THAI BUDDHIST AMULET RECOGNITION SYSTEM

WARANAT KITIYANAN

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE (COMPUTER SCIENCE)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2014**

Thesis
entitled

THAI BUDDHIST AMULET RECOGNITION SYSTEM

……
Mr. Waranat Kitiyanan
Candidate

……
Asst.Prof. Chomtip Pornpanomchai,
Ph.D. (Computer Science)
Major advisor

……
Assoc.Prof. Damras Wongsawang,
Ph.D. (Information Engineering)
Co-advisor

……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……
Asst.Prof. Sudsanguan Ngamsuriyaroj,
Ph.D. (Computer Science and Engineering)
Program Director
Master of Science Program in Computer Science
Faculty of Information and Communication Technology
Mahidol University

Thesis
entitled

THAI BUDDHIST AMULET RECOGNITION SYSTEM

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science (Computer Science)
on
August 8, 2014

……
Mr. Waranat Kitiyanan
Candidate

……
Assoc.Prof. Panjai Tantasanawong,
Ph.D. (Computer Science)
Chair

……
Asst.Prof. Chomtip Pornpanomchai,
Ph.D. (Computer Science)
Member

……
Assoc.Prof. Damras Wongsawang,
Ph.D. (Information Engineering)
Member

……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……
Assoc.Prof. Jarernsri L. Mitrpanont,
Ph.D. (Computer Science)
Dean
Faculty of Information and
Communication Technology
Mahidol University

ACKNOWLEDGEMENTS

The implementation of this research would not be successful without the great support from individuals.

I would like to thank Asst.Prof. Chomtip Pornpanomchai who advises me all along this work and the committee members: Assoc.Prof. Damras Wongsawang and Assoc.Prof. Panjai Tantasanawong for being my thesis committee and giving me very beneficial suggestions about this research.

Also, I give my thankfulness to Miss Pimprapai Leerasakultham for assisting in compilation of samples and Miss Benjamaporn Lurstwut who shared her experiences about researching and gave me valuable suggestions.

In addition, I would like to show gratitude to the faculty of Information and Communication technology, Mahidol University for the opportunity in this research area.

Most importantly, I am very grateful to my parents who always support me for everything so far.

Waranat Kitiyanan

THAI BUDDHIST AMULET RECOGNITION SYSTEM

WARANAT KITIYANAN 5537465 ITCS / M

M.Sc. (COMPUTER SCIENCE)

THESIS ADVISORY COMMITTEE : CHOMTIP PORNPANOMCHAI, Ph.D.
DAMRAS WONGSAWONG, Ph.D.

ABSTRACT

The objective of this research is to develop a computer system to recognize digital images of Thai Buddhist amulets by using image processing techniques and artificial neural networks (ANN). The scope is to recognize 100 kinds of powder Buddhist amulets. Moreover, the system is designed to tolerate three kinds of image transformation, which are rotation, size rescaling, and brightness adjustment.

To extract features from collected samples, the extraction process considers shape, color, and texture. In total, 54 feature values are calculated and fed into the recognition process. To recognize those features, there are two ANN models applied, which are perceptron and multi-layer perceptron. In the experiment, those two models are tested with a normal test set and transformed test sets.

From the experiment, on average, the perceptron can recognize the amulet images correctly at 97.33 percent for all test sets, and the multi-layer perceptron can classify them correctly at 100 percent. In addition, the rotated, resized, and brightness adjusted test sets do not cause significant errors in the system.

KEY WORDS: BUDDHIST AMULET / IMAGE RECOGNITION /
IMAGE PROCESSING / ARTIFICIAL NEURAL NETWORK

96 pages

ระบบรู้จำพระเครื่องไทย

THAI BUDDHIST AMULET RECOGNITION SYSTEM

วรณัฐ กิติยานันท์ 5537465 ITCS/M

วท.ม. (วิทยาการคอมพิวเตอร์)

คณะกรรมการที่ปรึกษาวิทยานิพนธ์ : ชมทิพ พรพนมชัย, Ph.D.,ดำรัส วงศ์สว่าง, Ph.D.

บทคัดย่อ

วัตถุประสงค์ของงานวิจัยนี้เพื่อพัฒนาระบบคอมพิวเตอร์ที่สามารถรู้จำ (Recognize) ภาพของพระเครื่องโดยการประยุกต์ใช้หลักวิชาของการประมวลภาพ (Image Processing) และระบบโครงข่ายประสาทเทียม (Artificial Neural Network) ซึ่งขอบเขตของกลุ่มตัวอย่างพระเครื่องในงานวิจัยนี้ถูกจำกัดให้เป็นพระชนิดผงจำนวน 100 องค์ โดยที่ระบบรู้จำถูกออกแบบให้ทนทานต่อการแปลงภาพพื้นฐาน 3 ชนิด ได้แก่ การหมุนภาพ การปรับขนาดภาพ และการปรับความสว่างของภาพ

เพื่อที่จะสกัดลักษณะเด่นของตัวอย่างภาพที่เก็บได้ระบบจะพิจารณาจาก 3 สิ่ง ได้แก่ รูปร่าง สี และลักษณะพื้นผิว ซึ่งโดยรวมแล้วค่าลักษณะเด่นทั้งหมด 54 ค่าจะถูกคำนวณ และป้อนเข้าไปในกระบวนการรู้จำ ในการรู้จำลักษณะเด่นเหล่านั้นระบบจะสามารถรู้จำได้ด้วยระบบโครงข่ายประสาทเทียม 2 รูปแบบ นั่นคือ เพอร์เซ็ปตรอน (Perceptron) และเพอร์เซ็ปตรอนแบบหลายชั้น (Multi-Layer Perceptron) จากนั้นทั้งสองรูปแบบจะถูกทดสอบในการทดลองด้วยชุดการทดลองแบบธรรมดาและแบบที่แปลงแล้ว

จากผลการทดลองพบว่าโดยเฉลี่ยแล้วเพอร์เซ็ปตรอนสามารถแยกแยะรูปภาพพระเครื่องได้ถูกต้อง 97.33 เปอร์เซ็น และในส่วนของเพอเซ็ปตรอนแบบหลายชั้นสามารถแยกแยะได้ถูกต้อง 100 เปอร์เซ็น นอกจากนั้นยังไม่พบว่าชุดการทดลองที่ถูกหมุน ถูกปรับขนาด และถูกปรับความสว่างก่อให้เกิดความผิดพลาดในระบบแบบมีนัยยะสำคัญ

96 หน้า

CONTENTS

CONTENTS (cont.)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

CHAPTER I
INTRODUCTION

1.1 Motivation

In Thailand, there are more than 90 percent of population who are Buddhists. Thailand and Buddhism are firmly related in many aspects. It has been influencing the Thai history, art, and culture for a long time. There are at least 25,000 Buddhist temples established but, they cannot connect Thai people and Buddhism together without other two things, which are monks and ceremony. Mostly, Thai Buddhist people believe that attending to ceremony led by monks comes up with luck and good things. After joining a ceremony, they may get a small Buddha image believed that it will bless and protect them from bad things as amulet.

Buddhist amulets are made in many types and styles. They have different size, shape, material, and the appearance of Buddha. It makes them unique and attractive to collectors who have faith and love to see Buddhist art. Because of the demand, Buddhist amulet becomes a trading community. As a result, amulet trader is a well-known occupation in Thailand.

Nowadays the trading is not only on the street, but it is on the internet also. Traders share their collections by using digital media rather than physical media such as amulet magazine. The traders like to post their collection on the internet via online services such as, internet forum, and social network, so they may get some interesting offers as well. In addition, search engines can find several results of Buddhist amulet images if there are some related keywords used. It indicates that putting amulets into digital form for trading is a usual way for traders to get trading easier and faster.

Basically, the traders have many kinds of Buddhist amulet, so the skill they need is to remember them. It is very important because traders have to manage their collection or share necessary information requested by other traders. Lacking of ability to remember those amulets is a problem for traders because they need to check the offered amulets if they the same as their own collections. Moreover, for new collectors

and traders, it is hard to check that the amulet's information is corresponding to the posted amulet's image. It is almost impossible to know all of amulet kinds. Technically, traders will have a trading only when they can verify the amulet kind and genuineness.

In the area of artificial intelligence, this kind of problem can be solved by using pattern recognition technique. A popular approach is called artificial neural network (ANN). ANN is a supervised learning machine inspired by the biological nervous system. It is used to classify data of digital images in several recognition experiments.

By the reasons, this research is motivated to invent a system, which is able to mitigate the problems of Buddhist amulet management inefficiency and non-experience trader. The goal is to collect a sample dataset of Buddhist amulet images then design, implement, and evaluate a recognition system using techniques of digital image processing and artificial neural network for classifying those amulet images. Additionally, the system has to be designed for easy use and acceptable result as well.

1.2 Problem Statement

It is not easy to remember many kinds of amulets even they are collectors. The collectors have their specialty, and they can remember only their collections that interest them. Basically, to study Buddhist amulets, collection books are necessary for searching and remembering. The collectors may have many books on their shelves; it is tough and take a long time to search. In addition, some kinds of amulets are similar, so new collectors can get confused in this case.

Unfortunately, people who are not amulet collectors but already have some amulets may not aware of the value in what they have. Moreover, they can be deceived by people who have a malicious intention.

1.3 Objectives

The objective of this research is to implement an automated system to recognize digital images of Buddhist amulets. The system has to be more tolerant to digital image transformation. After that, we will have an evaluation for the system to

assess the system's accuracy. Additionally, graphic user interface have to be created and provided to support user.

1.4 Scope of Work

This work is going to be executed under these below scopes:

- All of Buddhist amulets in this experiment are made of powder
- The system supports only digital image
- The software is windows based
- The system is based on digital image processing and artificial neural network
- In evaluation, accuracy rate of the system is focused

1.5 Thesis Organization

There are six chapters in this research including:

- Chapter I: Introduction – the chapter that describe about the background of what we are going to work with, the problems in the specific area, the goal of research, and project scope
- Chapter II: Literature Review – the discussion of researches found in the past and related to the problem in this work
- Chapter III: System Design and Methodology – the explanation of how our system is designed and methods in each component
- Chapter IV: Implementation – the presentation about how we apply chosen methods in software programming and how we assess the implemented system
- Chapter V: Experimental Result – The section that shows the result of system assessment
- Chapter VI: Conclusions and Suggestions – The last main part that sum up the result of this research and discuss the way for improvement.

CHAPTER II
LITERATURE REVIEW

In this chapter, the main content is the discussion of previous researches, which are relevant to this project. The purpose is to study about pieces involving to Buddhist amulet image recognition and similar problems in the past. Even their problem may be alike, the techniques are different. To study on this problem better, we need to explore techniques that they use to solve the problems. Hopefully, we would find some points that can be developed in our research, and the idea might influence to this research partially.

Before we are going to see the researches of Buddhist amulet recognition. We start with the coin recognition problems that are not too different from the Buddhist amulet recognition problem.

2.1 Coin Recognition Researches

The first research is “Rotation-Invariant Neural Pattern Recognition System with Application to Coin Recognition” [1]. M. Fukumi, et al. (1992) had an intention to distinguish 500 yen coin and 500 won coin without rotation sensitivity by using artificial neural network. The size of coin image in their research was 250×250 pixels. They had two main parts in their system, which were fixed network (box of slabs) and trainable network. Their model is shown in the Figure 2.1.

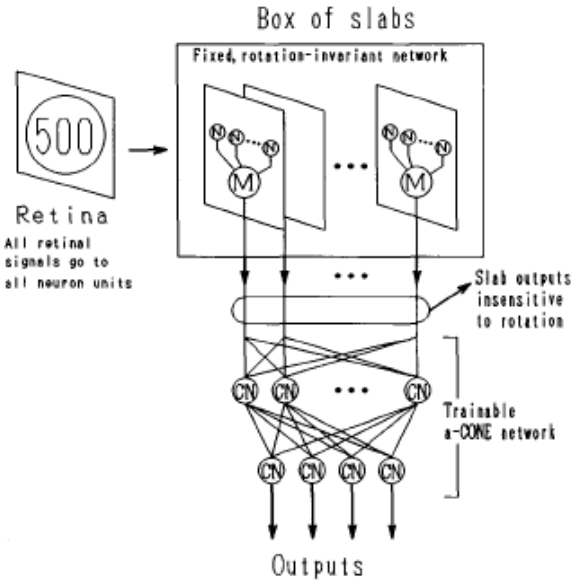


Figure 2.1 Fixed Network and Trainable Network [1]

To deal with rotation sensitivity, they proposed circular array instead of normal array to keep weights in each slap for fixed network. From the Figure 2.2, the circular array of weights with N = 12 could handle a coin at every 30-degrees rotation. Therefore, the number of slaps in fixed network would be increased accordingly to the number of degrees handled.

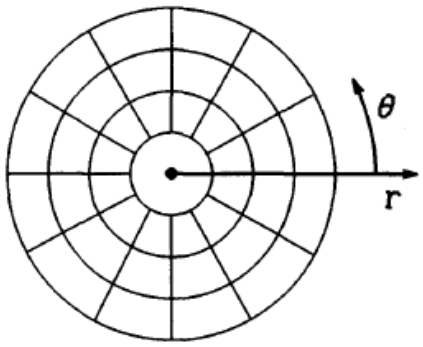


Figure 2.2 Circular Array [1]

They applied back propagation technique in their trainable network for training. After the network had been already trained, the output from fixed network would be fed into the trainable network for classification processing. From the Figure

2.3, the graph shown that 25 slabs can boost the accuracy rate to 100 percent if they used 10 samples for training.

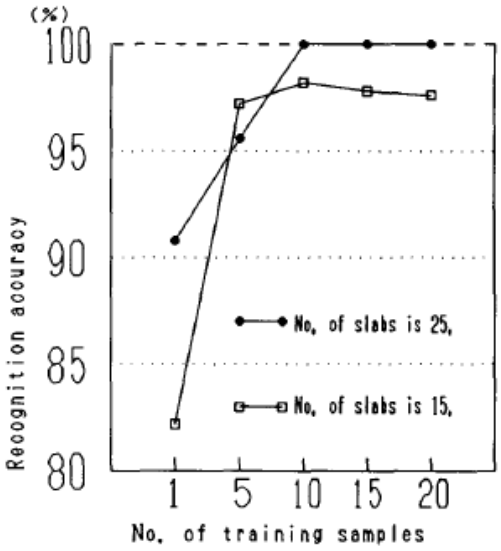


Figure 2.3 Accuracy Rates in Various Number of Samples [1]

Now we look at the research topic of "Design and Evaluation of Neural Networks for Coin Recognition by using GA and SA" [2]. This is another paper that applied artificial neural network to recognize coin. However, in the paper, Y. Mitsukura, et al. (2000) focused on how to design neural network with genetic algorithm (GA) and simulated annealing (SA).The problem was similar to the previous research since they would like to classify 500 yen coin and 500 won coin as well. For input signal, they divided the shape of coin into 512 sections (16 in radius direction and 32 in perimeter direction), and they analyzed those signals with Fourier transform technique. The accuracy of their recognition was 99.68 percent. That was very close to a perfect result.

R. Bremananth, et al. (2005) who proposed the research of "A New Approach to Coin Recognition using Neural Patter Analysis" [3] had an intention to recognize the value of coin in different currency because they aimed that it is going to a benefit in businesses. The research's scope was to recognize the numeral in Indian coin. It has three kinds of them, which are 1-rupee, 2-rupee, and 5-rupee. The main objective was to extract these numerals from coins. They claimed that the edge in coin numeral is difficult to detect. That made the edge detection process complicated. They

recommended to use hence statistical color threshold method solving this problem. After they had got the image of numeral, they put it into recognition process by using rotation-invariant technique. Their recognition technique used Gabor filter to carry itself out. Basically, they applied back propagation as learning method of their system. They mentioned that the system could support real-time application. According to the Figure 2.4, the success rate of their neural networks was 92.43 percent with 102 feature sets and 94.5 with 412 feature sets.

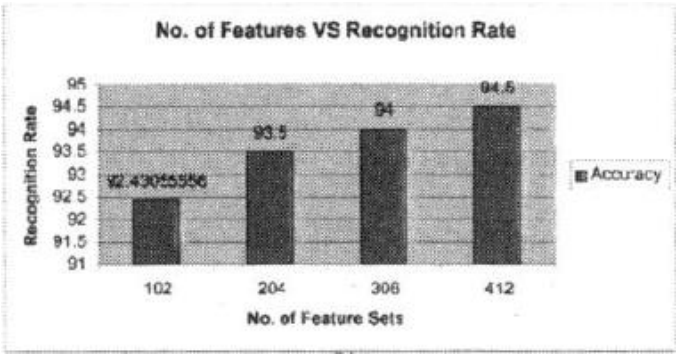


Figure 2.4 Success Rate in Various Numbers of Features [3]

Also, in "Intelligent Coin Identification System" [4], A. Khashman, et al. (2006) thought about coin identification to prevent confusion of coin physical similarities. They proposed Intelligent Coin Identification System (ICIS) to recognize coins in various-degree rotation. In Europe, they found a problem in slot machines. They accepted the new Turkish 1 Lira coin as a 2 euro coin even a 2 euro coin equaled to 4 new Turkish 2 Lira coin. The reason was the physical similarities of them. Their system used pattern averaging to generate inputs for neural network. Mainly, they train their system with images of euro coin in 3 degrees of ration, which are 30 degrees, 45 degrees, and 90 degrees. The neural network in their system was 3-layer back propagation network. It had 400 nodes for input layer, 25 nodes for hidden layer, and 2 nodes for output layer. The Figure 2.5 illustrates a topology to explain the system design of ICIS.

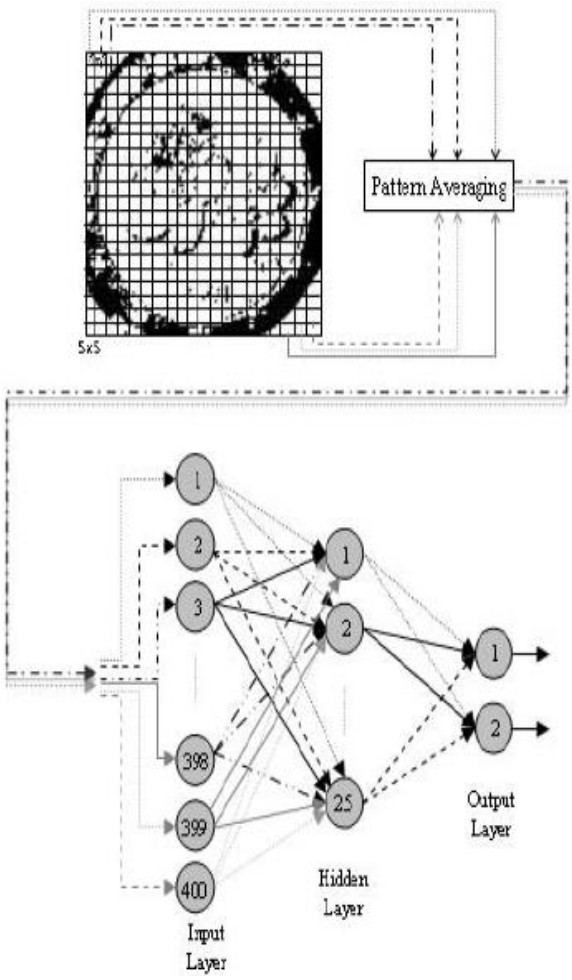


Figure 2.5 ICIS Topology [4]

From the testing from 80 coins, 96.3 percent of them or 77 coins were identified correctly, and the most successful was the result of 90-degree rotated coins.

In "Coin Recognition by using Artificial Neural Network" [5], T. N. Nimbhorkar and M. M. Bartere (2006) propose a research about coin classification as well, and they would like to work with Indian coins. They mentioned about the textons. The textons are fundamental micro structures of natural images. They claimed that textons are smallest elements creating texture in many images. Thy use two theories, which are Discrete Cosine Transform and Discrete Wavelet Transform to find those textons. Procedurally, their process was going like other researches. The process started with getting RGB values of an input image and converting to a greyscale image. After that, the image enhancement was applied before the system divided the enhanced image into 400 segments. They calculated average values from

each segment and fed them to their trained network. Unfortunately, they did not mention about the architecture of their neural network, but it is sure that the input layer size must be 400 nodes.

Coin recognition had been being interested. C. Cai-ming, et al. (2006) proposed a method to recognize coin with rotation tolerance in "A Coin Recognition System with Rotation Invariance" [6]. Fourier transform was used in their work. The concept was to get grey intensity values of a coin image in the radius directions. They kept them in every angle and project them on polar coordinates displayed in the Figure 2.6.

Figure 2.6 Projection of Radii in Coordinates [6]

The projected signals were calculated in the equation of Fourier Transform to analyze them in frequency domain. The coins with the same type would have similar amplitude sets like an example in the Figure 2.7 even they had different degrees of rotation.

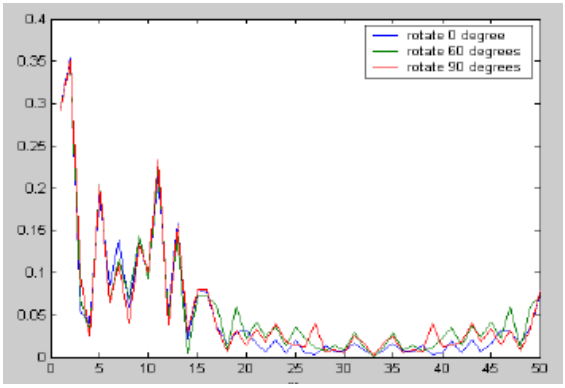


Figure 2.7 Fourier Coefficients of Original and Rotated [6]

Then they put those amplitude set into their neural network to recognize them. In the experiment, they had 900 coin images with three types of them. 200 of each type were managed to be the train set, so there would be 600 images for train set and 300 images for test set. 252 images or 83.3 percent of test set were classified correctly.

2.2 Amulet Recognition Researches

Now, we have already seen the researches about recognition pretty much. Then it is will be about the researches that have correspondent problems to this project.

P. Thumwarin et al. (2006) had an intention to build a system for recognizing Thai currency coins. They proposed a research named "A Robust Coin Recognition Method with Rotation Invariance" [7]. The point is the dataset they used. It did not contain only the images of currency coins, but it had amulet-coin images also. The examples of amulet coins are displayed in the figure 2.8.

Figure 2.8 Amulet Coins [7]

The technique they used to extract features was to collect grey-level intensities from many circles in different radii of a coin and analyze them with Fourier transform. The coefficient values are rotation-invariant features that can be matched by using Euclidean distance equation. For the test, they compared 69 coins to a reference amulet coin, and the system could classify between matched and mismatched type for 100 percent.

C. Korbuakaew (2007) presented a recognition system for powder Buddhist amulet in "Identification of Amulets with Special Feature Matching" [8]. His motivation came from a willing to conserve Thai Buddhist art. For the scope, his dataset contained only images of powder amulet of Buddha. To extract special features from amulet images, he used the technique of canny edge detection to proceed. 140 images were used as archetypes that input images can refer to. For matching process, he used correlation coefficient to perform. According to his experiment, the system's accuracy rate is 93.5 percent.

J. Wongkorsub et al. (2010) introduced "Buddhist amulet recognition system (BARS)" [9]. The system is dedicated to recognize amulet coins. 53 types of coins are collected in there dataset. To proceed feature extraction, the gray-scale conversion was used. The gray-scale values of each pixel were treated as features. The difference of gray-scale intensities between an input image and reference images were calculated to find correlation of them. Low difference value represented high correlation. 80 percent of testing data were classified correctly.

N. Srisupornwattana et al. (2013) published "Buddhist amulet coin recognition by using genetic algorithm (BACRS)" [10]. Mainly, the researchers considered two things as features, which are texture and edge. They applied canny edge detection to trace amulet's edges and grey level co-occurrence matrix (GLCM) to analyze amulet's texture. In matching phase, the system used a combination rule-based technique and genetic algorithm to classify their input images. In addition, they provide a graphic user interface for convenience in usage. Their GUI is shown in the figure 2.9. 130 styles of amulets were tested, and 91.53 percent of them are matched.

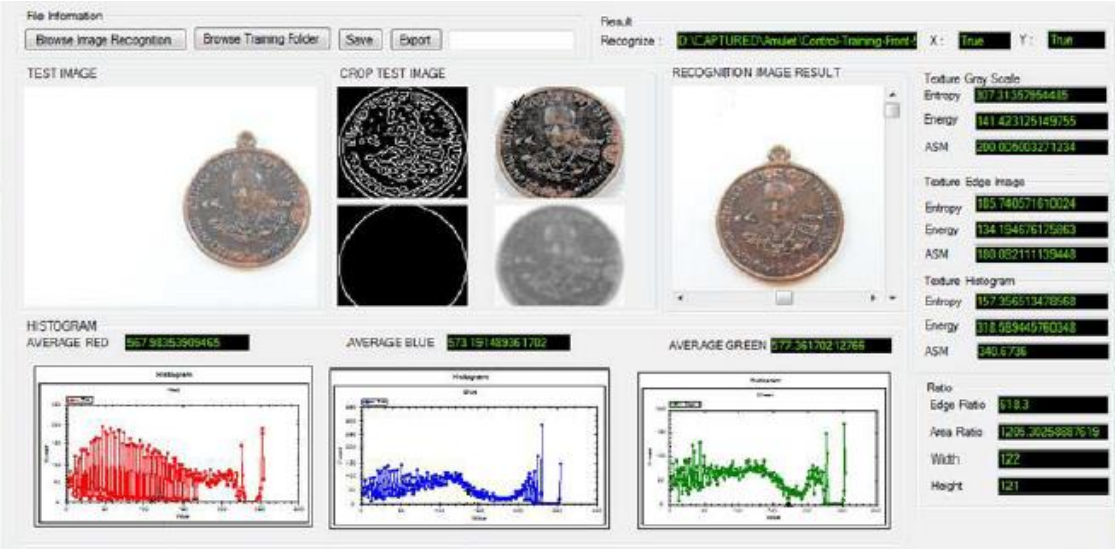


Figure 2.9 BACRS Graphic User Interface [10]

2.3 Analysis

According to the referred researches, the researchers implemented systems that were able to recognize Buddhist amulet with various techniques. Their result were also acceptable. However, there was no one who tried to apply ANN with Buddhist amulet problem. Moreover, considering on their feature extraction techniques, those methods did not provide robustness to their system. It was noticeable that their features were very sensitive to image transformation such as rotation, size scaling, and brightness adjustment. These transformation types are very common in real situation because users may enhance their pictures by using editor program. In addition, users cannot fix the same setting all the time to take an amulet picture. Just only 10-degree missing in amulet orientation, 10-centimeter difference of distance between camera and object, or dynamic brightness compensation can cause errors on those systems. At this point, the objective is not only to recognize amulet, but the system has to be less-or-non sensitive in those situations. The next section will introduce design and strategies to deal with those problems and explain the way to recognize the amulets.

CHAPTER III
DESIGN AND METHODOLOGY

This chapter is to describe the idea of design and process in the amulet recognition system. For system design, firstly, it is overview picture of the recognition system, which are used to classify amulet. Then the following part is an explanation about system architecture. The part will show main components in the software of the recognition system. The last part are descriptions for processes in each components.

3.1 System Overview

According to the Figure 3.1, at the first point, the sample images of Buddhist amulets are taken by using a digital camera or smart device to train the amulet recognition system. After training has been done, then user send his Buddhist amulet image to the system for getting recognizing result. The trained recognition system will classify the input image from user and return classification result via graphic user interface to user.

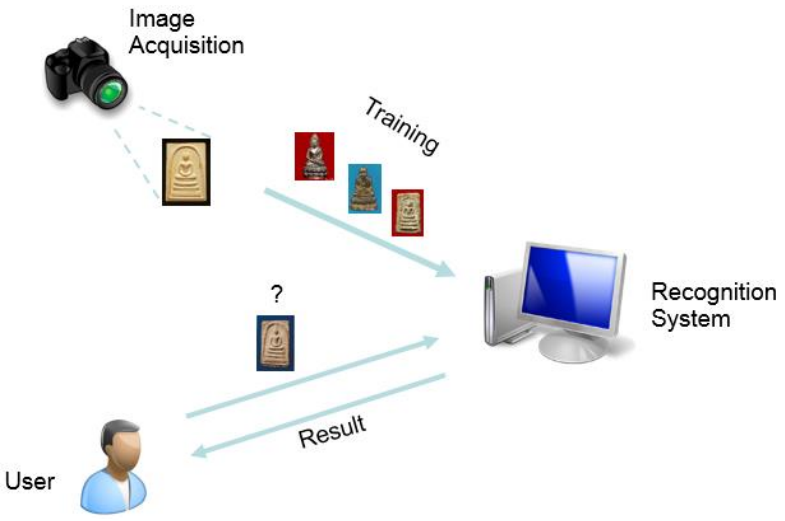


Figure 3.1 System Overview

3.2 Software Architecture

This part is going to be about each component in the Buddhist amulet recognition software. The details of processes in each module will be discussed in sequence. To make it easier to understand, we can present the software's components and their relationship with a structure chart in Figure 3.2.

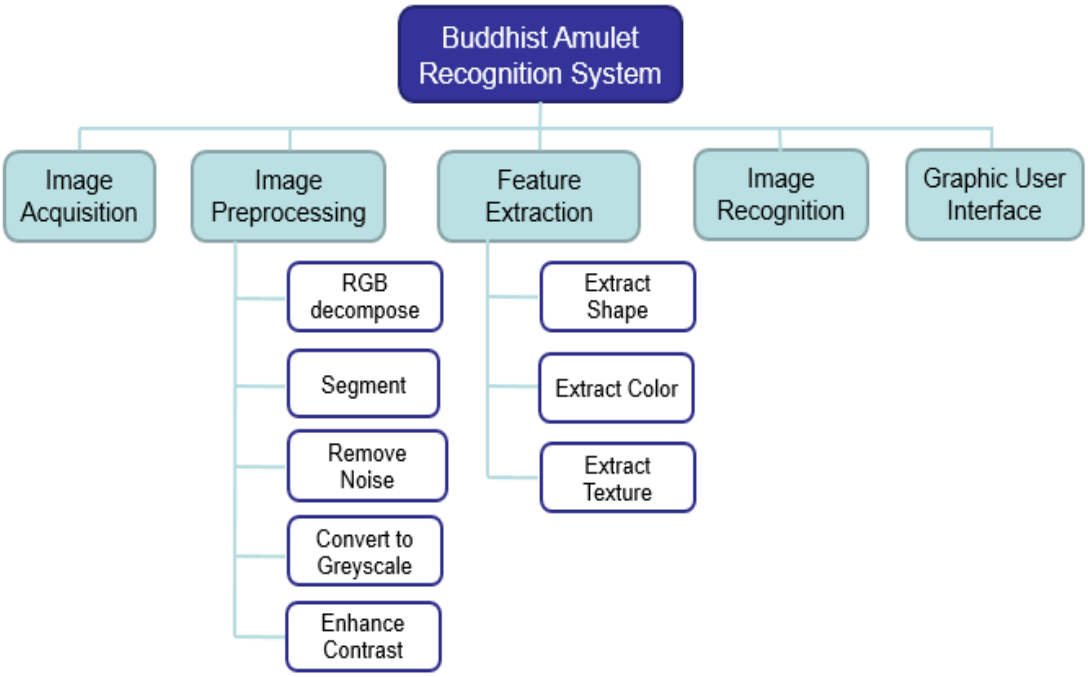


Figure 3.2 Structure Chart

According to the structure chart, the Buddhist Amulet Recognition System consists of five main modules, which are image acquisition, image preprocessing, feature extraction, image recognition, and graphic user interface. Sequentially, the first four modules work together to proceed full recognition process, but the graphic user interface module provide interaction between software and user.

Each module can be described briefly by following these points.

- Image acquisition – the first step of all to get an image of Buddhist amulet in digital form
- Image preprocessing – the procedure to prepare fed image in appropriate form and be ready for calculation

- Feature extraction – the process that extract feature values from an already prepared image
- Image recognition – the module to recognize images based on artificial neural network
- Graphic user interface – the component generating graphic user interface to support user in term of interaction with software

Looking at the image preprocessing module, it consists of five sub modules which are:

- RGB decompose – to separate digital image into RGB components
- Segment – to justify the area of object and background
- Remove noise – to remove noisy pixels in the image
- Convert to greyscale – to convert the color image to greyscale image
- Enhance contrast – to adjust contrast of image

Also, feature extraction module have its sub modules, which are:

- Extract shape – to get feature values of shape
- Extract color – to get feature values of color
- Extract texture – to get feature values of texture

Those referred modules have their own processing method. The next section will explain about the methodologies of each module and the related theories applied with them.

3.3 Methodology

This section is the explanation of process in each module. They are described about methods and theories related. The idea in this part can be applied in other platform because it is the description of concept and strategy.

3.3.1 Image Acquisition

To take an image of Buddhist amulet, we need a digital camera or smart device that have built-in camera. An amulet has to be placed on a plain blue screen showing its front side in any orientation. The camera's angle is bird-eye view or 90

degrees to the surface as shown in the Figure 3.3. It is suggested to have the amulet in the frame of camera at least 25 percent of whole picture with complete shape. For suitability, the camera can be moved nearer or further from the amulet because the system does not concern about distance between camera and amulet. During taking the picture, flash is needed to eliminate shadow. The brightness compensation can be reduced if the image is too bright caused by flash light. The mentioned setting is an imitation of amulet collector's photography; the comparison is in the Figure 3.4.

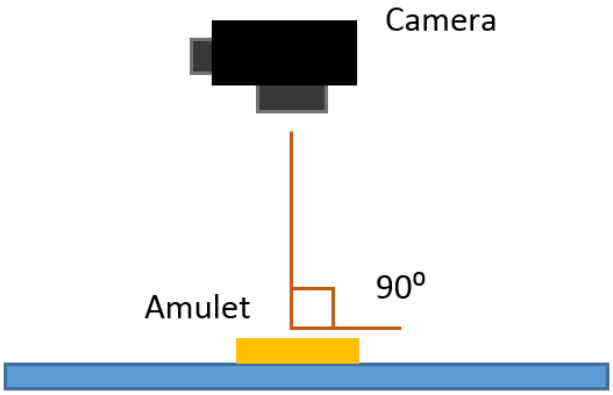


Figure 3.3 Camera's Angle

Figure 3.4 Collector's Setting and Imitation Setting

3.3.2 Image Preprocessing

The image preprocessing is to prepare the amulet image before feature extraction process. This module helps the system to get important data and work more accurately. There are five sub modules in this part, which are RGB decomposition, segmentation, noise removal, grey-scale conversion, and contrast enhancement.

In **RGB decomposition**, the input image is formed into RGB color components. Basically, RGB color model was invented to sense, represent, and display images in electronics devices. The idea is that we have primitive colors and they can be mixed up to reproduce a huge space of colors. The primitive colors consist of three components, which are red component, green component and blue component. The cube of color space in RGB model is shown in the Figure 3.5. The X, Y, and Z axes are mapped to red, green, and blue respectively.

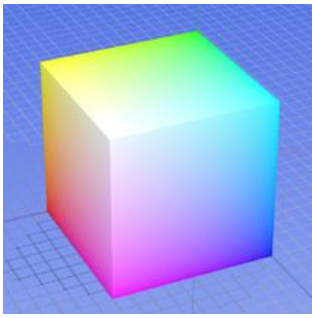
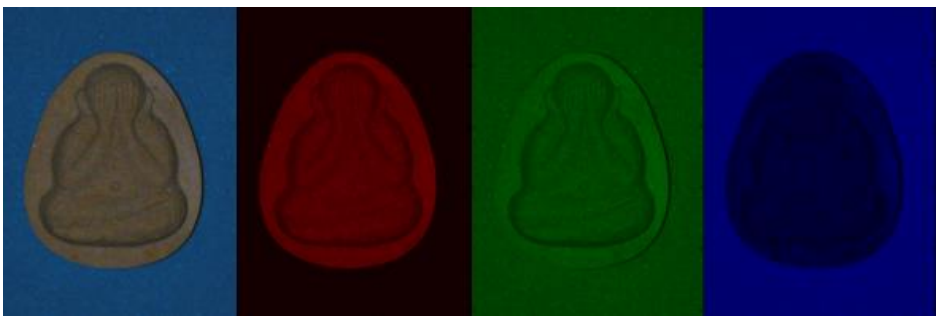


Figure 3.5 RGB Color Space

According to the RGB color model, a color image can be represented into 3 components as shown in the Figure 3.6 that (a) is original image, and the other images are its RGB components. Those 3 components are *m*-by-*n* matrices where *m* is image's height and *n* is image's width containing intensity values of their color for every pixel. The value of intensity is in an interval from 0 to 255. By this sub module, the system produces 3 intensity matrices of red, green, and blue channel to be processed in the next module.



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

Figure 3.6 RGB Components

Segmentation is to determine the object's area for proceeding in other modules. From the image acquisition module, the image background is blue, so it can be indicated that any pixels, which have intensity value in blue channel larger than red channel and green channel are pixels of background. To calculate background indicator, the equation 1 can be used.

$$Ind_{i,j} = \left(B_{i,j} - R_{i,j}\right) + (B_{i,j} - G_{i,j}) \quad (1)$$

Where i = image's row index

j = image's column index

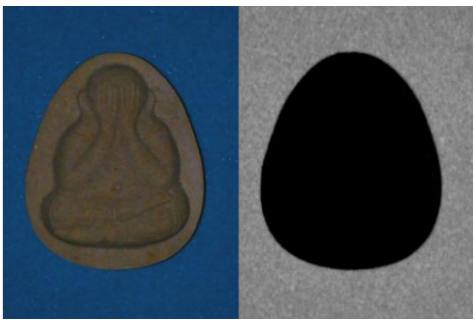
R = red value

G = green value

B = blue value

Grey = grey-level value.

From the Figure 3.7, the picture (b) is the background indicator of the picture (a). The high value of indicator (brighter area) represents background area. The low value (darker area) might be the object.

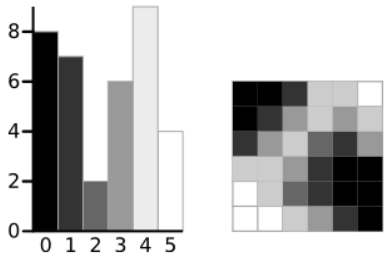


(a) (b)

Figure 3.7 Background Indicator

To justify that, it is needed to use Otsu's method to find a threshold value that separate background and object. The Otsu's threshold is the value that makes the sum of background variance and foreground variance as least as possible. For good understanding, we will show an example to explain the Otsu's method.

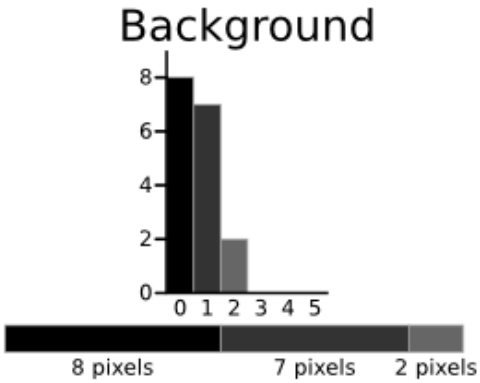
In the Figure 3.8, we assume that the picture (a) is the histogram of 6X6 image with only 6 greyscale level in the picture (b).



(a) (b)

Figure 3.8 Example for Otsu's Method [11]

Then we need to calculate weight and variance of foreground and background for every single possible threshold value. The Figure 3.9 and Figure 3.10 show the way of calculation at threshold = 3.



$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

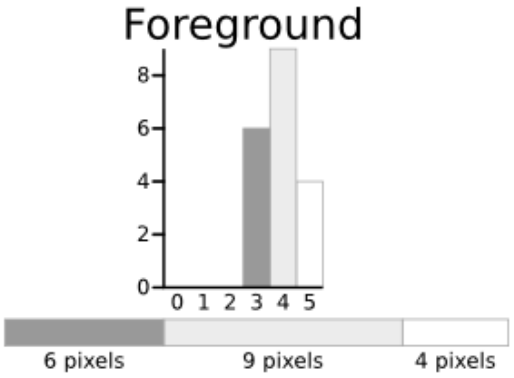
$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$$\text{Variance} \quad \sigma_b^2 = \frac{((0-0.6471)^2\times 8)+((1-0.6471)^2\times 7)+((2-0.6471)^2\times 2)}{17}$$

$$= \frac{(0.4187\times 8)+(0.1246\times 7)+(1.8304\times 2)}{17}$$

$$= 0.4637$$

Figure 3.9 Weight and Variance Calculation of Background [11]



$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

$$\text{Mean} \quad \mu_f = \frac{(3\times 6)+(4\times 9)+(5\times 4)}{19} = 3.8947$$

$$\text{Variance} \quad \sigma_f^2 = \frac{((3-3.8947)^2\times 6)+((4-3.8947)^2\times 9)+((5-3.8947)^2\times 4)}{19}$$

$$= \frac{(4.8033\times 6)+(0.0997\times 9)+(4.8864\times 4)}{19}$$

$$= 0.5152$$

Figure 3.10 Weight and Variance Calculation of Foreground [11]

After that, it is needed to calculate 'Within-Class Variance'. The Within-Class Variance can be simply computed the sum of the two variances multiplied by their associated weights. In the case that threshold value is 3, the Within-Class Variance computation is explained in the Figure 3.11.

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152$$

$$= 0.4909$$

Figure 3.11 Within-Class Variance Calculation [11]

In the example, there are six possible threshold values from 0 to 5, and we need find the Within Class Variance of them all. The figure 3.12 shows the results of within-class variance for all possible threshold values. From the calculation, we can know that the Within Class Variance at threshold = 3 is the lowest, which is the chosen value for Otsu's threshold.

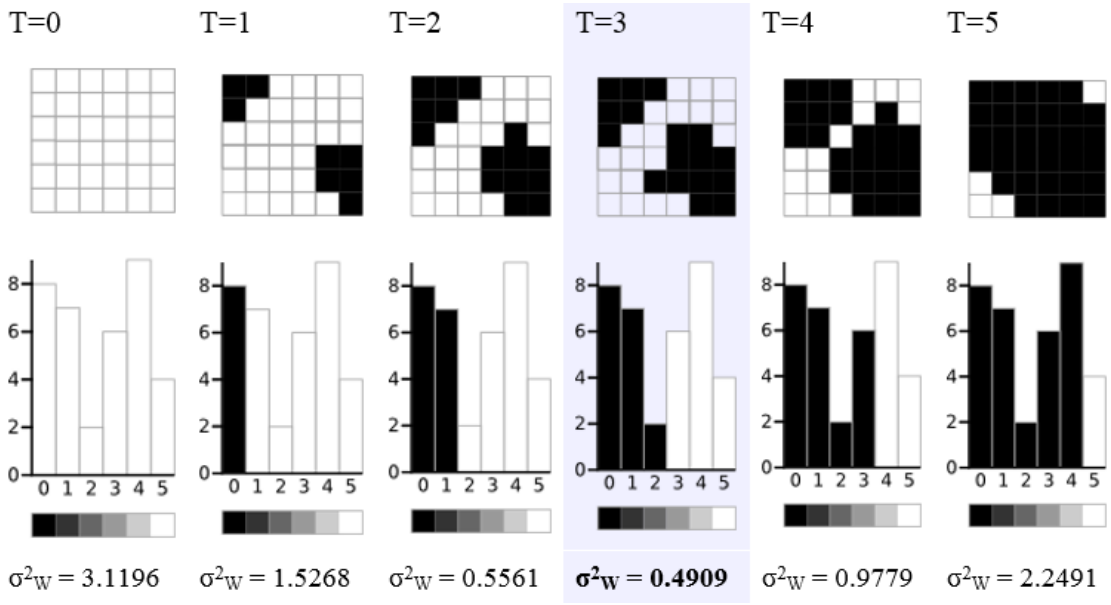
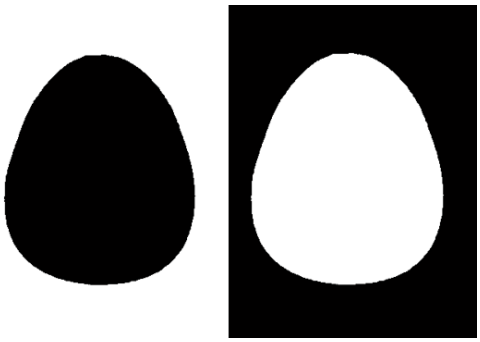


Figure 3.12 Within-Class Variance by each Threshold Value [11]

After Otsu's threshold are calculated, every pixel containing background indicator value higher than Otsu's threshold are mapped to be 1 or white otherwise they are 0 or black. That is called binary conversion. Then, those values have to be inverted, so the object's area is turned to be white, and the background is black instead. The Figure 3.13 is the example of binary image (a) and inversion (b).



(a) (b)

Figure 3.13 Binary image and Complemented Image

The segmented object's area still has some noise at its perimeter, so the median filter is used to remove them. **The noise removal** module uses median value of neighbor pixels to replace the interest pixel. The result is the segmented image that

have smooth object's perimeter. From this point, the white area given by this module is called as region of interest (ROI).

Grey-scale conversion is needed to analyze the texture of Buddha's image. The output from RGB the decomposition module is computerized again in this part. The output grey-scale image of this part can be produced by using equation 2.

$$Grey_{i,j} = 0.2989R_{i,j} + 0.5870G_{i,j} + 0.1140B_{i,j} \quad (2)$$

Where i = image's row index

j = image's column index

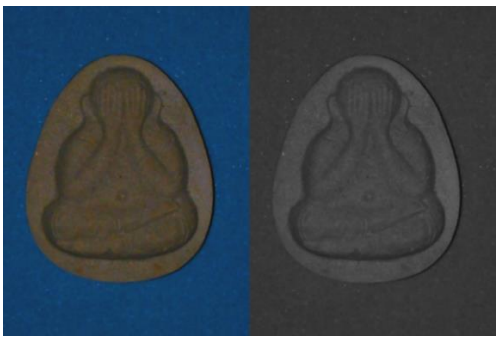
R = red value

G = green value

B = blue value

Grey = grey-level value

This equation is the way to convert color image to grey-scale image retaining image's luminance producing a result as the Figure 3.14 that (b) is the greyscale version of (a).



(a) (b)

Figure 3.14 Greyscale Conversion

For **contrast enhancement**, the sub module is to adjust the contrast of grey-scale image derived from grey-scale conversion. To enhance gray-scale contrast, a technique called histogram equalization is applied. The first is to create a histogram of the grey-scale image. Then it is applied with equation 3.

$$NewGrey_{i,j} = floor\left(255 \sum_{n=0}^{Grey_{i,j}} \frac{f_n}{N}\right) \quad (3)$$

Where i = image's row index

j = image's column

NewGrey = new grey value

Grey = original grey value

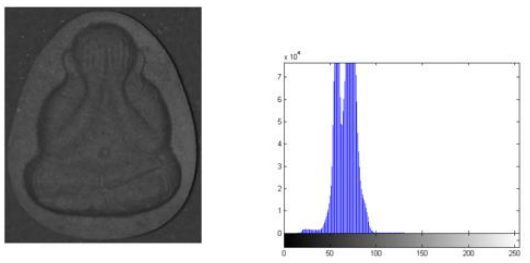
N = total number of pixels

f_n = number of pixels with intensity n

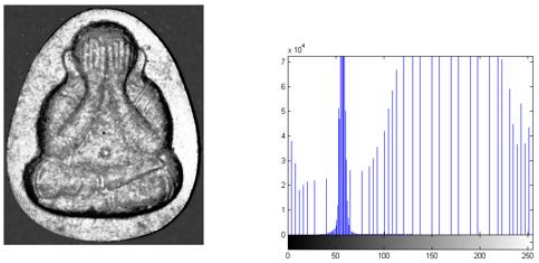
n = 1, 2, 3, …, 255

In addition, the process of histogram equalization in the contrast enhancement module is performed in region of interest defined by the enhanced segmented image only. The pixels outside the region are not counted.

The Figure 3.15 shows an example of histogram equalized image, but the background is not involved in the calculation. The picture of (b) is the contrast enhanced version of the (a) picture. It is noticeable that the intensities of the amulet in the example is stretched by using the technique.



(a)



(b)

Figure 3.15 Histogram Equalization

Claimed by A. Setkov et al, even the contrast is increased, it preserves the rank order of those pixels. Therefore, it is invariant to brightness adjustment because the regular brightness adjustment is a monotonous color change.

3.3.3 Feature Extraction

Feature extraction is the process that analyzes the data given by the image preprocessing module and come up with feature values. There are three types of focused feature, which are shape, color, and texture.

In this module, region of interest is very crucial because the background's data is not needed, and its completeness make the system extract the shape accurately. Incomplete segmentation will make this part works incorrectly.

In the **shape extraction**, there is two techniques needed to extract shape feature efficiently, which are centroid contour distance and Fourier transform. Those technique is a combination that provide rotational and scaling invariance in shape extraction. The centroid contour distance is the series of distance between object's centroid to all points on object's edge, and the equation # and the equation # show the mathematical way to find the centroid of the object and the distance between it and the object's perimeter.

$$Centroid = (\frac{\sum x}{N}, \frac{\sum y}{N}) \quad (4)$$

Where x = x coordinate in the amulet's area

y = y coordinate in the amulet's area

N = total number of coordinate in the amulet's area

$$Distance = \sqrt{(x_{centroid} - x_{perimeter})^2 + (y_{centroid} - y_{perimeter})^2} \quad (5)$$

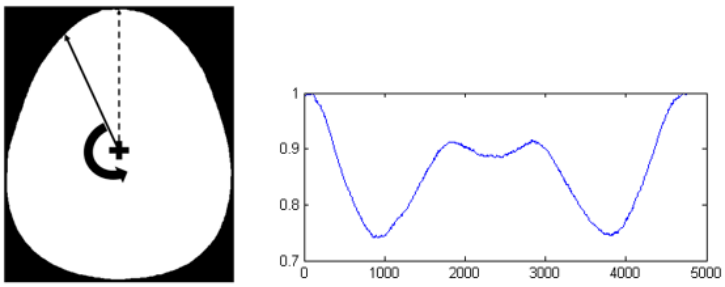
Where $x_{centroid}$ = x coordinate of the centroid

$y_{centroid}$ = y coordinate of the centroid

$x_{perimeter}$ = x coordinate of the perimeter

$y_{perimeter}$ = y coordinate of the perimeter

The series of distance can be traced around the amulet's perimeter like the (a) illustration and plotted as a graph of signal as (b) in the Figure 3.16.



(a) (b)

Figure 3.16 Centroid Contour Distance Tracing

The signal can be analyzed by using Fourier transform. The technique is to transform the signal from spatial domain into frequency domain. The peak in each term of frequency can be used describing the shape of the signal. The sequence of the peak values is called as Fourier descriptor (*FDT*).The equation 6 explains the calculation of the *FDT*.

$$FDT_n = \frac{\sum_{t=0}^{N-1} d_t e^{\frac{-j2\pi nt}{N}}}{N} \quad (6)$$

Where N = total number of perimeter points

t = perimeter point

d = centroid distance at t

n = number of term = 0, 1, 2, …, 30.

After *FDT* is calculated, its amplitude is Coefficient value that represents the shape of signal. Excluding term 0, we merely use the first 30 terms of frequency because it is sufficient to describe shape. The frequency terms that are higher than this will give too much details and noise. The normalized form of FDT is tolerant to scaled signal. We can normalize those terms by dividing them with term 0. For instance, we show a 30 terms of normalized FDT from the Figure 3.16 as a graph in the Figure 3.17. As a result, now, we have 30 feature values describing the shape of the input amulet.

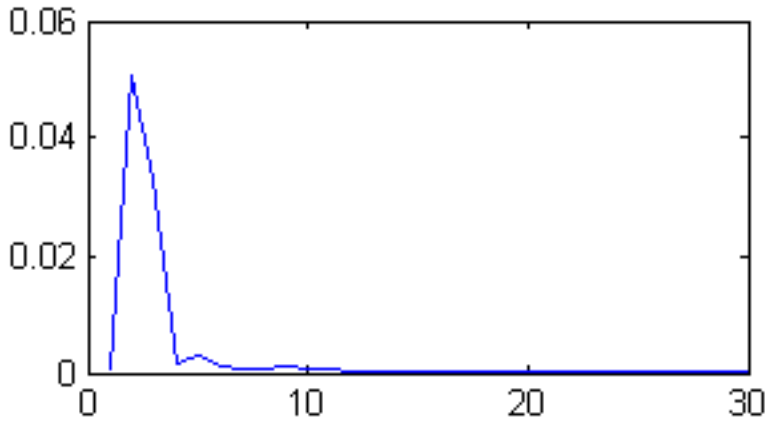


Figure 3.17 Normalized FDT

The DFT has a crucial a potential advantage. It is invariant to phrase shifting and amplitude rescaling. It means that the FDT's result is the same even the shape of the amulet is rotated or resized. The Figure 3.18 shows the comparison DFTs of the original shape, the rotated shape, and the resized shape.

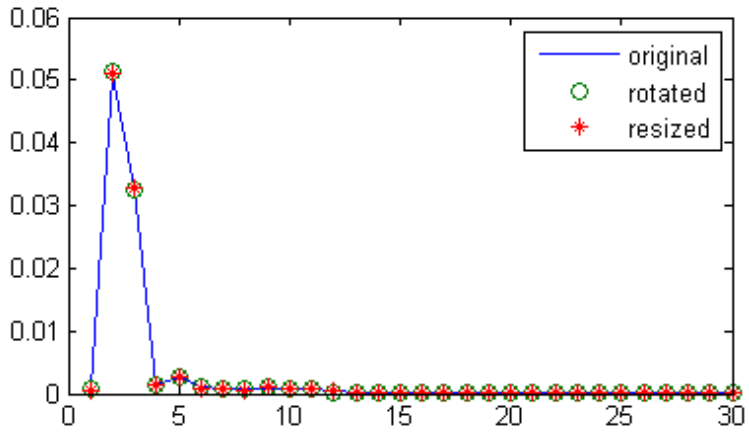


Figure 3.18 DFT Comparison of Original and Transformed Shape

In the **color feature extraction** module, The RGB colors in the region of interest is calculated into six features in this part, which are red-green difference, blue-red difference, green-blue difference standard deviation of red, standard deviation of green, and standard deviation of blue. Those can be expressed in the equation 7, 8, 9, 10, 11, and 12.

$$Red\ and\ Green\ Diff. = \mu_{Red} - \mu_{Green} \quad (7)$$

$$Blue\ and\ Red\ Diff. = \mu_{Blue} - \mu_{Red} \quad (8)$$

$$Green\ and\ Blue\ Diff. = \mu_{Green} - \mu_{Blue} \quad (9)$$

$$Red\ Standard\ Deviation = \sqrt{\frac{\sum_{i=1}^{n}(R_i - \mu_{Red})^2}{n-1}} \quad (10)$$

$$Green\ Standard\ Deviation = \sqrt{\frac{\sum_{i=1}^{n}(G_i - \mu_{Green})^2}{n-1}} \quad (11)$$

$$Blue\ Standard\ Deviation = \sqrt{\frac{\sum_{i=1}^{n}(B_i - \mu_{Blue})^2}{n-1}} \quad (12)$$

Where μ_{Red} = Mean of red intensities in the amulet's area

μ_{Green} = Mean of green intensities in the amulet's area

μ_{Blue} = Mean of blue intensities in the amulet's area

i = sequence of pixel in the amulet's area

R_i = Red intensity at i

G_i = Green intensity at i

B_i = Blue intensity at i

To explain 'difference' feature more clearly, for example, red-green difference is value of red value subtracted by green value, so, if the green value is larger than red value, the red-green difference is negative. That reflects how much more intense green is over red.

Those color features are used because they are invariant even the brightness of image is adjusted. The reason is that the differences and standard deviations of them are relative values. They stay being constant since the intensities of red, green, and blue are shifted together equally. This way can mitigate dynamic brightness compensation problem.

In the section of **texture extraction**, the strategy is to divide the grey-scale image in pieces, but regular dividing cannot provide robustness to the system. The tactic is to use circles to divide it into ring-based regions. Using the object's centroid as the center, draw 9 circles around it where d is longest centroid contour distance of the amulet by following radii which are: r, 0.8*r, 0.7*r, 0.6*r, 0.5*r, 0.4*r, 0.3*r, 0.2*r, and 0.1*r. The circles divide the amulet into nine rings as shown in Figure 3.19. Each ring's grey values are calculated to find mean and standard deviation.

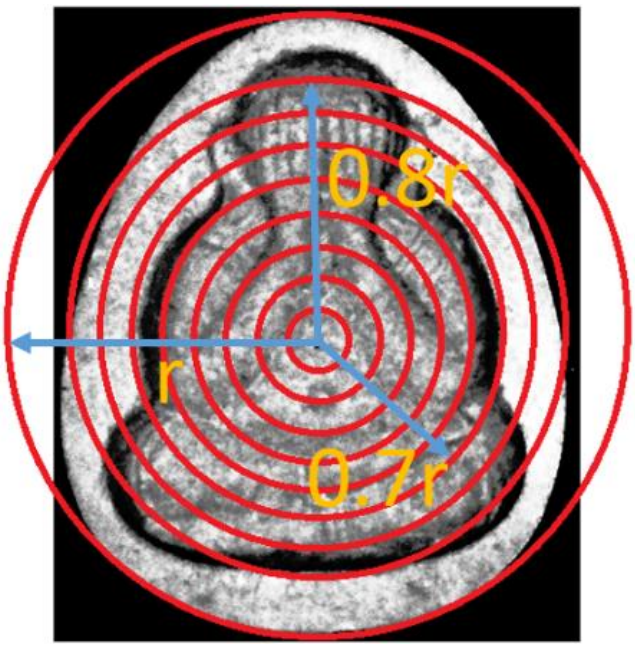


Figure 3.19 Circular Dividing

To sum up this part, there are totally 54 features derived from feature extraction process, which are 1-30 coefficient values (30 feature values), red-green difference, blue-red difference, green-blue difference, standard deviation of red, standard deviation of green, standard deviation of blue, mean values of ring 1-9 (9 feature values), and standard deviation values of ring 1-9 (9 feature values).

3.3.4 Image recognition

For recognition process, artificial neural network is chosen (ANN). The technique is mathematical model inspired by biological nervous system in animal brain.

In the concept of ANN, the network consists of many cells called ‘neurons’ interconnected to each other with ‘synapses’. The neurons and synapses work together. The electric signal from one neuron stimulates other neurons via synapses. If the synapses between them are strong, the neurons will be more sensitive to the incoming signal.

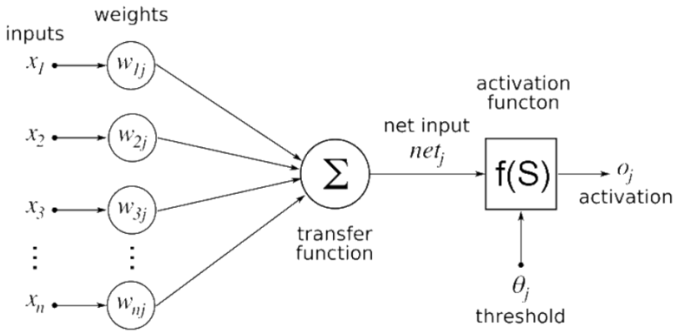


Figure 3.20 Feed Forward Process

From the concept, it comes up with the model in the Figure 3.20. The input values is sent into input neural nodes as input neurons. They are multiplied by weight of each connection before sending to the output node. The weights represent how strong of synapse between input neurons and output neurons. The output node will sum those weighted values including bias and calculate them by activation function for its output.

There are many different models of ANN, but two basic models are claimed in this research, which are perceptron and multi-layer perceptron. The perceptron is an ANN that has only two layers, which are input layer and output layer. Unlike perceptron, multi-layer perceptron has more than two layers, which are input layer, hidden layer(s), and output layer as shown in the Figure 3.21.

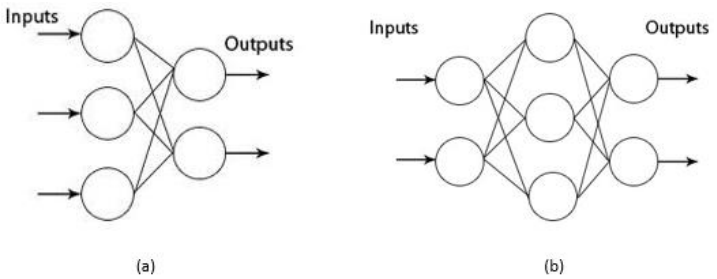


Figure 3.21 Perceptron and Multi-layer Perceptron

To design the perceptron and multi-layer perceptron for this research, the number of input nodes has to be 54 because of the number of involved features. The output nodes number is referred to the number of amulet classes in the experiment's scope which is 100. Each output nodes represent 1 amulet class from 100 classes, so the activated node is the result of the network. The result is concluded as unknown amulet if there is no output node activated. The activation function of perceptron and multi-layer perceptron are described in equation 13 and 14 respectively where x = sum of weighted input. The equation 13 is called as hard limiter that produce a value of 0 or 1. The equation 6 is non-linear function called Logistic Sigmoid function. Its value is in the interval [0 , 1].

$$f(x) = \begin{cases} 1, if\ x \geq 0 \\ 0\ if\ x < 0 \end{cases} \tag{13}$$

$$f(x) = \frac{1}{1+e^{-x}} \tag{14}$$

Where x = Weighted summation of inputs

Because the perceptron and multi-layer perceptron are supervised learning machine, they need expected result to learn. Back-propagation training is a common method in their learning process. In each iteration, the input data for training is passed into the network. Then it comes up with outputs and errors. The errors and gradients are calculated to adjust weights depending on each connected output nodes. How much changes of weights is related to learning factor. At the end of all iteration, it is needed to measure the network's performance by using performance function such as mean absolute error (MAE), or mean squared error (MSE). In this experiment, mean absolute error function is used for the perceptron and mean squared error for multi-layer perceptron. The process is running iteratively until the network reaches a stopping condition. For examples, the criteria can be to define maximum iteration or minimum MSE. In this experiment, the network is trained by 1,000 images of 100 amulet kinds (10 images per kind). Those train-set amulet images are original images taken by image acquisition process without any transformation and adjustment.

3.3.5 Graphic User Interface

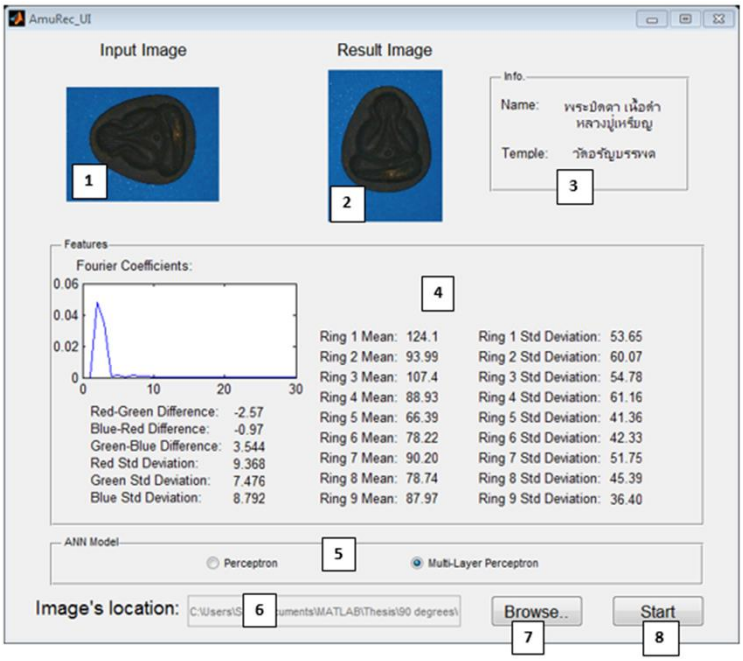


Figure 3.22 Graphic User Interface

From the Figure 3.22, there are six parts in the system's graphic user interface, which are 1) input image section, 2) result image section, 3) result information section, 4) feature values section, 5) ANN model section, 6) image location, 7) browse button, and 8) start button. To use the interface, the user needs to click at browse button select (7) an amulet picture in JPEG format. Then, the system will show the picture in the section of input image (1) with path of the file in image location (6) and calculate its features displayed on feature values section (4). To select model of neural network, the user can choose by clicking radio button in ANN model section (5). After that, the user has to click at start button (8) to initialize matching process. Finally, the result image is shown in result image section (2) with its information in result information section (3).

CHAPTER IV
IMPLEMENTATION

This chapter is discussions about four parts, which are materials, programming, training, and testing. In the materials part, it is description about necessary hardware and software for this research. The second part of this chapter is programming. It is the section that explain program algorithm and coding in each module. For the third, the training process is explained. The content focuses on train set, training configuration, and training performance. The last discussion for this chapter is testing that contains details of the test set and the evaluation method used to measure the amulet recognition system.

4.1 Hardware and Software Specification

The amulet pictures are taken by using mobile phone's camera (Sony Xperia M Dual). The image resolution is set to 2592×1944. For the computer, it will perform based on following specifications: 1) CPU Intel® Core™ i5-2410M Processor (3M Cache, up to 2.90 GHz), 2) Memory DDR3 4GB, and 3) Hard disk 640 GB. The needed software in implementation is MATLAB R2013a.

4.2 Programming

The explanation for this part has two kinds, which are flow chart and programming code. The flowcharts describe the procedures occurring in the software. Anyway, the algorithms can be applied with other platforms. In addition, we show the coding implemented in the platform of MATLAB that is a suitable software for digital image processing and artificial intelligence area.

4.2.1 Image Acquisition

In this module, the system requests the file location from the user, and the system will access to the file system to copy the file into the memory. The procedure is shown in the Figure 4.1.

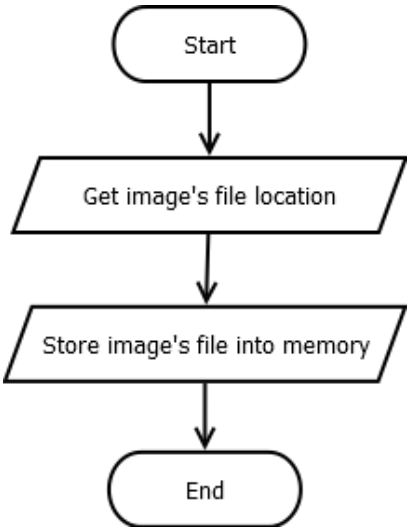


Figure 4.1 Image Acquisition Flowchart

In MATLAB, we can use the instruction in the Figure 4.2 to follow the flowchart in the Figure 4.1.

```
im = imread(imFileName);
```

Figure 4.2 Image Acquisition Coding

4.2.2 Image Preprocessing

According to the Figure 4.3, to prepare the input image, we need to get the components of red, blue, and green for the first. Then we calculate background indicators and convert them to binary. After that, we invert the binary image before enhancing with median filter. For the last part in this module, the system convert the image into grey-scale image, and it adjust the image's contrast by histogram equalization.

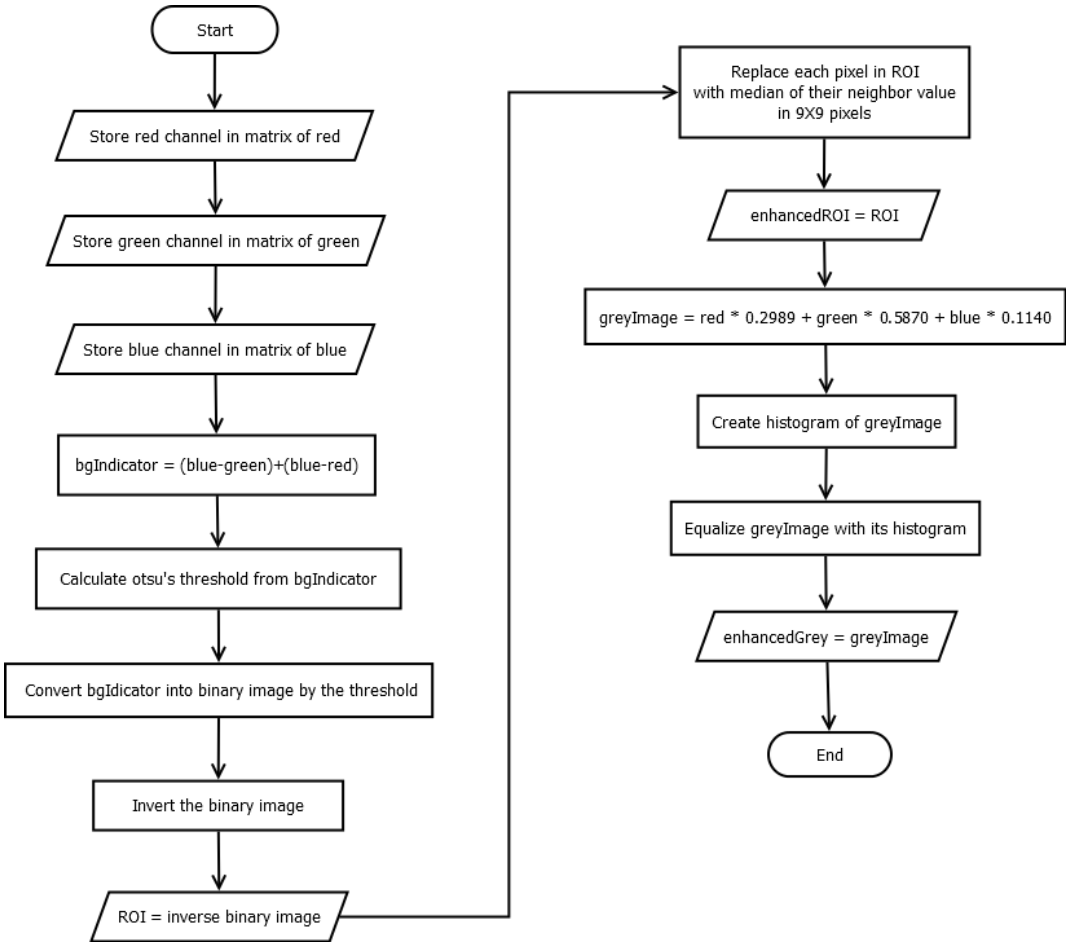


Figure 4.3 Image Preprocessing Flowchart

The process can be implemented in MATLAB following instructions in the Figure 4.4.

```
%------RGB Decomposition--------
redIm = im(:,:,1);
greenIm = im(:,:,2);
blueIm = im(:,:,3);

%------Segmentation---------
ROI = segmentation(redIm, greenIm, blueIm);

%------Grayscale Conversion--------
grayIm = rgb2gray(im);

%------------Enhancement----------------------------
enhancedBinROI = medfilt2(ROI,[9,9]);
enhancedGray = roifilt2(grayIm, enhancedBinROI, 'histeq');
```

Figure 4.4 Image Preprocessing Coding

```
function out = segmentation(red, green, blue)
        bgIndicator = (blue-green)+(blue-red);
        binary = im2bw(bgIndicator);
        ROI = imcomplement(binary);
        out = ROI;
end
```

Figure 4.5 Segmentation Function

In addition, the Figure 4.5 describe about coding in 'segmentation' function that calculate background indicator, convert to binary image, and complement the image.

4.2.3 Feature Extraction

In the feature extraction module, we will explain its sub modules separately because they have many details to be shown.

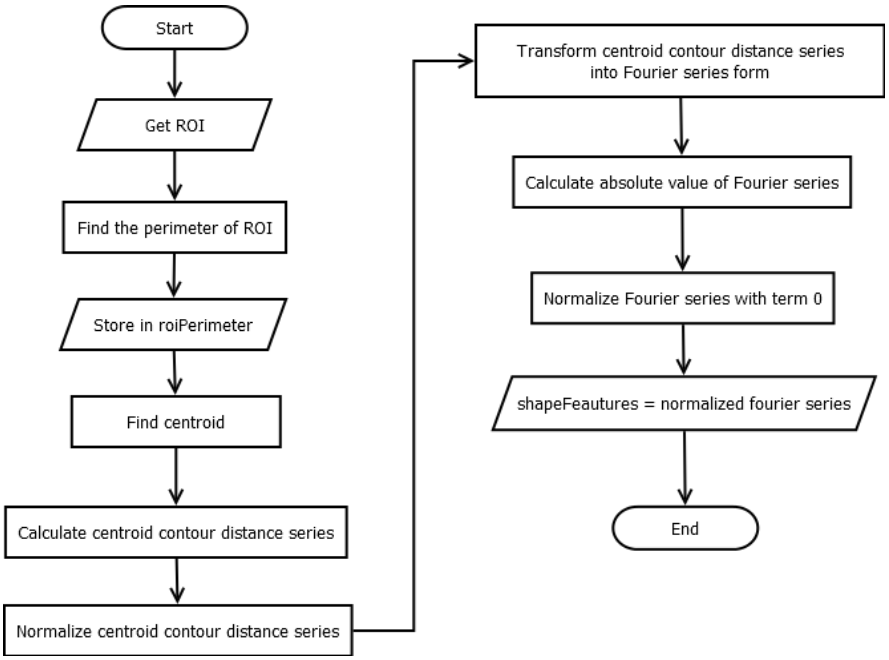


Figure 4.6 Shape Extraction Flowchart

From the Figure 4.6, the process of shape extraction begins by getting ROI then find its perimeter and centroid. After the perimeter and centroid are found, the system calculate the distance between them in each point of perimeter. The derived series of centroid contour is transformed into Fourier coefficients and be normalized.

```
imPerim = bwperim(enhancedBinROI);
regProps = regionprops(enhancedBinROI,'centroid');
centroid = regProps.Centroid;
centroidX = round(centroid(1));
centroidY = round(centroid(2));
distanceList = getContDistance(imPerim, centroidX, centroidY);
normalizedDistanceList = distanceList/max(distanceList);
fourierDesc = fft(normalizedDistanceList);
FDT = abs(fourierDesc);
normalizedFDT = FDT/FDT(1);
first30FDT = normalizedFDT(2:31);
```

Figure 4.7 Shape Extraction Coding

The Figure 4.7 shows the code of shape extraction process in MATLAB platform following the process in the Figure 4.6.

In the code, there are sub process called ‘getContDistance’. It is a function to get series of centroid contour distance. Its process is based on the Figure 4.8.

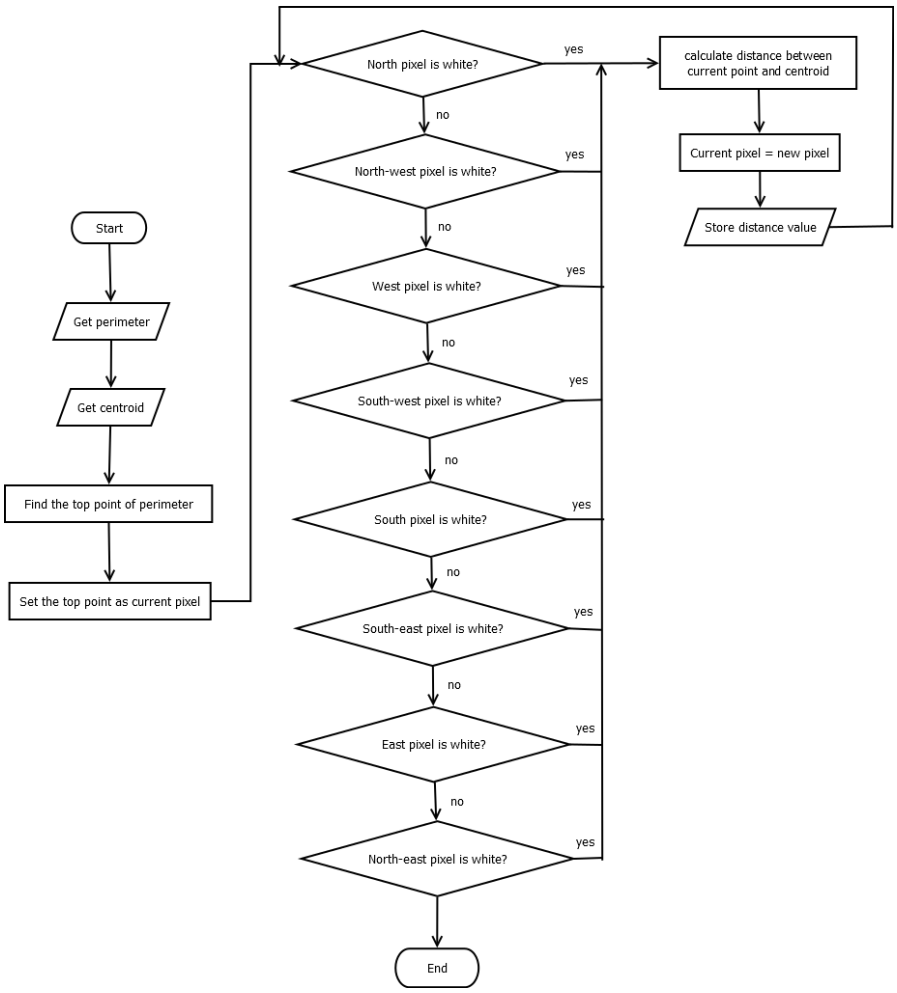


Figure 4.8 Centroid Contour Distance Tracing Flowchart

```
function out = getContDistance(inputIm, centroidX, centroidY)
  distanceList = [];
  firstX = centroidX;
  firstY = find(inputIm(:,centroidX),1,'first');
  currentX = firstX;
  currentY = firstY;
  converged = false;
  while ~converged
      if inputIm(currentY-1, currentX) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentY = currentY-1;
      elseif inputIm(currentY-1, currentX-1) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentY = currentY-1;
          currentX = currentX-1;
      elseif inputIm(currentY, currentX-1) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentX = currentX-1;
      elseif inputIm(currentY+1, currentX-1) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentY = currentY+1;
          currentX = currentX-1;
      elseif inputIm(currentY+1, currentX) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentY = currentY+1;
      elseif inputIm(currentY+1, currentX+1) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentY = currentY+1;
          currentX = currentX+1;
      elseif inputIm(currentY, currentX+1) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentX = currentX+1;
      elseif inputIm(currentY-1, currentX+1) == 1
          distance = pdist([centroidX centroidY ; currentX currentY]);
          distanceList = [distanceList distance];
          inputIm(currentY,currentX) = 0;
          currentY = currentY-1;
          currentX = currentX+1;
      else
          converged = true;
      end
  end
  out = distanceList;
```

Figure 4.9 Centroid Contour Distance Tracing Function

The procedure in the Figure 4.8 can be wrote in MATLAB as shown as in the Figure 4.9.

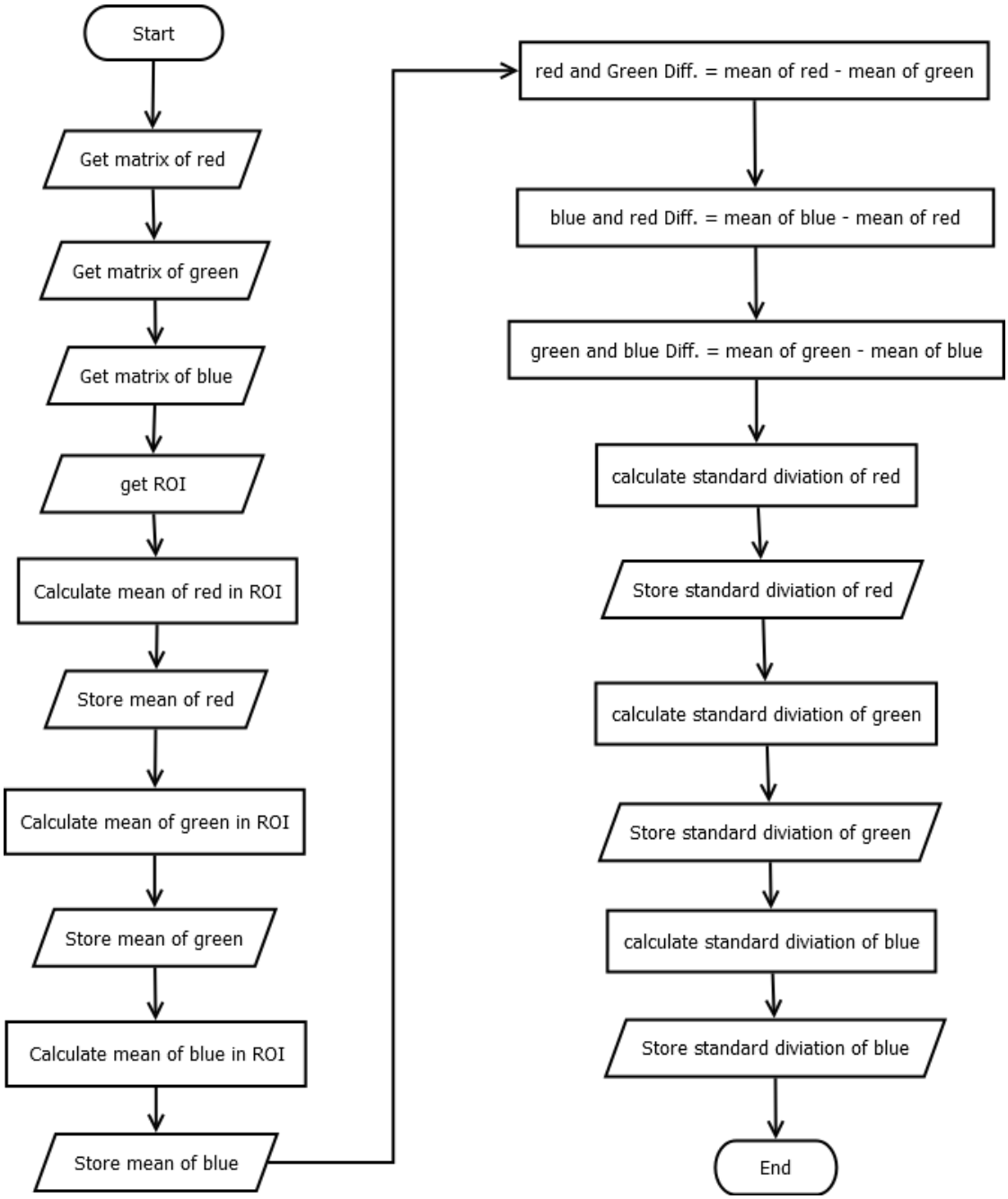


Figure 4.10 Color Extraction Flowchart

After the process of shape extraction is done, the system starts the process of color extraction as shown in the Figure 4.10.

```
%-------------get color features---------------------

vectorRed = double(redComposition(:));
vectorGreen = double(greenComposition(:));
vectorBlue = double(blueComposition(:));
meanRed = mean(vectorRed(enhancedBinROI));
meanGreen = mean(vectorGreen(enhancedBinROI));
meanBlue = mean(vectorBlue(enhancedBinROI));
redGreenDiff = meanRed-meanGreen;
blueRedDiff = meanBlue-meanRed;
greenBlueDiff = meanGreen-meanBlue;
stdRed = std(vectorRed(enhancedBinROI));
stdGreen = std(vectorGreen(enhancedBinROI));
stdBlue = std(vectorBlue(enhancedBinROI));
```

Figure 4.11 Color Extraction Coding

According to the Figure 4.11, the color extraction process is described in the MATLAB coding following the flow chart.

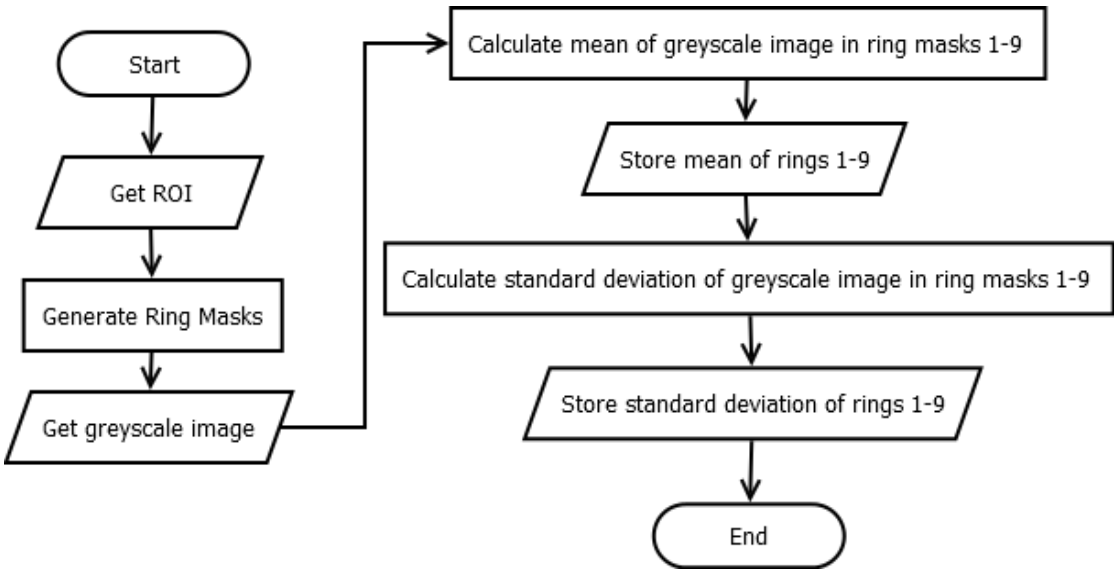


Figure 4.12 Texture Extraction Flowchart

The last part of the feature extraction module is the texture extraction process. Based on the Figure 4.12, the process starts with getting ROI and generate ring masks to divide the grey-scale image into 9 rings. After we have obtained those rings, they will be used as filters to calculate mean values and standard deviation values in divided regions.

```
%-------------get texture--------------------

circularMask = circularCrop(enhancedBinROI, centroidX, centroidY,
max(distanceList));
vectorGray = double(enhancedGray(:));
meanRing = [];
stdRing = [];
      for i = 1:9
          meanRing(i) = mean(vectorGray(circularMask(:,:,i)));
          stdRing(i) = std(vectorGray(circularMask(:,:,i)));
      end
out = [first30FDT redGreenDiff blueRedDiff greenBlueDiff stdRed
stdGreen stdBlue meanRing stdRing];
```

Figure 4.13 Texture Extraction Coding

The Figure 4.13 shows the code of texture extraction process implemented in the system.

It can be seen that the 'circularCrop' function generates the ring masks before calculating mean and standard deviation. The process inside the function is shown in the Figure 4.14.

At the beginning, the function get ROI, longest centroid contour distance, and centroid. Then the system create the circles that their radii are equal to 80 percent, 70 percent, 60 percent, 50 percent, 40 percent, 30 percent, 20 percent, and 10 percent of longest centroid contour distance. To create ring 1, the system subtract ROI with 80-percent circle. For ring 2, the system subtract ROI with 70-percent circle and ring 1. The next ring is created from subtracting ROI with 60-percent circle, ring 1, and ring 2. The fourth ring is generated by subtract ROI with 50-percent circle, ring 1, ring 2, and ring 3. The ring 5 is subtraction of ROI with 40-percent circle, ring 1, ring 2, ring 3, and ring 4. For ring 6, it is subtraction of ROI with 30-percent circle, ring 1, ring 2, ring 3, ring 4, and ring 5. Subtracted ROI with 20-percent circle, ring 1, ring 2, ring 3, ring 4, ring 5, and ring 6 is ring 7. Ring 8 is produced by subtracting ROI with 10-percent circle, ring 1, ring 2, ring 3, ring 4, ring 5, ring 6, and ring 7. For the last one, the ring 9 is 10-percent circle.

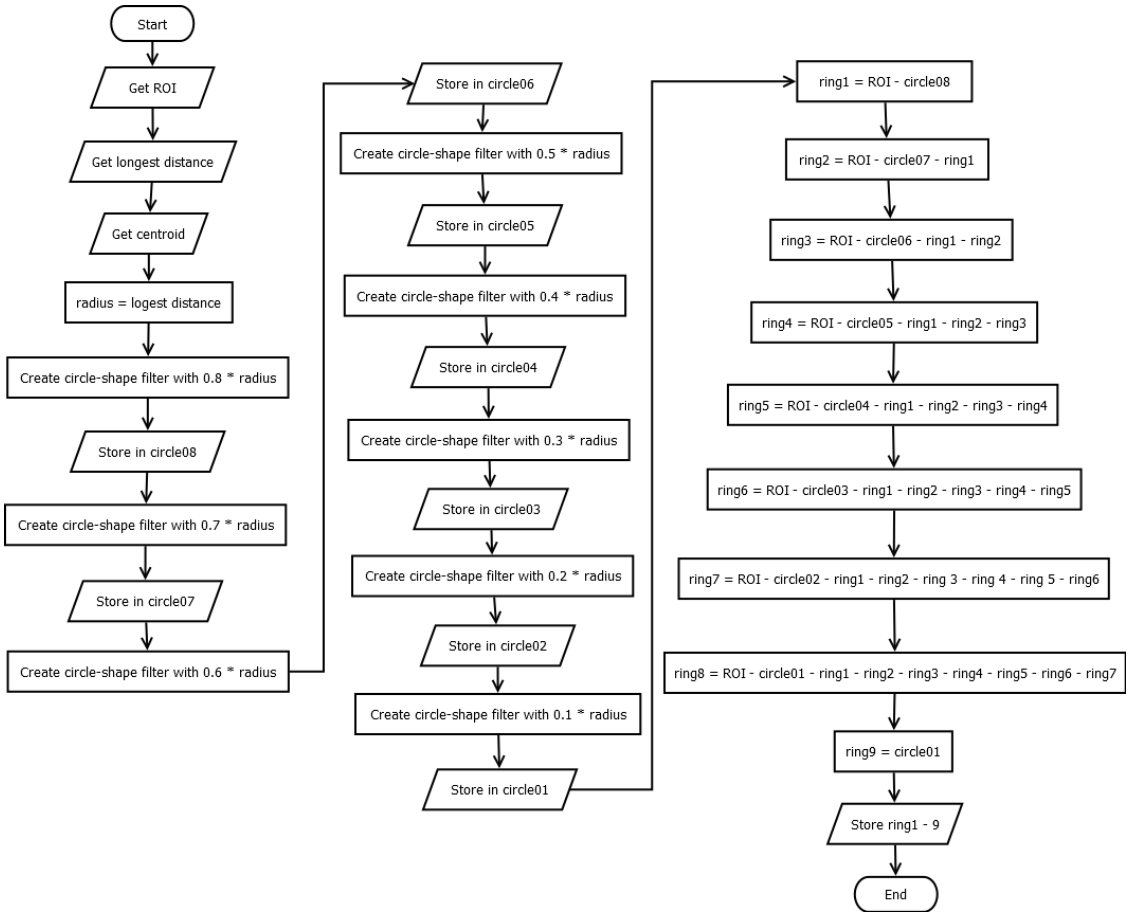


Figure 4.14 Circular Dividing Flowchart

The flowchart is applied into the code of ‘outputMask’ function as shown in the Figure 4.15.

```
function outputMask = circularCrop( inputBin, centroidX, centroidY,
radius )
      imageSize = size(inputBin);
      ci = [centroidY, centroidX, 0.8*radius, 0.7*radius, 0.6*radius,
0.5*radius, 0.4*radius, 0.3*radius, 0.2*radius, 0.1*radius];
      [xx,yy]= ndgrid((1:imageSize(1))-ci(1),(1:imageSize(2))-ci(2));
      mask08 = uint8((xx.^2 + yy.^2)<ci(3)^2);
      mask07 = uint8((xx.^2 + yy.^2)<ci(4)^2);
      ask06 = uint8((xx.^2 + yy.^2)<ci(5)^2);
      mask05 = uint8((xx.^2 + yy.^2)<ci(6)^2);
      mask04 = uint8((xx.^2 + yy.^2)<ci(7)^2);
      mask03 = uint8((xx.^2 + yy.^2)<ci(8)^2);
      mask02 = uint8((xx.^2 + yy.^2)<ci(9)^2);
      mask01 = uint8((xx.^2 + yy.^2)<ci(10)^2);
      outputMask(:,:,1) = logical(uint8(inputBin) - mask08);
      outputMask(:,:,2) = logical(uint8(inputBin) - mask07 -
uint8(outputMask(:,:,1)));
      outputMask(:,:,3) = logical(uint8(inputBin) - mask06 -
uint8(outputMask(:,:,1)) - uint8(outputMask(:,:,2)));
      outputMask(:,:,4) = logical(uint8(inputBin) - mask05 -
uint8(outputMask(:,:,1)) - uint8(outputMask(:,:,2)) -
uint8(outputMask(:,:,3)));
      outputMask(:,:,5) = logical(uint8(inputBin) - mask04 -
uint8(outputMask(:,:,1)) - uint8(outputMask(:,:,2)) -
uint8(outputMask(:,:,3)) - uint8(outputMask(:,:,4)));
      outputMask(:,:,6) = logical(uint8(inputBin) - mask03 -
uint8(outputMask(:,:,1)) - uint8(outputMask(:,:,2)) -
uint8(outputMask(:,:,3)) - uint8(outputMask(:,:,4)) -
uint8(outputMask(:,:,5)));
      outputMask(:,:,7) = logical(uint8(inputBin) - mask02 -
uint8(outputMask(:,:,1)) - uint8(outputMask(:,:,2)) -
uint8(outputMask(:,:,3)) - uint8(outputMask(:,:,4)) -
uint8(outputMask(:,:,5)) - uint8(outputMask(:,:,6)));
      outputMask(:,:,8) = logical(uint8(inputBin) - mask01 -
uint8(outputMask(:,:,1)) - uint8(outputMask(:,:,2)) -
uint8(outputMask(:,:,3)) - uint8(outputMask(:,:,4)) -
uint8(outputMask(:,:,5)) - uint8(outputMask(:,:,6)) -
uint8(outputMask(:,:,7)));
      outputMask(:,:,9) = logical(mask01);
```

Figure 4.15 Circular Dividing Coding

4.2.4 Image Recognition

This module is an important module because it is the module that proceed classification from the fed input patterns.

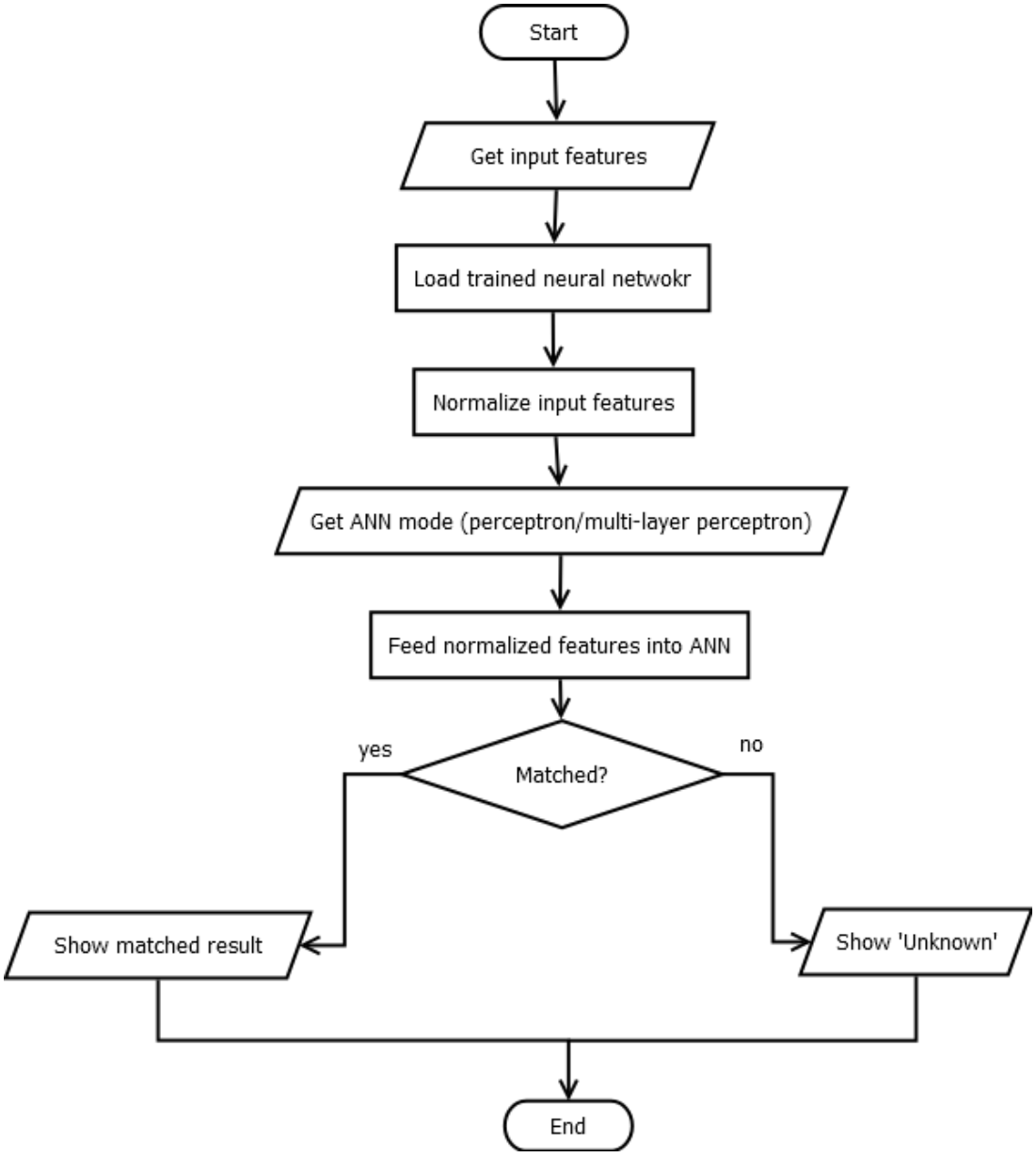


Figure 4.16 Image Recognition Flowchart

The process need to feature values and trained networks for the first. Then the feature values will be normalized before they are sent into the chosen network. If the values is matched with a class in the set of trained amulets, the information of the amulet will be shown. Nevertheless, the result is ‘unknown’.

```
load('trainedNet');
range = maxFeatures - minFeatures;
features=handles.features;
scaledFeatures = (features-minFeatures)./range;
outputs = [];
v1 = get(handles.radiobutton1, 'Value');
v2 = get(handles.radiobutton2, 'Value');
      if v1 == 1
        outputs = Zperceptron(scaledFeatures');
        outputs = round(outputs);
      end
      if v2 == 1
        outputs = Z54net(scaledFeatures');
        outputs = round(outputs);
      end
      if sum(outputs)==1
        [v i]=max(outputs);
        amuName = nameList(i,1);
        templeName = nameList(i,2);
        set(handles.text81,'String',amuName);
        set(handles.text82,'String',templeName);
        imResultName = strcat(sprintf('%04d',i*15),'.jpg');
        imResultPath = strcat('./result_image/r',imResultName);
        imResult = imread(imResultPath);
        axes(handles.axes2);
        imshow(imResult);
      else
        imResult = imread('./result_image/r0000.jpg');
        axes(handles.axes2);
        imshow(imResult);
        set(handles.text81,'String','unknown');
        set(handles.text82,'String','unknown');
```

Figure 4.17 Image Recognition Coding

The Figure 4.17 show the implementation of image recognition coding in MATLAB following the designed process in the flowchart.

4.2.5 Graphic User Interface

This part is description of graphic user interface (GUI) implementation. The GUI is the way that user can interact with the system. Its process is explained from flowchart in the Figure 4.18.

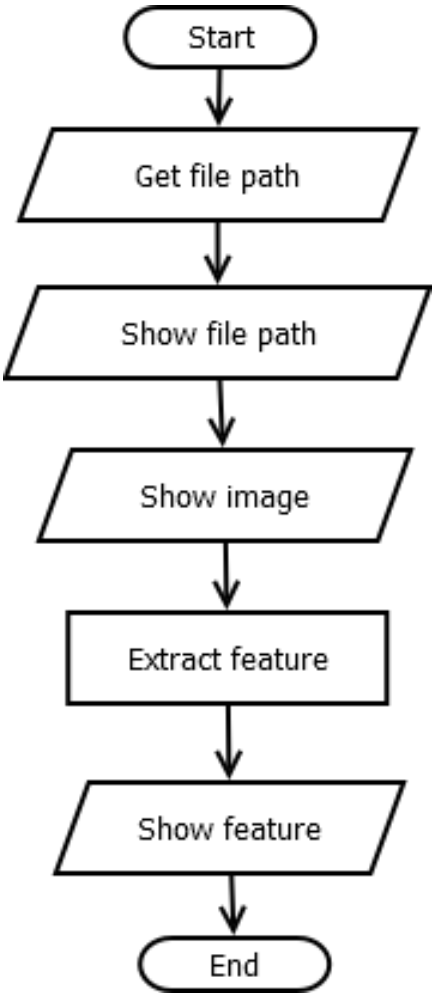


Figure 4.18 Browse Button Flowchart

The flow chart shows that the system get file path and show to the user with its image. Then the extract feature process is called, and its result will be displayed to the user as well.

```
function pushbutton1_Callback(hObject, eventdata, handles)
[FileName,PathName] = uigetfile('*.jpg','Select an image');
axes(handles.axes1);
set(handles.edit1,'String',[PathName FileName]);
imshow([PathName FileName]);
features = extractF([PathName FileName]);
axes(handles.axes3);
plot(features(1:30));
set(handles.text35,'String',features(31));
set(handles.text36,'String',features(32));
set(handles.text37,'String',features(33));
set(handles.text38,'String',features(34));
set(handles.text39,'String',features(35));
set(handles.text40,'String',features(36));
set(handles.text41,'String',features(37));
set(handles.text42,'String',features(38));
set(handles.text44,'String',features(39));
set(handles.text45,'String',features(40));
set(handles.text54,'String',features(41));
set(handles.text55,'String',features(42));
set(handles.text56,'String',features(43));
set(handles.text57,'String',features(44));
set(handles.text58,'String',features(45));
set(handles.text68,'String',features(46));
set(handles.text69,'String',features(47));
set(handles.text70,'String',features(48));
set(handles.text71,'String',features(49));
set(handles.text72,'String',features(50));
set(handles.text73,'String',features(51));
set(handles.text74,'String',features(52));
set(handles.text75,'String',features(53));
set(handles.text76,'String',features(54));
handles.features = features;
guidata(hObject,handles)
```

Figure 4.19 Browse Button Coding

4.3 Training

In this experiment, the train set is samples of 1,000 Buddhist amulet images. It consists of 100 kinds of amulets with 10 images per each. The name list with class number of the amulets is shown in the Table 4.1.

Table 4.1 Amulet Name List

No.	Name	Pronunciation
1.	พระปิดตา เนื้อดำ หลวงปู่เหรียญ	Pra-pid-ta-nua-dam-luang-poo-rian
2.	พระปิดตา หลวงปู่เหรียญ	Pra-pid-ta-luang-poo-rian
3.	พระสมเด็จ หลวงปู่เหรียญ	Pra-som-dej-luang-poo-rian
4.	พระนาคปรก หลวงปู่เหรียญ	Pra-nark-prok-luang-poo-rian
5.	ภาพพิมพ์ หลวงปู่เหรียญ	Parp-pim-luang-poo-rian
6.	หลวงปู่เหรียญ วัดอรัญบรรพต	luang-poo-rian-wat-arun-ya-ban-pod
7.	หลวงปู่ทวด วัดช้างไห้	luang-poo-tuad-wat-chang-hai
8.	หลวงปู่ทวด วัดช้างไห้	luang-poo-tuad-wat-chang-hai
9.	สมเด็จ ส.ธ.ประจำวัน	Som-dej-sor-tor-pra-jam-wan
10.	สมเด็จ ส.ธ.ประจำวัน	Som-dej-sor-tor-pra-jam-wan
11.	หลวงพ่อเปิ่น วัดลางบางพระ	Luang-por-pern-wat-lang-bang-phra
12.	พระครูอุดมวิสุทธิธรรม วัดทุ่งนาใหม่	Phra-kru-udom-wi-sut-ti-tam
13.	สมเด็จวัดบวรนิเวศ วัดบวรนิเวศ	Som-dej-wat-ba-worn-ni-wed
14.	พระมงคลบพิตร รุ่นกรุงเก่า	Phra-mong-kol-bor-pid
15.	พระมงคลบพิตร รุ่นกรุงเก่า	Phra-mong-kol-bor-pid
16.	พระมงคลบพิตร รุ่นกรุงเก่า	Phra-mong-kol-bor-pid
17.	หลวงปู่ภู วัดอินทร์	Luang-poo-phu-wat-in
18.	เจ้าคุณนร. วัดเทพศิรินทร์	Chao-kun-norn-wat-thep-sirin
19.	พระพุทธชินราช วัดต้นสน	Phra-put-ta-chin-na-rat-wat-ton-son
20.	หลวงพ่อโหน่ง วัดอัมพวัน	Luang-por-nong-wat-am-pa-wan

Table 4.1 Amulet Name List (cont.)

No.	Name	Pronunciation
21.	พระสายรุ้ง	Phra-sai-roong
22.	หลวงพ่อวัดโบสถ์น้อย วัดอรุณอัมรินทร์	Luang-por-wat-bot-noi
23.	หลวงพ่อหยอด วัดแก้วเจริญ	Luang-por-yod-wat-kaew-cha-roen
24.	หลวงพ่อเงิน วัดบางคลาน	Luang-por-ngern-wat-bang-klan
25.	หลวงพ่อเงิน วัดสามต้น	Luang-por-ngern-wat-sam-ton
26.	สมเด็จเมล็ดบัว วัดเจริญธรรม	Som-dej-ma-led-bua-wat-cha-roen-tham
27.	พระรอดขีดข้าง วัดลำพูน	Phra-rod-keed-kang-wat-lam-poon
28.	พระผงสุพรรณ หลวงพ่อดี วัดพระรูป	Phra-pong-su-pan-luang-por-dee
29.	สมเด็จจิตรดา วัดท่าพระ	Som-dej-jit-ra-da-wat-ta-phra
30.	หลวงพ่อเพี้ยง วัดท่าถนน	Luang-por-piang-wat-ta-ta-non
31.	หลวงพ่อทบ วัดโบสถ์	Luang-por-tob-wat-bot
32.	พระครูสุนทร วัดลางบางพระ	Phra-kru-sun-thorn
33.	พระไพรีพินาศ ปี30 วัดบวรนิเวศ	Phra-pai-ree-pi-nat-wat-ba-worn-ni-wed
34.	พระรอด วัดหนองหอย	Phra-rod-wat-nong-hoi
35.	หลวงพ่อคล้าย วัดธาตุน้อย	Luang-por-klai-wat-tard-noi
36.	หลวงพ่อเสือ วัดบางแวก	Luang-por-seau-wat-bang-wag
37.	หลวงพ่อแพร วัดพิกุลทอง	Luang-por-prae-wat-pi-kul-thong
38.	หลวงพ่อคูณ วัดบ้านไร่	Luang-por-koon-wat-baan-rai
39.	พระผงขุนแผนหลวงปู่ดู่ วัดสะแก	Phra-pong-khun-paen-luang-poo-doo
40.	พระผงกระเบื้องหลังคาโบสถ์ วัดโสธร	Phra-pong-kra-buaong-lung-ka-bot

Table 4.1 Amulet Name List (cont.)

No.	Name	Pronunciation
41.	พระไพรีพินาศ วัดบวรนิเวศ	Phra-pai-ree-pi-nard-wat-ba-worn-ni-wed
42.	พระพิชิตมาร วัดบรมนิวาส	Phra-pi-chit-marn-wat-bor-rom-ni-wat
43.	พระสมเด็จสัมมาอะระหัง วัดนาคกลาง	Phra-som-dej-sam-ma-a-ra-hang
44.	พระสมเด็จวัดธงไชย วัดธงไชย	Phra-som-dej-wat-tong-chai
45.	พระสมเด็จมหาไชโย วัดมหาไชย	Phra-som-dej-ma-ha-chai-yo
46.	พระสมเด็จข้าวสารดำ วัดไพรสณห์	Phra-som-dej-kaw-sarn-dam
47.	พระเกสรดอกไม้ วัดเจดีย์หลวง	Phra-gay-sorn-dok-mai
48.	พระสมเด็จวัดพระศรีรัตนมหาธาตุ	Phra-som-dej-wat-phra-sri-ma-ha-tard
49.	หลวงปู่ทิม วัดพระยาว	Luang-poo-tim-wat-phra-yaw
50.	พระอุปถัมป์ วัดป่ามหาไชย	Phra-up-pa-tham-wat-pa-ma-ha-chai
51.	พระปิดตา หลวงพ่อเขียว วัดเขาอีส้าน	Phra-pid-ta-luang-por-kiew
52.	รูปเหมือน หลวงพ่อเขียว วัดเขาอีส้าน	Roop-meaun-luang-por-kiew
53.	พระสังกะจาย หลวงพ่อเขียว วัดเขาอีส้าน	Phra-sang-ka-jay-luang-por-kiew
54.	พระรอด วัดไผ่ล้อม	Phra-rod-wat-pai-lom
55.	พระนางพญา วัดไผ่ล้อม	Phra-nang-pa-ya
56.	พระผงสุพรรณ วัดไผ่ล้อม	Phra-pong-su-pan-wat-pai-lom
57.	พระซุ้มกอ วัดไผ่ล้อม	Phra-sum-kor-wat-pai-lom
58.	พระสมเด็จ วัดไผ่ล้อม	Phra-som-dej-wat-pai-lom
59.	หลวงพ่อสมชาย วัดเขาสุกิม	Luang-por-som-chai-wat-koa-su-kim
60.	พระรอด วัดลำพูน	Phra-rod-wat-lam-poon

Table 4.1 Amulet Name List (cont.)

No.	Name	Pronunciation
61.	หลวงพ่อพระธรรมจักร วัดแก้วแจ่มฟ้า	Luang-por-phra-tham-ma-jak
62.	หลวงปู่บาง จันทสโร วัดหนองพลับ	Luang-poo-bang-jan-ta-sa-roe
63.	หลวงพ่อมหาเงา วัดพระธาตุมหาเงา	Luang-por-wat-ma-ha-ngao
64.	พระปางเปิดโลก วัดเขาลั่นทม	Phra-pang-perd-lok
65.	หลวงพ่อเปิ่น วัดบางพระ	Luang-por-pern-wat-lang-bang-phra
66.	หลวงพ่อเปิ่น วัดบางหอ	Luang-por-pern-wat-bang-hor
67.	สมเด็จองค์บรมพระพุทธเจ้า วัดบวรนิเวศ	Som-dej-ong-bor-rom-phra-put-ta-chao
68.	หลวงปู่ทวด วัดช้างไห้	Luang-poo-tuad-wat-chang-hai
69.	สมเด็จพุทธาจารย์โต	Som-dej-put-ta-jan-to
70.	พระครูธรรมสรคุณ วัดกระทิง	Phra-kur-wi-sut-ti-tham-wat-kra-ting
71.	สมเด็จพุฒาจารย์โต	Som-dej-put-ta-jan-to
72.	พระรอดผงหลังใบโพธิ์	Phra-rod-pong-lung-bai-pho
73.	พระพุทธชินราชสามเหลี่ยม หลังอกเลา	Phra-put-ta-chin-na-rat-sam-liam
74.	พระซุ้มกอเนื้อผง หลังใบโพธิ์	Phra-sum-kor-nuao-pong-lun-bai-pho
75.	พระนางพญาหลวงปู่แหวน วัดสัมพันธวงศ์	Phra-nang-pa-ya-luang-poo-wan
76.	พระพุทธชินราช วัดศรีรัตนมหาธาตุ	Phra-put-ta-chin-na-rat-wat-ma-ha-tard
77.	พระสมเด็จ วัดสว่างโพธาราม	Phra-som-dej-wat-sa-wang-pho-ta-ram
78.	พระสมเด็จ วัดบุ่งยาง	Phra-som-dej-wat-bung-yang
79.	พระสมเด็จกลีบบัว วัดโนนสุวรรณ	Phra-som-dej-wat-non-su-wan
80.	พระปางเทศนา วัดทุ่งสะเดียง	Phra-pang-ted-sa-na-wat-toong-sa-diang

Table 4.1 Amulet Name List (cont.)

No.	Name	Pronunciation
81.	พระนาคปรก ใบมะขาม วัดยางเอน	Phra-nak-prok-bai-ma-kam
82.	ย่าโม วัดประชุมราษฎร์	Ya-mo-wat-pra-chum-rat
83.	กวนอิม วัดประชุมราษฎร์	Guan-im-wat-pra-chum-rat
84.	ราหู วัดประชุมราษฎร์	Ra-hu-wat-pra-chum-rat
85.	พระวิศนุ วัดประชุมราษฎร์	Phra-wit-sa-nu-wat-pra-chum-rat
86.	กรมหลวงชุมพร วัดประชุมราษฎร์	Grom-ma-luang-chum-porn
87.	พระเจ้าตาก วัดประชุมราษฎร์	Phra-chao-tak-wat-pra-chum-rat
88.	สมเด็จพุฒาจารย์โต วัดประชุมราษฎร์	som-dej-put-ta-jan-to-wat-pra-chum-rat
89.	รัชกาลที่ 5 วัดประชุมราษฎร์	Rat-cha-karn-tee-ha-wat-pra-chum-rat
90.	พระพรหม วัดประชุมราษฎร์	Phra-prom-wat-pra-chum-rat
91.	พระพิฆเนศ วัดประชุมราษฎร์	Phra-pik-ka-ned-wat-pra-chum-rat
92.	หลวงปู่ฤาษี วัดประชุมราษฎร์	Luang-poo-ru-see-wat-pra-chum-rat
93.	พระพุทธปางสมาธิ วัดประชุมราษฎร์	Phra-put-pang-sa-ma-ti-wat-pra-chum-rat
94.	พระปางสมาธิฐานกลีบบัว	Phra-pang-sa-ma-ti-tarn-gleeb-bua
95.	หลวงปู่เทพโลกอุดร วัดประชุมราษฎร์	Luang-poo-tep-lok-u-dorn-wat-pra-chum-rat
96.	หลวงปู่วัดบ้านแหลม วัดประชุมราษฎร์	Luang-poo-wat-baan-laem-wat-pra-chum-rat
97.	หลวงพ่อวัดไร่ขิง วัดประชุมราษฎร์	Luang-por-wat-rai-king-wat-pra-chum-rat
98.	พระพุทธนราวันตบพิตร	Phra-put-ta-na-ra-wan-bor-pid
99.	พระปิดตา วัดเขาพนก	Phra-pid-ta-wat-kao-pa-nok
100.	จตุคาม รามเทพ	Ja-tu-kam-ram-ma-tep

We do not change anything in those images. The images are original images as once they are taken from the camera. Their features are extracted and fed into the neural networks, and we let the system learn until its performance is stable.

We have trained neural networks with different settings. The purpose is to see how good the models work with the problem and find the best one.

Before going to see their performance, we need to know about training configuration. It is very crucial because it affects to effectiveness in classification rate significantly. Training without any setting causes Lack-of-training problems.

In the MATLAB, it has a toolbox for artificial neural network. We can run a training course with a few instructions.

```
x = [0 0 1 1; 0 1 0 1];
t = [0 1 1 1];
net = perceptron;
net = train(net,x,t);
```

Figure 4.20 Perceptron Training Script

The Figure 4.20 is an example of a MATLAB script for training a perceptron network. The x variable is the patterns that the perceptron has to learn. The t variable is targets of results. To create a perceptron, we need only one instruction in the third line. We just invoke 'perceptron' in MATLAB script to instantiate a network. To train the network, we have to call another function, which is 'train'. The 'train' function needs at least three parameters, such as network, input patterns, and targets.

The MATLAB does not have only the perceptron in the toolbox since we can invoke other networks that have more layers as in the Figure 4.21. The figure shows the script of training in 4.3-1 perceptron.

```
x = [0 0 1 1; 0 1 0 1; 1 0 0 1; 0 1 1 0];
t = [0 1 1 1];
hiddenLayerSize = 3;
mLayerNet = patternnet(hiddenLayerSize);
mLayerNet = train(mLayernet,x,t);
```

Figure 4.21 Multi-layer Network Training Script

In addition, there are more configuration parameters that can be fixed before training. The Figure 4.22 shows that we can the other parameters such as activation function, learning rate, performance goal, and maximum epochs

```
x = [0 0 1 1; 0 1 0 1; 1 0 0 1; 0 1 1 0];
t = [0 1 1 1];
hiddenLayerSize = 3;
mLayerNet = patternnet(hiddenLayerSize);
mlayernet.layers{1}.transferFcn = 'logsig'; % Activation Function
mlayernet.layers{2}.transferFcn = 'logsig'; % for output layer
mlayernet.trainParam.lr = 0.01; % Learning rate
mlayernet.trainParam.goal = 1e-4; % Performance goal
mlayernet.trainParam.epochs=2000; % Maximum Epoch
mLayerNet = train(mLayernet,x,t);
```

Figure 4.22 Example of Training Configuration

The mentioned setting influence to the result of significantly. For example, if the performance goal is set too high, or the number of maximum epoch is set too small, the training will be stopped too early. Probably, the network did not learn enough yet. The accuracy rate of the neural network might be lower than the expected. In opposite, too low performance goal and too epochs push the network into overtraining. Moreover, too high learning rate makes a chance that the network will not be converged in training. Therefore, it is important to fix those parameters carefully to the network more efficiently.

At the first time, we train a perceptron the script in the Figure 4.23. The default value of goal is 0, and it uses mean absolute error (MAE) as performance function. The network has 54 input nodes and 100 output nodes.

```
inputs = scaled_amu_train'; % training set (54-dimension input)
targets = amu_targets'; % target result (100 classes)
net = perceptron
net.trainParam.lr = 0.01;
net.trainParam.epochs = 3000;
net = train(net,inputs,targets);
```

Figure 4.23 Configuration of Perceptron

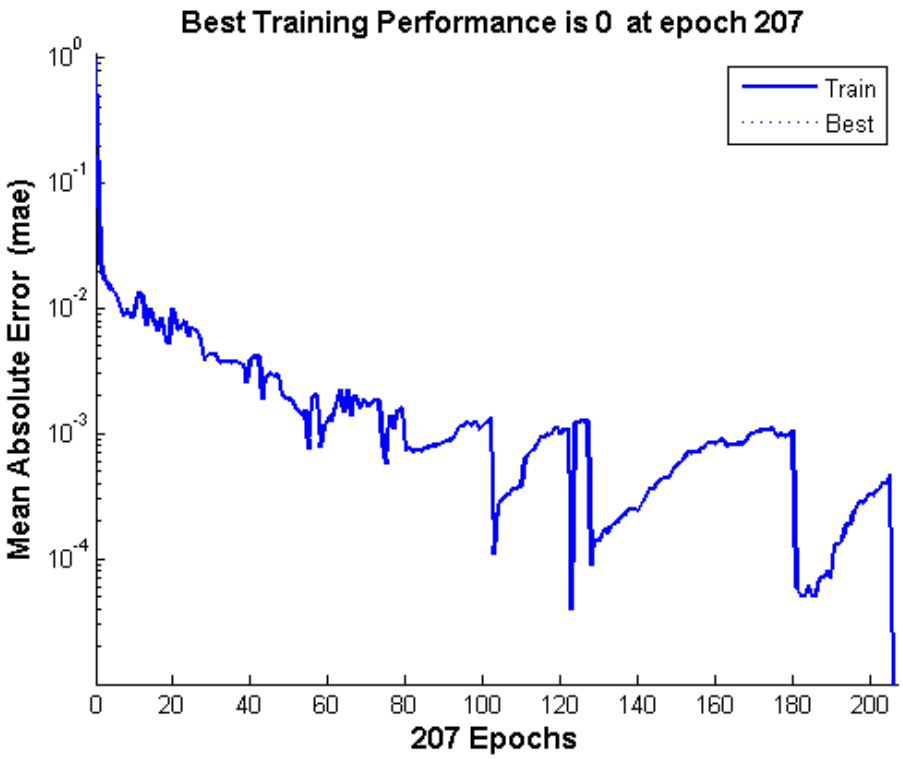


Figure 4.24 Perceptron's Performance

The graph in the Figure 4.24 shows performances of the perceptron in each epoch. It indicates the perceptron reach the goal at epoch 207.

After that, we train more three networks with multiple layers. The model of the networks are 54-54-100 perceptron. They are trained with different defined performance goal, which are 10^{-3}, 10^{-4}, and 10^{-5}. The performance function for those networks is mean square error (MSE). The training scripts for those three multi-layer neural networks are shown in the Figure 4.25, Figure 4.26, and Figure 4.27, and the performances of them are shown in the Figure 4.28, Figure 4.29, and Figure 4.30 respectively.

```
inputs = scaled_amu_train'; % prepared training set (54-dimension
input)

targets = amu_targets'; % target result (100 classes)

hiddenLayerSize = 54; % 54 hidden nodes

mLayerNet = patternnet(hiddenLayerSize);

mlayernet.layers{1}.transferFcn = 'logsig'; % Activation Function

mlayernet.layers{2}.transferFcn = 'logsig';

mLayerNet.trainParam.lr = 0.01;

Mlayernet.trainParam.goal = 1e-3; % Performanc goal

net.trainParam.epochs = 5000;

net = train(net,inputs,targets);
```

Figure 4.25 Configuration of 54-54-100 Perceptron (MSE = 10^{-3})

```
inputs = scaled_amu_train'; % training set (54-dimension input)

targets = amu_targets'; % target result (100 classes)

hiddenLayerSize = 54; % 54 hidden nodes

mLayerNet = patternnet(hiddenLayerSize);

mlayernet.layers{1}.transferFcn = 'logsig'; % Activation Function

mlayernet.layers{2}.transferFcn = 'logsig';

mLayerNet.trainParam.lr = 0.01;

Mlayernet.trainParam.goal = 1e-4; % Performanc goal

net.trainParam.epochs = 7000;

net = train(net,inputs,targets);
```

Figure 4.26 Configuration of 54-54-100 Perceptron (MSE = 10^{-4})

```
inputs = scaled_amu_train'; % training set (54-dimension input)

targets = amu_targets'; % target result (100 classes)

hiddenLayerSize = 54; % 54 hidden nodes

mLayerNet = patternnet(hiddenLayerSize);

mlayernet.layers{1}.transferFcn = 'logsig'; % Activation Function

mlayernet.layers{2}.transferFcn = 'logsig';

mLayerNet.trainParam.lr = 0.01;

Mlayernet.trainParam.goal = 1e-5; % Performanc goal

net.trainParam.epochs = 10000;

net = train(net,inputs,targets);
```

Figure 4.27 Configuration of 54-54-100 Perceptron (MSE = 10^{-5})

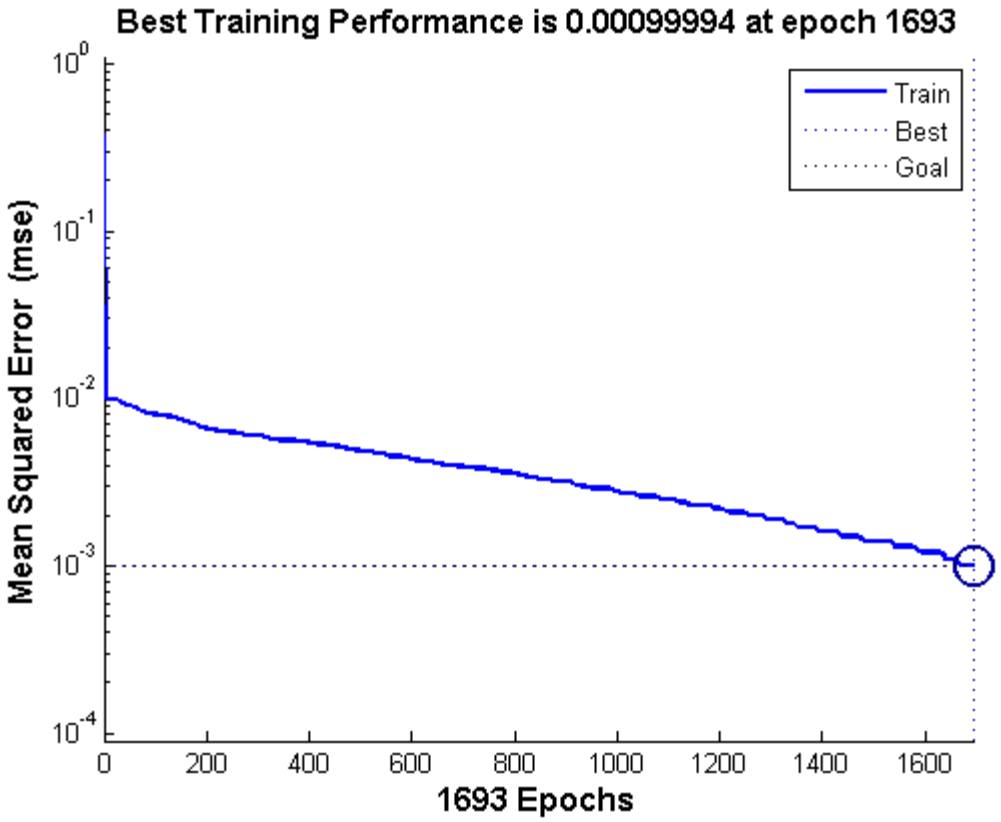


Figure 4.28 54-54-100 Perceptron's Performance (MSE = 10^{-3})

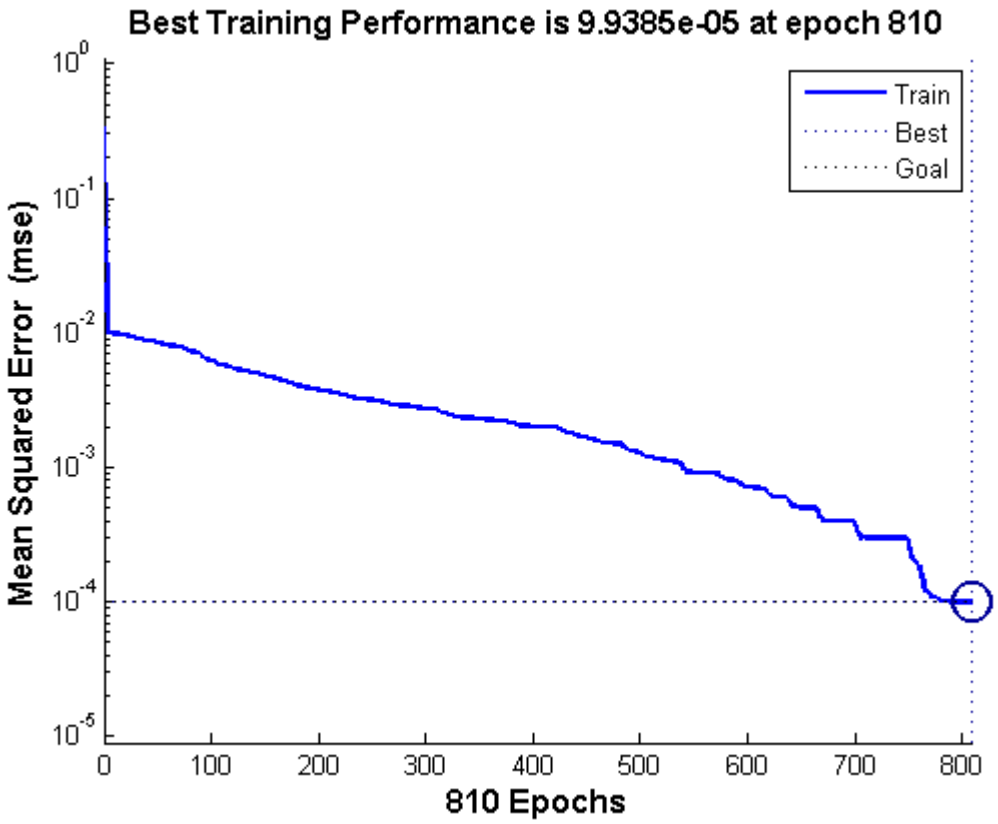


Figure 4.29 54-54-100 Perceptron's Performance (MSE = 10^{-4})

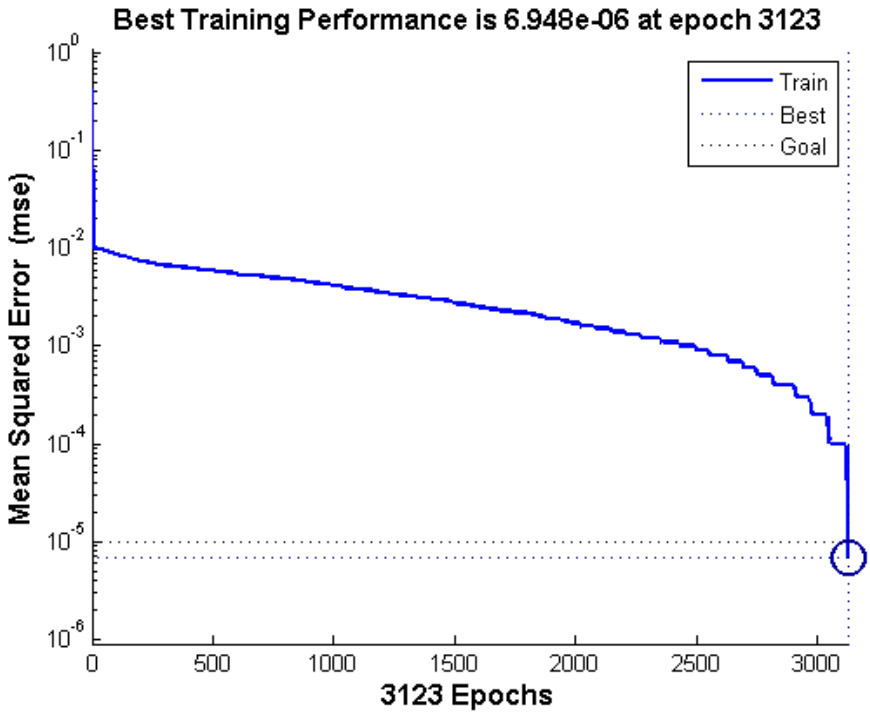


Figure 4.30 54-54-100 Perceptron's Performance (MSE = 10^{-5})

4.4 Testing

In testing phase, the test set consists of 103 kinds of amulet – 100 for known kinds, 3 for unknown kinds, and 5 samples for each kind. Therefore, the number of images in test set is 515. However, the original test set is not enough to test the system; it needs transformed test sets also. There are seven types of transformed test set in this experiment, which are 90-degree rotated, 180-degree rotated, 270-degree rotated, 20-percent bigger, 20-percent smaller, 20-percent brighter, and 20-percent darker. Those types are transformed from the original test set entirely. An example of transformed test set is shown in Figure 4.31. The picture of (a) is an example of normal test set. The (b), (c), (d), (e), (f), (g), and (h) are examples of 90-degree rotated, 180-degree rotated, 270-degree rotated, 20-percent bigger, 20-percent smaller, 20-percent brighter, and 20-percent darker test set respectively.

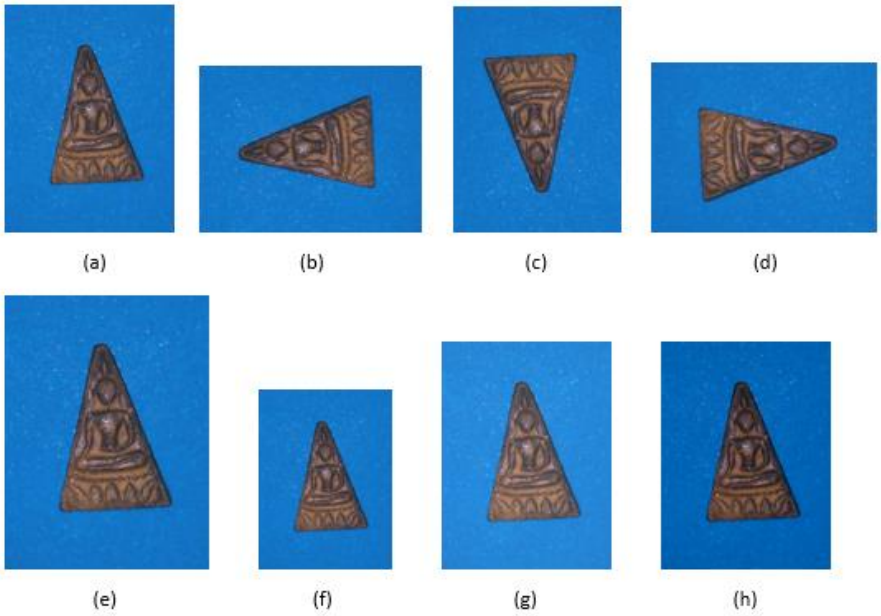


Figure 4.31 Examples of Normal Test Set and Transformed Test Set

The test sets are used to test the trained models of perceptron and multi-layer perceptron networks. The accuracy rate of them will indicate their effectiveness to recognize amulet images.

CHAPTER V
EXPERIMENTAL RESULT

We must record the results from the graphic user interface compared with the expected results. There are three possible results in this system, which are matched result, mismatched result, and unknown result. The examples of them are given in the figure 5.1, 5.2 and 5.3.

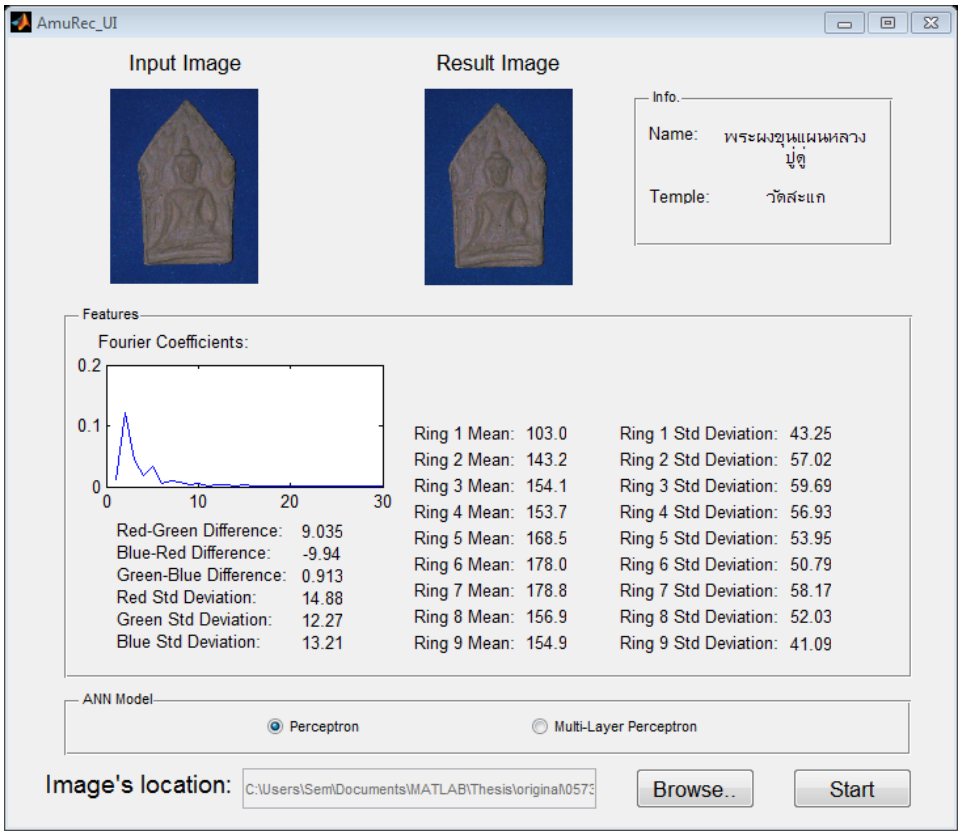


Figure 5.1 Matched Result

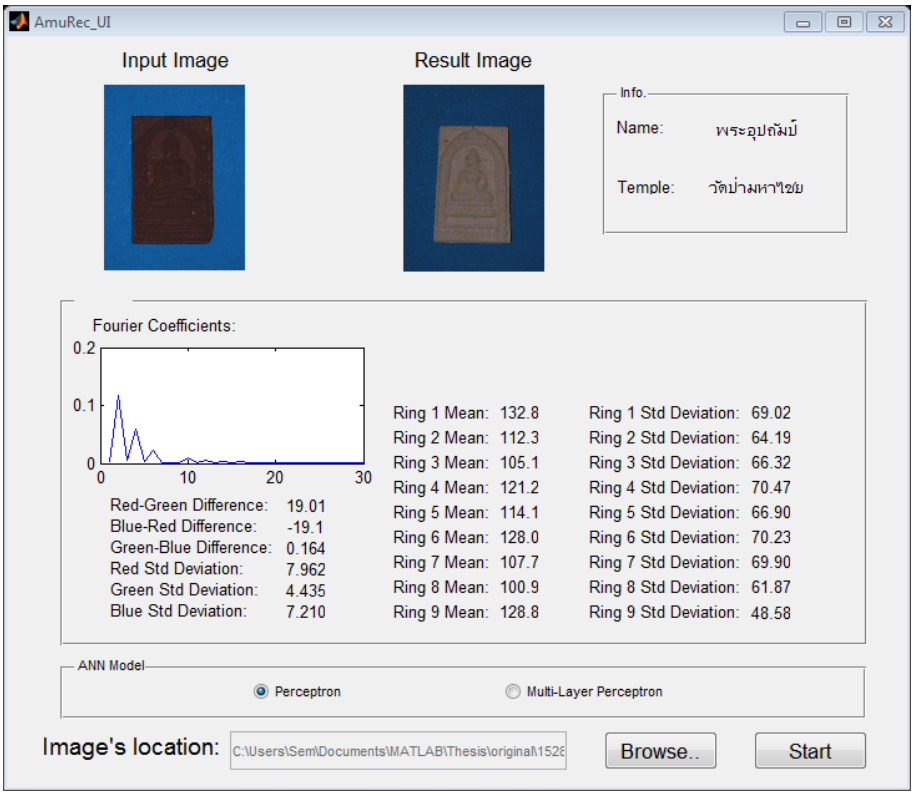


Figure 5.2 Mismatched Result

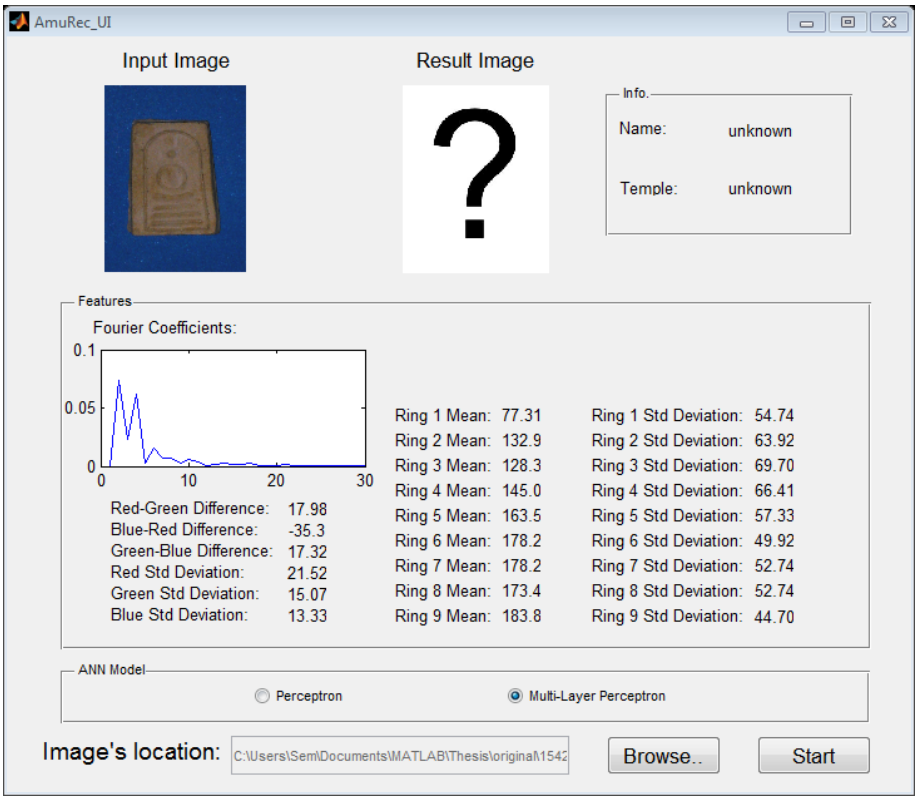


Figure 5.3 Unknown Result

Firstly, the perceptron is tested with normal test set. It can classify 489 images from 500 images correctly for known type, and 14 images from 15 images for unknown type. Totally, its accuracy rate is 97.67 percent as shown in the table 5.1.

Table 5.1 Perceptron's Result

Type of Dataset	Correct Matching	Accuracy Rate
Known	489/500	97.8%
Unknown	14/15	93.33%
Total	503/515	97.67%

For multi-layer perceptron, based on the number input and output nodes, the number of hidden nodes should be between them. Moreover, in the perceptron's experiment, the result is quite good, so a small hidden layer is probably enough. According to the assumption, 54-node hidden layer is used because it is the smallest between the size of input layer and output layer. Three trained 54-54-100 perceptron in different MSE values are tested with normal test set. The result of them is shown in Table 5.2. It is shown that 54-54-100 perceptron at 10^{-5} of MSE can classify the normal test set correctly at 100 percent.

Table 5.2 Network Performance of 54-54-100 Perceptron with Normal Test Set

Mean Squared Error	Accuracy Rate
10^{-3}	89.32%
10^{-4}	98.06%
10^{-5}	100%

For rotated test sets, the accuracy rates of the perceptron are 97.09 percent for 90-degree rotated, 97.67 percent for 180-degree rotated, and 97.28 percent for 270-degree rotated. For 20-percent bigger and 20-percent smaller test set, the accuracy rates are 97.67 percent and 95.53 percent respectively. In the case of brightness

adjustment, they are 97.67 percent for brighter test set and 98.06 percent for darker test set.

Also, the 54-54-100 perceptron where MSE = 10^{-5} is tested with the transformed test sets, and the results comparing to ordinary perceptron is expressed in Table 5.3, Table 5.4, and Table 5.5. From the results, the multi-layer can recognize the test sets with transformation correctly at 100 percent.

Table 5.3 Accuracy Rate of Perceptron and 54-54-100 Perceptron with Normal and Rotated Test Set

Model	**Normal**	**90-Degree**	**180-Degree**	**270-degree**
Perceptron	97.67%	97.09%	97.67%	97.28%
54-54-100 Perceptron (MSE = 10^{-5})	100%	100%	100%	100%

Table 5.4 Accuracy Rate of Perceptron and 54-54-100 Perceptron with Normal and Scaled Test Set

Model	**Normal**	**20% Bigger**	**20% Smaller**
Perceptron	97.67%	97.67%	95.53%
54-54-100 Perceptron (MSE = 10^{-5})	100%	100%	100%

Table 5.5 Accuracy Rate of Perceptron and 54-54-100 Perceptron with Normal and Brightness Adjusted Test Set

Model	**Normal**	**20% Brighter**	**20% Darker**
Perceptron	97.67%	97.67%	98.06%
54-54-100 Perceptron (MSE = 10^{-5})	100%	100%	100%

CHAPTER VI
CONCLUSION AND SUGGESTION

6.1 Conclusion

To conclude, it can be observed that the proposed feature extraction methods are able to mitigate the effect of transformed test sets. Normally, the result of transformed test sets should be worse than normal test sets significantly, but it does not occur in this experiment. According to the experiment, the system is tolerant to rotation, scaling, and brightness adjustment noticeably.

Moreover, the experimental result shows that the multi-layer perceptron can classify the samples amulet better than the ordinary perceptron. Even the average accuracy rate of perceptron is 97.33 percent, but 54-54-100 perceptron can perform at 100 percent in all prepared test sets. The cause of errors in perceptron is that its output nodes are activated more than one nodes. The problem occurs because some amulets are similar to the other amulets in the test set. Their features cannot be classified by linear separator. Multi-layer perceptron is more efficient because of its hidden layer. The hidden layer provides more separators for each class, so it can classify the similar classes more efficiently.

The Thai Buddhist amulet recognition system can satisfy the research objectives very well. It can recognize those provided amulet images and tolerate to basic transformations that can be met in real situation as well. Moreover, the system provide a graphic user interface for user's convenience.

6.2 Problems and Limitations

Along this research, we have realized that to acquire samples is difficult because many of them are rare and expensive. We can buy only the cheap and unpopular Buddhist amulets. We need to borrow some of them from some collectors,

and that is inconvenient to work. We need more support to collect more samples of amulets.

Moreover, the Buddhist amulets do not have their official name, so we do not have a standard to present their information properly.

6.3 Future Works

In the future, the size of sample data should be expanded to support more amulets. The presentation of amulet information should be standardized and more complete. The result could be shown in ranking of matching rate to suggest other results in case of mismatched result. Certainly, there are some rooms for other techniques for this problem such as support vector machine. Crucially, we need to focus on implementation of more practical system and the chance that the recognition system for Buddhist amulet would be a commercial product.

REFERENCES

1 M. Fukumi, S. Omatsu., F. Takeda, T. Kosaka (1992). Rotation-Invariant Neural Pattern Recognition System with Application to Coin Recognition. IEEE Transactions on Neural Networks, 3(2), 8.

2 Y. Mitsukura, M. Fukumi, N. Akamatsu. (2000). Design and Evaluation of Neural Networks for Coin Recognition by using GA and SA. Paper presented at the Proceedings of the IEEE-INNS-ENNS International Joint Conference, Como.

3 R. Bremananth, B. Balaji, M. Sankari, and A. Chitra. (2005). A New Approach to Coin Recognition using Neural Patter Analysis. Paper presented at the IEEE Indicon 2005, Chennai.

4 A. Khashman, B. Sekeroglu, K. Dimililer. (2006). Intelligent Coin Identification System. Paper presented at the International Symposlum on Intelligent Control, Munich.

5 T. N. Nimbhorkar, M. M. Bartere (2006). Coin Recognition by using Artificial Neural Network. International Journal of Computer, Information Technology and Bioinformatics, 1(4), 4.

6 C. Cai-ming, Z. Shi-qing, C. Yue-fen. (2010). A Coin Recognition System with Rotation Invariance. Paper presented at the International Conference on Machine Vision and Human-machine Interface, Kaifeng.

7 P. Thumwarin, S. Malila, P. Janthawong, and W. Pibulwej. (2006). A Robust Coin Recognition Method with Rotation Invariance. Paper presented at the Communications, Circuits and Systems Proceedings, 2006 International Conference, Guilin.

8 C. Korbuakaew (2007). Identification of Amulets with Special Feature Matching. (Master of Science), Silpakorn University. (45307303)

9 J. Wongkorsub, T. Pornaudomdaj, P. Vessawasdi, and C. Pornpanomchai. (2010). Buddhist Amulet Recognition System (BARS). Paper presented at the Computer and Network Technology (ICCNT), 2010 Second International Conference, Bangkok.

10 N. Srisuporn and C. Pornpanomchai (2013). Buddhist amulet coin recognition by genetic algorithm. Paper presented at the Computer Science and Engineering Conference (ICSEC), 2013 International, Nakorn Pathom.

11 A. Greensted (2010, 17th June 2010). Otsu Thresholding. From http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

APPENDIX

KNOWN AMULETS

In this part, we presents the information of 100 known amulets that are used to display in our system. The information is about the image, name, and temple name of the amulet.

No. 1

Name: พระปิดตา เนื้อดำ หลวงปู่เหรียญ

Temple: วัดอรัญบรรพต

No. 2

Name: พระปิดตา หลวงปู่เหรียญ

Temple: วัดอรัญบรรพต

No. 3

Name: พระสมเด็จ หลวงปู่เหรียญ

Temple: วัดอรัญบรรพต

No. 4

Name: พระนาคปรก หลวงปู่เหรียญ

Temple:วัดอรัญบรรพต

No. 5

Name: ภาพพิมพ์ หลวงปู่เหรียญ

Temple:วัดอรัญบรรพต

No. 6

Name: หลวงปู่เหรียญ

Temple:วัดอรัญบรรพต

No. 7

Name: หลวงปู่ทวด

Temple: วัดช้างไห้

No. 8

Name: หลวงปู่ทวด

Temple: วัดช้างไห้

No. 9

Name: สมเด็จ ส.ธ.ประจำวัน

Temple: -

No. 10

Name: สมเด็จ ส.ธ.ประจำวัน

Temple:-

No. 11

Name: หลวงพ่อเปิ่น

Temple: วัดลางบางพระ

No. 12

Name: พระครูอุดมวิสุทธิธรรม

Temple: วัดทุ่งนาใหม่

No. 13

Name: สมเด็จวัดบวรนิเวศ

Temple:วัดบวรนิเวศ

No. 14

Name: พระมงคลบพิตร รุ่นกรุงเก่า

Temple:

No. 15

Name: พระมงคลบพิตร รุ่นกรุงเก่า

Temple:

No. 16

Name: พระมงคลบพิตร รุ่นกรุงเก่า

Temple: -

No. 17

Name: หลวงปู่ภู

Temple: วัดอินทร์

No. 18

Name: เจ้าคุณนร.

Temple: วัดเทพศิรินทร์

No. 19

Name: พระพุทธชินราช

Temple:วัดต้นสน

No. 20

Name: หลวงพ่อโหน่ง

Temple: วัดอัมพวัน

No. 21

Name: พระสายรุ้ง

Temple: -

No. 22

Name: หลวงพ่อวโรสถ์น้อย

Temple: วัดอรุณอัมรินทร์

No. 23

Name: หลวงพ่อหยอด

Temple: วัดแก้วเจริญ

No. 24

Name: หลวงพ่อเงิน

Temple: วัดบางคลาน

No. 25

Name: หลวงพ่อเงิน

Temple: วัดสามต้น

No. 26

Name: สมเด็จเมล็ดบัว

Temple: วัดเจริญธรรม

No. 27

Name: พระรอดขีดข้าง

Temple: วัดลำพูน

No. 28

Name: พระผงสุพรรณ หลวงพ่อดี

Temple: วัดพระรูป

No. 29

Name: สมเด็จจิตรดา

Temple: วัดท่าพระ

No. 30

Name: หลวงพ่อเพี้ยง

Temple: วัดท่าถนน

No. 31

Name: หลวงพ่อทบ

Temple: วัดโบสถ์

No. 32

Name: พระครูสุนทร

Temple: วัดลางบางพระ

No. 33

Name: พระไพรีพินาศ ปี 30

Temple: วัดบวรนิเวศ

No. 34

Name: พระรอด

Temple: วัดหนองหอย

No. 35

Name: หลวงพ่อคล้าย

Temple: วัดธาตุน้อย

No. 36

Name: หลวงพ่อเสือ

Temple: วัดบางแวก

No. 37

Name: หลวงพ่อแพร

Temple:วัดพิกุลทอง

No. 38

Name: หลวงพ่อคูณ

Temple: วัดบ้านไร่

No. 39

Name: พระผงขุนแผนหลวงปู่ดู่

Temple: วัดสะแก

No. 40

Name: พระผงกระเบื้องหลังคาโบสถ์

Temple: วัดโสธร

No. 41

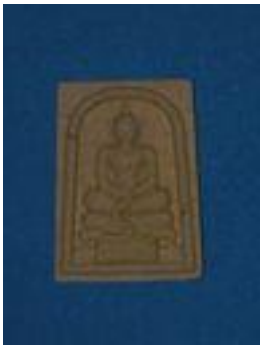
Name: พระไพรีพินาศ

Temple: วัดบวรนิเวศ

No. 42

Name: พระพิชิตมาร

Temple: วัดบรมนิวาส



No. 43

Name: พระสมเด็จสัมมาอะระหัง

Temple: วัดนาคกลางวรวิหาร

No. 44

Name: พระสมเด็จวัดธงไชย

Temple: วัดธงไชย

No. 45

Name: พระสมเด็จมหาไชโย

Temple: วัดมหาไชย

No. 46

Name: พระสมเด็จข้าวสารดำ

Temple: วัดไพรสณห์

No. 47

Name: พระเกสรดอกไม้

Temple: วัดเจดีย์หลวง

No. 48

Name: พระสมเด็จวัดพระศรีรัตนมหาธาตุ

Temple: วัดพระศรีรัตนมหาธาตุ

No. 49

Name: หลวงปู่ทิม

Temple: วัดพระยาว

No. 50

Name: พระอุปถัมป์

Temple: วัดป่ามหาไชย

No. 51

Name: พระปิดตา หลวงพ่อเขียว

Temple: วัดเขาอีส้าน

No. 52

Name: รูปเหมือน หลวงพ่อเขียว

Temple: วัดเขาอีส้าน

No. 53

Name: พระสังกะจาย หลวงพ่อเขียว

Temple: วัดเขาอีส้าน

No. 54

Name: พระรอด

Temple: วัดไผ่ล้อม

No. 55

Name: พระนางพญา

Temple: วัดไผ่ล้อม

No. 56

Name: พระผงสุพรรณ

Temple: วัดไผ่ล้อม

No. 57

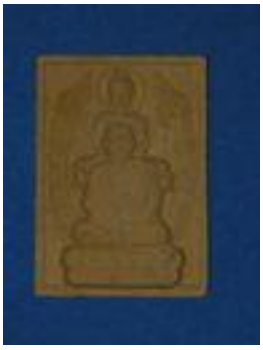
Name: พระซุ้มกอ

Temple: วัดไผ่ล้อม

No. 58

Name: พระสมเด็จ

Temple: วัดไผ่ล้อม



No. 59

Name: หลวงพ่อสมชาย

Temple: วัดเขาสุกิม

No. 60

Name: พระรอด

Temple: วัดลำพูน

No. 61

Name: หลวงพ่อพระธรรมจักร

Temple: วัดแก้วแจ่มฟ้า

No. 62

Name: หลวงปู่บาง จันทสโร

Temple: วัดหนองพลับ

No. 63

Name: หลวงพ่อมหาเงา

Temple: วัดพระธาตุมหาเงา

No. 64

Name: พระปางเปิดโลก

Temple: วัดเขาลั่นทม

No. 65

Name: หลวงพ่อเปิ่น

Temple: วัดบางพระ

No. 66

Name: หลวงพ่อเปิ่น

Temple: วัดบางหอ

No. 67

Name: สมเด็จองค์บรมพระพุทธเจ้า หลวงปู่ทวด

Temple: วัดบวรนิเวศ

No. 68

Name: หลวงปู่ทวด

Temple: วัดช้างไห้

No. 69

Name: สมเด็จพุทธาจารย์โต

Temple: -

No. 70

Name: พระครูธรรมสรคุณ

Temple: วัดกระทิง

No. 71

Name: สมเด็จพุฒาจารย์โต

Temple: -

No. 72

Name: พระรอดผงหลังใบโพธิ์

Temple: -

No. 73

Name: พระพุทธชินราชสามแหลี่ยม หลังอกเลา

Temple: วัดศรีรัตนมหาธาตุ

No. 74

Name: พระซุ้มกอเนื้อผง หลังใบโพธิ์

Temple: -

No. 75

Name: พระนางพญาหลวงปู่แหวน

Temple: วัดสัมพันธวงศ์

No. 76

Name: พระพุทธชินราช

Temple: วัดศรีรัตนมหาธาตุ

No. 77

Name: พระสมเด็จ

Temple: วัดสว่างโพธาราม

No. 78

Name: พระสมเด็จ

Temple: วัดบุ่งยาง

No. 79

Name: พระสมเด็จกลีบบัว

Temple: วัดโนนสุวรรณ

No. 80

Name: พระปางเทศนา

Temple: วัดทุ่งสะเดียง

No. 81

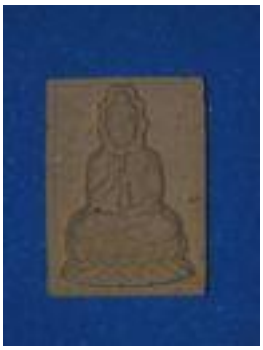
Name: พระนาคปรก ใบมะขาม

Temple: วัดยางเอน

No. 82

Name: ย่าโม

Temple: วัดประชุมราษฎร์



No. 83

Name: กวนอิม

Temple: วัดประชุมราษฎร์

No. 84

Name: ราหู

Temple: วัดประชุมราษฎร์

No. 85

Name: พระวิศนุ

Temple: วัดประชุมราษฎร์

No. 86

Name: กรมหลวงชุมพร

Temple: วัดประชุมราษฎร์

No. 87

Name: พระเจ้าตาก

Temple: วัดประชุมราษฎร์

No. 88

Name: รูปเหมือนสมเด็จพุฒาจารย์โต

Temple: วัดประชุมราษฎร์

No. 89

Name: รัชกาลที่ 5

Temple: วัดประชุมราษฎร์

No. 90

Name: พระพรหม

Temple: วัดประชุมราษฎร์

No. 91

Name: พระพิฆเนศ

Temple: วัดประชุมราษฎร์

No. 92

Name: หลวงปู่ฤาษี

Temple: วัดประชุมราษฎร์

No. 93

Name: พระพุทธปางสมาธิ

Temple: วัดประชุมราษฎร์

No. 94

Name: พระพุทธปางสมาธิฐานกลีบบัว

Temple: วัดประชุมราษฎร์

No. 95

Name: หลวงปู่เทพโลกอุดร

Temple: วัดประชุมราษฎร์

No. 96

Name: หลวงปู่วัดบ้านแหลม

Temple: วัดประชุมราษฎร์

No. 97

Name: หลวงพ่อวัดไร่ขิง

Temple: วัดประชุมราษฎร์

No. 98

Name: พระพุทธนราวันตบพิตร

Temple: -

No. 99

Name: พระปิดตา

Temple: -

No. 100

Name: จตุคาม รามเทพ

Temple: -

UNKNOWN AMULETS

In this part, it is to show the pictures of unknown amulets that we use in the experiment. There are three kinds of them, which are used to measure the precision of system also.

No. 1

Name: Unknown

Temple: Unknown

No. 2

Name: Unknown

Temple: Unknown

No. 3

Name: พระสมเด็จ หลวงปู่เหรียญ

Temple: วัดอรัญบรรพต

BIOGRAPHY

NAME	Mr. Waranat Kitiyanan
DATE OF BIRTH	30 November 1989
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Mahidol University, 2011-2012: The Degree of Bachelor of Science (Information and Communication Technology) Mahidol University, 2013-2014: The Degree of Master of Information and Communication Technology (Computer Science)
HOME ADDRESS	3599/90 Charoenrat Road Bang Khlo, Bang Kho Lam Bangkok Tel. 09-0938-3623 E-mail : waranat.kit@student.mahidol.ac.th