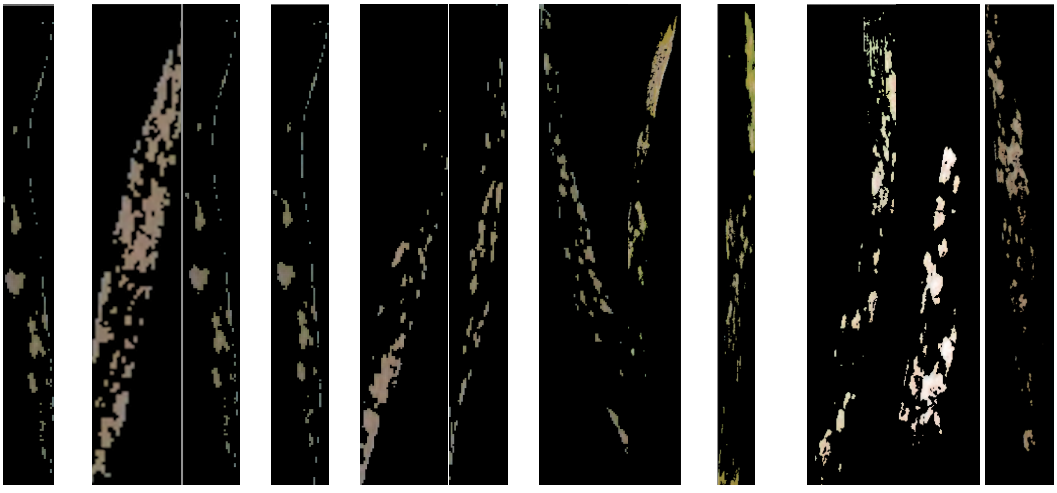


Figure 5.6: Leaf segmentation results of Category 4

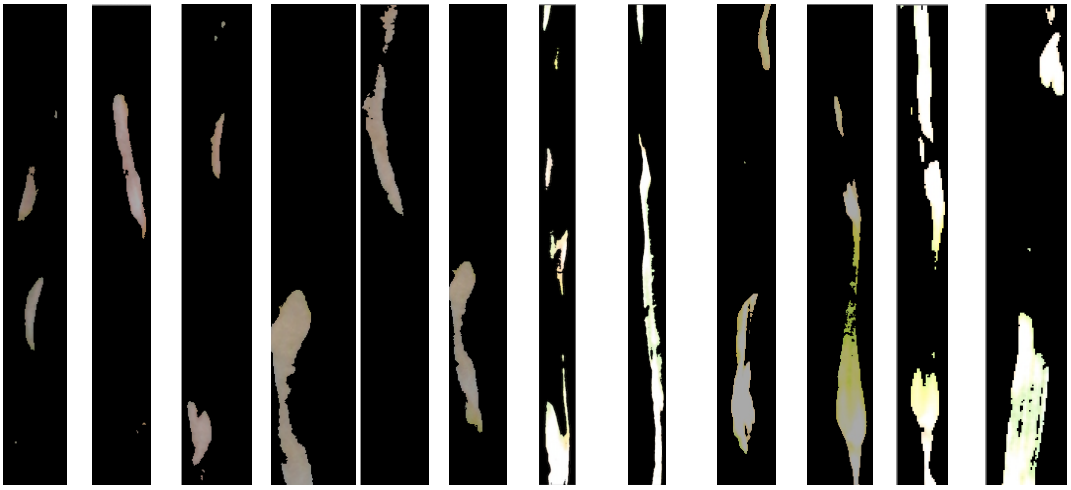


Figure 5.7: Leaf segmentation results of Category 4

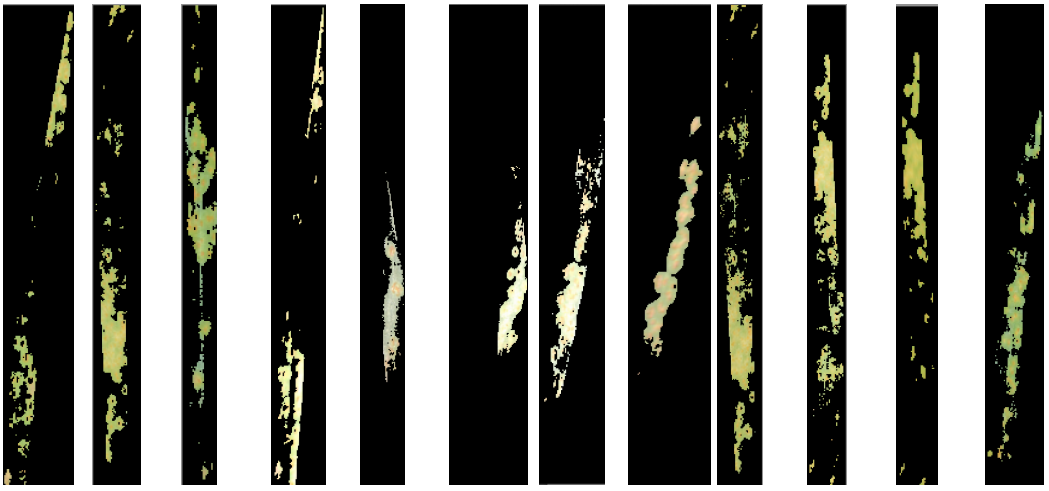
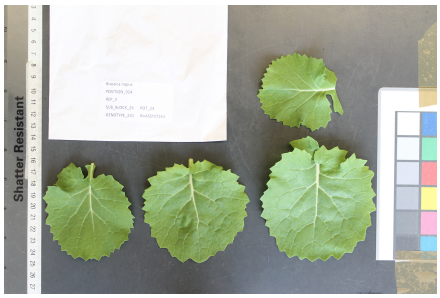
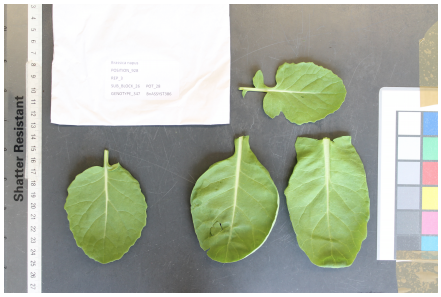


Figure 5.8: Leaf segmentation results of Category 4

(a) Category 1



(b) Category 2



(c) Category 3

(d) Category 4

Figure 5.9: Some tesing leaves in other categories

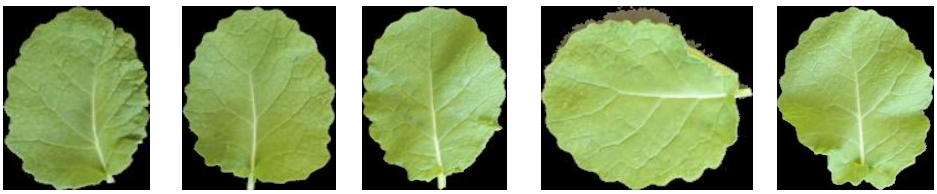


Figure 5.10: Leaf segmentation results of Category 1

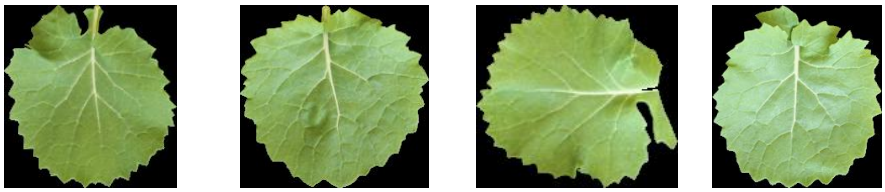


Figure 5.11: Leaf segmentation results of Category 2

Figure 5.12: Leaf segmentation results of Category 3

Figure 5.13: Leaf segmentation results of Category 4

However the leaves in one category of them shown in Figure 5.6 has a little problems on precise segmentation. The segmentation results are shown in Figure 5.7. If you look at the results, you can find that it has a little defect which was not segmented out. Because the shadow part in the background are too similar to the leaf, so for this situation, the leaf still can be segmented out, but with some defect. So it is better to ask the user to take picture with different background colour compared with leaves.

Figure 5.14: Some tesing leaves in Category 5

Figure 5.15: Leaf segmentation results of Category 4

Overall, the test result shows that the segmentation implemented in this project are suitable for other kinds of leaves. Although it cannot be said that the segmentation would work perfectly for all kinds of leaves in any condition, it is generalised for most kinds of green leaves in not very complex background.

5.2 Classification Test and Evaluation on different learinig algorithms

In the previous chapter, we use different learning methods to train and classify the training data, so we need to compare and evaluate which one is better. Note features extracted from whole leaf and from segmented disease spot have been trained respectively, we need to find out of which recognition accuracy is better. Before starting to evaluate them, it is necessary to introduce some evaluation concepts and criteria used in this part.

5.2.1 Evaluation Concepts

In order to evaluate the classification results on different learning algorithms, we need some measures and standards to judge the accuracy of the results. In this section, we will talk about some simple evaluation approach to help us choose the best classification method.

Cross Validation

Cross validation is one of the most widely used method to improve the generalisation of the results and avoid overfitting. Usually a learning algorithm is trained for a set of data with corresponding target until it got a satisfied result to predict output for other input data. However, if the learning is performed too long, it will be particularly adjust to the some points or features which are highly correlative with the target function. The training error will decrease continuously, but the validation error for new input may start to increase. Thus the learning will be overfitting and lose the generalisation to other new input. To solve this problem, cross validation is often used. N-fold Cross validation is a statistical practise of portioning the data into N different subsets such that initially a single subset is used to be the validation set which aims to estimate the generalisation error and the other subset are used to be the training set together for adjusting the parameters of the classifiers. After that the subsequent subset will be used as validation set and the others to be the training set and so on. The process shows in Figure 5.8.

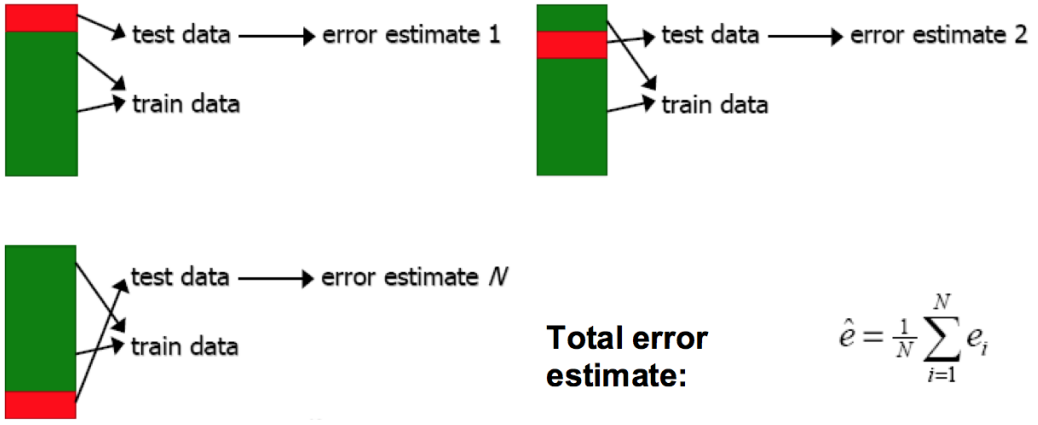


Figure 5.16: N-Fold Cross Validation

Confusion Matrix

Confusion matrix can be a useful validation tool to show the linear result of a classifier on a given input. Each row of the matrix represents the actual class the input data should be and each column represents the output predicted by the classifier. One benefit of it is that it can clearly show the confusing classification between two classes by a classifier. For instance, the Table 5.1 shows the example of the confusion matrix of the recognition between apple, orange and banana. There are 4 actual apples, and three of them are predicted as oranges. Three of 7 actual oranges are predicted as apples. For the banana, the result is very good that only one of 12 actuarial banana are mis-predicted to the orange. The confusion matrix shows the classifier has troubles on differentiating apples and oranges, however works on banana very well. In addition the diagonal can show the accurate predictions.

	Apple	Orange	Banana
Apple	6	4	0
Orange	3	4	1
Banana	0	1	11

Table 5.1: A example of Confusion Matrix

Recall and Precision Rates

Consider a training problem on a binary classification, there are only positive and negative examples with a binary classifier. True Positive (TP) / False Negative (FN) means the examples that are positive assigned to positive (true classification) / negative (false classification). Similarly, True Negative (TN) / False Positive (FP) are the positive examples assigned to negative (true classification) / positive (false classification). In this case, the most simple and obvious way to measure the performance is calculating the correction prediction rate which are the sum of TP and TN divided by the sum of FP and FN, whereas sometimes it may be a misleading measure when comparing which classifier is better. As the tables shown below in Table 5.2, classifier B is obviously better than classifier A. However the performance measure method has a problem for Table 5.3 Although the recognition of classifier B is much better than A, it may not be able to recognise positive data due to the misclassification of all the actual positives, hence we cannot say that B is better than A. At that time some other measures can be introduced to solve that problem.

Classifier	TP	TN	FP	FN	Error Rate
A	50	50	50	50	50%
B	74	74	26	26	26%

Table 5.2: classification scenario 2

Classifier	TP	TN	FP	FN	Error Rate
A	50	150	150	50	50%
B	0	300	0	100	25%

Table 5.3: classification scenario 1

Recall and precision are methods to measure the quality of the information retrieval process. Recall describes the completeness of the process by calculating the number of retrieved positive data divided by the number of the total positive data including the ones not retrieved. Precision describes that the accuracy of the retrieval via calculating the number of the correctly retrieved positive data divided by the number of all the correctly retrieved data including the correctly retrieved negative data. They can be used to measure the performance of the classifier.

F_α-measure

Although the recall and precision can be individually used, $F_\alpha$is a combination of these two values to measure the quality of the classifier based the which value the user are more interested in. The F_α formula is shown below in Equation 5.1.For our project, we use F_1 measures which means α is set to one shown in Equation 5.2.

$$F_\alpha = (1+\alpha)\frac{recall * precision}{\alpha * precision + recall}; \tag{5.1}$$

$$F_1 = 2\frac{recall * precision}{precision + recall}; \tag{5.2}$$

5.2.2 Decision Trees

Leaf Features Trainning

Next we will use the evaluation concepts introduced in las section to measure to performance of different learning methods. First we should measure the decision trees algorithm. Decision trees are trained on leaves features and disease spot features respectively with 10-fold cross validation and the confusion matrix of leaves features is shown is Figure 5.9. It shows that the recognition rate are 84.4% and there are some confusions between Powdery Mildew and Septoria, where 23.2% are misclassified on the Powdery Mildew and 29.0% are misclassifies on Septoria, while other diseases are predicted pretty well. The average recall, precision and F_1 Measure are shown in Table 5.4. The average precision of Powdery Mildew and average recall of Septoria are relatively bad which are less than 80%. The F_1 measures of these are also obviously worse than the other two.

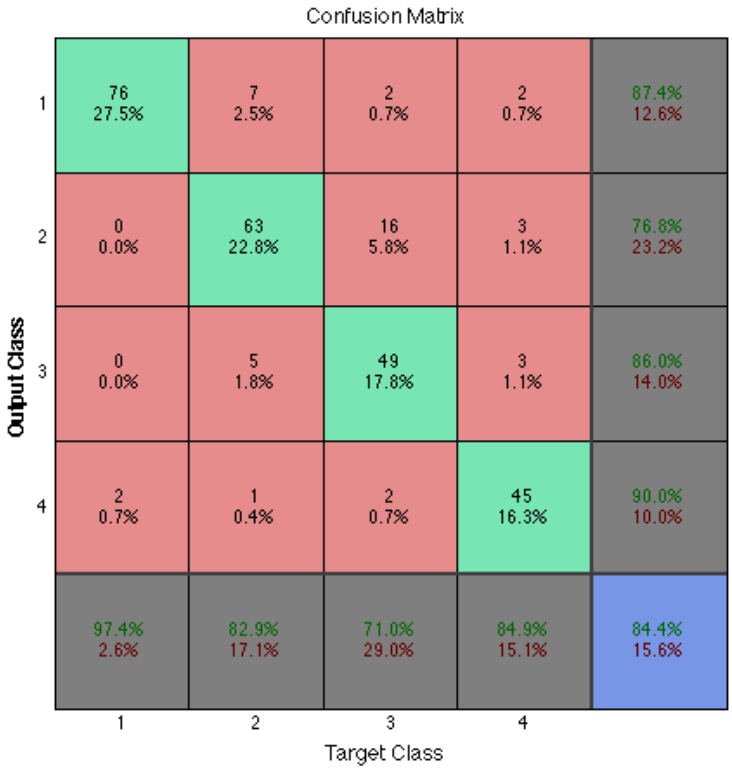


Figure 5.17: Confusion matrices of decision trees on leaf features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.9750	0.8321	0.7048	0.8467
Average precision	0.8954	0.7915	0.8964	0.9124
Average F_1 Measure	0.9289	0.8003	0.7551	0.8669

Table 5.4: Decision Trees Evaluation on leaf

Disease Spot Features Training

We will look at the evaluation of decision trees training on disease spot features. The confusion matrix of the cross validation result are shown in Figure 5.10 below. The overall recognition rate rate are 85.0% which is similar to the results of decision trees training on leaf features. The confusion matrix shows the training has problem in distinguishing among the three diseases but classify natural leaf without disease very well. The average recall, precision and F_1 Measure are shown in Table 5.5. The average precision of Powdery Mildew and average recall of Septoria are relatively bad which are less than 82%. The F_1 measures of these are also obviously worse than the other two. This experiment result is similar to the leaf features training with decision tree above.

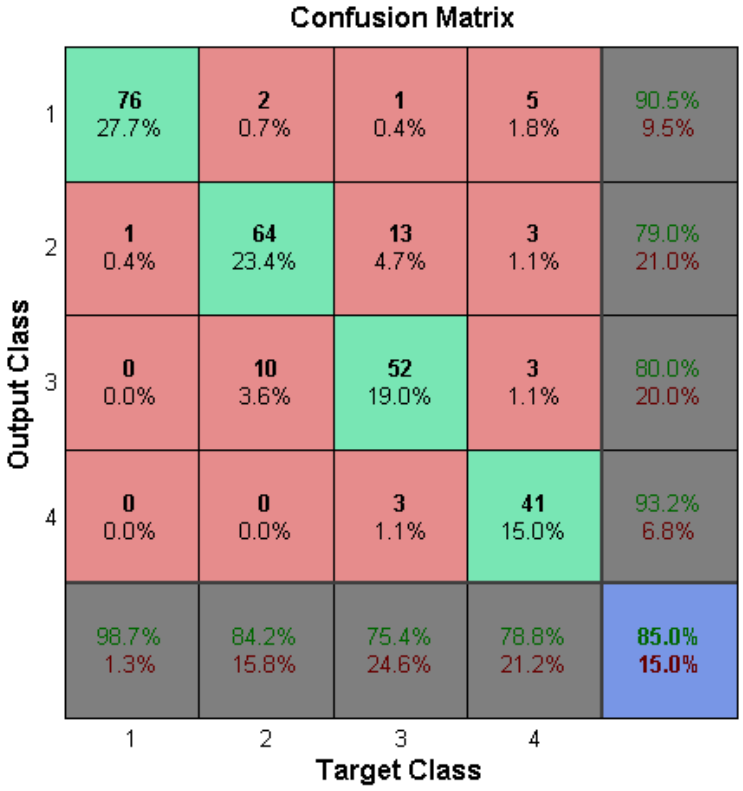


Figure 5.18: Confusion matrices of decision trees on disease features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.9875	0.8411	0.7548	0.7933
Average precision	0.9172	0.8010	0.8161	0.9417
Average F_1 Measure	0.9482	0.8154	0.7775	0.8449

Table 5.5: Decision Trees Evaluation on disease spot

5.2.3 Artificial Neural Network

Leaf Features Trainning

The results blew shows the evaluation of artificial neural network training on leaf features. The confusion matrix of the cross validation result are shown in Figure 5.11 below. The overall recognition rate are 87.3% and confusion matrix shows the distinguishing ability of the classifier is nearly balanced without outstanding performance for every category. The average recall, precision and F_1 Measure are shown in Table 5.6. The average precision of Powdery Mildew and Septoria are relatively not very good which are less than 80%, but their recall is better. The F_1 measures of three kinds of disease are not very good around 80%, the result of disease free leaves are much better around 90%.

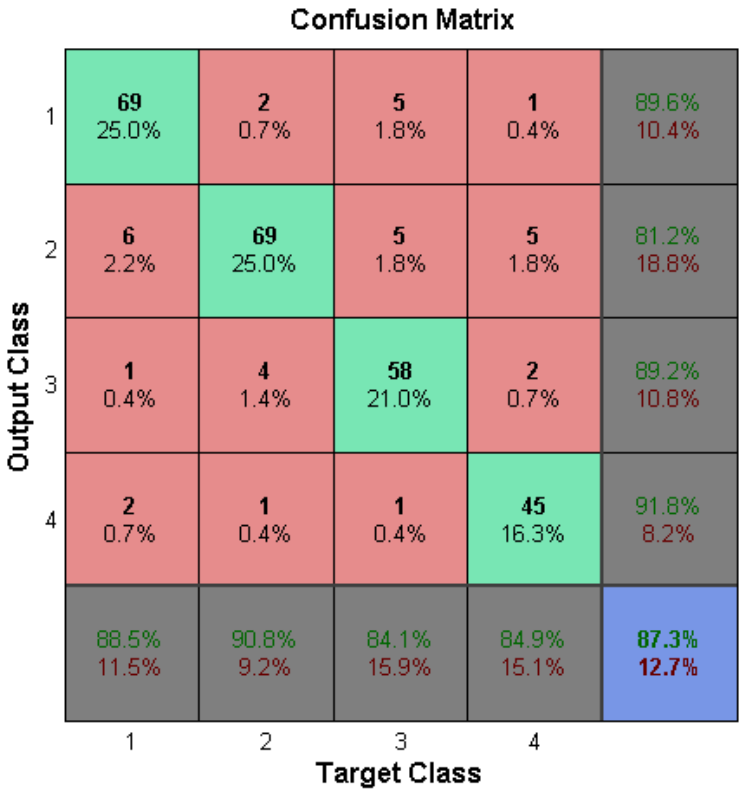


Figure 5.19: Confusion matrices of Artificial Neural Network on leaf features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.9750	0.7857	0.7976	0.8400
Average precision	0.8547	0.8335	0.8839	0.8238
Average F_1 Measure	0.9027	0.7900	0.8173	0.8146

Table 5.6: Artificial Neural Network Average Evaluation on leaf

Disease Spot Features Training

The results blew shows the evaluation of artificial neural network training on disease spot features. The confusion matrix of the cross validation result are shown in Figure 5.12 below. The overall recognition rate are 78.1% which are much worse than the results of artificial neural network training on leaf features. The confusion matrix shows the distinguishing ability of the classifier is nearly balanced without outstanding performance for every category except the disease free is better. The average recall, precision and F_1 Measure are shown in Table 5.7. Both of the average precision and recall of Powdery Mildew and Septoria are relatively bad which are less than 80%. The F_1 measures of three kinds of disease are bad around 70%. Overall the experiment result on disease spot features is much worse than the leaf features training with Artificial Neural Network above.

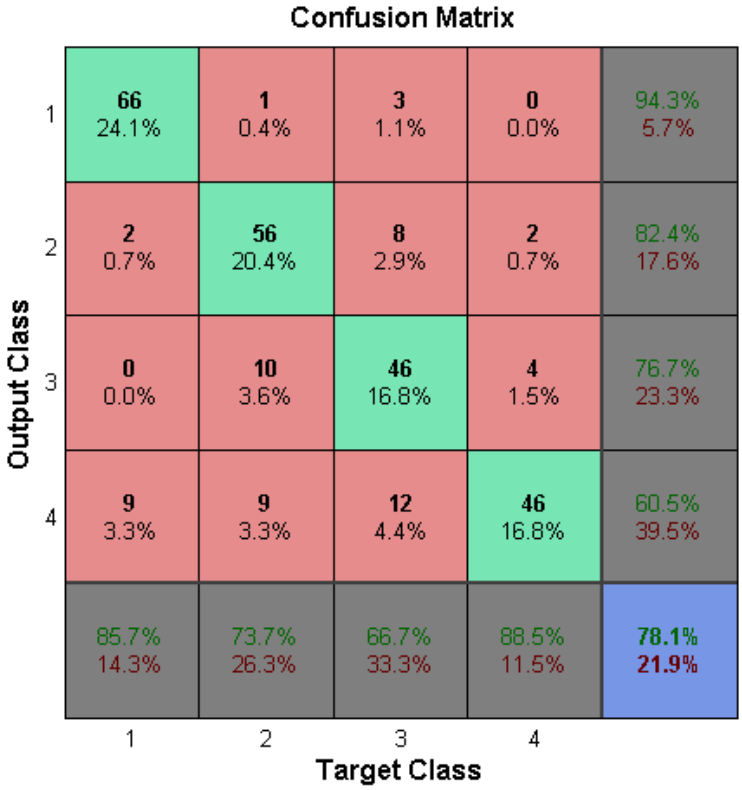


Figure 5.20: Confusion matrices of Artificial Neural Network on disease spot features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.8464	0.7411	0.6667	0.8867
Average precision	0.8578	0.7606	0.7261	0.7938
Average F_1 Measure	0.8492	0.7353	0.6768	0.8007

Table 5.7: Artificial Neural Network Average Evaluation on disease spot

5.2.4 Support Vector Machine

Leaf Features Trainning

The results blew shows the evaluation of Support Vector Machine training on leaf features. The confusion matrix of the cross validation result are shown in Figure 5.13 below. The overall recognition rate are 94.2% and confusion matrix shows the distinguishing ability of the classifier is nearly balanced with outstanding performance for every category. The average recall, precision and F_1 Measure are shown in Table 5.8. The average precision of Powdery Mildew and Septoria are relatively not low which are around 80%, and average recall of disease free leaves and ones with Wheat rust are relatively low around 93%. The F_1 measures of all these four class are good around 94%, overall the performance of the method is very good which is better than the previous methods.

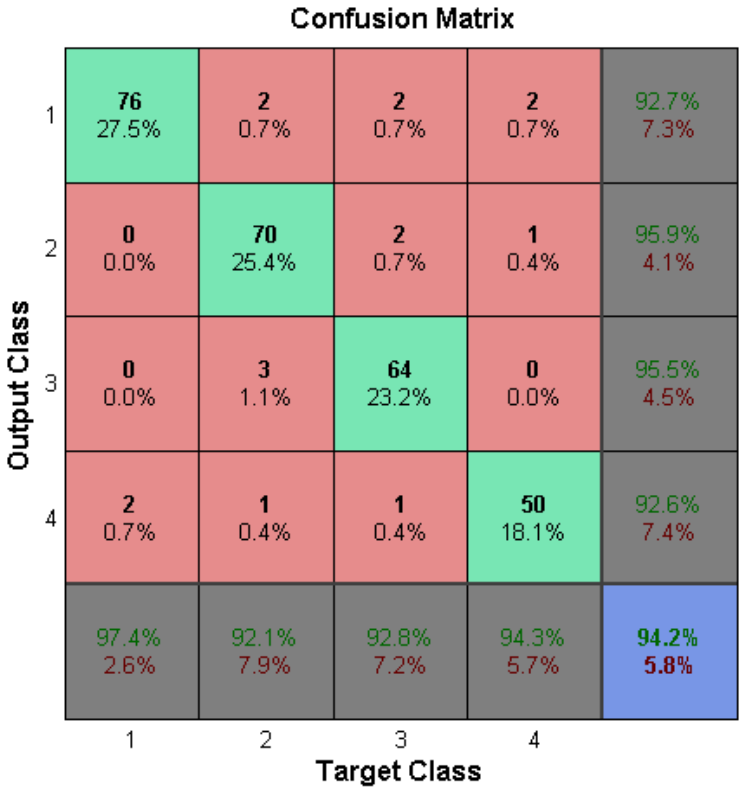


Figure 5.21: Confusion matrices of Support Vector Machine on leaf features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.9750	0.9232	0.9286	0.9400
Average precision	0.9328	0.9621	0.9653	0.9357
Average F_1 Measure	0.9513	0.9389	0.9411	0.9317

Table 5.8: Support Vector Machine Average Evaluation on leaf

Disease Spot Features Training

The results blew shows the evaluation of Support Vector Machine on disease spot features. The confusion matrix of the cross validation result are shown in Figure 5.14 below. The overall recognition rate are 90.9% which are a little worse than the results of Support Vector Machine training on leaf features. The confusion matrix shows the distinguishing problem between Powdery Mildew and Septoria, whose misclassification rate are obviously higher than the other two. The average recall, precision and F_1 Measure are shown in Table 5.8. Both of the average precision and recall of Powdery Mildew and Septoria are relatively worse which are less than 90% but still good. The F_1 measures of Powdery Mildew and Septoria are relatively worse than the other two around 85%. Overall the experiment result on disease spot features is a little worse than the leaf features training with Support Vector Machine above.

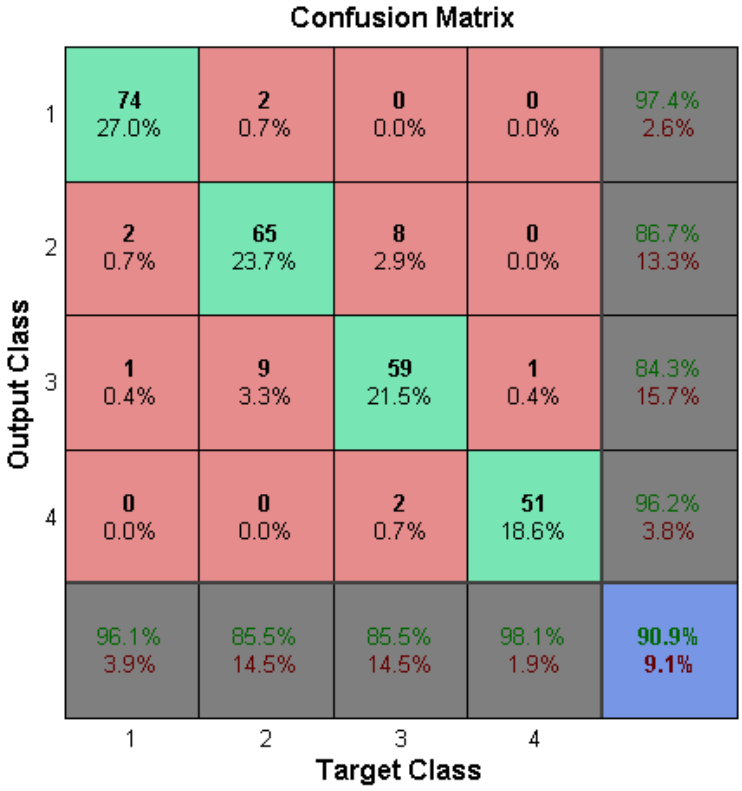


Figure 5.22: Confusion matrices of Support Vector Machine on disease spot features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.9625	0.8554	0.8571	0.9800
Average precision	0.9750	0.8732	0.8488	0.9633
Average F_1 Measure	0.9683	0.8607	0.8472	0.9709

Table 5.9: Support Vector Machine Average Evaluation on disease

5.2.5 Random Forest

Leaf Features Trainning

The results blew shows the evaluation of Random Forest training on leaf features. The confusion matrix of the cross validation result are shown in Figure 5.15 below. The overall recognition rate are 96.4% which is the best result of all the others. The confusion matrix shows the distinguishing ability of the classifier is nearly balanced with extremely outstanding performance for every category. The average recall, precision and F_1 Measure are shown in Table 5.10. The average precision and recall of all of these categories are over 90% and even over 95%. The F_1 measures of all these four class are extremely outstanding over 94% and average 96%, overall the performance of the method is extremely good which is the best among all the methods.

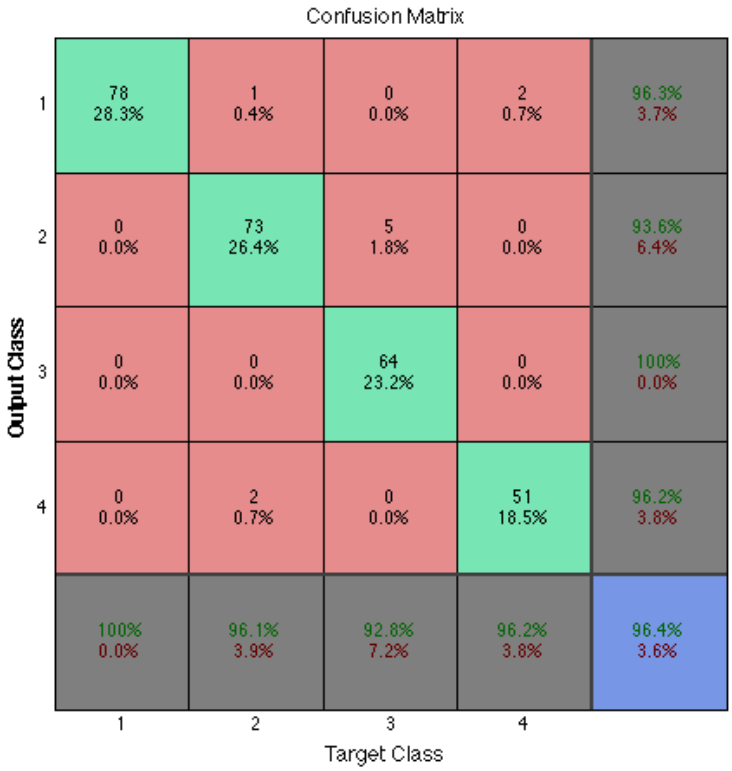


Figure 5.23: Confusion matrices of Random Forest on leaf features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	1.0000	0.9589	0.9286	0.9600
Average precision	0.9675	0.9453	1.0000	0.9714
Average F_1 Measure	0.9822	0.9484	0.9603	0.9624

Table 5.10: Random Forest Average Evaluation on leaf

Disease Spot Features Training

The results blew shows the evaluation of Random Forest on disease spot features. The confusion matrix of the cross validation result are shown in Figure 5.16 below. The overall recognition rate are 88.3% which are worse than the results of Random Forest training on leaf features. The confusion matrix shows the distinguishing problem between Powdery Mildew and Septoria, whose misclassification rate are obviously higher than the other two. The average recall, precision and F_1 Measure are shown in Table 5.11. Both of the average precision and recall of Powdery Mildew and Septoria are relatively worse which are less than 90% and some less than 80%. The F_1 measures of Powdery Mildew and Septoria are relatively worse than the other two around 80%. Overall the experiment result on disease spot features is a worse than the leaf features training with Random Forest above.

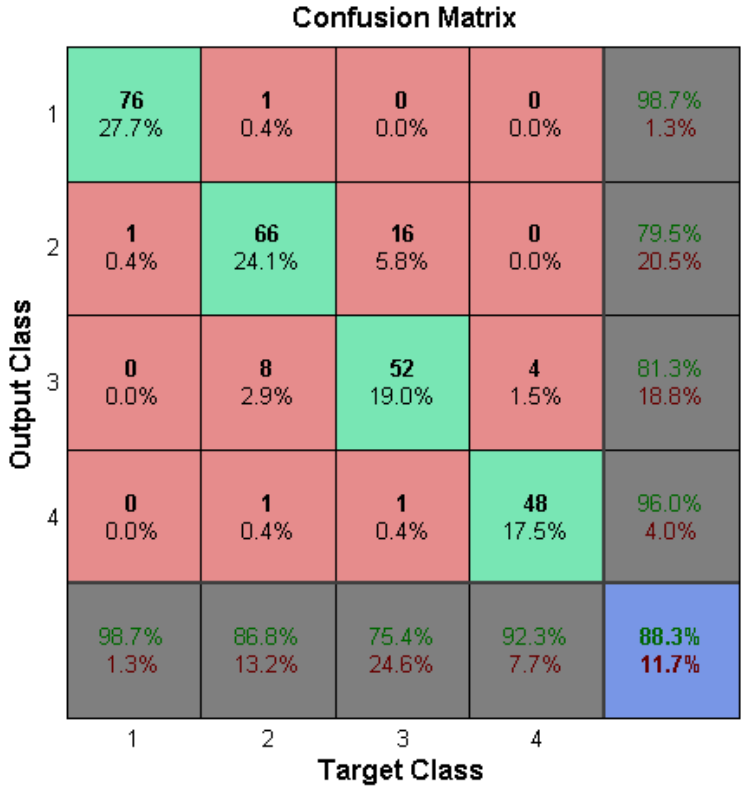


Figure 5.24: Confusion matrices of Random Forest on disease spot features

	Disease Free#1	Powdery Mildew #2	Septoria#3	Wheat Rust#4
Average recall	0.9857	0.8696	0.7571	0.9233
Average precision	0.9889	0.8000	0.8332	0.9633
Average F_1 Measure	0.9864	0.8286	0.7815	0.9396

Table 5.11: Random Forest Average Evaluation on disease spot

5.2.6 Overall Learining Evaluation

Learninig Method	Measure	Disease Free	Powdery Mildew	Septoria	Wheat Rust
DT on leaf	Recall	0.9750	0.8321	0.7048	0.8467
	Precision	0.8954	0.7915	0.8964	0.9124
	F_1 Measure	0.9289	0.8003	0.7551	0.8669
DT on disease	Recall	0.9875	0.8411	0.7548	0.7933
	Precision	0.9172	0.8010	0.8161	0.9417
	F_1 Measure	0.9482	0.8154	0.7775	0.8449
ANN on leaf	Recall	0.9750	0.7857	0.7976	0.8400
	Precision	0.8547	0.8335	0.8839	0.8238
	F_1 Measure	0.9027	0.7900	0.8173	0.8146
ANN on disease	Recall	0.8464	0.7411	0.6667	0.8867
	Precision	0.8578	0.7606	0.7261	0.7938
	F_1 Measure	0.8492	0.7353	0.6768	0.8007
SVM on leaf	Recall	0.9750	0.9232	0.9286	0.9400
	Precision	0.9328	0.9621	0.9653	0.9357
	F_1 Measure	0.9513	0.9389	0.9411	0.9317
SVM on disease	Recall	0.9625	0.8554	0.8571	0.9800
	Precision	0.9750	0.8732	0.8488	0.9633
	F_1 Measure	0.9683	0.8607	0.8472	0.9709
RF on leaf zq	Recall	1.0000	0.9589	0.9286	0.9600
	Precision	0.9675	0.9453	1.0000	0.9714
	F_1 Measure	0.9822	0.9484	0.9603	0.9624
RF on diseaes	Recall	0.9857	0.8696	0.7571	0.9233
	Precision	0.9889	0.8000	0.8332	0.9633
	F_1 Measure	0.9864	0.8286	0.7815	0.9396

Table 5.12

Chapter 6

Conclusion

This chapter summarises the features and evaluation of the software that is overall successful on the project specification. System limitation will be talked about to show what more functions could be added and which part of the application can be optimised. After that I will personally reflect myself on this project in the personal reflection part. Finally future work will be discussed on the possible extension of the project if further development need to be done.

6.1 Summary

The development of the mobile application has met its specification successfully. For the given leaves with three categories of disease, the application can recognise all them. The segmentation program can successfully segment the leaves and diseases spot out with few defects. The machine learning methods based on the extracted features work very well. The cross validation was done which aims to prove the generalisation of the learning method. Both the leaf features and disease spot features are trained via four different kinds of popular machine learning methods, and the best result are got from leaf features training with Random Forest whose average error is 96.4%. The high cross validation results of recognition shows the application is generalised for the other untrained leaves with these tree kinds of diseases. The result can demonstrate that the both features extracted and the classification method are appropriate for this application. Furthermore. the result will be much higher if more training samples are given in the future. In addition, the implementation of the java software are also tested with JUnit Test and all of them passed. The operation is easy for user to start to use the application. The user evaluation results show the high user-friendly.

Overall the project is successful for identifying the plants disease based on its leaf appearance. It can be a central prototype to be used by the agricultural industry to improve the product of crops or by gardeners to recognise the disease and avoid it next time.

6.2 System Limitation

Although the system has many positive aspects and successfully achieved all the goals on the specification, it could still be enhanced to be a publishable application. In terms of the segmentation, the leaves cannot be segmented into very good result if the background is too complex which is nearly a unavoidable problem. However, the application can add a new function to allow the user to cut their interest part out, then the result should be acceptable. The application can also make the user input the EM segmentation parameter if they would like to. For example, if the use choose to partition the image into two sub-images, but the result is not very good, then user can try partition it into three. For the feature extraction, more complicated feature can be tried in experiments. It is a complicated part, many papers discuss about that in different complex method. However, implementing all the candidate algorithms is no practical given a limited amount of time. The system only trained three categories disease of the same plant with limited amount of images which can not lead generally used in reality. But the project is just a research in this research field, the recognition could be more generalised and better if more images are given and trained in the future.

6.3 Personal Reflection

On one hand, it is challengeable to implement such an mobile application to identify the disease in such complex background, particular for me without image processing background before. Since no same application exists in software store and many researchers are studying this topic, it is impossible for a undergraduate student to build a perfect application about that. However, I worked hard and tried my best to learn and research some image processing and machine learning technique so that the application was implemented which achieved the goals in the specification. On the other hand, with few requirement and constraints, the goal are clear and the system design is flexible.

During the process of doing the project, I strictly followed the plan to complete each task weekly except that I overestimate my speed of finishing the dissertation. In my original plan for the last two month, the penultimate one is planed for writing the dissertation and the last is for optimisation the system with adding some more functions. However, it is obvious a miscalculation since I spent the whole last two months on writing the dissertation and so I didnt have enough time to optimise application. Another mistake is that I didnt record each part during the implementation so that I have to recall each part with much difficulty and even do the same things again for adding some resource in the dissertation.

The project brought valuable experience, so that I learnt lots of things which have a profound influence to me, both on research and programming skills. Before doing the project, I have no knowledge on computer vision and machine learning. The process makes me very interested on the machine learning and probably choose to study this field in my postgraduate course. I also learnt how to use MatLab to process image and data, and knows how to use the statistics and machine learning functions built in the MatLab. It also taught me task planning and time man-

agement which would be much helpful in my future research career. My reading and self-learning skills are improved by reading academic papers and writing skills are enhanced by writing the dissertation. Through the project, self-motivation and perseverance are also mixed with my character.

6.4 Future Work

Many extensional functions can be added to the application which could make it being a real application in a Google Store or App Store. Other types of software such as website could also be implemented to make the program more conveniently to attain. The application should allow the user to cut the leaf region out if the background in too big or too complex, which would be very help to segment the leaf out successfully. More other advanced features can be implemented and tested to optimise the algorithm to get a better result. Other categories of leaves with different diseases should be trained via experiments to make the application applicable to as many categories of plants as possible. The application software can be built in some additional functions such as viewing histories, saving locations and so on.

Bibliography

Agarwal, Gaurav, Belhumeur, Peter, Feiner, Steven, Jacobs, David, Kress, W John, Ramamoorthi, Ravi, Bourg, Norman A, Dixit, Nandan, Ling, Haibin, Mahajan, Dhruv, *et al.* . 2006. First steps toward an electronic field guide for plants. *Taxon*, **55**(3), 597–610.

Belhumeur, Peter N, Chen, Daozheng, Feiner, Steven, Jacobs, David W, Kress, W John, Ling, Haibin, Lopez, Ida, Ramamoorthi, Ravi, Sheorey, Sameer, White, Sean, *et al.* . 2008. Searching the worlds herbaria: A system for visual identification of plant species. *Pages 116–129 of: Computer Vision–ECCV 2008*. Springer.

Bishop, Christopher M, *et al.* . 2006. *Pattern recognition and machine learning*. Vol. 1. springer New York.

Casanova, Dalcimar, Florindo, Joao Batista, Gonçalves, Wesley Nunes, & Bruno, Odemir Martinez. 2012. IFSC/USP at ImageCLEF 2012: Plant Identification Task. *In: CLEF (Online Working Notes/Labs/Workshop)*.

Dhawan, Atam P. 1990. A review on biomedical image processing and future trends. *Computer Methods and Programs in Biomedicine*, **31**(3), 141–183.

Duda, Richard O, Hart, Peter E, & Stork, David G. 2012. *Pattern classification*. John Wiley & Sons.

Kumar, Neeraj, Belhumeur, Peter N, Biswas, Arijit, Jacobs, David W, Kress, W John, Lopez, Ida C, & Soares, João VB. 2012. Leafsnap: A computer vision system for automatic plant species identification. *Pages 502–516 of: Computer Vision–ECCV 2012*. Springer.

Neto, Joao Camargo, Meyer, George E, & Jones, David D. 2006. Individual leaf extractions from young canopy images using Gustafson–Kessel clustering and a genetic algorithm. *Computers and electronics in agriculture*, **51**(1), 66–85.

Parker, Jim R. 2010. *Algorithms for image processing and computer vision*. John Wiley & Sons.

Prince, Simon JD, & Prince, Simon Jeremy Damion. 2012. *Computer vision: models, learning, and inference*. Cambridge University Press.

Pydipati, R, Burks, TF, & Lee, WS. 2006. Identification of citrus disease using color texture features and discriminant analysis. *Computers and electronics in agriculture*, **52**(1), 49–59.

Sanyal, P, & Patel, SC. 2008. Pattern recognition method to detect two diseases in rice plants. *The Imaging Science Journal*, **56**(6), 319–325.

Shearer, Scott A, Holmes, RG, *et al.* . 1990. Plant identification using color co-occurrence matrices. *Transactions of the ASAE*, **33**(6), 2037–2044.

Yanikoglu, Berrin A, Aptoula, Erchan, & Tirkaz, Caglar. 2012. Sabanci-Okan System at ImageClef 2012: Combining Features and Classifiers for Plant Identification. *In: CLEF (Online Working Notes/Labs/Workshop).*