MULTI CLASSIFICATION OF ORCHID IMAGE

MR.PIPAT JAIDEE

MR.SARAN LERTRAT

MR.SITTAKARN MAIDEE

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE, INTERNATIONAL PROGRAM

FACULTY OF SCIENCE

KING MONKUT'S INSTITUE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2009

Project Title	Multi Classification of Orchid image	
Personal Details	Mr.Pipat	Jaidee
	Mr.Saran	Lertrat
	Mr.Sittakarn	Maidee
Degree	Bachelor of Science	
Program	Computer Science, Internaltional Program	
Academic Year	2009	
Advisor	Mr. Suntana Oudomying	

Abstract

We implemented two neural network models for classifying 10 species of Dendrobium orchids. We compare two models between using grayscales technique and wavelet transform technique for training our neural network. To provide color information to our networks, we supply it with R-G-B of the picture. Our result showed that the techniques worked well on querying image.

Acknowledgement

We would like to express my gratitude to all those who gave us the possibility to complete this project.

The authors wish to express our deep gratitude to Mr. Suntana Oudomying our advisor, for this valuable guidance, attention, and encouragement throughout this research.

We are greatly appreciated all the professors who have given invaluable knowledge while we were studying in the department of computer, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang.

We are also would like to give the special thanks to master of telecommunication engineering students and all of our friends at the department who had encouraged us.

Mr.Pipat Jaidee

Mr.Saran Lertrat

Mr.Sittakarn Maidee

Table of Contents

Chapter 1

Introduction

1.1 Rational

Researchers have been proposing techniques for query multimedia data. It is now common to tag data for being searched. Tagging, in fact, is a text matching between tagged terms and search keywords. Ideally, researchers wish that computers are able to recognize pictures or videos the way human do. Once the recognition ability is accomplished, querying multimedia data would be a simple task as computers would understand that users want. Tagging would be applied to multimedia data only for more abstract information.

Whether we can make computers exactly understand how human recognize multimedia data is trivial here. We rely on machine learning concept to create semantic abstraction of the multimedia representation. Computers see a red apple on a yellow table as pixels of red and yellow. Suppose a computer successfully classifies such images from the others and we express rules that it uses on synthesis features extracted from raw data. Would the rule make sense to human? Yet, again, machine learning is an ideal solution we use today. Researchers have been classifying classes of multimedia data in hope that we, one day, could have a computer to recognize all classes of multimedia data known to human.

Dendrobium orchid is one of the most important cut flower exports of Thailand. Regarding to nurnerous varieties and hybrids come in different breeds. People simply call it an orchid, though they often prefer to be able to call it properly. Not knowing the object name in an image limits users to correctly refer to what they are looking for. The other limitation to today multimedia query is that it is not easy to describe the appearance of an orchid. Thus, we believe multimedia query soon will allow users to query by an image as well text because it will be easier for users to query.

1.2 Objective

We propose an application for classifying Dendrobium orchid using neural network. The model should be able to be extended for other kinds of flowers as well. We expect our result to be able for extending it to tolerate noises such that the application becomes practical.

1.3 The scope of the study

Image classification using neural network always poses a few challenges for example, what kind of features would yield best accuracy rate, what is the best topology, etc. To focus only on the challenges, we limit our study to

1. Orchid in Dendorbium species
2. Use 10 species and 100 different pictures of each species.
3. Picture size greater than 256x256 pixels.
4. in picture full shape of orchid
5. No scaling nor orientation feature is covered.
6. No noise in picture

The remaining content of this document is organized as allow. Chapter 2 will describe about the approach of neural network and image processing literature. Chapter 3 will describe the structure of application. Chapter 4 will show the experiment and result of the experiment. Finally chapter 5 will be the conclusion and the extension of what could be improved from this research.

Chapter 2

Rational and review literature

2.1 Image processing

Image processing is the field of research concerned with the development of computer algorithms working on digitized images. The range of problems studied in image processing is large, encompassing everything from low-level signal enhancement to high-level image understanding. In general, a chain of tasks solves image-processing problems. This chain, shown in the picture image 2-1 processing chain, outlines the possible processing, needed from the initial sensor data to the outcome (e.g. a classification or a scene description). The pipeline consists of the steps of pre-processing, data reduction, segmentation, object recognition and image understanding. In each step, the input and output data can either be images (pixels), measurements in images (features), and decisions made in previous stages of the chain (labels) or even object relation information (graphs).

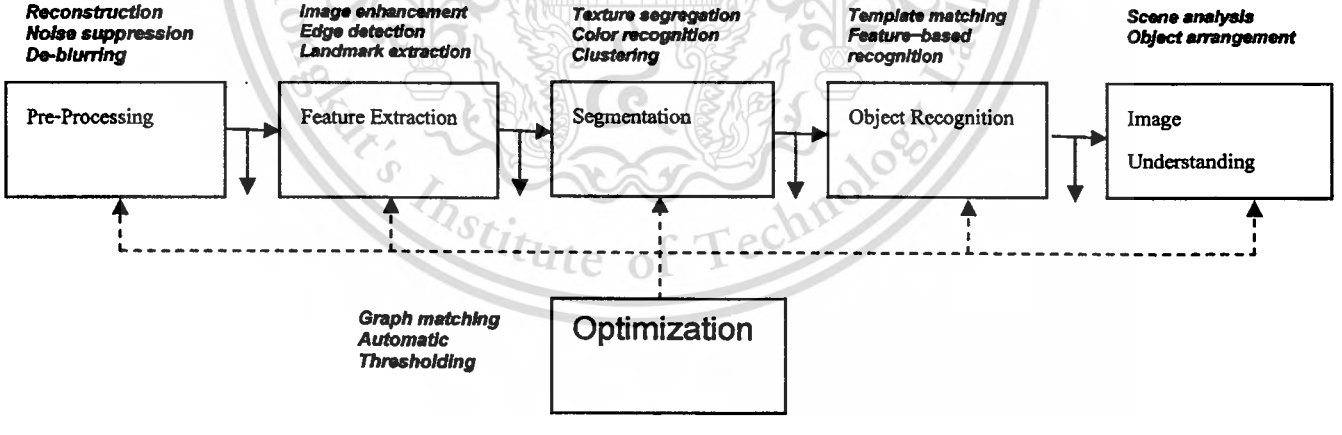


Figure 2-1: The image processing chain

All this leads to the idea that nonlinear algorithms that can be trained, rather than designed, might be valuable tools for image processing. To explain why, a brief introduction into artificial neural networks will be given first.

2.2 Artificial neural networks (ANNs)

An artificial neural network is a system based on the operation of biological neural networks, in other words, is an emulation of biological neural system. Why would be necessary the implementation of artificial neural networks? Although computing these days is truly advanced, there are certain tasks that a program made for a common microprocessor is unable to perform; even so a software implementation of a neural network can be made with their advantages and disadvantages.

Advantages:

- A neural network can perform tasks that a linear program cannot.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- A neural network learns and does not need to be reprogrammed.
- It can be implemented in any application.
- It can be implemented without any problem.

Disadvantages:

- The neural network needs training to operate.
- The architecture of a neural network is different from the architecture of microprocessors therefore need to be emulated.
- Requires high processing time for large neural networks.

Another aspect of the artificial neural networks is that there are different architectures, which consequently require different types of algorithms, but despite to be an apparently complex system, a neural network is relatively simple.

2.3 Definitions and Style

Artificial neural networks (ANN) are among the newest signal-processing technologies in the engineer's toolbox. The field is highly interdisciplinary, but our approach will restrict the view to the engineering perspective. In engineering, neural networks serve two important functions: as pattern

classifiers and as nonlinear adaptive filters. We will provide a brief overview of the theory, learning rules, and applications of the most important neural network models. Definitions and Style of Computation an Artificial Neural Network is an adaptive, most often nonlinear system that learns to perform a function (an input/output map) from data. Adaptive means that the system parameters are changed during operation, normally called the training phase. After the training phase the Artificial Neural Network parameters are fixed and the system is deployed to solve the problem at hand (the testing phase).

The Artificial Neural Network is built with a systematic step-by-step procedure to optimize a performance criterion or to follow some implicit internal constraint, which is commonly referred to as the learning rule. The input/output training data are fundamental in neural network technology, because they convey the necessary information to "discover" the optimal operating point. The nonlinear nature of the neural network processing elements (PEs) provides the system with lots of flexibility to achieve practically any desired input/output map.

In artificial neural networks, the designer chooses the network topology, the performance function, the learning rule, and the criterion to stop the training phase, but the system automatically adjusts the parameters. So, it is difficult to bring a priori information into the design, and when the system does not work properly it is also hard to incrementally refine the solution. But ANN-based solutions are extremely efficient in terms of development time and resources, and in many difficult problems artificial neural networks provide performance that is difficult to match with other technologies.

2.4 Applications of ANNs

Image processing literature contain s■umerous applications of the above types of ANNs and various other, more specialized models. Below, we will give a broad overview of these applications, without going into specific ones. Furthermore, we will only discuss application of ANNs directly to pixel data.

2.4.1 Pre-processing

An image can consist of image reconstruction (building up an image from a number of indirect sensor measurements) and/or image restoration (removing aberrations introduced by the sensor, including noise). To perform pre-processing, ANNs have been applied in the following ways:

- Optimization of an objective function specified by a traditional preprocessing approach
- Approximation of a mathematical transformation used in reconstruction, by regression
- General regression/classification, usually directly on pixel data (neighborhood input, pixel output).

For image reconstruction, regression (feed-forward) ANNs can be applied. Although some successful applications are reported in literature, it would seem that these applications call for more traditional mathematical techniques, because a guaranteed performance of the reconstruction algorithm is essential.

Regression or classification ANNs can also be trained to perform image restoration directly on pixel data. In literature, for a large number of applications, non-adaptive ANNs were used. Where ANNs are adaptive, their architectures usually differ much from those of the standard ANNs: prior knowledge about the problem is used to design them. This indicates that the fast, parallel operation of ANNs, and the ease with which they can be embedded in hardware, can be important factors in choosing for a neural implementation of a certain pre-processing operation. However, their ability to learn from data is apparently of less importance.

2.4.2 Enhancement and feature extraction

After pre-processing, the next step in the image processing chain is extraction of information relevant to later stages. In its most generic form, this step can extract low-level information such as edges, texture characteristics etc. This kind of extraction is also called image enhancement, as certain general (perceptual) features are enhanced. As enhancement algorithms operate without a specific application in mind, the goal of using ANNs is to

outperform traditional methods either in accuracy or computational speed. The most well known enhancement problem is edge detection, which can be approached using classification feed-forward ANNs. Some modular approaches, including estimation of edge strength or de-noising, have been proposed. Morphological operations have also been implemented on ANNs, which were equipped with shunting mechanisms (neurons acting as switches).

Feature extraction entails finding more application-specific geometric or perceptual features, such as corners, junctions and object boundaries. For particular applications, even more high-level features may have to be extracted, e.g. eyes and lips for face recognition. Feature extraction is usually tightly coupled with classification or regression; what variables are informative depends on the application, e.g. object recognition. Some ANN approaches therefore consist of two stages, possibly coupled, in which the first ANN extracts features and the second ANN performs object recognition. If the two are completely integrated, it can be hard to label a specific part as a feature extractor

2.4.3 Segmentation

Segmentation is partitioning an image into parts that are coherent according to some criterion: texture, color or shape. When considered as a classification task, the purpose of segmentation is to assign labels to individual pixels or vexes. Classification feed-forward ANNs and variants can perform segmentation directly on pixels, when windows extracted around their position represent pixels. More complicated modular approaches are possible as well, with modules specializing in certain subclasses or invariance. Hierarchical models are sometimes used, even built of different ANN types, e.g. using a SOM to map the image data to a smaller number of dimensions and then using a feed-forward ANN to classify the pixel.

Again, a problem here is that ANNs are not naturally invariant to transformations of the image. These transformations will have to be removed beforehand, the training set will have to contain all possible transformations, or invariant features will have to be extracted from the image first. For a more detailed overview of ANNs applied to image segmentation.

2.4.4 Object recognition

Object recognition consists of locating the positions and possibly orientations and scales of instances of classes of objects in an image (object detection) and classifying them (object classification). Problems that fall into this category are e.g. optical character recognition, automatic target recognition and industrial inspection. Object recognition is potentially the most fruitful application area of pixel-based ANNs, as using an ANN approach makes it possible to roll several of the preceding stages (feature extraction, segmentation) into one and train it as a single system.

Many feed-forward-like ANNs have been proposed to solve problems. Again, invariance is a problem, leading to the proposal of several ANN architectures in which connections were restricted or shared corresponding to desired invariance. More involved ANN approaches include hierarchical ANNs, to tackle the problem of rapidly increasing ANN complexity with increasing image size; and multi-resolution ANNs, which include context information.

2.4.5 Image understanding

Image understanding is the final step in the image processing chain, in which the goal is to interpret the image content. Therefore, it couples techniques from segmentation or object recognition with the use of prior knowledge of the expected image content (such as image semantics). As a consequence, there are only few applications of ANNs on pixel data. These are usually complicated, modular approaches.

A major problem when applying ANNs for high level image understanding is their black-box character. Although there are proposals for explanation facilities and rule extraction, it is usually hard to explain why a particular image interpretation is the most likely one, another problem in image understanding relates to the amount of input data. When, e.g., seldom-occurring images are provided as input to a neural classifier, a large number of images are required to establish statistically representative training and test sets.

2.4.6 Optimization

Some image processing (sub) tasks such as stereo matching can best be formulated as optimization problems, which may be solved by Hopfield Neural Network (HNN;

Improvement for Color Image Recognition). HNNs have been applied to optimization problems in reconstruction and restoration, segmentation, (stereo) matching and recognition.

Mainly, HNNs have been applied for tasks that are too difficult to realize with other neural classifiers because the solutions entail partial graph matching or recognition of 3D objects. A disadvantage of HNNs is that training and use are both of high computational complexity.

2.5 ANNs for image processing

As was discussed above, dealing with nonlinearity is still a major problem in image processing. ANNs might be very useful tools for nonlinear image processing:

- Instead of designing an algorithm, one could construct an example data set and an error criterion, and train ANNs to perform the desired input/output mapping;
- The network input can consist of pixels or measurements in images; the output can contain pixels, decisions, labels, etc., as long as these can be coded numerically – no assumptions are made. This means adaptive methods can perform several steps in the image processing chain at once;
- ANNs can be highly nonlinear; the amount of nonlinearity can be influenced by design, but also depends on the training data

2.6 Feed-forward networks in matlab

The feed-forward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

Feed-forward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons (Show in figure 2-2). Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the range –1 to +1.

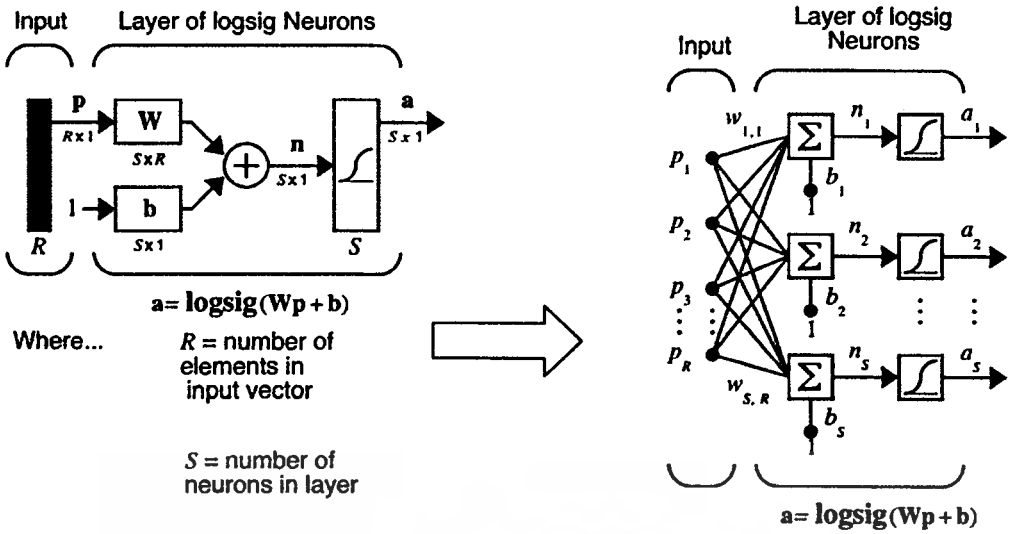


Figure 2-2: A one-layer network with *R*

On the other hand, if you want to constrain the outputs of a network (such as between 0 and 1), then the output layer should use a sigmoid transfer function (such as logsig).

2.6.1 Creating a Network

Create the network object. The function newff creates a feed-forward network. It requires three arguments and returns the network object. The first argument is a matrix of sample R-element input vectors. The second argument is a matrix of sample S-element target vectors. The sample inputs and outputs are used to set up network input and output dimensions and parameters. The third argument is an array containing the sizes of each hidden layer.

Two-layer feed-forward networks can potentially learn virtually any input-output relationship; feed-forward networks with more layers might learn complex relationships more quickly.

The function newcf creates cascade-forward networks. These are similar to feed-forward networks, but include a weight connection from the input to each layer and from each layer to the successive layers.

2.6.2 Simulation

The function sim simulates a network. sim takes the network input p and the network object net and returns the network outputs a. Using sim to simulate the network created above for a single input vector:

p = [1; 2]; a = sim (net, p)

a=-3.6686

Where p is vector if use [,] input value is change to pararell which requite node for net

2.6.3 Training

Once the network weights and biases are initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The training process requires a set of examples of proper network behavior—network inputs p and target outputs t. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function net.performFcn. The default performance function for feed-forward networks is mean square error, mse — the average squared error between the networks outputs and the target outputs. The basic back propagation training algorithm, in which the weights are moved in the direction of the negative gradient.

2.6.4 Faster Training

The two back propagation training algorithms which are gradient descent, and gradient descent with momentum, are often too slow for practical problems. This section discusses several high-performance algorithms that can converge from ten to one hundred times faster than the algorithms discussed previously. All the algorithms in this section operate in batch mode and are invoked using train.

Levenberg-Marquardt (trainlm)

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix (the square matrix of second-order partial derivatives of a function). When the performance function has the form of a sum of squares (as is typical in training feed-forward networks), then the Hessian matrix can be approximated as $H = J^T J$ and the gradient can be computed as $g = J^T e$ Where J is the Jacobian matrix (the matrix of all first-order partial derivatives of a vector-valued function) that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard back propagation technique that is much less complex than computing the Hessian matrix.

2.8 Wavelet Transform

All image formats gone through here use an image transform, quantization and coding. All of these are described for the different formats in question. Transforms mentioned below can be separable extended to two dimensions for applications to image processing. Therefore, results in one dimension only for the sake of simplicity. The application process will describe this separation process later. Value m for block dimension (or more precisely, the number of channels) will be use for our wavelet transforms we will always have m = 2. It will associate a transform with m filters, so that for m = 2, then it will have only two filters. With respect to signal processing, these are to be interpreted as low-pass high-pass filters.

If we denote the signal by x, the block dimension (or number of chanels) describes the size of the block partitioning of the signal. JPEG applies only block transforms, meaning that if the signal is split into blocks x = x[i] (each x[i] a vector of dimension m), the transformed signal y = y[i] is given by

$$y[i] = A*x[i]$$

for some m × m matrix A. This way, the influence from different blocks for instance for pixels on block boundaries is lose. This is what gives rise to the blockades artifact in JPEG.

2.8.1 Two-Dimension Discrete Wavelets

Two-Dimension Discrete Wavelets (dwt2) command performs single-level two-dimensional wavelet decomposition with respect to either a particular wavelet ('*wname*', see wfilters for more information) or particular wavelet decomposition filters (Lo_D and Hi_D) you specify.

$$[cA, cH, CV, cD] = dwt2(X,'wna.ne')$$

Computes the approximation coefficients matrix cA and details coefficients matrices cH, cV, and cD (horizontal, vertical, and diagonal, respectively), obtained by wavelet decomposition of the input matrix X. The '*wname*' string contains the wavelet name.

Another syntax for dw2 is

$$[cA, cH, CV, cD] = dwt2(X, Lo\_D, Hi\_D)$$

Computes the two-dimensional wavelet decomposition as above, based on wavelet decomposition filters that you specify.

- Lo_D is the decomposition low-pass filter.
- Hi_D is the de composition hig h-pass filter.

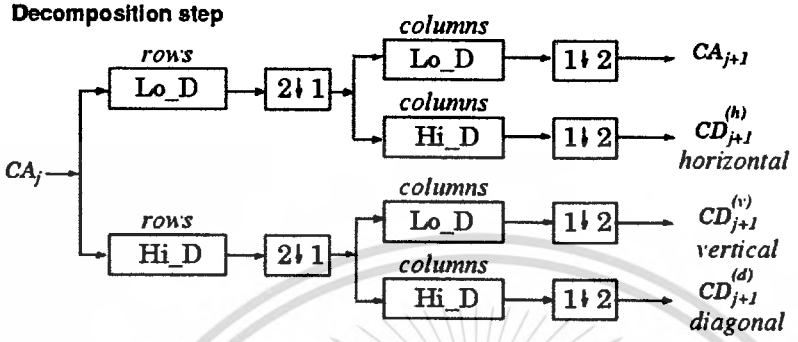


Figure 2-3: Two-dimension DWT

Where $\boxed{2\downarrow 1}$ Downsample columns: keep the even indexed columns

$\boxed{1\downarrow 2}$ Downsample row: keep the even indexed rows

rows
X Convolve with filter X the rows of the entry

column
X Convolve with filter X the columns of the entry

Initialization $CA_0 = s$ for the decomposition initialization

2.7 Discussion

One of the major advantages of ANNs is that they are applicable to a wide variety of problems. There are, however, still caveats and fundamental problems that require attention. Some problems are caused by using a statistical, data-oriented technique to solve image-processing problems; other problems are fundamental to the way ANNs work.

A problem with data-oriented approaches a problem in the application of data-oriented techniques to images is how to incorporate context information and prior knowledge about the expected image content. Prior knowledge could be knowledge about the typical shape of objects one wants to detect, knowledge of the spatial arrangement of textures or objects or of a good approximate solution to an optimization problem. The key to restraining the highly flexible learning algorithms ANNs lies in

with prior knowledge. However, most ANN approaches do not even use the prior information that neighboring pixel values are highly correlated. The latter problem can be circumvented by extracting features from images first, by using distance or error measures on pixel data, which do take spatial coherency into account, or by designing an ANN with spatial coherency or contextual relations between objects in mind. On a higher level, some methods, such as hierarchical object recognition ANNs can provide context information.

In image processing, classification and regression problems quickly involve a very large number of input dimensions, especially when the algorithms are applied directly on pixel data. This is problematic, as ANNs to solve these problems will also grow, which makes them harder to train. However, the most interesting future applications (e.g. volume imaging) promise to deliver even more input. One way to cope with this problem is to develop feature-based pattern recognition approaches; another way would be to design architecture that quickly adaptively down samples the original image.

Finally, there is a clear need for thorough validation of the developed image processing algorithms. Unfortunately, only few of the publications about ANN applications ask the question whether an ANN really is the best way of solving the problem. However, neural networks with their remarkable ability to derive meaning from complicated or imprecise data can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Often, comparison with traditional methods is neglected.

Although the approximation capabilities of ANNs have been proven, a few ANNs problems is mentioned here. For example, although feed-forward ANNs with two hidden layers can approximate any (even discontinuous) function to an arbitrary precision, result convergent, e.g., are lacking. The combination of initial parameters, topology and learning algorithm determines the performance of an ANN after its training has been completed. Furthermore, there is always a danger of overtraining an ANN, as minimizing the error measure occasionally does not correspond to finding a well-generalizing ANN.

Another problem is how to choose the best ANN architecture. Although there is some work on model selection, no general guidelines exists which guarantee the best trade-off between model bias and

variance for a particular size of the training set. Training unconstrained ANNs using standard performance measures such as the mean squared error might even give very unsatisfying results. This could be the reason why in a number of applications, ANNs were not adaptive at all or heavily constrained by their architecture.

ANNs suffer from what is known as the black-box problem: the ANN, once trained, might perform well but offers no explanation on how it works. That is, given any input a corresponding output is produced, but it cannot be easily explained why this decision was reached, how reliable it is, etc. In some image processing applications, e.g., monitoring of (industrial) processes, electronic surveillance, biometrics, etc. a measure of the reliability is highly necessary to prevent costly false alarms. In such areas, it might be preferable to use other, less well performing methods that do give a statistically profound measure of reliability.

The discussions of ANNs to image processing problem are as following;

- The choice of ANN architecture;
- The use of prior knowledge about the problem in constructing both ANNs and training sets;
- The black-box character of ANNs.

Chapter 3

Multi Classification of Orchid image

This chapter will cover designs we have for this project. These designs are application design. In our project, we categorized process of developing our application into 3 parts: System analysis process, system design process and solution. System analysis explains the idea behind this project while system design process explains technical designs and solution explain how we produce model.

3.1 System Analysis

3.1.1 Main Idea

In our project, we use neural network and matlab to create a model. First model we got idea from research: *Image Recognition by Neural Network [1]*, this research use gray scales and region of interest to help neural network recognize face. Second model we got idea from research: *Histograms, Wavelets Neural Networks Applied to Image Retrieval* [2], this research use wavelet transform and rgb picture to help neural network recognize the plane.

We combine 2 main ideas together to build our model. And we want to compare 2 models which one provides a good result. So our models are

- Grayscales Model

Using Mean, S.D from grayscales and maximum value of histogram

- Wavelet transform Model

Using Mean and S.D from wavelet transform and maximum value of histogram

3.2 System Design

3.2.1 Overall diagram

This section explains the overall process of each of our models. There are 2 processes involved to it pre-image and neural network. Pre-image is resizing image, transform and portion image before we feed it to neural network. Neural network is used for generate the output. Figure 3.1 and 3.2 show overall diagram for grayscales model and wavelet model.

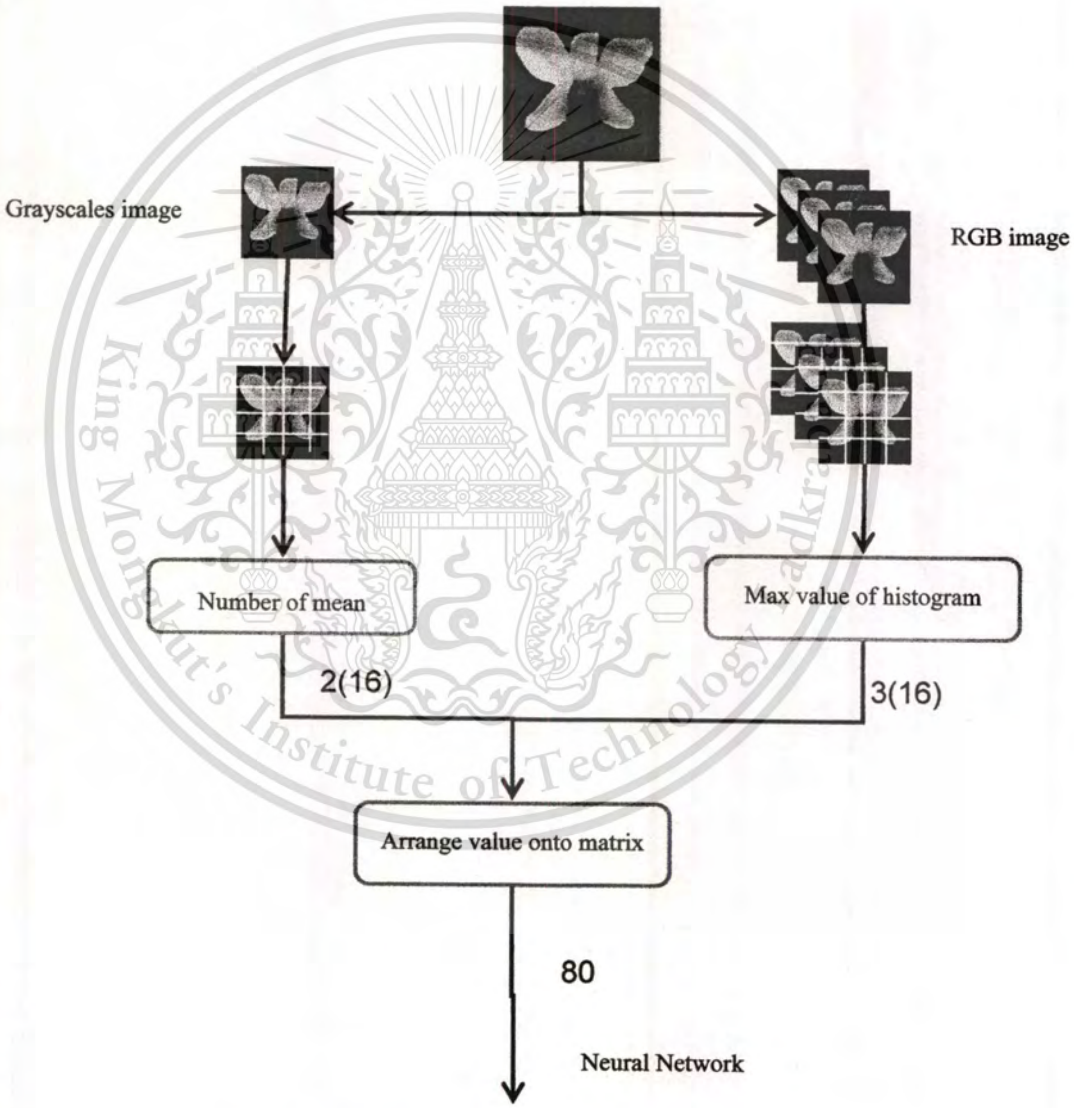


Figure 3.1 show Grayscales model

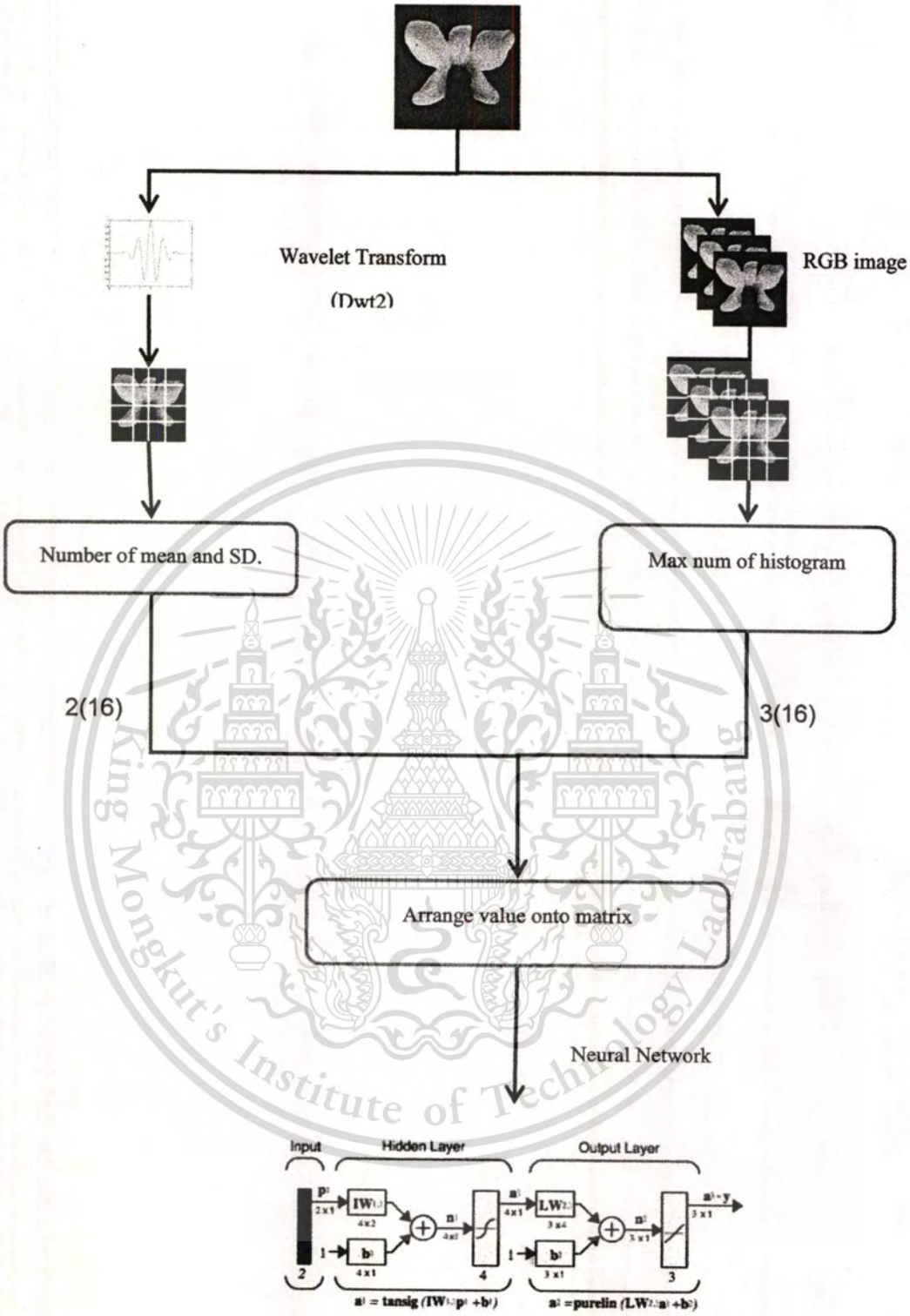


Figure 3.2 show Wavelet transform model

3.3 Solution

3.3.1 Input

We use 256x256 pixel pictures and transform it to grayscale which, a neural network cannot accumulate the amount of input node. To reduce size of it we portion the grayscale picture to 16 blocks (Figure 3.3).The list of 10 dendrobium types use in the experiment is listed in appendix A.

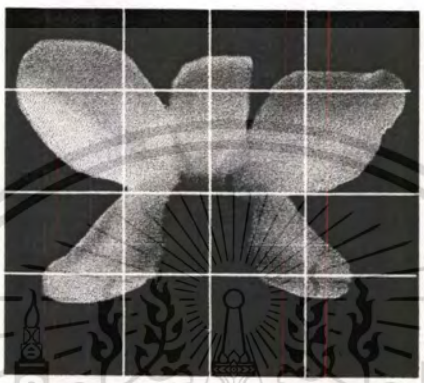


Figure 3.3 portion images to 16 blocks

We find value of Mean and S.D for each block. In a wavelet transform we transform grayscales picture into wavelet before segment it. Since grayscale show the black-white color in picture, we transform normal picture to R-G-B picture for color of orchid (Figure 3.4) to supply color picture.

Figure 3.4 Transform normal picture to R-G-B picture

Then we segment to 16 blocks and find maximum value of histograms. Now we have 32 values from Mean and S.D of picture and 48 values from maximum value of histogram. We arrange it into matrix and feed it to neural network. It have input 80 node, hidden layer is 45 nodes and output is 10 nodes.

3.3.2 Output

Our output we set value between 0 and 1.

- 1 represent correct put
- 0 is represent wrong output

3.4 Model code

In this section, we present the pseudo code of our application in matlab

3.4.1. Resizing image into 256x256 pixels.

```
image=imresize(a,[256 256]);
```

3.4.2. Converting normal image into 2 model Grayscales image and find max histograms in RGB image.

- Grayscales image

```
g= rgb2gray(image);
```

- Wavelet Transform

```
[cAr,cHr,cVr,cDr] = dwt2(ga{i,j,k},'db1');
```

- Find max histograms red,green and blue

```
hist_r{i,j,k}=max(imhist(xa{i,j,k}));
hist_g{i,j,k}=max(imhist(ya{i,j,k}));
hist_b{i,j,k}=max(imhist(za{i,j,k}));
```

3.4.3. Dividing image into 16 segments

```
ga(i,:,:)=mat2cell(g,[64 64 64 64],[64 64 64 64]);
xa(i,:,:)=mat2cell(x,[64 64 64 64],[64 64 64 64]);
ya(i,:,:)=mat2cell(y,[64 64 64 64],[64 64 64 64]);
```

3.4.4. Analyzing means and S.D for one block in grayscale image

- Mean for each block

```
aa{i,j,k}=mean(mean(image{i,j,k}));
```

S.D for each block

```
bb{i,j,k}=std(std(image{i,j,k}));
```

3.4.5. Rearranging any value onto the matrix form

3.4.5.1 Add value into 16 segments

```
abb{i,j,k}=[am{i,j,k};bm{i,j,k};hist_r{i,j,k};hist_g{i,j,k};hist_b{i,j,k}];
ab{i,j,k}=abb{i,j,k}(:);


cr{i}=[ab{i,1,1};ab{i,1,2};ab{i,1,3};ab{i,1,4};ab{i,2,1};ab{i,2,2};ab{i,2,3};
       ab{i,2,4};ab{i,3,1};ab{i,3,2};ab{i,3,3};ab{i,3,4};ab{i,4,1};ab{i,4,2};
       ab{i,4,3};ab{i,4,4}];
```

3.4.5.2 Convert 16 segments to 1 matrix

```
cr_1=cell2mat(cr_1);

cr_1
```

3.4.6 Constructing a neural network

To create a neural network, we suppose to use neural network pattern recognize (newpr)

```
net_1=newpr(cr_1,t,45);
```

3.4.7 Training Neural network

To train a neural network for at most 10,000 iterations with the performance goal of 0.001 and the minimum performance gradient of 1e-50, use the following code.

```
net_1.trainParam.epochs=10000;
net_1.trainParam.goal=0.001;
net_1.trainParam.min_grad=1e-50;
net_1=train(net_1,cr_1,t);
```

Chapter 4

Result

We used 50 picture of white background and 100 picture of black background of each species for training both models. For each trained model, we tested it by 30 white background pictures dedicated for testing. We repeat the experiment with 30 background pictures. Therefore the totals of 4 results are classified into 2 categories based on the background color of the test dataset. We trained both model with different background color (black and white) expecting the shape would be distinct from the back ground. The results are measured by accuracy and confidence of the matching output for each of experiment mentioned.

4.1. Confusion matrix of pictures with black background color

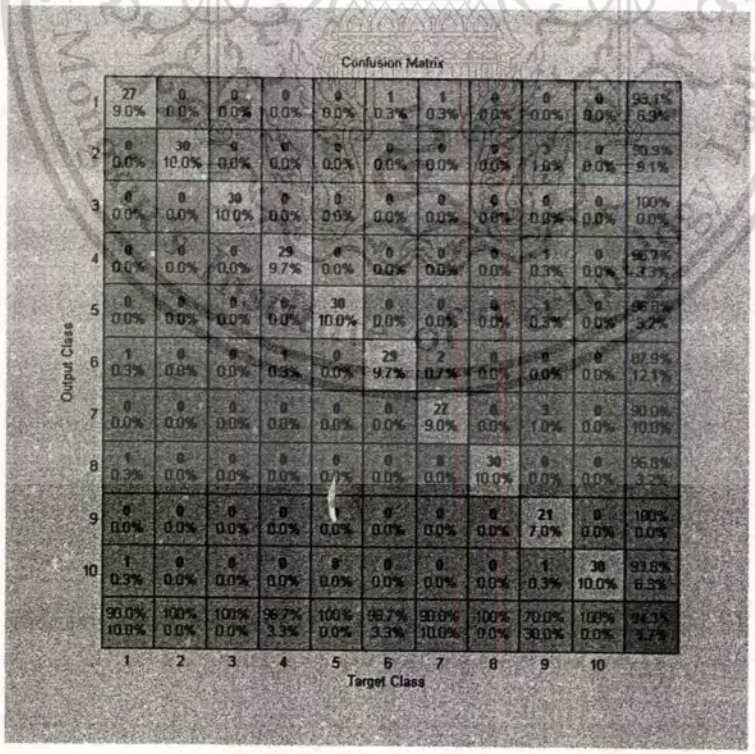


Figure 4.1 Confusion matrixes Black background [Grayscales model]

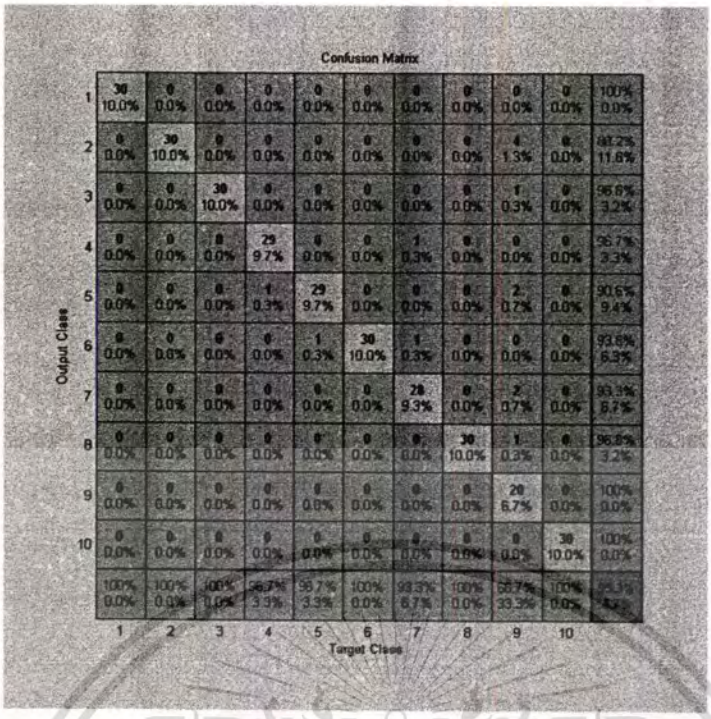


Figure 4.2 Confusion matrixes Black background [Wavelet model]

4.2. Confusion matrix of pictures with white background color

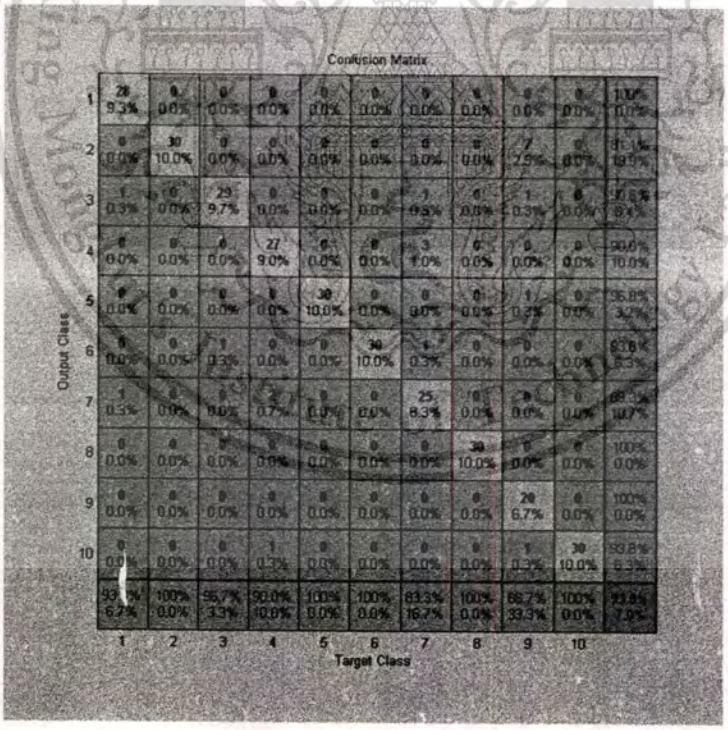


Figure 4.3 Confusion matrixes White background [Grayscale model]

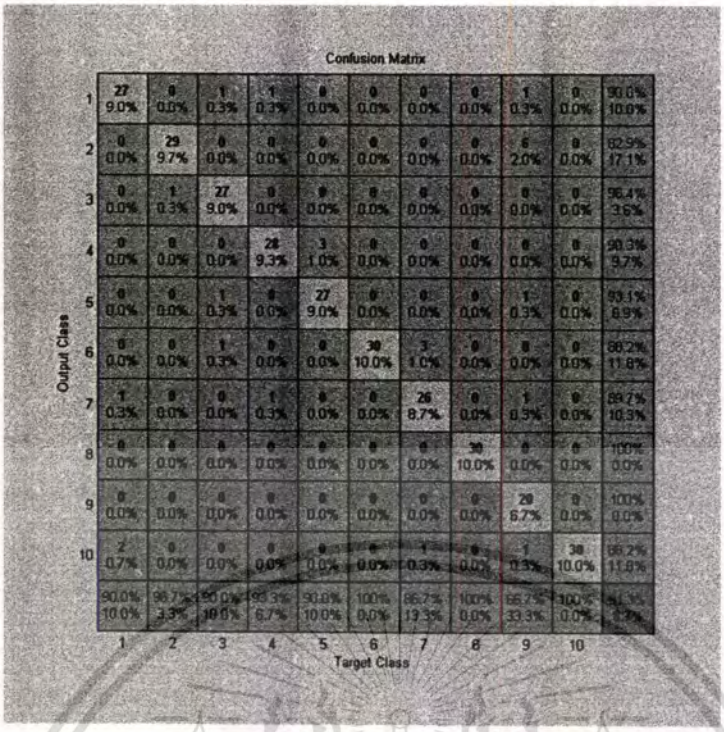


Figure 4.4 Confusion matrixes White background [Wavelet model]

From Figure 4.1 Dendrobium type 2, 3, 5 and 8 were correctly classified 30 out of 30 while Dendrobium type 4, 6 scored 29 out of 30.Dendobrium type 1,7 accuracy rate was 27 out of 30. Dendrobium type 9 score was 21 out of 30. Among 4 models type 9 (Daeng Supab) causes difficulty for classifation.

From the experiment(simulating the train models), the application choose incorrect answer to main and choose the correct answer to minor role and value of confident between them is very similar. Thus, we defined new metric name "come in 2nd," which is the number of correct dendrobium's score come in 2nd .The fact that is incorrectly classified is not our focus. Instead we want to focus on the minimum score difference between the classified dendrobium and the input. In other word, we want to find the threshold that value between the classified dendrobium and the input that might require further investigation to improve the model accuracy. For example

- On Dendobium type 6, the data have incorrectly 1 type out of 30. It chooses type 1 to answer but correct answer is come in minor role. Different confident value between Dendrobium type 1 and type 6, on data number 29, is 0.763307

- On Dendobium type 9, the data have incorrectly 1 type out of 30. It chooses type 10 answer and the correct answer is come in minor role. Different confident value between Dendrobium type 10 and type 9, on data number 1, is 0.082732

The detail comparing display in table 4.1

	Grayscale model		Wavelet model	
Hidden layer Neuron	45 unit		45 unit	
Learning rate	0.001		0.001	
Epochs	199		677	
Training time (seconds)	21.8188		45.8450	
Background color	Black	White	Black	White
Accuracy	94.3%	93%	95.3%	91.3%
Confident	95.1%	94%	96.2%	93.6%
Number of 2nd position	4	7	2	9
Time for test	0.1915	0.1801	0.177	0.1784

Table 4.1 Comparing between grayscale model and wavelet model

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

Our models can classify 10 breeds of orchid using by same neural network and same number of hidden layer. In wavelet model we used wavelet transform function for image compression before splitting image to 4x4 sub images; our result showed that the wavelet model has more accuracy and more confidence than grayscale model.

5.2 Recommendations

There are many areas where we can improve our system. For example

1. If the training set is huge, our neural network can classified breed of orchid more accuracy and more confident. But it will take long time to train and using more memory.
2. If system has more feature like shape detection, color matching, it will be more intelligent
3. User can fix the epoch, goal of training.
4. User can develop the program to be web application.
5. If the data set has noise (picture has the background) ,we believe that using ROI feature to cut their noise off would improve the model's performance

Reference

1. A. C. Gonzalez-Garcia, J. H.Sossa-Azuela , E. M. Felipe-Riveron and O. PogrebnyakTechnologic Institute of Toluca, Image Recognition by Neural Network

2. Wichada inkeaw,Chianmai University,Image Recognition by Neural Network,2007

3. King-sun fu, fellow, IEEE, and Azriel Rosenfeld, FELLOW, IEEE,' Pattern Recognition and Image Processing', IEEE transactions on computers, VOL. C-25, NO. 12, DECEMBER 1976

4. M. Soriano, L. Garcia' Fluorescent image classification by major color histograms and a neural network', National Institute of Physics, University of the Philippines, Quezon City 1101, Philippines

5. Anna Bosch ,Computer Vision Group University of Girona, Andrew Zisserman, Dept. of Engineering Science University of Oxford, 'Image Classification using Random Forests and Ferns'

6. Sonsorot kannasatan and warasok winad, ' Face Detection using Eigen Face and Correlation Analysis', Engineering Transactions (Group A), Vol. 2, No. 1, Jan-April 1999

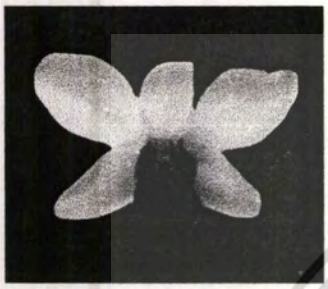
7. B. Bechtel, A. Ringeler & J. Böhner – Segmentation for Object Extraction of Trees, 'segmentation for object extraction of trees using matlab and saga '

8. Sung-Ho Hong a, Rae-Hong Park b, Seungjoon Yang c, Jun-Yong Kim ,' Image interpolation using interpolative classified vector quantization' Department of Electronic Engineering, Sogang University, C.P.O. Box 1142, Seoul 100-611, Republic of Korea

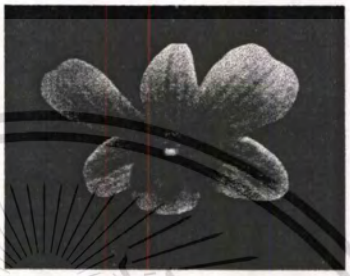
9. www.mathworks.com/

Appendix A

10 species of Dendrobium 1.44" 1.66"



1. Chamming White



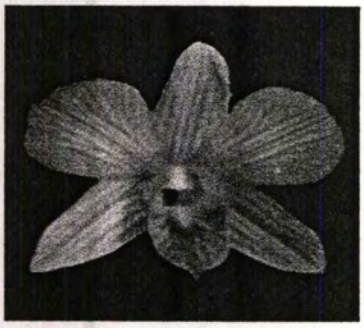
2.Suree Peach

3.Intuwong

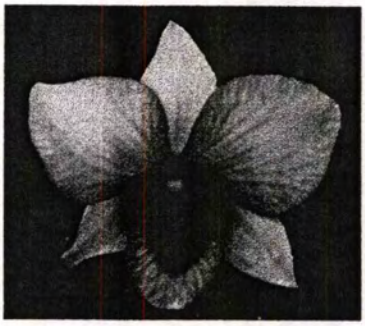
4.Young Star

5.D. Dimond Star

6.Lena



7.Lai Sirin



8. Shompoo AeamOn

9.Daeng Supab

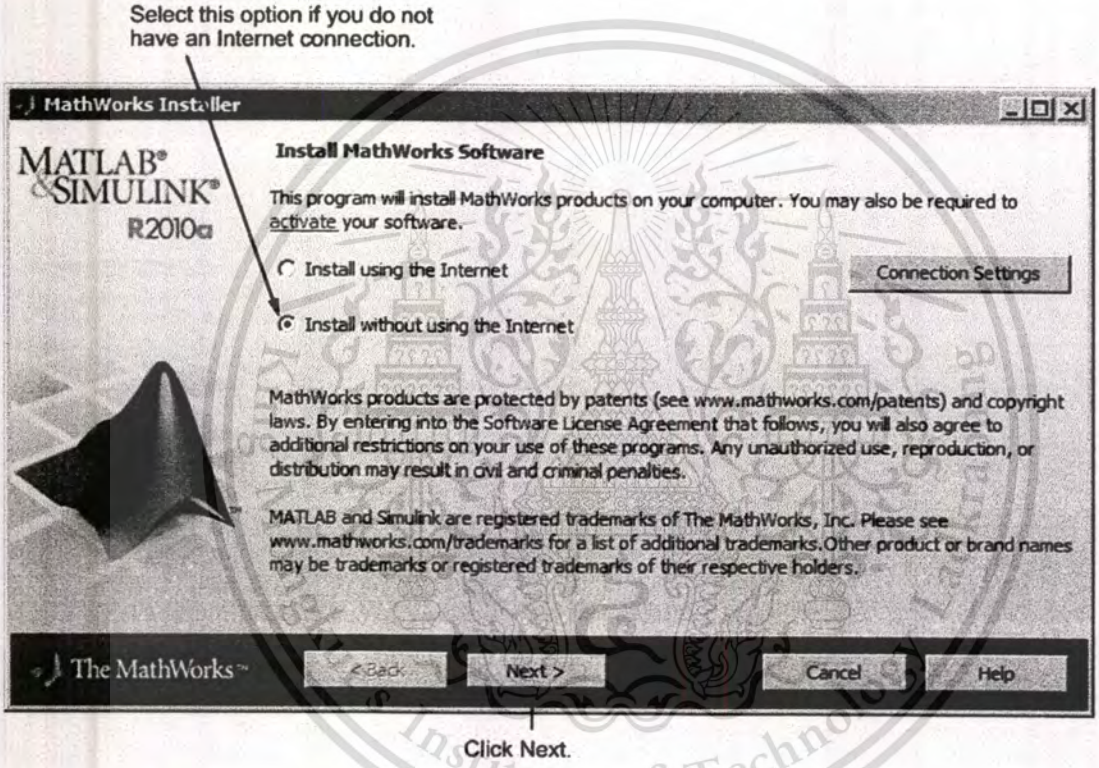
10.Lemon Green

Appendix B

Matlab Installer

Insert the DVD into the DVD drive connected to your system or double-click the installer file you downloaded from the MathWorks Web site. The installer should start automatically.

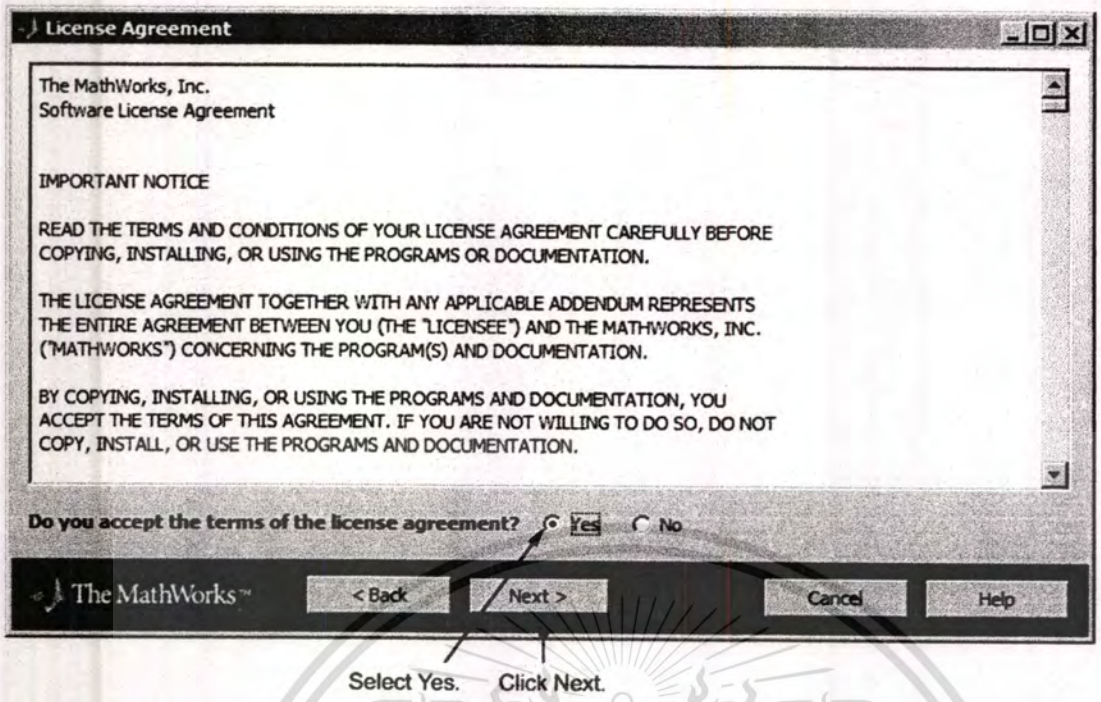
If you do not have an Internet connection, select the **Install without using the Internet** option and click **Next**.



Review the License Agreement

Review the software license agreement and, if you agree with the terms, select **Yes** and click **Next**.

After the installation is complete, you can view or print the license agreement using the file license.txt located in the top-level installation folder.

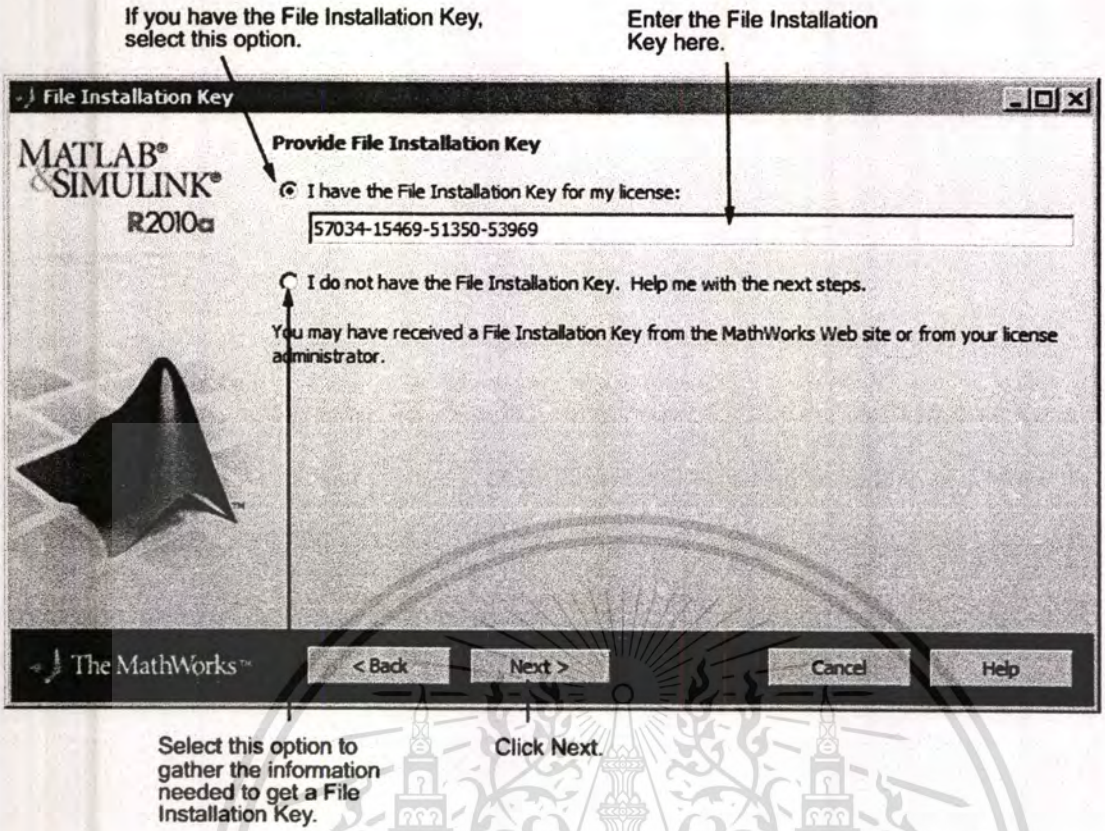


Specify the File Installation Key

If you do not have an Internet connection, and choose to install manually, the installer displays the File Installation Key dialog box. A File Installation Key identifies the products you can install.

If you have the key, select the **I have the File Installation Key for my license** option, enter the File Installation Key, and click **Next**. The administrator contact on a license can retrieve the File Installation Key from the License Center at the MathWorks Web site.

If you do not have the key, select the **I do not have the File Installation Key** option and click **Next**. The installer will provide you with the information you need to get a key.



If You Do Not Have the File Installation Key

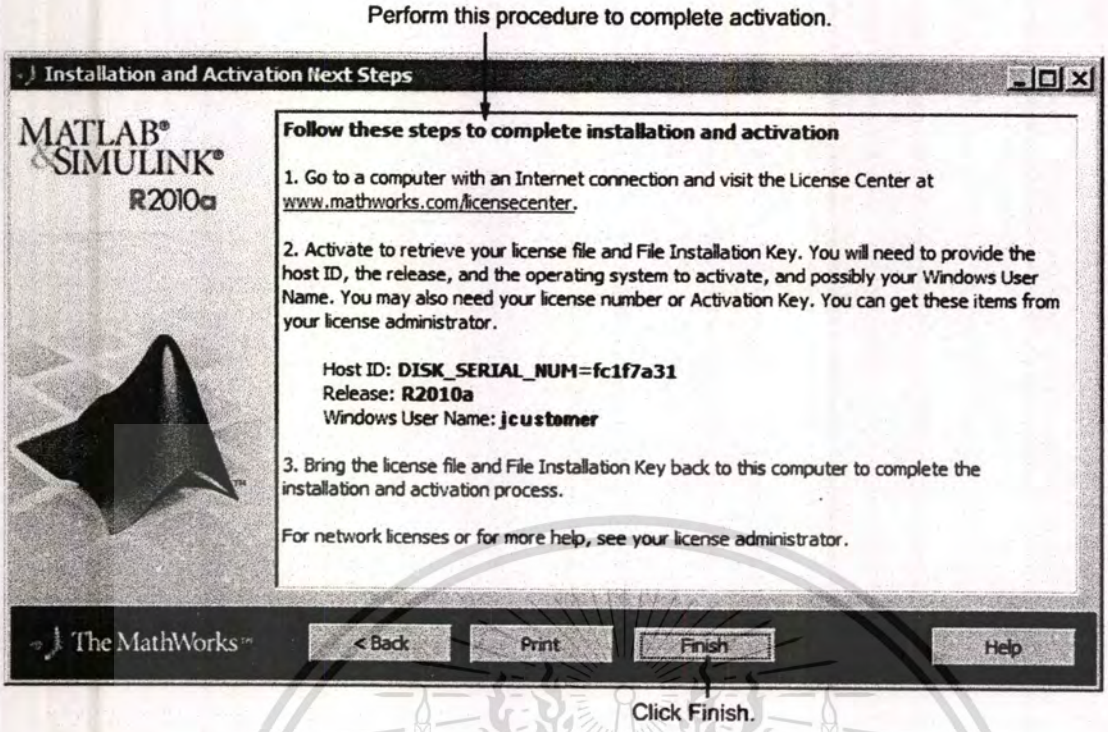
If you choose the **I do not have the File Installation Key** option, the installer displays the Installation and Activation Next Steps dialog box. This dialog box contains the information you need to retrieve your File Installation Key from the License Center at the MathWorks Web site, including:

Host ID

Release number (e.g., R2010a)

Operating system user name (Note that user names are case-sensitive in activation.)

Save the information displayed in this dialog box. For example, you can print a copy by clicking **Print**. Take the information to a computer with an Internet connection and visit the License Center at the MathWorks Web site. The MathWorks uses this information to generate a File Installation Key and a License File. You must have this information with you when you return to the computer on which you want to install and activate the software. To exit the activation process, click **Finish**.

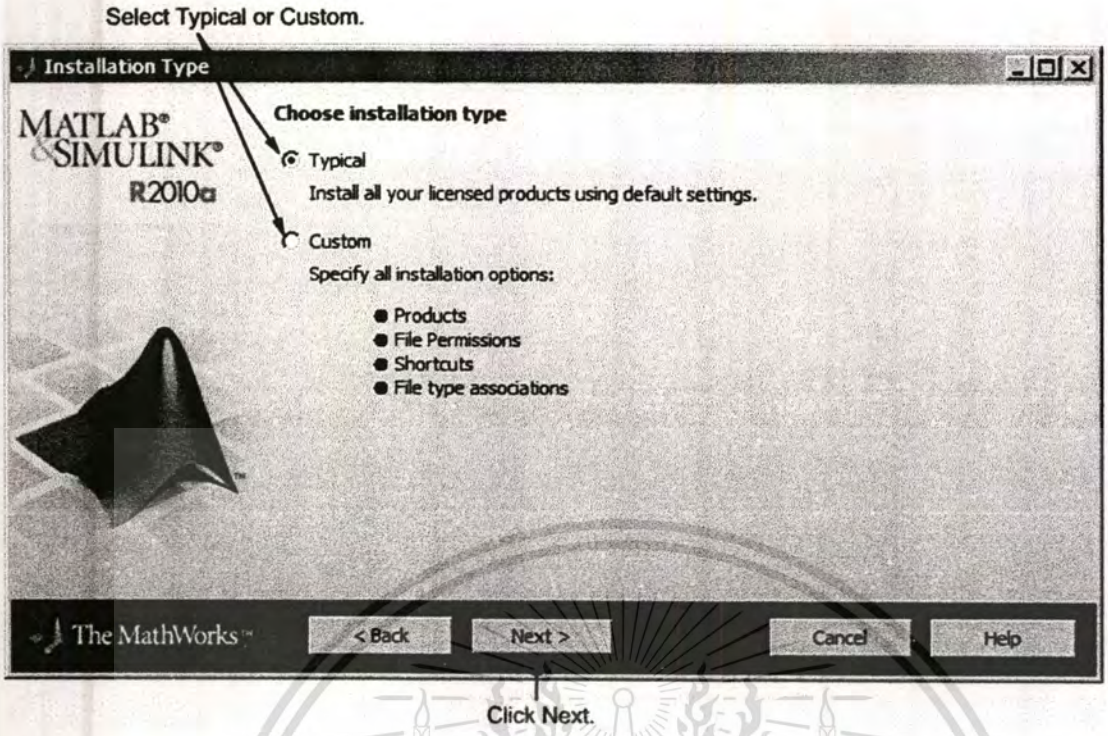


Choose the Installation Type

In the Installation Type dialog box, specify whether you want to perform a Typical or Custom installation and click **Next**.

Choose **Typical** if you have an Individual or Group license and do not need to specify which products you want to install and do not need to access any installation options.

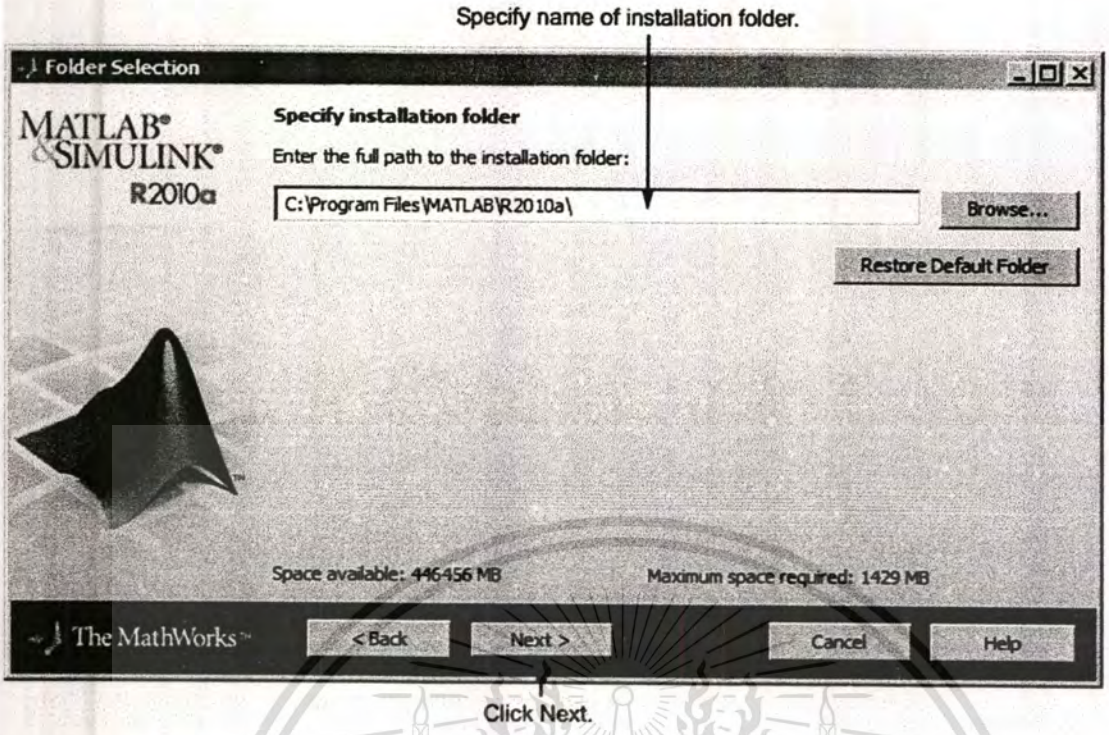
Choose **Custom** if you need to specify which products to install, need access to installation options, or need to install the license manager (network license options only).



Specify the Installation Folder

Specify the name of the folder where you want to install MathWorks products. You can accept the default installation folder or specify the name of a different installation folder. If the folder doesn't exist, the installer creates it.

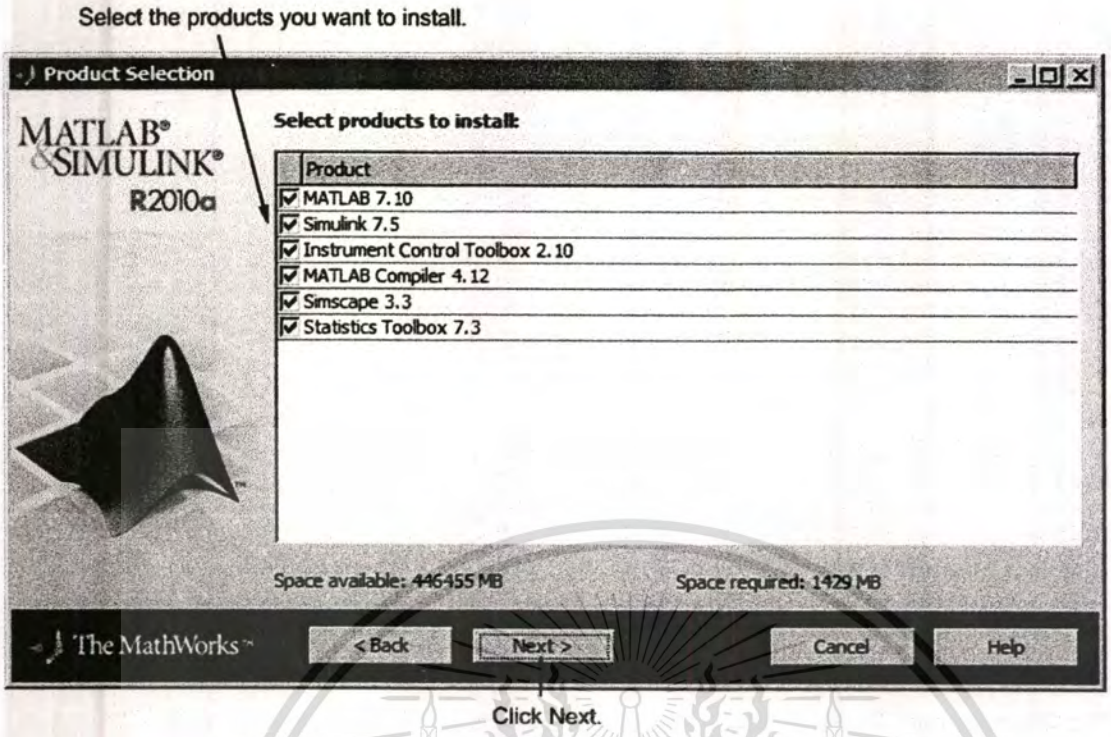
When specifying a folder name, do not specify a name that contains the @ sign, an exclamation point (!), the percent character (%), the plus sign (+), or the dollar sign character ($). The full path of the installation folder must not include a folder named private. If you make a mistake while entering a folder name and want to start over using the default folder name, click **Restore Default Folder**. To continue with the installation, click **Next**.



Specify Products to Install (Custom Only)

If you are performing a custom installation, you can specify which products you want to install in the Product Selection dialog box. This dialog box lists all the products associated with the license you selected or with the Activation Key you specified. In the dialog box, all the products are preselected for installation. If you do not want to install a particular product, clear the check box next to its name.

After selecting the products you want to install, click **Next** to continue with the installation.



Specify Installation Options (Custom Only)

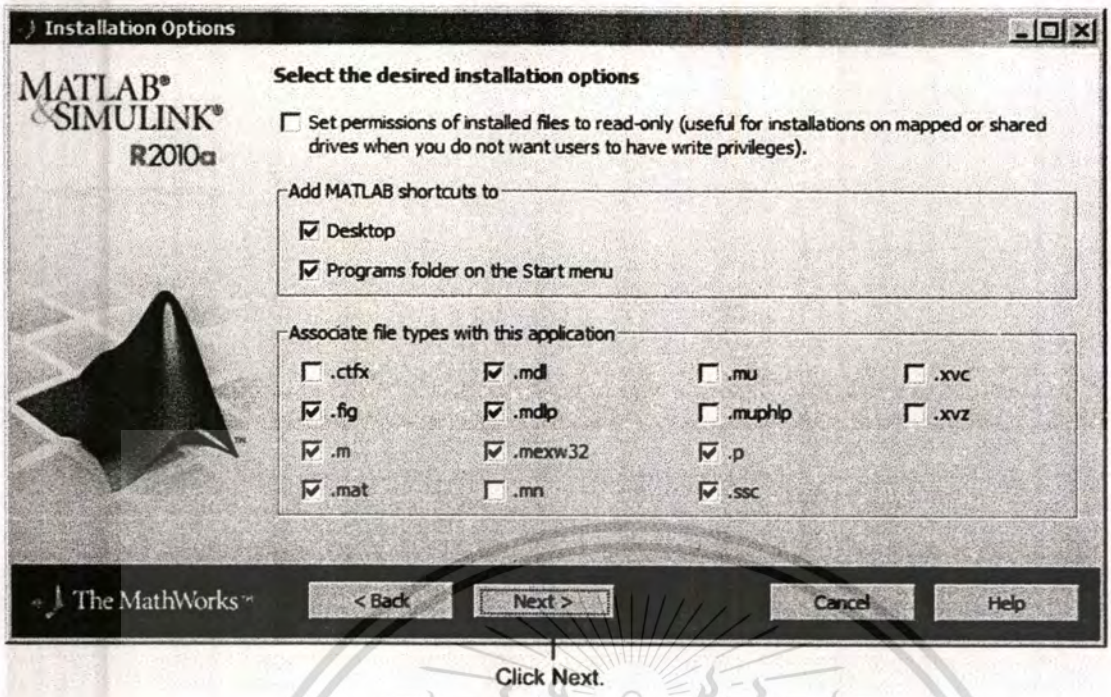
For Custom installations, you can specify several installation options, including:

Setting the permissions of all installed files to read only.

Determining whether the installer puts shortcuts for MATLAB software in the **Start** menu and on the desktop.

Specifying which files the operating system associates with MATLAB, based on their file extension. For example, if you associate files with the .m file extension with MATLAB, the operating system identifies the type of these files as MATLAB M-file. The installer preselects the extensions associated with products you are installing.

After selecting installation options, click **Next** to proceed with the installation.



Click Next.

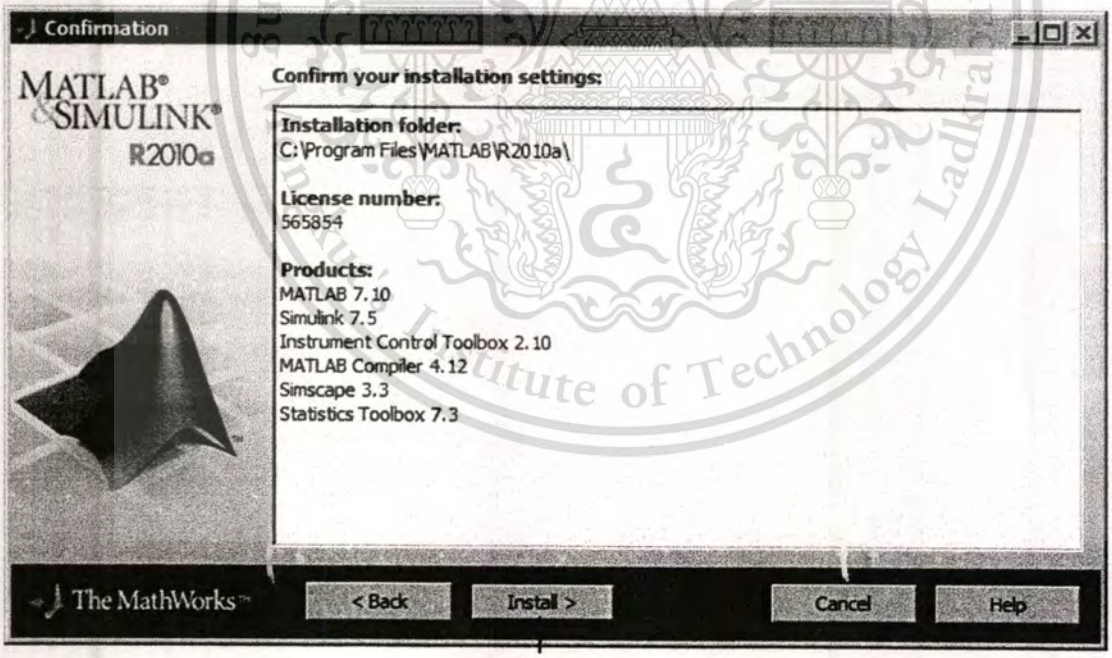
The following table provides brief descriptions of these file extensions.

File Extension	Description
.ctfx	MATLAB Compiled Application
.fig	MATLAB Figure
.m	MATLAB Code
.mat	MATLAB Data
.mdl	Simulink Model
.mdlp	Simulink Protected Model
.mex*	MATLAB MEX. This extension is platform specific: .mexw32 or .mexw64
.mn	MuPAD Notebook
.mu	MuPAD Code

File Extension	Description
.muphlp	MuPAD Help
.p	MATLAB P-code
.ssc	Simscape Model
.xvc	MuPAD Graphics
.xvz	MuPAD Graphics

Confirm Your Choices and Begin Copying Files

Before it begins copying files to your hard disk, the installer displays a summary of your installation choices. To change a setting, click **Back**. To proceed with the installation, click **Install**.



Click Install.

As it copies files to your hard drive, the installer displays a status dialog box to show the progress of the installation.

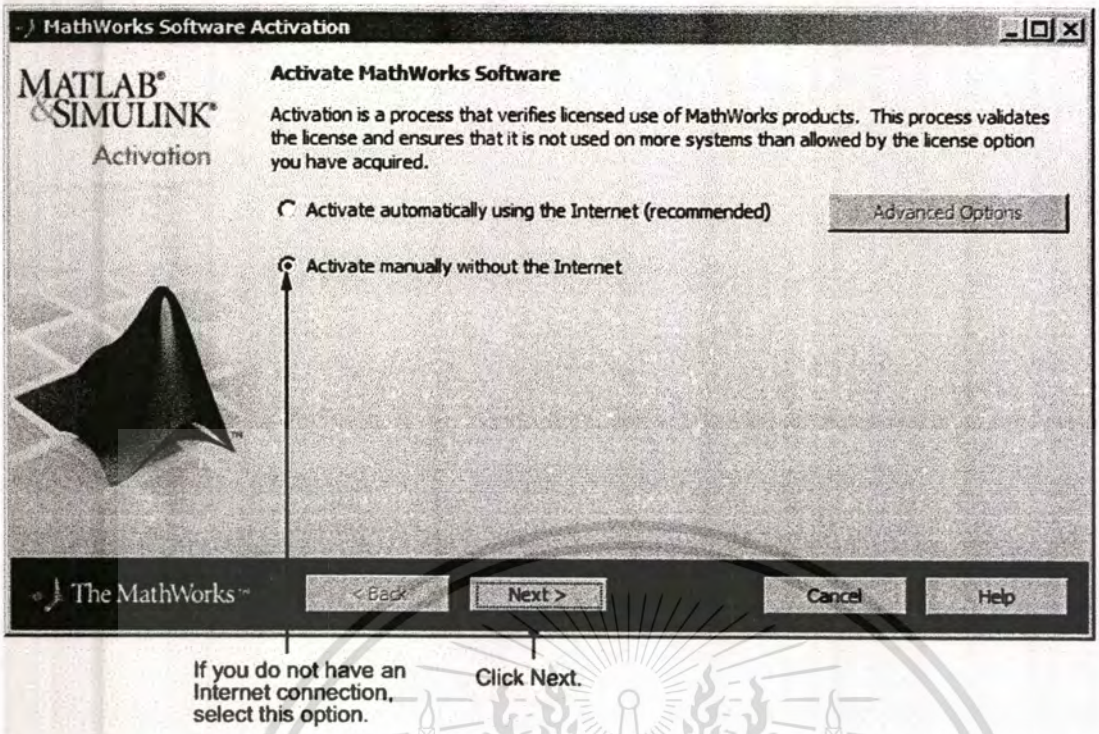
Complete the Installation

When the installation successfully completes, the installer displays the Installation Complete dialog box. In this dialog box, you can choose to activate the software you just installed. You cannot use the software you installed until you activate it. The MathWorks recommends activating immediately after installation. If you logged in to your MathWorks Account during installation, your log-in session continues into the activation process. Click **Next** to proceed with activation.

If you choose to exit the installer without performing activation, clear the **Activate MATLAB** option and click **Finish** (the button label changes). You can activate later using the activation application.

Activate Your Installation

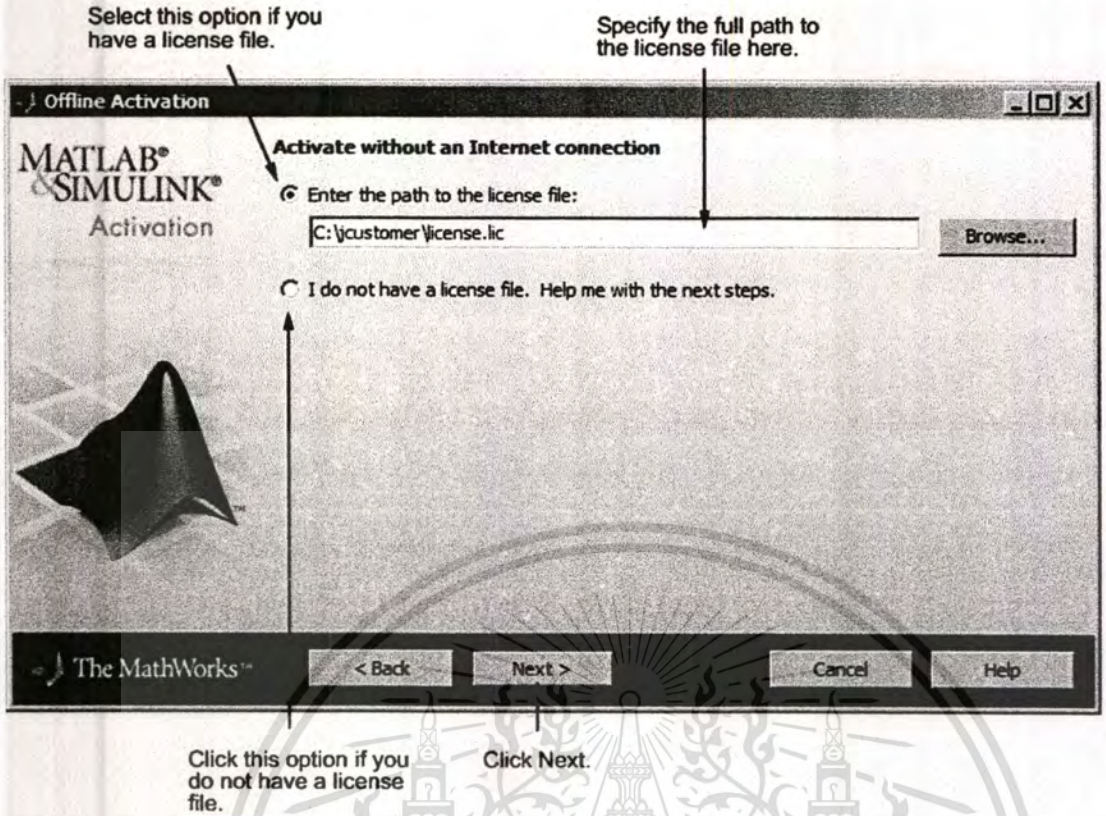
Because you were not logged in to your MathWorks Account during installation, or you started the activation application independently, you must choose whether to activate automatically or manually. Select the **Activate manually without the Internet** option and click **Next**.



Specify the Path to the License File

To activate without an Internet connection, you must have a License File. The License File identifies which products you can run. The administrator contact on the license can retrieve the License File from the License Center at the MathWorks Web site.

Select the **Enter the path to the License File** option and enter the full path of your License File in the text box (or drag and drop the file) and click **Next**. If you do not have your License File, select the **I do not have a license file** option and see If You Do Not Have a License File for more information.



If You Do Not Have a License File

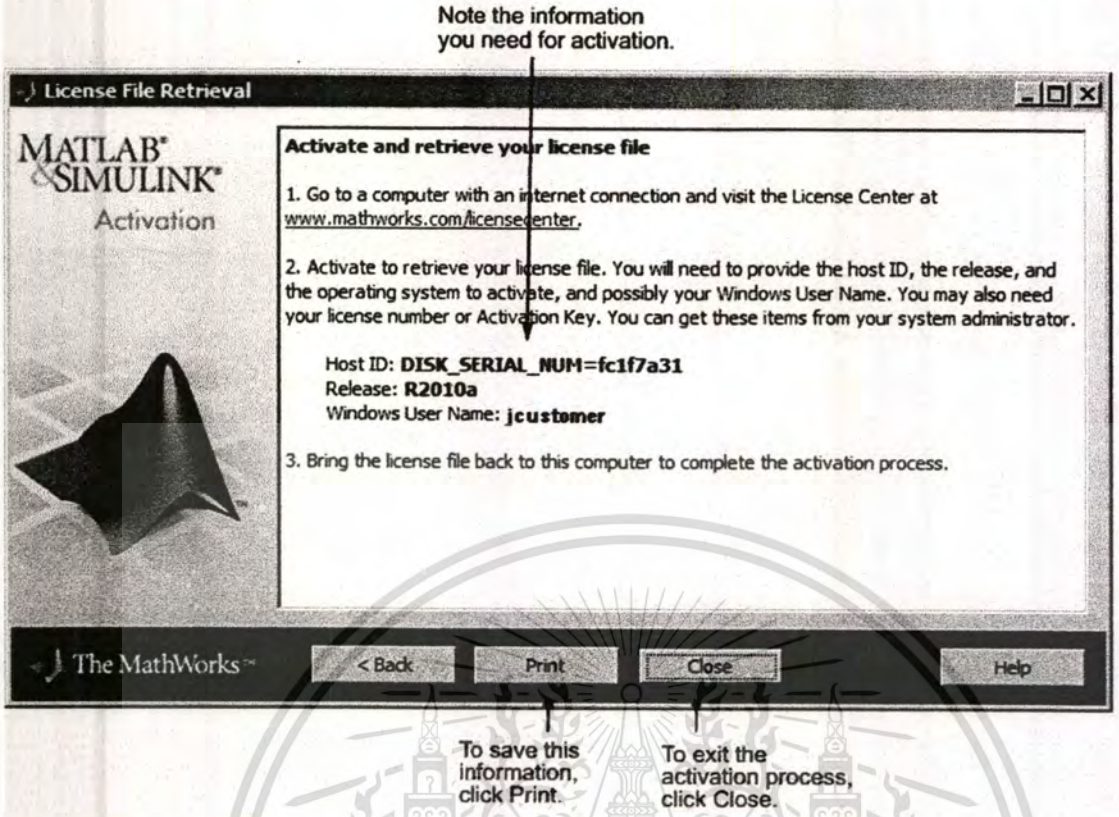
If you are activating manually and do not have your License File, the License File Retrieval dialog box explains how to get your License File and finish activation. The dialog box displays the information you will need to get your License File, including:

Host ID

Release number (e.g., R2010a)

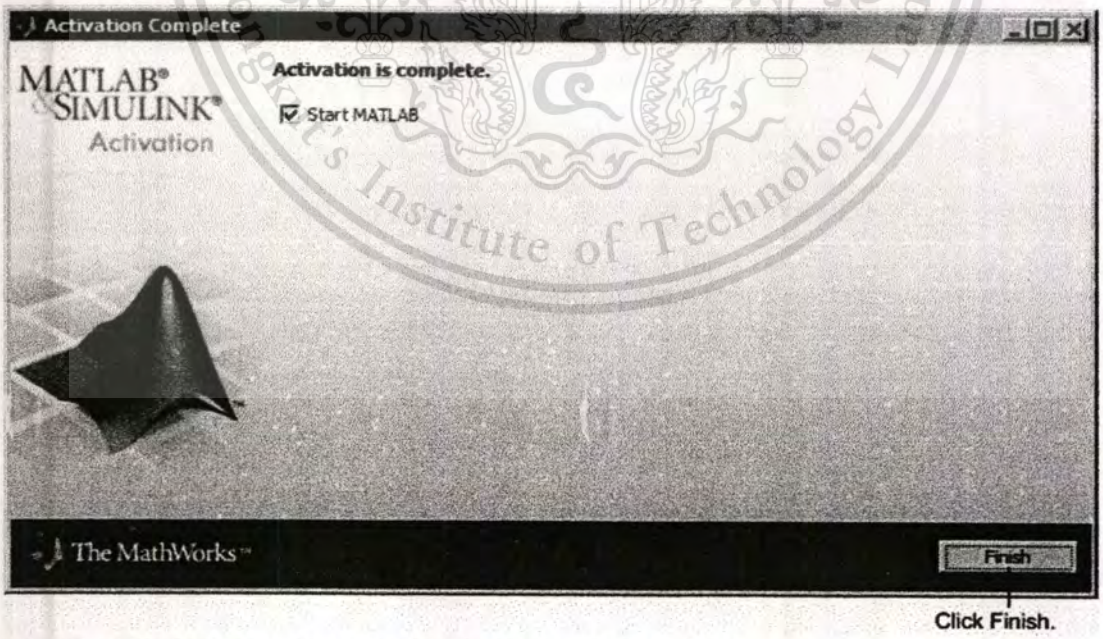
Operating system user name (Note that user names are case-sensitive in activation.)

Save the information displayed in this dialog box. For example, you can print a copy by clicking **Print**. Take the information to a computer with an Internet connection and visit the License Center at the MathWorks Web site. The MathWorks uses this information to generate a File Installation Key and a License File. You must have this information with you when you return to the computer on which you want to install and activate the software. To exit the activation application, click **Finish**.



Complete the Activation

After activating your installation, you can run The MathWorks software. If you do not want to run MATLAB now,



Clear the **Start MATLAB** option and click **Finish** to exit the activation process.

Appendix C

Brief matlab commands usage.

1. clear - Remove items from workspace, freeing up system memory

Syntax	Example
clear name	clear all

(Remove all item form workspace; you can specific on variable name)

2. clc - Clear Command Window

Syntax	Example
Clc	Clc

(Clear all command window)

3. imwrite - Write image to graphics file

Syntax	Example
imwrite(A,filename)	Imwrite(image,'c:\data\image1.jpg'); (export the variable name image to jpg file in c:\data\ name image1)

4. imread - Read from graphics file

Syntax	Example
A = imread(filename)	image=imread('c:\data\image.jpg'); (image.jpg must be in c:\data\ )

5. imresize – resize image

Syntax	Example
B = imresize(A, [mrows ncols])	image=imresize(image1,[256 256]);

(Resize image name 'image1' to 256x256 keep in variable name 'image')

6. rgb2gray - Convert RGB image or colormap to grayscale

Syntax	Example
I = rgb2gray(RGB)	g_image = rgb2gray(image);

(Change rgb image name 'image' to grayscale image name 'g_image')

7. mat2cell - Divide matrix into cell array of matrices

Syntax	Example
c = mat2cell(x, m, n)	X = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15; 16 17 18 19 20] X = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 C = mat2cell(X, [2 2], [3 2]) C = [2x3 double] [2x2 double] [2x3 double] [2x2 double] C{1,1} C{1,2} ans = ans = 1 2 3 4 5 6 7 8 9 10

	C{2,1} C{2,2} ans = ans = 11 12 13 14 15 16 17 18 19 20

8. cell2mat - Convert cell array of matrices to single matrix

Syntax	Example
m = cell2mat(c)	Combine the matrices in four cells of cell array C into the single matrix, M: C = {[1] [2 3 4]; [5; 9] [6 7 8; 10 11 12]} C = [1] [1x3 double] [2x1 double] [2x3 double] C{1,1} C{1,2} ans = ans = 1 2 3 4 C{2,1} C{2,2} ans = ans = 5 6 7 8 9 10 11 12 M = cell2mat(C) M = 1 2 3 4 5 6 7 8 9 10 11 12

9. newpr-Create pattern recognition network

Syntax	Example
net = newpr(P,T,S)	load simpleclass_dataset

P-input vector T-target vector S-size of hidden node	net = newpr(simpleclassInputs,simpleclassTargets,20);

10. train - Train neural network

Syntax	Example
net1=train(net,P,T) net-Network P-Network inputs T-Network targets	load simpleclass_dataset net = train(net,simpleclassInputs,simpleclassTargets);

11. sim -Simulate neural network

Syntax	Example
[Y,E,perf] = sim(net,P,T) net-Network P-Network inputs T-Network targets Y-Network output E-Network errors Perf-Network performance	Load simpleclass_dataset simpleclassOutputs = sim(net,simpleclassInputs);

Appendix D

Brief matlab Cook book

Case 1: Simple network with 2 input node and 1 output node for OR gate function

P = [0 0 1 1; 0 1 1 0]; Input value

P	0	0	1	1
	0	1	1	0

T = [0 1 1 1]; Output value

T	0	1	1	1

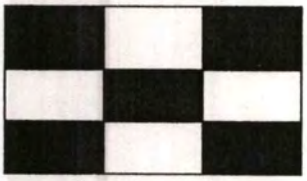
net = newff(P,T); create a neural network

net = train(net,P,T); Train a neural network

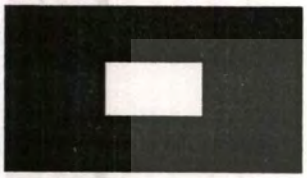
simulation = sim(net,P) Simulate a neural network

Remark : If input is vector value (p=[0 0;0 1;1 1; 1 0]), output must be vector value (T=[0;1;1;1])same.

Case 2 : Classifying X and O characters



A = [101; 010; 101]



B = [111; 101; 111]

A=A(:);	Arrange A into a vector form (101010101)
B=B(:);	Arrange B into a vector form (111101111)
P=[A,B];	Input value
T = [1 0;0 1];	Output value
net = newff(P,T,5, 'logsig');	create a neural network with 5 hidden nodes and using log-sig to transfer function
net = train(net,P,T);	Train the network
simulation = sim(net,P)	Simulate a neural network

Remark : The network will read a value of each input A,B one digit at a time until the end before learning the predefined output in T.

Case 3: Cropping and resizing for image processing

256x256 pixel

```
image=imread('c:\data\image.jpg');
image=imcrop(image);
image= imcrop(image,[139 273 135 239]);
```

139 is X minimum ,273 is Y minimum ,135 is width and 239 is high of box.

```
image = imresize(image,[256 256]);
```

Resize image to 256x256 pixel.

```
image=rgb2gray(image);
```

Convert image to grayscales image.

```
image=image(:);
```

Arrange image into a vector form(65536x1).

```
image=double(image);
```

Convert unit8 to double.

Remark 1: The coordinate for cropping the image used in this code is obtained manually from matlab GUI. (The output is of size 135 x 239 pixels prior to scaling to 256 x 256)

Remark 2:The neural network is always required a mathematical value as its input. Because the color value in unit8 format is not eligible format, therefore, a conversion to double is required.

Remark 3: If you have more than one picture ,you can use for -loop to store picture value.

Example : For-loop

```
for(i=1:256)
    for(j=1:256)
        b = j+((i-1)*256);
        a(b,1)=image(i,j);
    end
end
```

Case 4: The face recognition using neural network.

For the face recognize 10 pictures and 10 hidden nodes will be used. Program will categorized process into 2 parts. First part; we prepare data to input to neural network and convert the picture to gray scales image. On the picture, we specific point on eyes nose and month. Second part; create the neural network by using 'pic_msd' to store 10 pictures and using 'target' to rearranging any value onto the matrix form. Show the code of face recognition on below.

First Part : Image Process

Image = imread('pic.jpg');	Load image name pic.jpg to keep in variable image
Image=rgb2gray(image);	Change color image to grayscale image

Eye part

Eye=image(34:42,11:49);	Crop the picture specific eyes
Eye=double(eye);	Change the data to double for input to neural network
a=mean(eye);	Find the mean from any rolls
b=std(eye);	Find the s.d. from any columns
c=[a;b];	Store data a and b into c
Eye_msd=c(:);	Arrange the data to 1 column

Nose part

Nose=image(42:56,19:41);	Crop the picture specific nose
Nose=double(nose);	Change the data to double for input to neural network

a=mean(nose);	Find the mean from any rolls
b=std(nose);	Find the s.d. from any columns
C=[a;b];	Store data a and b into c
Nose_msd=c(:);	Arrange the data to 1 column

Mouse part

Mouth=image(56:68,19:41);	Crop the picture specific nose
Mouth=double(mouth);	Change the data to double for input to neural network
A=mean(mouth);	Find the mean from any rolls
B=std(mouth);	Find the s.d. from any columns
C=[a;b];	Store data a and b into c
Mouth_msd=c(:);	Arrange the data to 1 column
Pic_msd=[eye_md;nose_msd;mouth_msd];	Combine any value into 'pic_msd'

Second Part : Neural Network.

net=newff(pic_msd,target,10);	Create neural network with 10 hidden node
net = train(net,pic_msd,target);	Train Neural network
output=sim(net,pic_msd);	Simulate output

Remark : The neural network have limited space , you would reduce size of input before feed into neural network .