# IMAGE-BASED THAI AMULET RECOGNITION

**BY**

**THANACHAI SAUTHANANUSUK**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING (INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS)**

**SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY**
**THAMMASAT UNIVERSITY**

**ACADEMIC YEAR 2016**

# IMAGE-BASED THAI AMULET RECOGNITION

**BY**

**THANACHAI SAUTHANANUSUK**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING (INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS)**

**SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY**
**THAMMASAT UNIVERSITY**

**ACADEMIC YEAR 2016**

IMAGE-BASED THAI AMULET RECOGNITION

A Thesis Presented

By

THANACHAI SAUTHANANUSUK

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING (INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS)

Approved as to style and content by

Advisor and Chairperson of Thesis Committee ______________________

(Assoc. Prof. Dr. Chalie Charoenlarpnopparut)

Committee Member and
Chairperson of Examination Committee ______________________

(Assoc. Prof. Dr. Toshiaki Kondo)

Committee Member ______________________

(Dr. Pished Bunnun)

Committee Member ______________________

(Prof. Dr. Kaneko Hirohiko)

AUGUST 2017

# Abstract

IMAGE-BASED THAI AMULET RECOGNITION

by

THANACHAI SAUTHANANUSUK

Bachelor of Engineering (Electrical Engineering), Chulalongkorn University, 2011
Master of Engineering (Information and Communication Technology for Embedded Systems), Sirindhorn International Institute of Technology, Thammasat University, 2017

This thesis presents a way to determine the kind of amulet from taken picture. The pre-processing and amulet segmentation are processed with the plain color background and segmentation method is based on grayscale conversion and edge detection to locate high contrast area in the image. The amulet kind determination makes use of two feature, first is edge feature from prewitt edge detection and second is texture feature called local ternary pattern (LTP) which indicate differrent in image intensities. The accuracy of template matching are 77% when using Canny edge feature and 99% when using LTP feature.

**Keywords**: Image Processing, Template Matching, Amulet Recognition

# Acknowledgements

Firstly, the author would like to express his deepest sincere gratitude to Assoc. Prof. Chalie Charoenlarpnopparut, the author’s thesis advisor for his continuous support of the author’s study. His patience, motivation, and guidance helped the author all the time of research and thesis writing.

Beside his advisor, the author would like to thank the rest of his thesis committee: Assoc. Prof. Toshiaki Kondo, Prof. Kaneko Hirohiko, Dr. Pished Bunnun for their comment and useful suggestion.

Finally, the author would like to thank Thailand Advanced Institute of Science and Technology (TAIST), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology, and Sirindhorn International Institute of Technology (SIIT), Thammasat University (TU) for financial support this thesis research.

# Table of Contents

# List of Tables

# List of Figures

**Figures** **Page**

# Chapter 1
# Introduction

## 1.1 Background of study and Statement of problem

Thai amulet is accessory with Buddha shape and named after local Buddhism priest in Thailand. There are many kinds and variations, which are difficult to distinguish. Thai amulet can be highly valuable asset especially classical ones which has limit number of its own kind. Many amulet clubs in Thailand trade the amulet fluently. For non-specialist, knowing the kind of the amulet they are dealing with will help in price setting for trading in the market and estimating asset value for the investor who keeps it.

However, it is not easy because prior knowledge about amulet characteristic require long time study to master. Sometimes, even the specialist who has been familiar with Buddhist amulets for a long time still does not know all of them. Some kinds of amulet have word carved on the shape but for the rest those there are not any word on the shape, all information for non-specialist people is what they see as shown in Figure 1.1 and that leads to study about amulet recognition from taken picture.

Figure 1.1 Example of difficult to distinguish amulet for non-specialist

## 1.2 Purpose of study

Because amulet has very similar to coin in size and there is not much recent work about actual amulet, the majority of this study is coin classification method and the rest is study about feature extraction then propose an image-based approach for determine the kind of amulet from taken picture to help people know more about the amulet. The list of publications written from this study by the author can be found in appendix A.

## 1.3 Expected Outcome

The expected outcome of this study is the amulet recognition system that is easy for the user with the unknown amulet to try and use it. Since the target group user is non-specialist people, the author expect that the system can make more people to have interest and want to know more about amulet.

.

## 1.4 Scope and limitation

This study consist 33 kinds of amulet. The focus distance of camera for image acquisition should be at most 10 cm. The procedures in this study cannot work with image with complex background.

## 1.5 Structure of the thesis

The organization of the thesis is as follow, Chapter 2 describes the related work to this thesis. Chapter 3 explains the detail of image preprocessing in amulet segmentation and feature extraction using in this research is also described. Chapter 4 shows experiment result and discussion. Chapter 5 demonstrates the amulet recognition system from proposed algorithm. Finally, Chapter 6 concludes the study of this thesis and suggestion for future improvement.

# Chapter 2
# Literature Review

Because Thai amulet is not well known outside Thailand, there are very few works related to the amulet. Nevertheless, there are many researches about coin recognition those can be adopted to apply with amulet recognition. Since coin and amulet are very similar in object size.

Minoru Fukumi *et al.* [1] used rotated image for neural network training to create rotation invariant coin pattern recognition system. The system consists of two parts, the slabs of preprocessor and rotation invariant network called a-CONE. Each slab of the preprocessor contains sigmoid neuron units, which produce single output to the trainable multilayer neural-network a-CONE. To determine the weight of neuron units in order to achieve rotation invariant, the circular coordinate system used by the preprocessor is represented by radius and corresponding angle. With 12 equal segments in one circumference area, slab outputs can be insensitive to every 30 degrees rotation of an input pattern. The experiment was conducted with 500 Japanese yen coin and 500 Korean won coin.

Michael Nolle *et al.* [2] propose real-time system called Dagobert that can recognize 39 classes of coin and can reject unknown class, which is not recognized. There are mechanical parts, which can put the coin on and take the coin out from the conveyor belt. The system is equipped with two additional sensors measuring the thickness as well as the rough diameter of the current coin, which are used to trigger the image processing. For the coin detection, it is assumed that the conveyor belt used as background is homogenous and always darker (or brighter) than the coins. This makes a simple automatic threshold operation suffices for the segmentation. The center of gravity, perimeter and convex hull of the coin can be obtained easily. After applied edge detector, the system use binary string created from edge pixel as recognition feature.

Seth McNeill *et al.* [3] use vector quantization of edge feature image and kmeans algorithm to train machine learning system for classification. The data collection process was done using several background colors such as black, white, red and blue. The extensive segmentation testing show that red is the best background color for this step. The data set used in the experiment consists of approximately 400 images with similar rotation, lighting and size. Once the segmentation and cropping are done, the segmented coin images go through vertical edge detection and thresholding. The histogram of the image is constructed using vector quantization centroids. Bayes classifier is used in coin classification.

Marco Reisert *et al.* [4][5] used Hough transform to segment coin image, then shift coin center to origin position, scaling image so that coin radius is equal to 1 and use gradient image and fast Fourier transform to generate feature. The registration approach is to align two coins, which have maximal number of collinear gradient vectors. The gradient magnitude is completely neglected. The computation of the induced similarity measure can be done efficiently in the Fourier domain after the gradient directions are quantized. The classification is realized with a simple nearest neighbor classification method. The confidence value measurement is additional rejection criteria to meet the requirement of reducing false positive rate. The confidence value was computed from similarity scores, thickness, radius and angle pose differences.

Chomtip Pornpanomchai *et al.* [6] work with actual amulet, not another class of coin like all the works mentioned before. The system use correlation value to determine the recognition result. The correlation value is computed from the difference of grayscale value of all pixels between two images. The images in this work were taken in controlled environment (black box and spotlight).

Velu *et al.* [7] present Indian Coin counting system. The system can recognize the coins and sum up the total value in term of Indian National Rupee (INR). There are several assumptions in this system such as the coins should move on a conveyor belt like in Michael Nolle *et al.*, there has to be proper lighting focused on the coin, both sides of the coin have to be collected and all parameter of the coin must be measured accurately.

# Chapter 3
# Methodology

This chapter presents the procedures for amulet segmentation using grayscale conversion and Prewitt edge detection then describes the feature extraction using in this research. The similarity measurement using edge feature and texture feature is computed separately. The system diagram is shown in Figure 3.1.

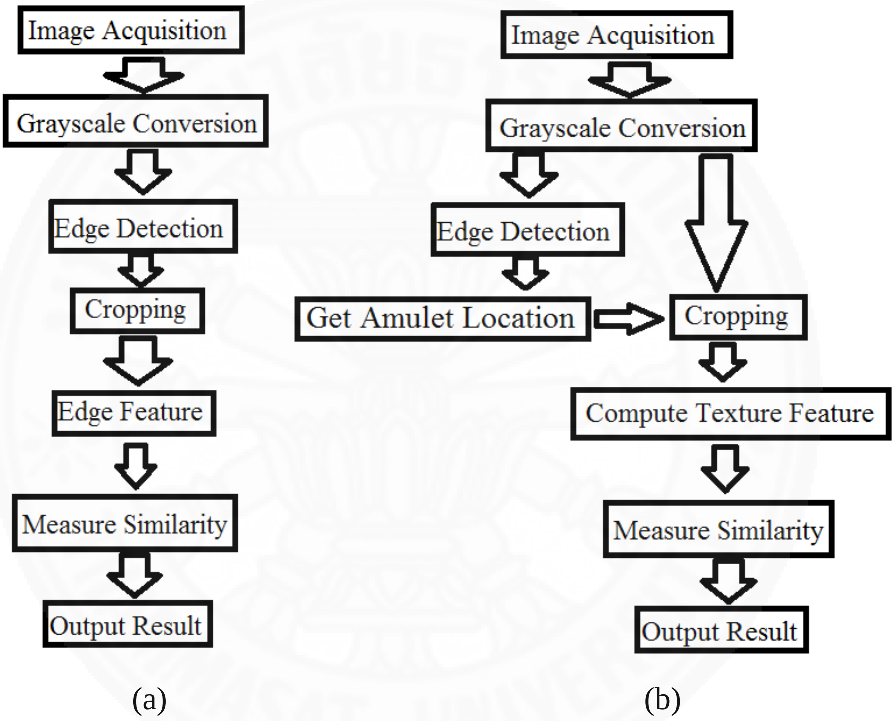



Figure 3.1 System diagram (a) using edge feature (b) using texture feature

## 3.1 Image acquisition

For the image acquisition, the condition is amulet has to be placed on the plain color background. The background color should be contrast with the amulet color so that background can be easily wiped out in segmentation process. In this work, the author places the amulets on red color paper, green color paper, pink color paper, blue color paper, yellow color paper, orange color paper and white color paper. The pictures used as dataset in this work are taken by mobile phone camera and compact camera. Some of sample data are shown in Figure 3.2.

Figure 3.2 Initial condition of the amulet images

**3.2 Grayscale conversion**

To make luminance intensity of the images to be easy to distinguish, the amulet image is converted from RGB color to grayscale color using grayscale conversion formula in Eq. (3.1).

$$Grayscale = 0.299 * R + 0.587 * G + 0.114 * B \quad (3.1)$$

Where

R is red color intensity in RGB image,

G is green color intensity in RGB image,

B is blue color intensity in RGB image.

The result of grayscale conversion is shown in Figure 3.3.

Figure 3.3 Grayscale conversion

### 3.3 Prewitt edge detection

To indicate the high contrast area of the amulet image, Prewitt edge detection [8] is applied to grayscale image of the amulet. The binary image with only edge pixels remain is shown in Figure 3.4.

Figure 3.4 Prewitt edge detection

### 3.4 Post processing

From the ideal edge image, the exact location of the amulet can be shown by finding the leftmost, rightmost, topmost, lowermost white pixel but there still are some noise white pixels which make segmentation go wrong. As shown in the right subfigure of Figure 3.4, the leftmost, rightmost, topmost, lowermost white pixel is not the exact location of amulet. To fix this problem, 2-D linear FIR filters were applied before getting amulet location. The masks of the filters are shown in Figure 3.5. The images filtered by those masks are shown in Figure 3.6.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(a)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.1 | 0 | 0 | 0 |

(b)

Figure 3.5 7x7 Masks (a) horizontal (b) vertical

(a) (b)

Figure 3.6 Noise reduction of edge images

(a) After apply vertical line filter (b) After apply horizontal line filter

After getting amulet position, the grayscale image is cropped like shown in Figure 3.7 before doing feature extraction.

Figure 3.7 Cropped amulet image

### 3.5 Edge feature retrieval

Beside the grayscale image from section 3.2, the Prewitt edge detection image from section 3.3 is also cropped after getting amulet position from section 3.4. The cropped image is shown Figure 3.8.

Figure 3.8 Cropped amulet edge image

### 3.6 Local binary pattern

LBP [9] is texture feature derived from difference in the grayscale value between center pixel and neighborhood pixels. The area of neighbor system can be square with any odd number length but the greater distance, the correlation between center pixel and neighbor decrease. LBP feature can be deriving from the following formula.

$$T(n,c) = \begin{cases} 1, g(n) - g(c) \geq 0 \\ 0, g(n) - g(c) < 0 \end{cases} \tag{3.2}$$

Where

T(n,c) is binary code value of neighbor pixel n compared to center pixel c,

g(n) is grayscale value of neighbor pixel n,

g(c) is grayscale value of center pixel c,

For feature representation, if the grayscale value of neighbor pixel is not less than one of center pixel, the feature is represented by ‘1’ otherwise the feature is represented by ‘0’ like shown in Figure 3.9.

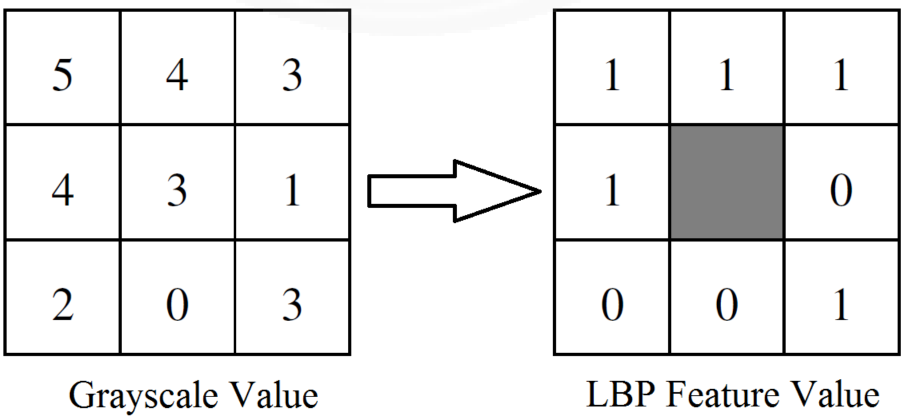



Figure 3.9 LBP derivation

In Figure 3.9, the example LBP calculation gives 8 bits of binary which can be represented by binary number ‘10001111b’.

With the larger neighborhood area, the more neighbor pixel stay in range. Figure 3.10 show example of 3x3 and 5x5 neighborhood area.

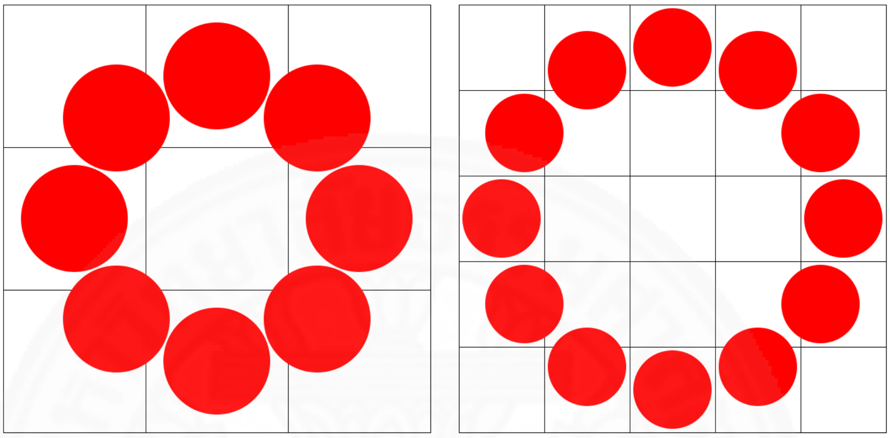


Figure 3.10 3x3 and 5x5 neighbor area

### 3.7 Local ternary pattern

LTP [10] is the improved version of LBP, which has ‘-1’ feature representative in additional form LBP. LTP feature is designed to increase LBP reliability in difficult lighting condition. LTP feature can be deriving from the following formula.

$$T(n,c) = \begin{cases} 1, g(n) - g(c) > t \\ 0, |g(n) - g(c)| \leq t \\ -1, g(n) - g(c) < t \end{cases} \quad (3.3)$$

Where

T(n,c) is ternary code value of neighbor pixel n compared to center pixel c,

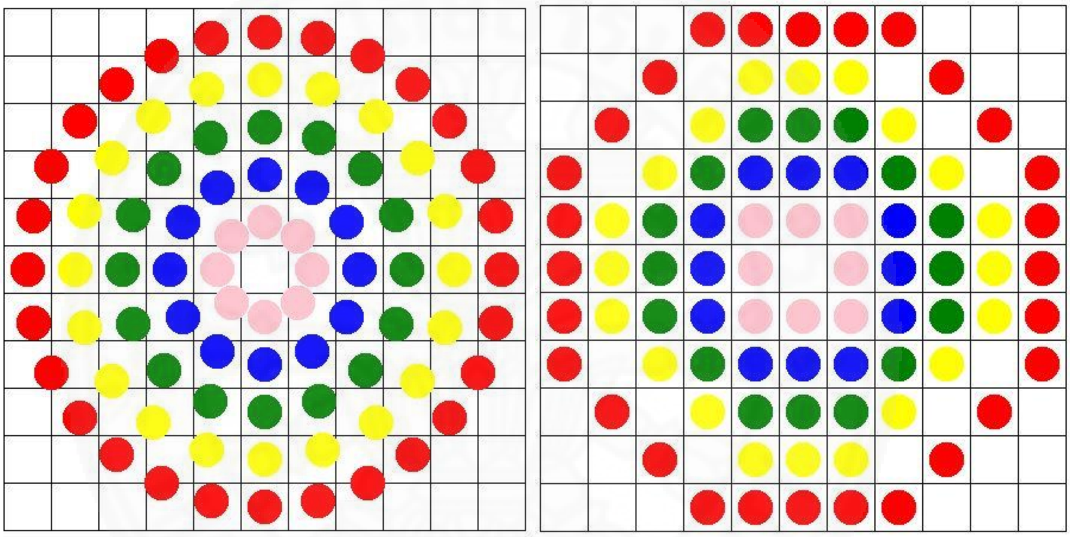
g(n) is grayscale value of neighbor pixel n,

g(c) is grayscale value of center pixel c,

t is threshold value.

### 3.8 Texture feature retrieval

The amulet segmented image from section 3.6 is resized to 54x54, 90x90, 126x126, 162x162, 198x198 pixels depends on the grid size for collecting 3x3, 5x5, 7x7, 9x9, 11x11 grid size LBP and LTP features respectively. The resized image is divided to 18x18 areas. In each area, LBP and LTP feature were collected. The neighbor pixels to collect LBP and LTP features per one grid area are shown in Figure 3.11.



(a) (b)

Figure 3.11 Texture feature collection per one grid area

(a) Show by radius distance from center (b) Show by actual pixel location

The example of 3*3 grid size LBP derivation is shown in Figure 3.12 and Figure 3.13.

| | | |
|---|---|---|
| 27 | 118 | 200 |
| 230 | 144 | 207 |
| 56 | 38 | 244 |

⇨

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | | 1 |
| 0 | 0 | 1 |

Figure 3.12 3*3 grid size LBP derivation in one grid area

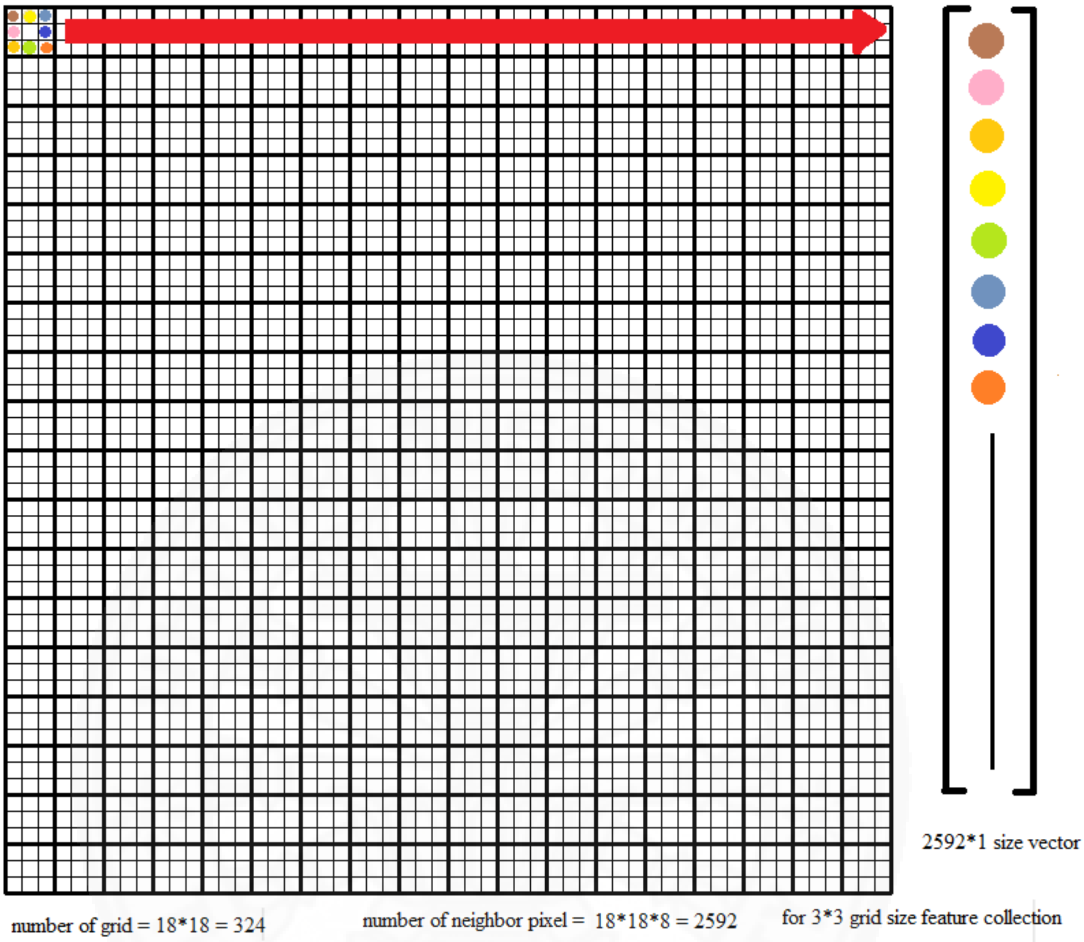



Figure 3.13 vector feature derivation from 54*54 pixel image

### 3.9 Experiment Setup

The experiments were done by using MATLAB on a laptop with 2.4GHz CPU and 4GB RAM. The template matching was conducted on 33 kinds of amulet with 630 images per kind and using 5-fold cross validation technic with each fold has 126 images per kind as shown in Figure 3.14. The 126 images of each kind in a fold consist of 18 images per each color background of 7 background colors used in this work. For the 18 images of each kind on the same color background, 9 images were collected in daylight and 9 images were collected in fluorescent light. The example image of amulet on each of background color is shown in Figure 3.15-3.21.

| Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
|---|---|---|---|---|
| Type1 126images | Type1 126images | Type1 126images | Type1 126images | Type1 126images |
| Type2 126images | Type2 126images | Type2 126images | Type2 126images | Type2 126images |
| Type3 126images | Type3 126images | Type3 126images | Type3 126images | Type3 126images |
| Type4 126images | Type4 126images | Type4 126images | Type4 126images | Type4 126images |
| Type5 126images | Type5 126images | Type5 126images | Type5 126images | Type5 126images |
| Type6 126images | Type6 126images | Type6 126images | Type6 126images | Type6 126images |
| Type7 126images | Type7 126images | Type7 126images | Type7 126images | Type7 126images |
| Type8 126images | Type8 126images | Type8 126images | Type8 126images | Type8 126images |
| Type9 126images | Type9 126images | Type9 126images | Type9 126images | Type9 126images |
| Type10 126images | Type10 126images | Type10 126images | Type10 126images | Type10 126images |
| Type11 126images | Type11 126images | Type11 126images | Type11 126images | Type11 126images |
| Type12 126images | Type12 126images | Type12 126images | Type12 126images | Type12 126images |
| Type13 126images | Type13 126images | Type13 126images | Type13 126images | Type13 126images |
| Type14 126images | Type14 126images | Type14 126images | Type14 126images | Type14 126images |
| Type15 126images | Type15 126images | Type15 126images | Type15 126images | Type15 126images |
| Type16 126images | Type16 126images | Type16 126images | Type16 126images | Type16 126images |
| Type17 126images | Type17 126images | Type17 126images | Type17 126images | Type17 126images |
| Type18 126images | Type18 126images | Type18 126images | Type18 126images | Type18 126images |
| Type19 126images | Type19 126images | Type19 126images | Type19 126images | Type19 126images |
| Type20 126images | Type20 126images | Type20 126images | Type20 126images | Type20 126images |
| Type21 126images | Type21 126images | Type21 126images | Type21 126images | Type21 126images |
| Type22 126images | Type22 126images | Type22 126images | Type22 126images | Type22 126images |
| Type23 126images | Type23 126images | Type23 126images | Type23 126images | Type23 126images |
| Type24 126images | Type24 126images | Type24 126images | Type24 126images | Type24 126images |
| Type25 126images | Type25 126images | Type25 126images | Type25 126images | Type25 126images |
| Type26 126images | Type26 126images | Type26 126images | Type26 126images | Type26 126images |
| Type27 126images | Type27 126images | Type27 126images | Type27 126images | Type27 126images |
| Type28 126images | Type28 126images | Type28 126images | Type28 126images | Type28 126images |
| Type29 126images | Type29 126images | Type29 126images | Type29 126images | Type29 126images |
| Type30 126images | Type30 126images | Type30 126images | Type30 126images | Type30 126images |
| Type31 126images | Type31 126images | Type31 126images | Type31 126images | Type31 126images |
| Type32 126images | Type32 126images | Type32 126images | Type32 126images | Type32 126images |
| Type33 126images | Type33 126images | Type33 126images | Type33 126images | Type33 126images |

Figure 3.14 Experiment data setup

Figure 3.15 Amulet on white background

Figure 3.16 Amulet on red background

Figure 3.17 Amulet on blue background

Figure 3.18 Amulet on green background

Fig. 3.19 Amulet on yellow background

Figure 3.20 Amulet on pink background

Figure 3.21 Amulet on orange background

For template matching using edge feature, the cropped amulet edge image from section 3.5 is resized to 200 pixels width and 200 pixels height, which is the size of all edge templates. The similarity between testing image and template image is computed from logical comparison of binary image. The formula is shown in equation Eq. (3.3). The template with highest similarity comparing with testing image is determined to be recognition result.

$$s = \sum_{i=0}^{200} \sum_{j=0}^{200} (A_{i,j} \equiv B_{i,j}) \tag{3.3}$$

Where

S is similarity value of two images A and B,

$A_{i,j}$ is grayscale value of binary image A at row I and column j,

$B_{i,j}$ is grayscale value of binary image B at row I and column j,

For template matching using texture feature, the feature collected from section 3.8 is kept in form of 2592x1, 3888x1, 5184x1, 6480x1, 9072x1 size vector for 3x3, 5x5, 7x7, 9x9, 11x11grid size LBP and LTP features respectively. The similarity between testing image and template image is calculated from dot product of texture feature from both images as shown in Figure 3.22.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = (1)(1)+(0)(1)+\ldots+(0)(0)+(1)(0)$$

Figure 3.22 Similarity calculation

# Chapter 4
# Result and Discussion

This chapter begins with numerical result of every method. The criteria for deciding the result is k-nearest-neighbor (kNN) which is to choose the majority of k best matches to be the result when k is positive odd integer. The majority of 1, 3, 5, 7, 9 best matches are the result for 1NN, 3NN, 5NN, 7NN, 9NN respectively. The next part of this chapter is comparison of computation time of all method used in this study.

For the edge feature, segmentation process use only Prewitt edge detection but this chapter will also show matching result using 4 different kinds of edge feature such as Sobel edge[11], Canny edge[12], Laplacian edge[13]. Beside edge features and texture features, the author also compares computation time and result with famous general purpose feature called SIFT keypoint descriptor[14]. The SIFT keypoint descriptor is computed using opencv library version 2.4.13 binding with python language. The template with the most matched descriptor is chosen to be recognition result.

The percentage accuracy of matching result is calculated from finding ratio between number of test image with correct recognition result and number of all test images and then multiply the ratio by 100.

Table 4.1 shows Percentage accuracy of matching result using Sobel edge feature. The average accuracy of 5 folds cross validation is about 33.79% at 1NN and the accuracy is decrease when increase k of kNN.

Table 4.1 Percentage accuracy of matching result using Sobel edge feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 28.2407 | 36.4198 | 26.5432 | 42.9012 | 34.8765 | 33.79628 |
| 3NN | 35.3395 | 32.2531 | 20.0617 | 31.4015 | 25.4630 | 28.90376 |
| 5NN | 25.6173 | 28.8580 | 20.0617 | 22.3765 | 22.5309 | 23.88888 |
| 7NN | 22.6852 | 25.0000 | 22.3765 | 18.0556 | 21.6049 | 21.94444 |
| 9NN | 18.6752 | 24.6914 | 20.9877 | 14.9691 | 18.5185 | 19.56838 |

Table 4.2 shows Percentage accuracy of matching result using Prewitt edge feature. The average accuracy of 5 folds cross validation is about 34.84% at 1NN and the accuracy is decrease when increase k of kNN but still is higher than accuracy using Sobel edge at every k of kNN.

Table 4.2 Percentage accuracy of matching result using Prewitt edge feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 29.4753 | 37.5000 | 27.7778 | 44.1358 | 35.3395 | 34.84568 |
| 3NN | 36.8827 | 33.4877 | 19.9074 | 32.4074 | 25.6173 | 29.66050 |
| 5NN | 26.8519 | 29.9383 | 20.679 | 23.3025 | 23.6111 | 24.87656 |
| 7NN | 23.1481 | 26.2346 | 22.2222 | 18.8272 | 22.0679 | 22.50000 |
| 9NN | 19.1358 | 25.3086 | 21.2963 | 15.7407 | 18.5185 | 19.99998 |

Table 4.3 shows Percentage accuracy of matching result using Laplacian edge feature. The average accuracy of 5 folds cross validation is about 36.29% at 1NN and the accuracy is decrease when increase k of kNN. The accuracy using Laplacian edge is slightly higher than accuracy using Prewitt edge but Laplacian edge also requires computation time about 1.2 times compared to Prewitt edge.

Table 4.3 Percentage accuracy of matching result using Laplacian edge feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 33.6420 | 22.9938 | 31.7901 | 45.8333 | 47.2222 | 36.29628 |
| 3NN | 26.0802 | 37.0370 | 18.2099 | 32.2531 | 42.1296 | 31.14196 |
| 5NN | 19.1358 | 35.4938 | 13.5802 | 22.2272 | 34.4136 | 24.97012 |
| 7NN | 15.1235 | 27.9321 | 10.6481 | 20.1600 | 31.3272 | 21.03818 |
| 9NN | 23.1481 | 22.8395 | 8.79630 | 17.4383 | 26.6975 | 19.78394 |

Table 4.4 shows Percentage accuracy of matching result using Canny edge feature. The average accuracy of 5 folds cross validation is about 77.77% at 1NN. The accuracy using Canny edge is highest compared with all three other edge features but Laplacian edge also requires computation time about 2 times compared to Prewitt edge which is longest computation time of all edge feature in this study.

Table 4.4 Percentage accuracy of matching result using Canny edge feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 82.7160 | 73.4568 | 70.2160 | 82.5617 | 79.9383 | 77.77776 |
| 3NN | 72.6852 | 84.8765 | 65.2778 | 75.1543 | 72.9938 | 74.19752 |
| 5NN | 68.6728 | 79.4753 | 54.9383 | 68.5185 | 70.8333 | 68.48764 |
| 7NN | 64.8148 | 76.2346 | 46.9136 | 61.7284 | 66.0494 | 63.14816 |
| 9NN | 58.4877 | 70.2160 | 41.0494 | 55.7099 | 62.1914 | 57.53088 |

Table 4.5 shows Percentage accuracy of matching result using 3*3 grid size LBP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is hardly decrease when increase k of kNN.

Table 4.5 Percentage accuracy of matching result using 3*3 grid size LBP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 3NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 5NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 7NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 9NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |

Table 4.6 shows Percentage accuracy of matching result using 3*3 grid size LTP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.6 Percentage accuracy of matching result using 3*3 grid size LTP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 99.8457 | 99.8457 | 99.6914 | 99.8457 | 99.8457 | 99.81484 |
| 3NN | 99.8457 | 99.8457 | 99.6914 | 99.8457 | 99.8457 | 99.81484 |
| 5NN | 99.6914 | 99.8457 | 99.6914 | 99.8457 | 99.6914 | 99.75312 |
| 7NN | 99.6914 | 99.8457 | 99.6914 | 99.8457 | 99.8457 | 99.78398 |
| 9NN | 99.6914 | 99.8457 | 99.5370 | 99.5370 | 99.6914 | 99.6605 |

Table 4.7 shows Percentage accuracy of matching result using 5*5 grid size LBP feature. The average accuracy of 5 folds cross validation is about 99.87% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.7 Percentage accuracy of matching result using 5*5 grid size LBP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 100 | 100.0000 | 99.8457 | 99.8457 | 99.6914 | 99.87656 |
| 3NN | 100 | 99.8457 | 99.8457 | 99.8457 | 99.6914 | 99.84570 |
| 5NN | 100 | 99.8457 | 99.8457 | 99.8457 | 99.6914 | 99.84570 |
| 7NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 9NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |

Table 4.8 shows Percentage accuracy of matching result using 5*5 grid size LTP feature. The average accuracy of 5 folds cross validation is about 99.84% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.8 Percentage accuracy of matching result using 5*5 grid size LTP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 99.8457 | 99.8457 | 99.6914 | 100.0000 | 99.8457 | 99.84570 |
| 3NN | 99.2284 | 99.8457 | 99.6914 | 99.8457 | 99.6914 | 99.66052 |
| 5NN | 99.2284 | 99.8457 | 99.6914 | 99.6914 | 99.5370 | 99.59878 |
| 7NN | 99.2284 | 99.8457 | 99.6914 | 99.5370 | 99.2284 | 99.50618 |
| 9NN | 99.2284 | 99.6914 | 99.3827 | 99.3827 | 99.9198 | 99.52100 |

Table 4.9 shows Percentage accuracy of matching result using 7*7 grid size LBP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is hardly decrease when increase k of kNN.

Table 4.9 Percentage accuracy of matching result using 7*7 grid size LBP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 3NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 5NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 7NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 9NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |

Table 4.10 shows Percentage accuracy of matching result using 7*7 grid size LTP feature. The average accuracy of 5 folds cross validation is about 99.78% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.10 Percentage accuracy of matching result using 7*7 grid size LTP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 99.6914 | 99.8457 | 99.6914 | 99.8457 | 99.8457 | 99.78398 |
| 3NN | 99.0741 | 99.8457 | 99.6914 | 99.6914 | 99.6914 | 99.59880 |
| 5NN | 99.0741 | 99.8457 | 99.6914 | 99.5370 | 99.5370 | 99.53704 |
| 7NN | 99.0741 | 99.8457 | 99.5370 | 99.3827 | 99.0741 | 99.38272 |
| 9NN | 99.0741 | 99.6914 | 99.5370 | 99.3827 | 98.7654 | 99.29012 |

Table 4.11 shows Percentage accuracy of matching result using 9*9 grid size LBP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is hardly decrease when increase k of kNN.

Table 4.11 Percentage accuracy of matching result using 9*9 grid size LBP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 3NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 5NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 7NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 9NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |

Table 4.12 shows Percentage accuracy of matching result using 9*9 grid size LTP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.12 Percentage accuracy of matching result using 9*9 grid size LTP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 99.8457 | 99.8457 | 99.6914 | 99.8457 | 99.8457 | 99.81484 |
| 3NN | 99.0741 | 99.8457 | 99.6914 | 99.8457 | 99.5370 | 99.59878 |
| 5NN | 98.9198 | 99.6914 | 99.6914 | 99.6914 | 99.0741 | 99.41362 |
| 7NN | 99.0741 | 99.6914 | 99.5370 | 99.3827 | 98.9198 | 99.3210 |
| 9NN | 99.0741 | 99.3827 | 99.5370 | 99.2284 | 98.4568 | 99.1358 |

Table 4.13 shows Percentage accuracy of matching result using 11*11 grid size LBP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.13 Percentage accuracy of matching result using 11*11 grid size LBP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 3NN | 100 | 99.8457 | 99.8457 | 99.8457 | 99.6914 | 99.84570 |
| 5NN | 100 | 99.8457 | 99.8457 | 99.8457 | 99.6914 | 99.84570 |
| 7NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |
| 9NN | 100 | 99.8457 | 99.8457 | 99.6914 | 99.6914 | 99.81484 |

Table 4.14 shows Percentage accuracy of matching result using 11*11 grid size LTP feature. The average accuracy of 5 folds cross validation is about 99.81% at 1NN and the accuracy is slightly decrease when increase k of kNN.

Table 4.14 Percentage accuracy of matching result using 11*11 grid size LTP feature

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|---|
| 1NN | 99.8457 | 99.8457 | 99.6914 | 99.8457 | 99.8457 | 99.81484 |
| 3NN | 99.3827 | 99.8457 | 99.6914 | 99.8457 | 99.6914 | 99.69138 |
| 5NN | 99.2284 | 99.6914 | 99.6914 | 99.6914 | 99.3827 | 99.53706 |
| 7NN | 99.2284 | 99.6914 | 99.6914 | 99.3827 | 99.0741 | 99.41360 |
| 9NN | 99.2284 | 99.5370 | 99.5370 | 99.2284 | 98.6111 | 99.22838 |

Table 4.15 shows Percentage accuracy of matching result using SIFT keypoint descriptor. The accuracy of all 5 folds in 5 folds cross validation is 100% but also take tremendous computation time as much as 92 times compared with 3*3 grid size texture features.

Table 4.15 Percentage accuracy of matching result using SIFT feature

| Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Average |
|---|---|---|---|---|---|
| 100 | 100 | 100 | 100 | 100 | 100 |

Table 4.16 shows computation time of every method used in this work. The time measurement starts when testing image is read and end when recognition result is known. Among the edge features, Sobel edge takes shortest computation time. Prewitt edge takes computation time slightly more than Sobel edge. Canny edge takes longest computation time among the edge features which is almost two times of Prewitt edge. For the texture features, LBP amd LTP take almost equal computation

time at the same grid size. The computation times of LBP and LTP are increasing when increase grid size but at 11*11 grid size, the computation time is still faster than Sobel edge. The SIFT keypoint descriptor takes long computation time more than all features in this study which is almost two hours.

Table 4.16 Computation time in second

| | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | Total |
|---|---|---|---|---|---|---|
| Sobel | 460.9781 | 451.3504 | 455.6027 | 452.739 | 456.7211 | 2277.391 |
| Prewitt | 486.0912 | 485.7395 | 491.1928 | 486.0196 | 486.9185 | 2435.961 |
| Laplacian | 622.4656 | 621.3507 | 622.7242 | 620.3942 | 628.7392 | 3115.674 |
| Canny | 939.9226 | 959.9155 | 933.6992 | 948.0208 | 930.9655 | 4712.524 |
| LBP3*3 | 81.61101 | 82.09881 | 81.58735 | 82.1827 | 81.97508 | 409.4549 |
| LTP3*3 | 81.55889 | 81.69491 | 82.09307 | 81.86571 | 82.42746 | 409.6400 |
| LBP5*5 | 98.57156 | 97.49566 | 98.23928 | 97.54278 | 97.89867 | 489.7479 |
| LTP5*5 | 97.85553 | 97.61807 | 98.57082 | 98.49263 | 97.89867 | 490.4357 |
| LBP7*7 | 102.592 | 104.0623 | 100.7746 | 101.769 | 100.7897 | 509.9875 |
| LTP7*7 | 103.0894 | 102.9292 | 101.2395 | 102.0709 | 100.6193 | 509.9482 |
| LBP9*9 | 106.1316 | 105.435 | 104.7524 | 106.0734 | 105.2539 | 527.6462 |
| LTP9*9 | 105.6228 | 105.504 | 104.0907 | 103.9716 | 110.1446 | 529.3338 |
| LBP11*11 | 113.5277 | 113.5666 | 112.8898 | 112.4654 | 113.5911 | 566.0406 |
| LTP11*11 | 111.9601 | 114.0983 | 113.2247 | 112.3349 | 112.3882 | 564.0063 |
| SIFT | 7418.299 | 7295.848 | 7042.497 | 7700.427 | 7639.149 | 37096.22 |

Among the methods used in this study, 3*3 grid size LBP gives the second best performance in accuracy with not much different from SIFT keypoint descriptor which gives best accuracy. The 3*3 grid size LBP also gives best performance in computation time. Therefore, the author chooses 3*3 grid size LBP to develop MATLAB application which works with USB camera.

# Chapter 5
# Amulet Recognition in a Nutshell System

This chapter demonstrates the system called 'Amulet Recognition in a Nutshell' which is created to make user to be able to do amulet recognition with image from USB camera. The minimum focus distance of USB camera used with this system should not be longer than 10 cm. The system is developed on MATLAB platform. The system takes about 24 milliseconds computation time per one input image. The system screen when the user starts the system is shown in Figure 5.1.

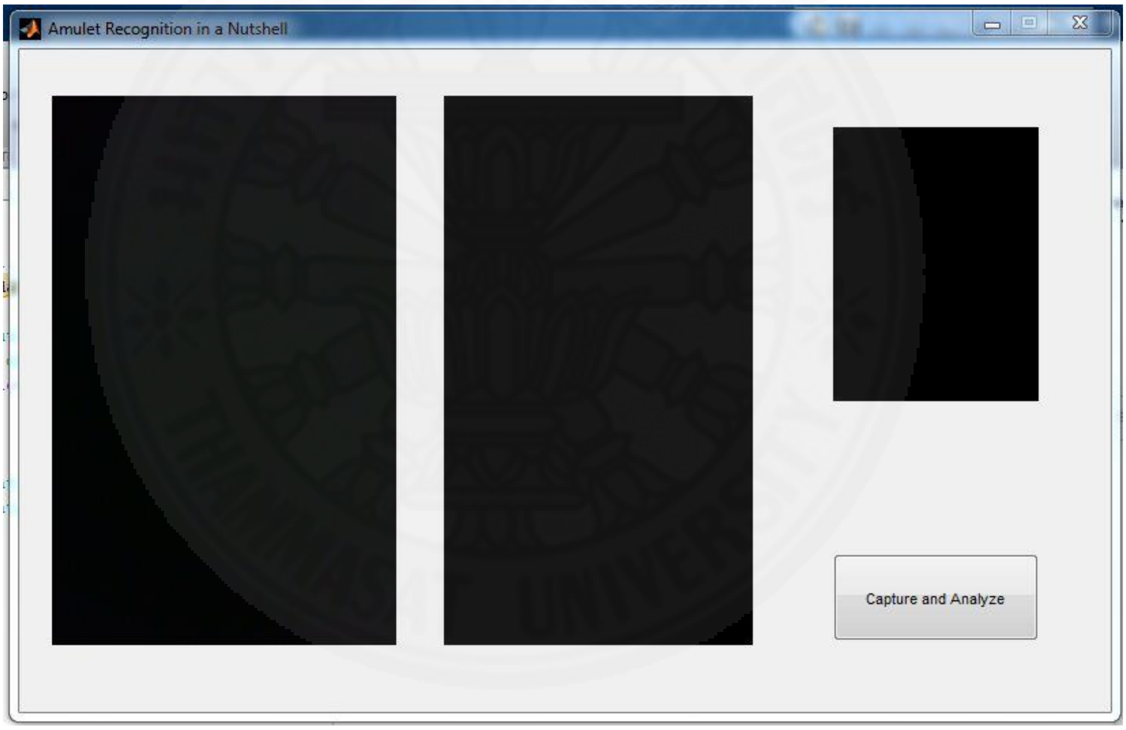



Figure 5.1 System screen

The left side of system screen contains real time video input from USB camera. When user clicks 'Capture and Analyze' button, input image at the moment is captured and processed as mentioned in chapter 3. The user can see grayscale segmented image in the middle of system screen and result of amulet recognition can be seen in the right side of system screen as shown in Figure 5.2 and Figure 5.3.

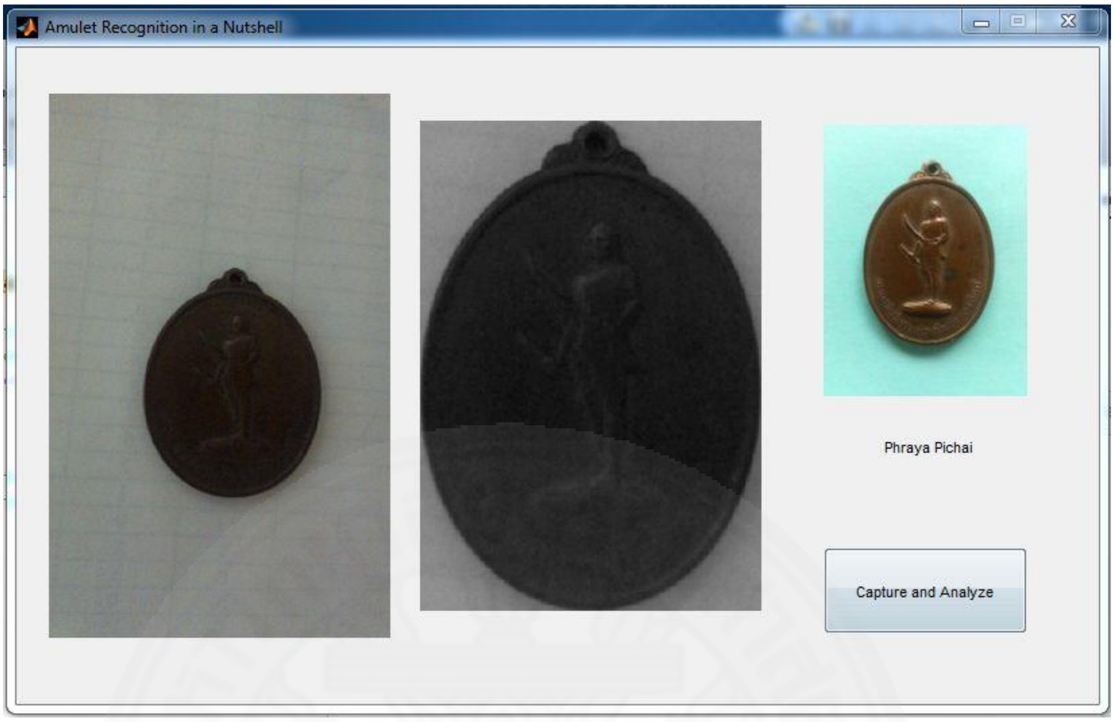



Figure 5.2 Correct recognition with Phraya Pichai

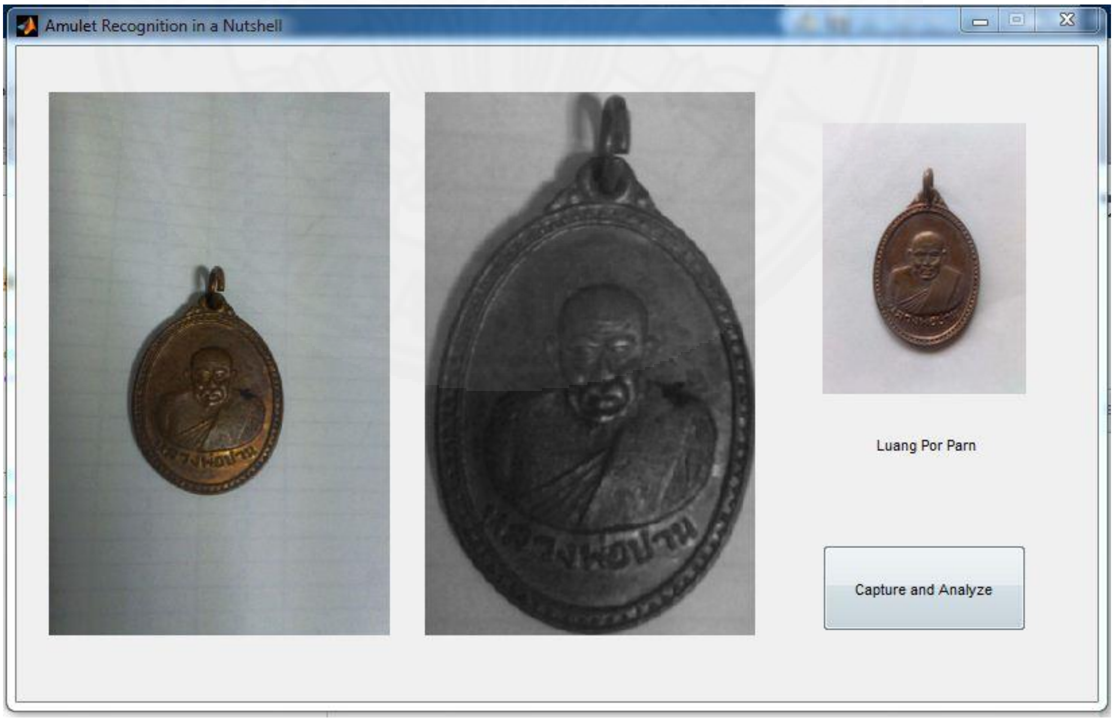



Figure 5.3 Correct recognition with Luang Por Parn

The system uses segmentation process mentioned in chapter 3 which requires the amulet to be placed on plain background to make background vanish when Prewitt edge detection is applied. The background color does not need to be white. The inappropriate lighting condition can cause shadow of camera and user's body which makes wrong segmentation. The wrong segmentation can make recognition also wrong as shown in Figure 5.4.

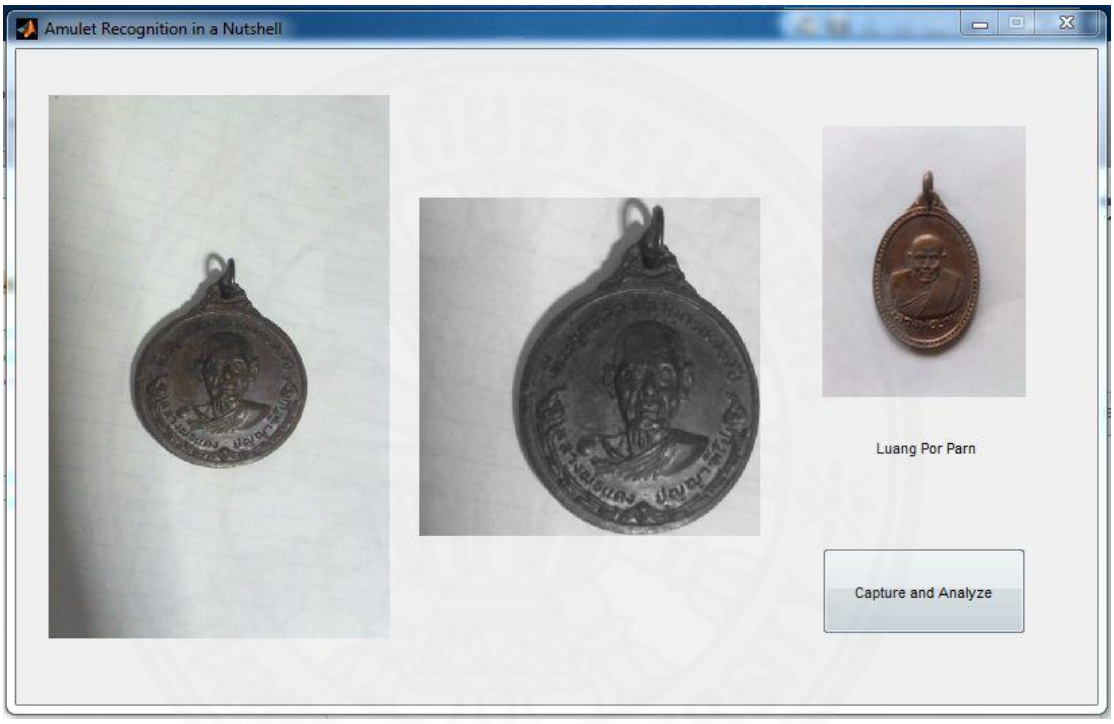



Figure 5.4 Wrong recognition

The author conducts out-of-sample test experiment of the system with 5 users. For each user, the testing was conduct with 5 kinds of amulet which the users select themselves. The users use the system the chosen amulet 100 times for each of amulet kind. The test experiment result of the system is shown in Table 5.1-5.5.The out-of-sample test result gives lower performance compared to the in-sample test result in chapter 4 because shadow of camera and user's body affect not only segmentation but also the luminance of amulet in the image.

Table 5.1 Test experiment conducted with user #1

| Amulet Name | Correct | Incorrect | Accuracy |
|---|---|---|---|
| Phraya Pichai | 91 | 9 | 91% |
| Luang Por Thongyu | 93 | 7 | 93% |
| Luang Poo Van | 95 | 5 | 95% |
| Luang Por Pew | 85 | 15 | 85% |
| Luang Por Noi | 88 | 12 | 88% |

Table 5.2 Test experiment conducted with user #2

| Amulet Name | Correct | Incorrect | Accuracy |
|---|---|---|---|
| Luang Por Daeng | 90 | 10 | 90% |
| Luang Poo Van | 95 | 5 | 95% |
| Luang Por Sothorn | 92 | 8 | 92% |
| Luang Por Su Kho | 91 | 9 | 91% |
| Luang Por Kaew | 88 | 12 | 88% |

Table 5.3 Test experiment conducted with user #3

| Amulet Name | Correct | Incorrect | Accuracy |
|---|---|---|---|
| Luang Por Parn | 86 | 14 | 86% |
| Luang Por Thongyu | 90 | 10 | 90% |
| Luang Por Daeng | 92 | 8 | 92% |
| Luang Poo Sarm | 94 | 6 | 94% |
| Luang Poo Van | 96 | 4 | 96% |

Table 5.4 Test experiment conducted with user #4

| Amulet Name | Correct | Incorrect | Accuracy |
|---|---|---|---|
| Luang Por Parn | 88 | 12 | 88% |
| Luang Por Si Kho | 90 | 10 | 90% |
| Luang Por Pew | 86 | 14 | 86% |
| Phraya Pichai | 85 | 15 | 85% |
| Luang Poo Van | 90 | 10 | 90% |

Table 5.5 Test experiment conducted with user #5

| Amulet Name | Correct | Incorrect | Accuracy |
|---|---|---|---|
| Praputtha Chinnarad | 88 | 12 | 88% |
| Luang Por Daeng | 95 | 5 | 95% |
| Luang Por Thongyu | 93 | 7 | 93% |
| Luang Poo Sarm | 90 | 10 | 90% |
| Luang Poo Duad | 87 | 13 | 87% |

# Chapter 6
# Conclusions and Recommendations

In this thesis, the author presented image-based amulet recognition method. The process begins with amulet segmentation. Amulet segmentation process does require the amulet to be placed on the plain background. The next process is feature extraction and the last process is template matching. The features used in this work are edge feature, which indicate high contrast area in the image, and texture feature, which describe object surface image. From comparison, texture feature give better result than edge feature. For possible improvement, the computation in feature matching process may be able to be simplified using artificial neural network with hidden layer. Furthermore, applying machine learning to this work can make adding more kinds of amulet to the system more easy considering this work consists only 33 kinds of amulet and there still are more many in the market.

# References

[1] Fukumi, M., Omatu, S., Takeda, F., and Kosaka, T. (1992). Rotation-invariant neural pattern recognition system with application to coin recognition. *Neural Networks, IEEE Transactions on*, *3*(2), 272-279.

[2] Nölle, M., Penz, H., Rubik, M., Mayer, K., Holländer, I., and Granec, R. (2003, December). Dagobert-a new coin recognition and sorting system. In *Proceedings of the 7th Internation Conference on Digital Image Computing-Techniques and Applications (DICTA'03), Syndney, Australia*.

[3] McNeill, S., Schipper, J., Sellers, T., and Nechyba, M. C. (2004). Coin Recognition using Vector Quantization and Histogram Modeling. In *2004 Florida Conference on Recent Advances in Robotics (FCRAR)*.

[4] Reisert, M., Ronneberger, O., and Burkhardt, H. (2006, September). An efficient gradient based registration technique for coin recognition. In *Proc. of the Muscle CIS Coin Competition Workshop, Berlin, Germany* (pp. 19-31).

[5] Reisert, M., Ronneberger, O., and Burkhardt, H. (2007). A fast and reliable coin recognition system. In *Pattern Recognition* (pp. 415-424). Springer Berlin Heidelberg.

[6] Pornpanomchai, C., Wongkorsub, J., Pornaudomdaj, T., and Vessawasdi, P. (2010, April). Buddhist amulet recognition system (BARS). In *Computer and Network Technology (ICCNT), 2010 Second International Conference on* (pp. 495-499). IEEE.

[7] Velu, C. M., Vivekanadan, P., and Kashwan, K. R. (2011). Indian coin recognition and sum counting system of image data mining using Artificial Neural Networks. *International Journal of Advanced Science and Technology*, *31*, 67-80.

[8] Prewitt, J. M. (1970). Object enhancement and extraction. *Picture processing and Psychopictorics*, *10*(1), 15-19.

[9] Mäenpää, T., and Pietikäinen, M. (2005). Texture analysis with local binary patterns. *Handbook of Pattern Recognition and Computer Vision*, *3*, 197-216.

[10] Tan, X., and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on*, *19*(6), 1635-1650.

[11] Vincent, O. R., & Folorunso, O. (2009, June). A descriptive algorithm for sobel image edge detection. In *Proceedings of Informing Science & IT Education Conference (InSITE)*(Vol. 40, pp. 97-107).

[12] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence,* (6), 679-698.

[13] Mlsna, P. A., & Rodriguez, J. J. (2005). Gradient and laplacian edge detection. In *Handbook of image and video processing*, Elsevier Inc..

[14] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91-110.

# Appendices

# Appendix A
## List of Publications

1. Sauthananusuk, T., Charoenlarpnopparut, C., Kondo, T., Bunnun, P., and Hirohiko, K. (2014). Thai Amulet Recognition Using Simple Feature. In I*nformation and Communication Technology for Embedded Systems (ICICTES2014), The International Conference on,* Ayutthaya, Thailand.

2. Sauthananusuk, T., Charoenlarpnopparut, C., Kondo, T., Bunnun, P., and Hirohiko, K. (2014). Thai amulet recognition based-on texture feature analysis. *Proceedings of Annual Conference on Engineering and Technology (ACEAT 2014).* Osaka, Japan (pp. 61-70).

# Appendix B

## Amulet segmentation MATLAB source code

```
h1=[0 0 0 0 0 0 0;
   0 0 0 0 0 0 0;
   0 0 0 0 0 0 0;
   0.1 0.1 0.1 0.1 0.1 0.1 0.1;
   0 0 0 0 0 0 0;
   0 0 0 0 0 0 0;
   0 0 0 0 0 0 0];

h2=[0 0 0 0.1 0 0 0;
   0 0 0 0.1 0 0 0;
   0 0 0 0.1 0 0 0;
   0 0 0 0.1 0 0 0;
   0 0 0 0.1 0 0 0;
   0 0 0 0.1 0 0 0;
   0 0 0 0.1 0 0 0];

im1=imread(['image directory']);
im2=rgb2gray(im1);
im3=imresize(im2,[720,540]);
im4=edge(im3,'prewitt');
ver = im2bw(imfilter(im4,h2,'replicate'));
hor = im2bw(imfilter(im4,h1,'replicate'));
lr=sum(ver);
ud=sum(transpose(hor));

for j = 1:540
   if lr(j)>0
      r=j;
   end
   if lr(541-j)>0
      l=541-j;
   end
   if ud(j)>0
      d=j;
   end
   if ud(721-j)>0
      u=721-j;
   end
end
for j = 541:720
   if ud(j)>0
      d=j;
```

```
    end
    if ud(721-j)>0
        u=721-j;
    end
end

im3=im3(u:d,l:r);
im3=imresize(im3,[720,540]);
```

# Appendix C

## Feature extraction MATLAB source code

```
im1=imread(['segmented image']);
im2=imresize(im1,[90,90]);
LBP=zeros(6480,1);
LTP=zeros(6480,1);
k=1;

for i=3:5:88
  for j=3:5:88
    if im2(i,j)<im2(i-2,j-1)
        LBP(k,1)=0;
        if im2(i,j)+10<im2(i-2,j-1)
            LTP(k,1)=-1;
        else
            LTP(k,1)=0;
        end
    else
        LBP(k,1)=1;
        if im2(i,j)-10>im2(i-2,j-1)
            LTP(k,1)=1;
        else
            LTP(k,1)=0;
        end
    end
    k=k+1;

    if im2(i,j)<im2(i-2,j)
        LBP(k,1)=0;
        if im2(i,j)+10<im2(i-2,j)
            LTP(k,1)=-1;
        else
            LTP(k,1)=0;
        end
    else
        LBP(k,1)=1;
        if im2(i,j)-10>im2(i-2,j)
            LTP(k,1)=1;
        else
            LTP(k,1)=0;
        end
    end
    k=k+1;
```

```
if im2(i,j)<im2(i-2,j+1)
   LBP(k,1)=0;
   if im2(i,j)+10<im2(i-2,j+1)
      LTP(k,1)=-1;
   else
      LTP(k,1)=0;
   end
else
   LBP(k,1)=1;
   if im2(i,j)-10>im2(i-2,j+1)
      LTP(k,1)=1;
   else
      LTP(k,1)=0;
   end
end
k=k+1;

if im2(i,j)<im2(i-1,j-2)
   LBP(k,1)=0;
   if im2(i,j)+10<im2(i-1,j-2)
      LTP(k,1)=-1;
   else
      LTP(k,1)=0;
   end
else
   LBP(k,1)=1;
   if im2(i,j)-10>im2(i-1,j-2)
      LTP(k,1)=1;
   else
      LTP(k,1)=0;
   end
end
k=k+1;

if im2(i,j)<im2(i-1,j-1)
   LBP(k,1)=0;
   if im2(i,j)+10<im2(i-1,j-1)
      LTP(k,1)=-1;
   else
      LTP(k,1)=0;
   end
else
   LBP(k,1)=1;
   if im2(i,j)-10>im2(i-1,j-1)
      LTP(k,1)=1;
   else
      LTP(k,1)=0;
```

```
    end
end
k=k+1;

if im2(i,j)<im2(i-1,j)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i-1,j)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i-1,j)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i-1,j+1)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i-1,j+1)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i-1,j+1)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i-1,j+2)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i-1,j+2)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
```

```
    if im2(i,j)-10>im2(i-1,j+2)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i,j-2)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i,j-2)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i,j-2)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i,j-1)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i,j-1)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i,j-1)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i,j+1)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i,j+1)
        LTP(k,1)=-1;
    else
```

```
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i,j+1)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i,j+2)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i,j+2)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i,j+2)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i+1,j-2)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i+1,j-2)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i+1,j-2)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i+1,j-1)
```

```
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i+1,j-1)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i+1,j-1)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i+1,j)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i+1,j)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i+1,j)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
end
k=k+1;

if im2(i,j)<im2(i+1,j+1)
    LBP(k,1)=0;
    if im2(i,j)+10<im2(i+1,j+1)
        LTP(k,1)=-1;
    else
        LTP(k,1)=0;
    end
else
    LBP(k,1)=1;
    if im2(i,j)-10>im2(i+1,j+1)
        LTP(k,1)=1;
    else
        LTP(k,1)=0;
    end
```

```
end
k=k+1;

if im2(i,j)<im2(i+1,j+2)
   LBP(k,1)=0;
   if im2(i,j)+10<im2(i+1,j+2)
      LTP(k,1)=-1;
   else
      LTP(k,1)=0;
   end
else
   LBP(k,1)=1;
   if im2(i,j)-10>im2(i+1,j+2)
      LTP(k,1)=1;
   else
      LTP(k,1)=0;
   end
end
k=k+1;

if im2(i,j)<im2(i+2,j-1)
   LBP(k,1)=0;
   if im2(i,j)+10<im2(i+2,j-1)
      LTP(k,1)=-1;
   else
      LTP(k,1)=0;
   end
else
   LBP(k,1)=1;
   if im2(i,j)-10>im2(i+2,j-1)
      LTP(k,1)=1;
   else
      LTP(k,1)=0;
   end
end
k=k+1;

if im2(i,j)<im2(i+2,j)
   LBP(k,1)=0;
   if im2(i,j)+10<im2(i+2,j)
      LTP(k,1)=-1;
   else
      LTP(k,1)=0;
   end
else
   LBP(k,1)=1;
   if im2(i,j)-10>im2(i+2,j)
```

```
            LTP(k,1)=1;
         else
            LTP(k,1)=0;
         end
      end
      k=k+1;

      if im2(i,j)<im2(i+2,j+1)
         LBP(k,1)=0;
         if im2(i,j)+10<im2(i+2,j+1)
            LTP(k,1)=-1;
         else
            LTP(k,1)=0;
         end
      else
         LBP(k,1)=1;
         if im2(i,j)-10>im2(i+2,j+1)
            LTP(k,1)=1;
         else
            LTP(k,1)=0;
         end
      end
      k=k+1;
   end
end
```