IMPLEMENTING CENTRE-SYMMETRIC LOCAL BINARY PATTERN (CS-LBP) IN RASPBERRY PI FOR IRIS RECOGNITION

MUHAMMAD FIRDAUS BIN ABDULLAH

UNIVERSITI SAINS MALAYSIA

2018

IMPLEMENTING CENTRE-SYMMETRIC LOCAL BINARY PATTERN (CS-LBP) IN RASPBERRY PI FOR IRIS RECOGNITION

by

MUHAMMAD FIRDAUS BIN ABDULLAH

Thesis submitted in partially fulfilment of the requirements for degree of

Bachelor of Engineering (Electronic Engineering)

JUNE 2018

ACKNOWLEDGEMENTS

First of all, I would like to thank the School of Electrical and Electronic Engineering of Universiti Sains Malaysia for giving me an opportunity to conduct my Final Year Project as I learned a lot along the way and for giving me the tools and knowledge to finish my thesis and project such as the seminars held and the equipment, software and hardware provided.

Next, I would like to thank my project supervisor, En. Ahmad Nazri Ali for all the guidance and advice given in order for me to finish this project. His guidance and suggestions had really help me finish my final year project. Furthermore, I would like to thank him for all the time and patience given from him throughout the course of this final year project.

Last but not least, I would like to thank my friends for the support and ideas given as I finish this final year project. The ideas they gave me in order for me to design the GUI, how to do all the testing and also how to finish my thesis would not be forgotten.

Table of Contents

Chapter 5: Conclusion

List of Tables

List of Figures

List of Abbreviations

CS-LBP	Centre-Symmetric Local Binary Pattern
GUI	Graphical User Interface
GPIO	General-Purpose Input/Output
HE	Histogram equalization
IDLE	Integrated Development and Learning Environment
LBP	Local Binary Pattern
RAM	Random-Access Memory
SD	Secure Digital
SSH	Secure Shell
SVM	Support Vector Machine
VNC	Virtual Network Computing

Abstract

The human iris, like fingerprints, has an amazing characteristic where it is unique to every person. Like fingerprints, iris is often use in biometric system for identification and security. Currently, there are many methods for image identification, enhancement, compression and many other image processing applications. The focus of this project is to apply a known descriptor called the Centre-Symmetric Local Binary Pattern (CS-LBP), for iris recognition. The CS-LBP method will be applied to an image of a human iris that had undergo segmentation and detection process for feature extraction. These features will describe all individual iris where it will then be tested using a classifier in order to determine the reliability of the applied method to classify different iris pattern with different people.

The CS-LBP is shown to have a few advantages given its computational-friendly operations. It describes the texture and grey-levels of an image by comparing the centre-symmetric pairs of opposite pixels in reference to the centre pixel. The descriptor also has tolerance to illumination changes, robustness on flat image areas and computational efficiency.

All this process will be implemented in a small, low-cost computer called the Raspberry Pi 3B. The purpose of implementing this iris recognition program into the Raspberry Pi is that the Raspberry Pi is a cheap yet powerful platform. The small size of the computer also helps with portability of the device if it can be further integrated into a hand carry-on device. This can be further applied into the security sector for executing an iris recognition program. A Graphical User Interface (GUI) will also be developed for an easy interaction between user and the program.

Abstrak

Seperti cap jari manusia, iris mata, mempunyai satu ciri yang menakjubkan di mana setiap manusia, mempunyai corak iris yang berbeza. Seperti cap jari, iris mata juga kerap di gunakan di dalam sistem biometric bagi tujuan pengenalan dan keselamatan. Pada masa kini, banyak cara yang boleh digunakan untuk pengenalan, penambahbaikan, mampatan dan banyak lagi bagi aplikasi di dalam bidang pemprosesan gambar. Tumpuan projek ini adalah menggunakan satu diskriptor gambar dipanggil Pusat-Simetri Corak Binari Tempatan (CS-LBP) untuk tujuan pengesahan iris. Kaedah CS-LBP ini akan digunakan terhadap gambar iris yang telah melalui proses segmentasi dan pengesanan untuk pengekstrakan ciri gambar tersebut. Ciri-ciri ini akan menerangkan corak setiap iris di mana semua ciri ini akan di uji menggunakan penguji klasifikasi bagi menetukan kebolehpercayaan kaedah ini.

Kaedah ini telah dibuktikan mempunyai beberapa kelebihan setelah mengambil kira ciri mesra-komputasi operasi kaedah ini. Ia menerangkan tekstur dan tahap kelabu sesuatu gambar dengan membandingkan pasangan pusat-simetri dua piksel yang bertentangan di mana piksel tengah berada di antara dua piksel tersebut. Deskriptor ini juga mempunyai toleransi terhadap pencahayaan dan kecekapan komputasi.

Semua proses ini akan dilaksanakan di dalam sebuah komputer kecil dan kos rendah dipanggil ‘Raspberry Pi’. Tujuan bagi pelaksanaan program pengenalan iris mata ini ke dalam ‘Raspberry Pi’ disebabkan ciri komputer ini yang berkos rendah tapi berkuasa. Saiz komputer yang kecil ini boleh membantu dari segi aspek mudah alih peranti. Hal ini boleh selanjutnya diaplikasikan di dalam sektor keselamatan bagi menjalankan program pengenalan iris mata ini. Antara muka pengguna grafik (GUI), juga akan dibangunkan agar memudahkan pengguna berinteraksi dengan program.

CHAPTER 1

Introduction

1.1 Research Background

There are many ways of implementing image recognition as many methods have been proposed throughout the years. However, some of these methods are very complex and hard to implement. No matter what, at the end of the day, the limiting factor of everything is cost, time and complexity of a method.

In this project, we will be implementing an image feature extraction method called Centre-Symmetric Local Binary Pattern (CS-LBP) in a low-cost computer called the Raspberry Pi 3B for human iris recognition. This computer is small (about the size of a credit card) and has powerful processing power taking into account its cost, size, organization and specification.

CS-LBP is a descriptor that describe the visual features of the contents in images. It is a descriptor that describes the interest region of an image where the features of the image are extracted and can then be further processed whether for recognition, classification or other relevant applications. CS-LBP combines the strengths of another descriptor called Scale-Invariant Feature Transform (SIFT) and a simple but efficient texture operator called Local Binary Pattern (LBP). This descriptor is proven to have several advantages such as tolerance to illumination changes, robustness on flat image areas and computational efficiency compared to SIFT.

In this project, the CS-LBP descriptor is going to be used to describe the image features and contents of the human's iris. The descriptor is going to be implemented using

Raspberry Pi, a small and low-cost computer with amazing processing power. The Raspberry Pi uses Broadcom BCM2837 64bit ARM Cortex-A53 Quad Core Processor SoC, running at 1.2GHz and has a 1 GB RAM.

The CS-LBP will be implemented and executed inside the Raspberry Pi by programming, using python language

1.2 Problem Statement

This project aims to implement the CS-LBP descriptor method into Raspberry Pi. However, unlike normal computer with high end hardware specifications, the Raspberry Pi is a low-cost computer with a limited storage (depending on the size of SD card used) and hardware specification such as Random-Access Memory (RAM). These hardware specifications may lead to limited amount of functionality such as maybe it is only able to store only a few people's iris images and features and may not be able to store a lot for it to really make a difference. This project is to determine just that.

Furthermore, this project will determine whether the Raspberry Pi can execute the CS-LBP algorithm efficiently and accurately. The CS-LBP will be used to extract features of an iris image that has undergo segmentation and normalization process. Segmentation is a process where the position and size of the iris and pupil are detected. The features will then be tested for classification using a classifier to test whether the extracted features from few different people are really different showing that the descriptor works well in the Raspberry Pi.

1.3 Objectives

The aim of this project is to successfully implement the CS-LBP descriptor method in Raspberry pi in order to extract the features of human iris for recognition.

The objectives are:

1) To implement Centre-Symmetry Local Binary Pattern (CS-LBP) for feature description of the human's iris in Raspberry Pi.
2) To develop a Graphical User Interface (GUI) for the implementation of CS LBP to ease extraction process.
3) To test the extracted features for classification using Support Vector Machine (SVM) program.

1.4 Research scope

This project only applies the CS-LBP descriptor to an image of iris that has already undergoes segmentation and detection. The features extracted using the descriptor will then be tested using a classifier in order to determine the reliability of implementing this descriptor in Raspberry Pi.

The descriptor will be implemented using Python language and its image processing library, the OpenCV. Furthermore, a GUI application will be developed for an easy interaction with the program.

1.5 Report structure

This section will explain the structure of this report. The report firstly consists of Chapter 1, the Introduction. This chapter explains the research background, problem statement, objectives and the research scope.

The second chapter, Literature Review provides insight on research, methods and information related in order to proceed with this project. This chapter explains the methods used in feature extraction, the platform for implementation which is the Raspberry Pi and also, the programming language and libraries needed in order to implement this project.

The third chapter, Methodology, will give a detailed flow of the overall project, how it can be done, how to implement the descriptor mentioned before, into Raspberry Pi and how to program the descriptor in Python with help of the OpenCV library and the Tkinter GUI library.

The fourth chapter, Results and Discussion, will give the results obtained from the project, and a discussion of these results. The chapter will show the output of the CS-LBP program, the Graphical User Interface (GUI) and the testing result using the SVM classifier program.

Next, fifth chapter, Conclusion, will conclude the project and also will touch a little bit about future works and improvements that can be made to this project.

Finally, readers can find the references, and all the appendices where Appendix A will show the CS-LBP program code in Python, Appendix B will show the results, Appendix C will show all the human iris image that will be tested in this project. Lastly, Appendix D will show the features extracted for each person.

CHAPTER 2

Literature Review

2.1 Introduction

All research regarding this project will be portrait in this chapter. The study will be based on how the project can be implemented. Research involves what kind of language that will be used to code the CS-LBP inside the Raspberry Pi. Furthermore, the research also involves the inner workings of the feature descriptor (CS-LBP) and how it is derived from previous works related to the field.

2.2 Characterization of interest region

Classification of only the interest region is important where this project will implement the description of the iris and only the iris of an individual. In order to achieve that, there are some methods or descriptors that had been proposed throughout the years. They are called Local Binary Pattern (LBP), Scale-Invariant Feature Transform (SIFT) and finally the highlight of this project, the Centre Symmetric Local Binary Pattern (CS-LBP) where it combines both the benefits of both descriptors for a more efficient descriptor.

2.3 Local Binary Pattern (LBP)

Interest region can be characterized or described very well by the distribution of its local features [2]. This method can help achieve just that. This method was proposed by the idea that local binary pattern plays an important role in defining local image texture

and these texture features can then be described by their occurrence in histogram [1]. Further image analysis can be done through the histogram. The proposed approach had been proven to be unaffected by grey-scale variation or in other words, their grey-scale monotonic transformation will not affect the description of features [1]. This means that illumination changes to the image will not affect the descriptor's performance. The descriptor also has been widely used in face recognition [2]. Another great benefit that this descriptor offers is that it only requires simple computational operations [1]. This shows that this descriptor is easy to be implemented and realize. This descriptor is also fast to compute [3]. Each pixel in an interest region of an image can be described by its neighbouring pixel's grey-levels. As described by equation (1) [3], the value will be set to one if the neighbouring pixel has higher or equal grey-levels, otherwise it will be zero [1][2][3]. Figure 2.1 will show how the LBP operator is being realize.

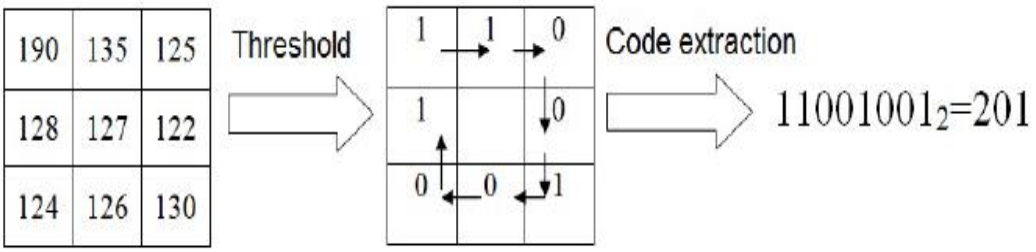


Figure 2.1: LBP operator [2]

$$LBP_{R,N}(x,y) = \sum_{i=0}^{N-1} s(n_i - n_c)2^i, \quad s(x) = \begin{cases} 1, & x \geqslant 0, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{(Equation 2.1)}$$

2.4 Scale-Invariant Feature Transform (SIFT)

The SIFT descriptor uses a 3D histogram where it represents gradient locations and orientation. These location and orientation has magnitude weighted with a Gaussian window overlaid over the region of interest [3].

2.5 Centre-Symmetric Local Binary Pattern (CS-LBP)

CS-LBP is a method proposed by Marko Heikkila, Matti Pietikainen, & Cordelia Schmid. It is a descriptor that implements the LBP method but more efficiently. The LBP produce a long histogram while the CS-LBP produces only half the value. LBP produces 256 (2^8) different binary patterns while CS-LBP only produces 16(2^4)[3]. These values describe the neighbouring pixels. The way CS-LBP works is that instead of taking the difference between the centre pixel and each of its neighbouring pixel like LBP, it takes and compares the centre-symmetric pairs of opposite pixels [3]. This can be further understood by referring to Figure 2.2.

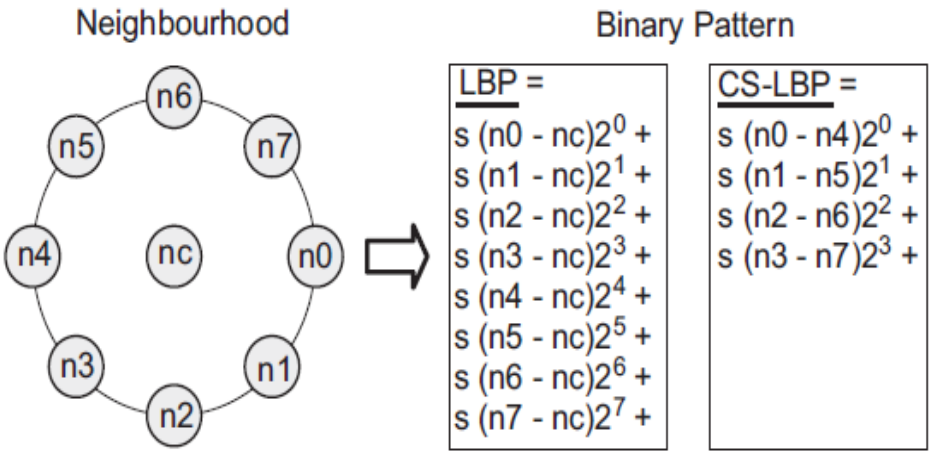


Figure 2.2: LBP and CS-LBP features for a neighbourhood of 8 pixels [3]

On flat images, the CS-LBP can be made more robust. This can be achieved with a small value, T by thresholding T with the grey-level difference as follows [3]:

$$CS\text{-}LBP_{R,N,T}(x,y) = \sum_{i=0}^{(N/2)-1} s(n_i - n_{i+(N/2)})2^i, \qquad \text{(Equation 2.2)}$$

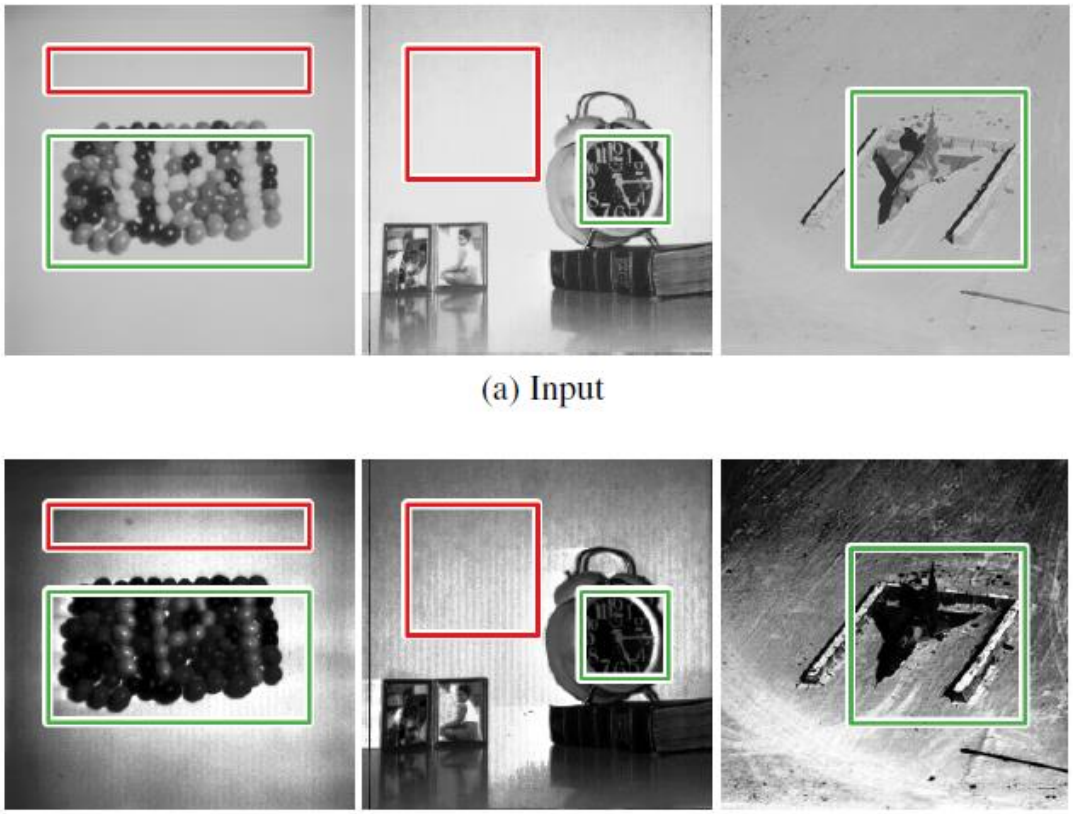
$$s(x) = \begin{cases} 1, & x > T, \\ 0 & \text{otherwise}, \end{cases} \qquad \text{(Equation 2.3)}$$

2.6 Histogram Equalization (HE)

Histogram equalization is one of the most popular and simplest image enhancement method to be implemented [11]. Because many image processing algorithms requires high quality inputs, image enhancement is needed to increase the contrast of an image with low dynamic range and hidden image details [4]. Figure 2.3 shows the effects of HE in a low contrast image.

Figure 2.3: a) Girl image b) Histogram equalized girl image [11]

This high contrast can provide good visual quality. However, the drawbacks of traditional HE is that it also increases the contrast of background noises [4]. This effect can be further portraited in Figure 2.4.



(a) Input

(b) Enhanced images obtained using HE

Figure 2.4: Appearance of enhanced contrast background noises (red) in HE images. [4]

However, this simple method can still be used if we implement CS-LBP for feature extraction since CS-LBP only describes an interest region which is the iris. Background noises will not affect the feature extraction.

HE is needed in this project as grayscale image from a Raspberry Pi camera can sometimes be in low contrast. Therefore, it needs to be enhanced before further image processing.

2.7 The Raspberry Pi

The Raspberry Pi is a low-cost computer. It is very small, a size of a credit card but with a rather powerful processing power. It can be plugged into a monitor or TV to access its desktop environment, just like a normal sized PC or laptop. Even so, it can still be accessed by other computer via the same network through PuTTy (a SSH and telnet client) where it will display the terminal of the Raspberry Pi so that we can interact with it. Other alternative to access its desktop environment is through VNC viewer where the Raspberry Pi's IP address will be required for access.

Raspberry Pi provides a great learning environment for people to learn programming languages such as Python (which will be used in this project) and familiarize themselves with inner workings of computers. It also provides GPIO pins where the computer can interact with the physical world by integrating end effectors such as servo motor, stepper motor, robot arm and so on.

2.7.1 Evaluation on the Effective Processing Time of Raspberry Pi 3B

In this section, readers are directed to ref [5]. This paper presents the comparison of effective processing time between a few platforms where one of it is Raspberry Pi 3B computer. The Raspberry Pi is low-cost, has friendly development environment, easy to configure and is great for research-specific applications.

The processing time of these platforms are compared by identifying fake images. The result shows that the processing time does not only depends on the image quality and dimension, but also the platform's and software facility such as the evaluated platform's organization, specifications and components.

The Raspberry Pi 3B shows great performance taking into account its system configuration, cost and size. Table 2.1 and Table 2.2 shows the compared platform and its running time.

Configuration	***Laptop1 (Matlab 2015a)***	***Laptop 2 (Visual Studio 12.0)***	***Raspberry Pi-3B (OpenCV + Cmake)***	***Raspberry Pi-3B (Qt Creator + C++)***
Micro processor	*Intel Core i5-5200U 2.2GhZ*	*Intel Core i3-M330U 2.13GhZ*	*BCM2837, 1.2GHz Quadcore ARMv8*	*BCM2837, 1.2GHz Quadcore ARMv8*
Memory	*4GB DDR3L (1600MHz)*	*4GB DDR3*	*1GB-LP DDR2 (900MHz)*	*1GB-LP DDR2 (900MHz)*
Platform	***Window 8.1, 64bits***	***Window 7 32bits***	***Raspbian***	***Raspbian***

Table 2.1: Differently tested software and hardware

Picture Size	**Processing Time (Seconds)**			
	Laptop1	**Laptop2**	***Raspberry Pi-3B (OpenCV + Cmake)***	***Raspberry Pi-3B (Qt Creator + C++)***
Flower.tiff 536x356	14	198	41	28
Garden.jpg 500x375	27	34	34	26
Pyramid.jpg 203x150	3.6	3	3	2
Library.jpg 737x492	48	734	136	69

Table 2.2: Running time of different systems

From both tables, it is concluded that the Raspberry Pi performs rather well with OpenCV library (which will be used in this project). The running time of the Raspberry Pi is lower than laptop 2 with an Intel Core i3 at 2.13 GHz compare to the Raspberry Pi which uses ARMv8 at 1.2GHz. Of course, the Raspberry Pi cannot match the running time of Laptop 1 which uses sophisticated hardware.

With that said, this shows that the Raspberry Pi is capable of image processing with average running time and high accuracy at a very affordable price [5]. Other than that, this study also managed to create a GUI application with the Qt Creator. However, this GUI creator is C++-based whereas this project uses Python where we will be using Tkinter as it has already been built-in with Python as a library.

The Raspberry Pi is found to have limited capabilities in processing high resolution images but are still reliable and low-cost in implementing ordinary image processing [5]. It is also easy to configure and can be further integrated into a hand-held image processing platform due to its size.

Further implementation of image processing in Raspberry Pi can also be referred to ref [6] [7] [8] [9] [10].

2.7.2 Python, OpenCV and Tkinter

In this project, the CS-LBP will be implemented inside the Raspberry Pi using the programming language Python. The language can be learned from many resources such as [14].

OpenCV is an open source library originally developed by Intel. It has many functions and algorithms for implementing computer vision and image processing. OpenCV library [11] can be install in Raspberry Pi. The library can then be used in Raspberry Pi's Python Integrated Development and Learning Environment (IDLE).

Apart from that, this project will create a Graphical User Interface (GUI) in order for the user of the CS-LBP program to interact with the program via the GUI. This GUI library is called Tkinter [13] where it is already built-in inside Python libraries upon installation. The upside of using Raspberry Pi is that upon installing Raspbian (Pi's operating system), the Python's IDLE is already installed.

CHAPTER 3

Methodology

3.1 Introduction

This chapter will elaborate on the methodology to be implemented in this project. The outline of this chapter will include all the flow and procedure of executing the project such as the overall flow regarding what needs to be done before implementing the CS-LBP descriptor for feature extraction and the flow to actually implement the descriptor for iris feature extraction. The CS-LBP algorithm and justifications on why all these methods and tools are used will also be elaborated.

3.2 Overall project flow

This section will describe the overall project flow. Figure 3.1 best describes the flow of this project. Before starting the project, research and literature review regarding and relating to the project must be done in order to understand the inner workings of the project. Areas to look into are image processing in general, the descriptor that are going to be used which is CS-LBP, the Raspberry Pi computer and its specifications, and also the programming language to be used along with libraries that can help us develop the project. In this project, the programming language that are going to be used is Python which already has its development environment built-in inside the Raspberry Pi image processing library, OpenCV along with the GUI library, Tkinter will also be used to help ease our programming during the implementation and coding of the CS-LBP descriptor and its GUI.

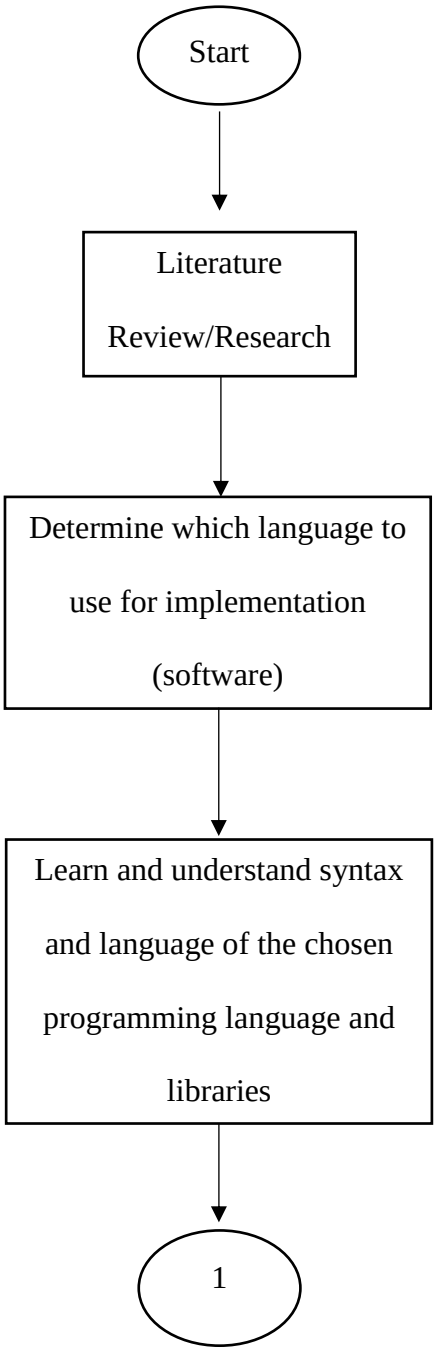


Figure 3.1: Overall project flow (cont)

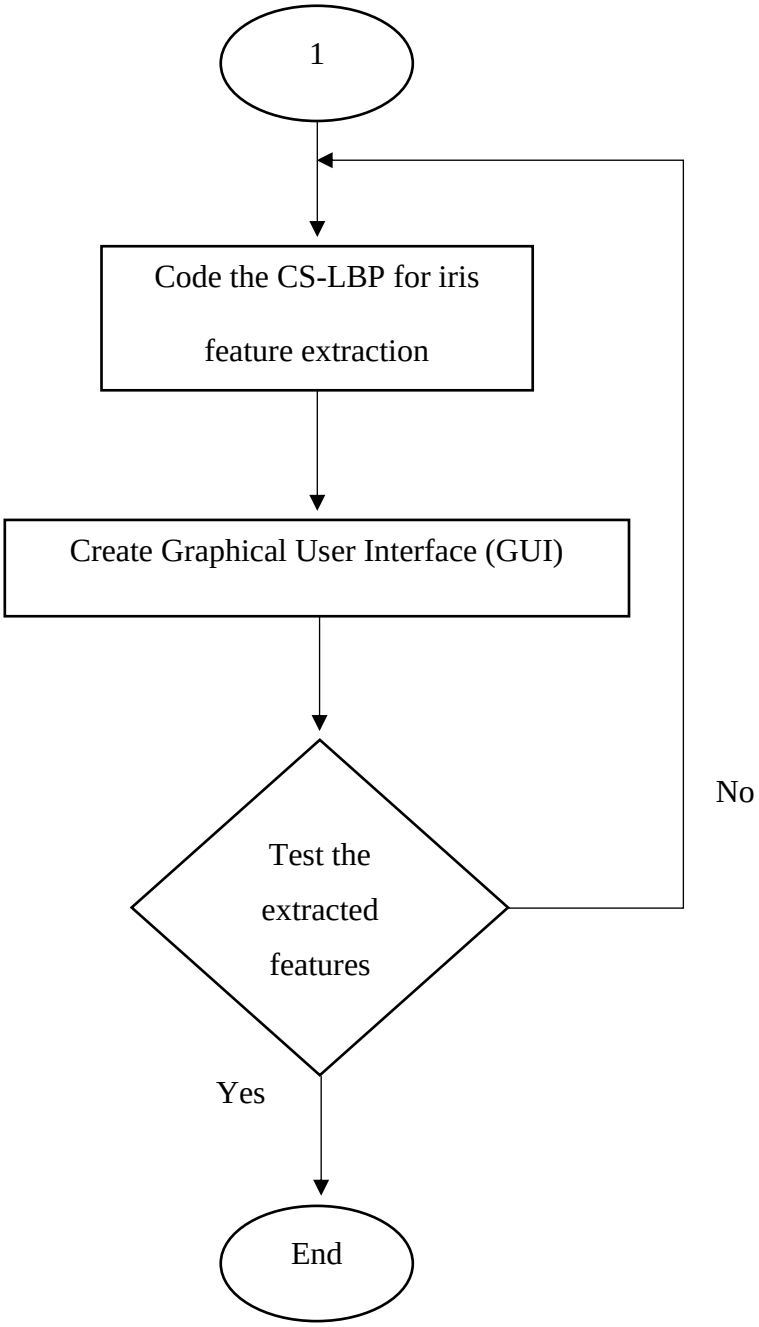


Figure 3.1: Overall project flow

Next, after determining the language and libraries that are going to be used to implement the program, the syntax and algorithms of the language and its libraries needs to be fully fathom in order for us to successfully implement the program into Raspberry Pi. To achieve this, readers are advised to refer to [11], [13] and [14].

After that, the next step is to code the CS-LBP descriptor in Python language. With the help of the OpenCV library, the coding process can be made much easier and faster such as reading the image array using the imread() and imshow() function. The focus of this project is to code the CS-LBP for the iris feature extraction. The algorithm and in-depth explanation about this method can be found in section 3.4.

Once the coding of the CS-LBP program has been done, the GUI for the program will be developed. This can be achieved using the Tkinter library. The GUI will allow ease of use to user. The GUI can then be used to extract the features of all the samples for testing. This can make the process to be faster and less tedious. The next step is to test the features extracted from different people using an SVM program, to determine whether the CS-LBP can be implemented inside the Raspberry Pi efficiently.

3.3 CS-LBP program flow

The flow for implementing the CS-LBP descriptor will be described in this section as shown in Figure 3.2. Firstly, the program should be able to load the image of an iris from the SD card of the Raspberry Pi into the CS-LBP program. Take note that the iris image has undergo segmentation and normalization process as shown in Figure 3.3. This image will then be used for further processing.

The person's name will be set for identification and the ROI will be set. The image of the iris will then be subjected into an enhancement process called the Histogram Equalization (HE) where it will cause the contrast of the image to be enhanced and clearer to the naked eye.

After that, the CS-LBP descriptor will be implemented where this descriptor will extract all the features of the iris image. The descriptor will give out the local binary pattern of the interest region of the iris. Figure 3.3 shows the interest region of the iris that will be described by the CS-LBP descriptor.

Next, the values of the local binary pattern will be saved into a text file where it will describe the number of frequency for each binary pattern values. These frequencies of values can also be represented in a histogram. There will be a total 10 person where each will be unique, with 7 samples each. Finally, the created text file from the program will be subject to testing. The features extracted in the files will be tested using a classifier.

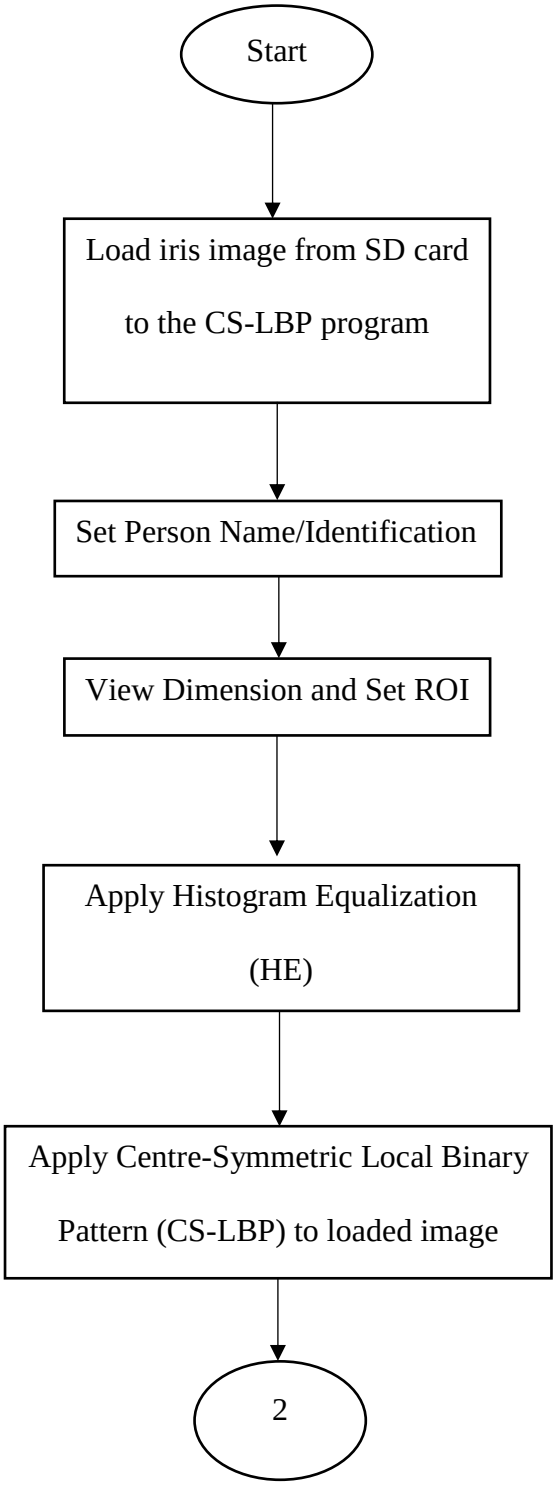


Figure 3.2: The flow of the CS-LBP program (continue)

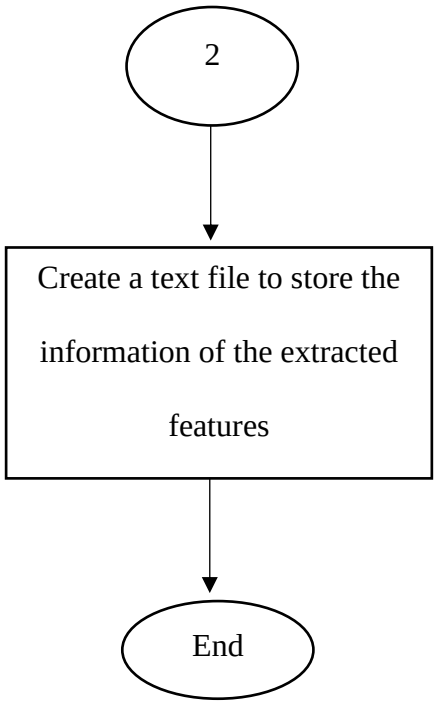


Figure 3.2 The flow of the CS-LBP program

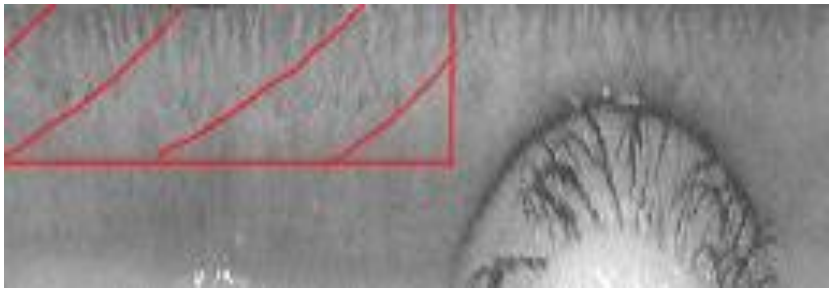
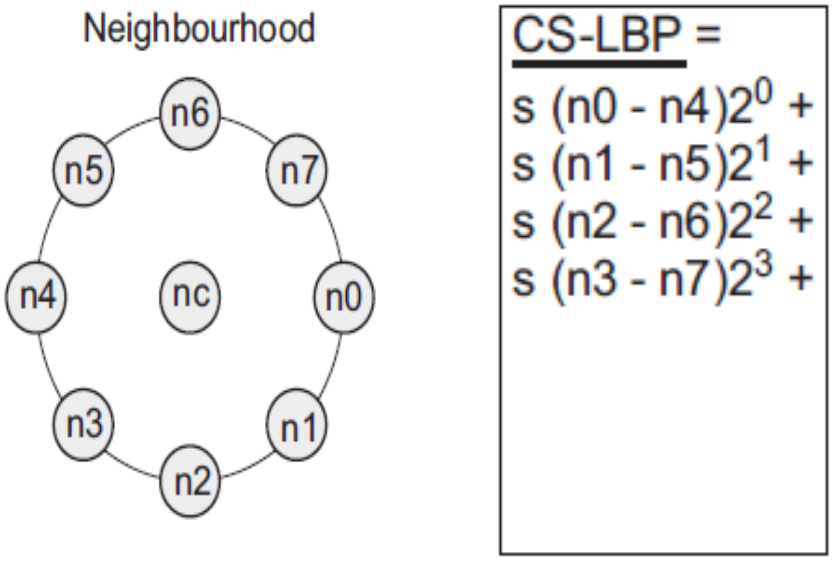


Figure 3.3: Image of an iris that has undergo segmentation and normalization (red area indicates interest region)

3.4 The CS-LBP algorithm

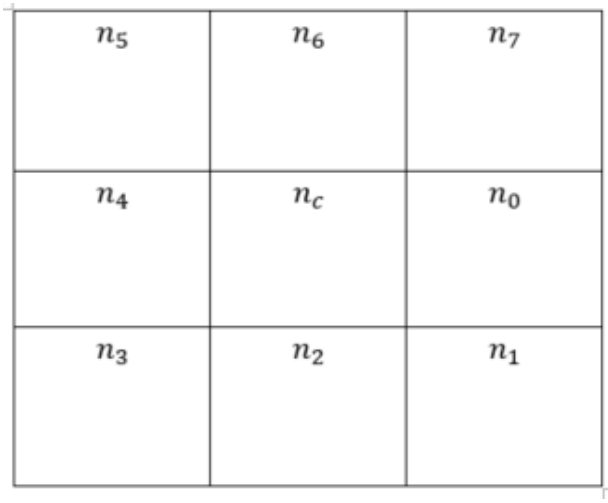
This section will show how the CS-LBP algorithm works. The CS-LBP uses the aforementioned equation (2) in Section 2.5. First of all, the algorithm will calculate the difference between centre-symmetric opposite pixel pairs. This can be clearly shown in Figure 3.4.



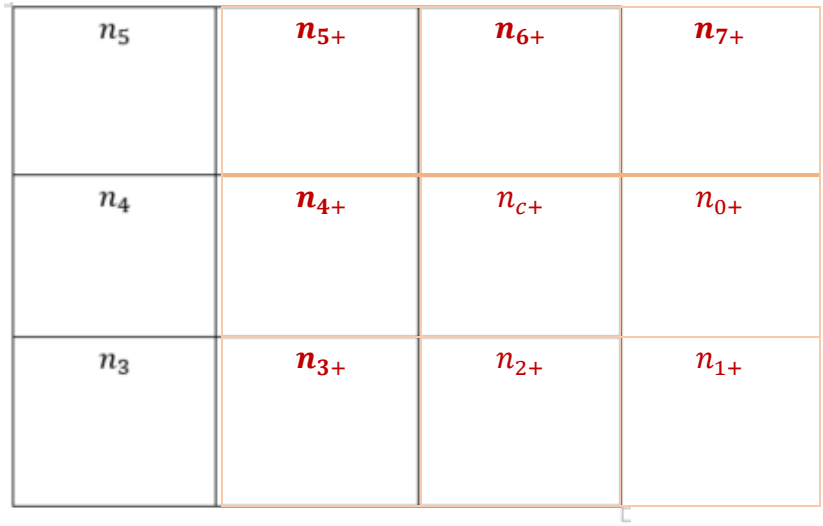
(a) (b)

Figure 3.4: (a) The CS-LBP neighbourhood pairs (b) CS-LBP operations

As in Figure 3.3, the interest region will be scanned one at a time from pixel to pixel. The centre pixel with grey-level, n_c , will act as reference point to apply the other neighbourhood pixels.



(a)



(b)

Figure 3.5: (a) Simplified CS-LBP neighbourhood

(b) next pixel CS-LBP value calculation

From Figure 3.5(a) the simplified diagram of the CS-LBP pixel neighbourhood can be shown. The way CS-LBP is calculated are as follows:

$$CS-LBP_{(x,y)} = s(n_0 - n_4)2^0 + s(n_1 - n_5)2^1 + s(n_2 - n_6)2^2 + s(n_3 - n_7)2^3$$

$$= s(x_1)2^0 + s(x_2)2^1 + s(x_3)2^2 + s(x_4)2^3$$

$$= CS-LBP_{(x,y)} < 16 \qquad \text{(Equation 3.1)}$$

Where, $s(x) = \begin{cases} 1, & x > 0 \\ 0, & otherwise \end{cases}$ (Equation 3.2)

Therefore, the value of $CS-LBP_{(x,y)}$ will never be above 16. This is because the CS-LBP compares opposite pixels instead of one-by-one like the LBP method. Next, the program will calculate the value of CS-LBP for the next centre pixel as shown in figure 3.5(b). This means that the program will calculate the next $CS-LBP_{(x+1,y)}$ value.

Since the CS-LBP calculate for every existing pixel value inside the interest region, therefore, the total number of values will be as follows:

$$Total\ number\ of\ values = height \times width \qquad \text{(Equation 3.3)}$$

The height and width represent the number of pixels. These values will then be turn into features by analysing the frequency for each value where it will then be represented in the form shown as follows:

$$CS-LBP\ value : Frequency \qquad \text{(Equation 3.4)}$$

The values that will exist are 0 to 16. Therefore, the features extracted that will be saved in a text file will be arranged as follows:

$$0: f_1 \quad 1: f_2 \quad 2: f_3 \quad 3: f_4 \quad \ldots\ldots\ldots\ldots \quad 16: f_{17} \qquad \text{(Equation 3.5)}$$

This distribution of values describes the interest region of the iris and should be unique to every person.

Figure 3.6 shows the execution flow of CS-LBP:

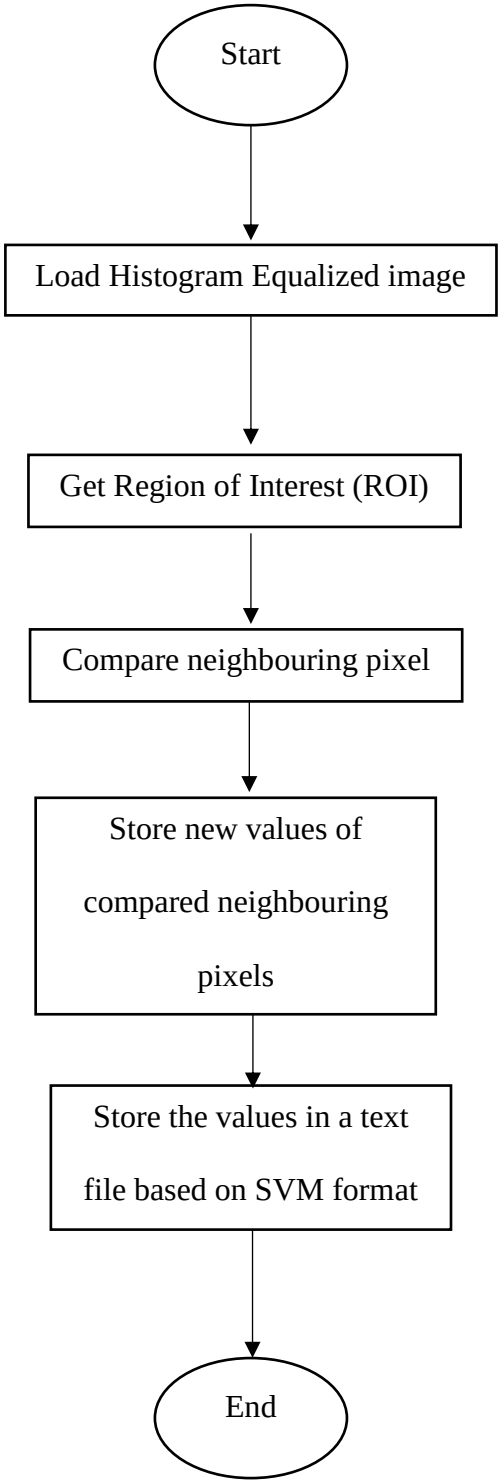


Figure 3.6: Centre-Symmetric Local Binary Pattern (CS-LBP) execution