Obstacle Avoidance System For Visually Impaired Person

Apipol Niyomsak 55090052
Karunyapas Dangruan 55090005
Pongrawee Jutadhammakorn 55090033
Siwatch Luxsameepicheat 55090049

Supervisor:
Dr.Chaiwat Nuthong

Project report submitted in partial fulfillment of the requirements for Software Project 1

Bachelor of Engineering Program in Software Engineering
Academic Year 2014, Semester 2
International College
King Mongkut's Institute of Technology Ladkrabang

Project Report

13016290 : Software Project 1

Academic Year 2014, Semester 2

B.Eng in Software Engineering

International College, King Mongkut's Institute of Technology Ladkrabang

Title: Obstacle Avoidance System For Visually Impaired Person

Authors:

Apipol Niyomsak 55090052

Karunyapas Dangruan 55090005

Pongrawee Jutadhammakorn 55090033

Siwatch Luxsameepicheat 55090049

Approved for submission

.....

(Dr.Chaiwat Nuthong)

Advisor

Date .....

Obstacle Avoidance System For Visually Impaired Person

Apipol Niyomsak 55090052
Karunyapas Dangruan 55090005
Pongrawee Jutadhammakorn 55090033
Siwatch Luxsameepicheat 55090049
Academic Year 2014

Abstract

A cane that blind people use today may not be sufficient to help them navigate throughout the environment. Especially when they have to be constantly aware of surrounding obstacles both mobile and immobile. By using only the cane, blind people often get their head injured by hitting head-level objects. The cane only provides floor-level awareness but not head-level awareness. Moreover, they could not tell whether the obstacle is human or not unless they move close enough. The aims of this study is to assist visually impaired person to avoid obstacles as many as possible with a portable device. By working together as a blind's cane complementary, the system assists blind in avoiding further obstacles the cane cannot reach. The modeling involved using ultrasonic sensors as a proximity detector, vibrators as a haptic feedback provider, and camera as the eye of object detection. The result showed that obstacles can be detected further away compared to the ordinary cane with an early haptic feedback through vibrators. 72 percents of human detection correctness for each image captured from a camera. An analysis showed that larger picture increase detection accuracy, but also a significant increase in execution time. An important factor in designing the system is to provide feedback to user as soon as possible which will limit the design. Further research is recommended to reduce execution time on human detection algorithm, also to increase detection range of the proximity sensors.

Acknowledgements

First of all we are thankful to Dr.Chaiwat Nuthong who is project supervisor for your financial and technical support and for providing necessary guidance concerning project implementation. His support are essential for implement the project in quality outcomes.

Apipol Niyomsak 55090052
Karunyapas Dangruan 55090005
Pongrawee Jutadhammakorn 55090033
Siwatch Luxsameepicheat 55090049

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Because blind people unable to see the way clearly. They have to use a cane to check for surrounding objects. They have to swing the cane around their vicinity in order to verify their surroundings before making a move. Sometimes, blind people get themselves injured due to unaware surface level difference on a floor or ceiling. Blind people have to able to use cane effectively. Skillful enough to identify and navigate through the environment.

1.2 Objectives

To assist visually impaired person navigate their way through the environment. Avoiding obstacle collision as many as possible using a portable device

1.3 Problem Statement

Blind people may not be able to assess far obstacles effectively and have to be constantly aware of surroundings.

1.4 Scope of Work

Monitoring distance of surrounding obstacles on head, hips, and differences in floor level. The system also able to detect presence of one person not a group of people according

to image captured from raspberry pi camera module. The person have to be looking at the camera and not too far away from the user. To narrow down scopes further, the system should perform indoors with no varying light condition.

1.5 Technical Background

LINUX SHELL SCRIPTING

Linux batch script is a shell script containing series of commands. Shell script allows programmer to program commands in chains and have the system execute them as script events. Similar to batch scripts, the difference is that shell script allows to use programming functions such as conditional statements, loop, and so forth. Shell script can interface directly with operating system where system's commands are accessible.

HAAR-LIKE FEATURE CASCADE – A FACIAL LEARNING ALGORITHM

Haar-feature-based cascade classifier is an image learning algorithm. The algorithm is looking for common features called Haar-like features. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle. White pixels represents bright part of a gray scale image whereas black pixels is the dark part of the image.

ULTRASONIC SENSOR

Ultrasonic sensors work on a principle similar to sonar which evaluate distance of a target by interpreting the echoes from ultrasonic sound waves. The sensor transmits an ultrasonic wave and produces an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.

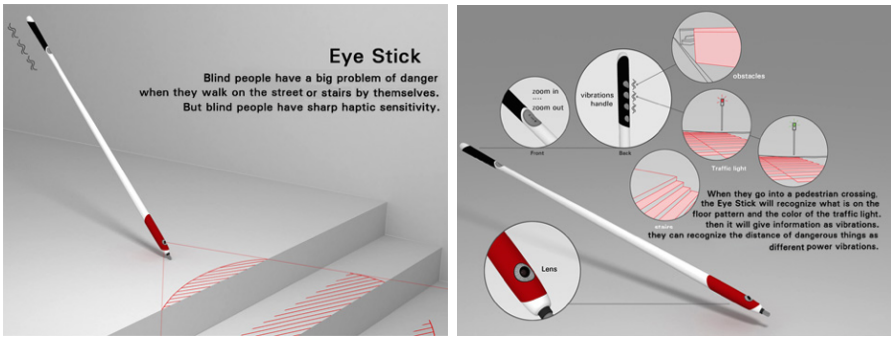
Chapter 2

Literature review and Overall Description

2.1 LITERATURE REVIEW

THE EYE STICK – WALKING STICK THAT SEES
Methodology - The Eye Stick is fitted with a sensor lens towards the bottom part, from where it picks up location bearings, like is the person nearing a staircase, or is he near the traffic lights. It then sends feedback to the blind commuter via sonic vibrations, communicating the scenario, so that the person can be aware of his surroundings and take his next step with confidence. Result - The blind person can recognize where they are through the Eye Stick. They can avoid dangerous things as well as locate their destination. The eye stick can also recognize traffic light signals, stair steps, and the subway station. Show in Figure 2.1a and 2.1b

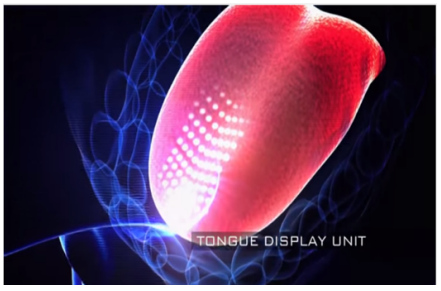
PLASTIC FANTASTIC BRAIN
Methodology – allow blind to see image from vibrations simulated from user's tongue. The equipment component are camera and tiny electric pinprict. When the blind wear a camera on their head. The camera will capture the object in front of them. When taking a tiny electric pinprict in user's mouth, it will draw a picture of an object directly on the surface of user's tongue. This information is represented in a form of a picture. Which is later translated by visual cortex of user's brain.
Result – Blind people use their tongue to locate an object and identify its shape. Show in figure 2.2a , 2.2b and 2.2c



(a) (b)

Figure 2.1: (a) and (b) The eye stick - Walking stick that sees

(a)



(b)

(c)

Figure 2.2: (a) , (b) and (c) Plastic Fantastic Brain

2.2 Overall Description

Obstacle Avoidance System is an embedded-system intended to assist a person with severe vision disabilities. Working as a cane complementary, the system is able to detect distance further than a cane can reach The system uses a proximity sensor attached at the belt. The system detects the closest object within its proximity and sends it to

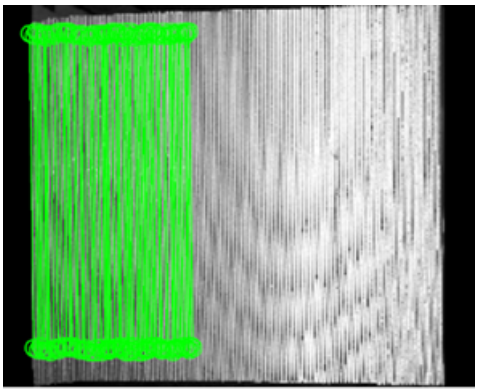


Figure 2.3: Edge detection

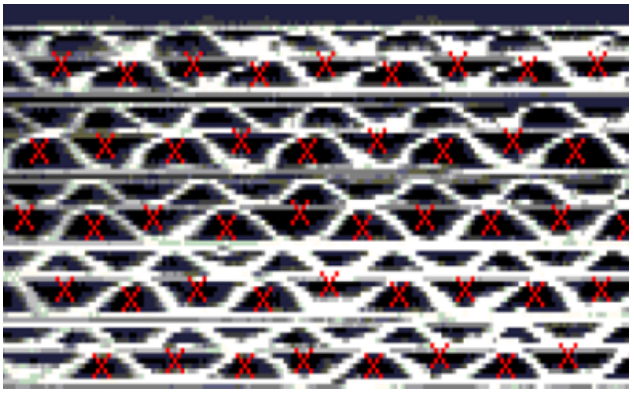


Figure 2.4: Center marking obtain from blob analysis operation

vibration sensor signaling an incoming obstacles. Moreover, the system can also detect possibility of human presence. It help ensures user that an object they are currently looking may likely be human. They can act accordingly

Method of Counting Thin Steel Plates Based on Digital Image Processing by Jingting Li and his team.[1] This project applied the method that offers a way to count piled thin steel plate for car chassis based on digital image processing and software designed for it. First, the image of the piled steel plates was enhanced quality. Then acquiring the region of interest (ROI) and detecting the lines with edge detection, horizontally shear the image against the tilting angle to adjust to vertical. Finally, count the number of plates in a Figure 2.3 by counting peaks that count with green lines. The method requires no large mechanical equipment and is efficient and accuracy.

Machine Vision System for Counting the Number of Corrugated Cardboard by Chatchai Suppitaksakul and his team [2]. This project presented a method for counting the number of corrugated cardboard that used machine vision system and image processing techniques. The cardboard edge images that were strip or slitter and fluted

or cut-off side were used for detection and counting. The gray level of edge images which obtained from the line scan camera were converted to binary images using threshoding. After that the strip lines of the slitter side were extracted by the first-order derivative and morphology then processed with mode for counting. For the cut-off side, the fluted areas of cardboard were identified and measured using Blob analysis. Then the centers of the detected area were marked with red 'x' and counted as the number of cardboards as shown in a Figure 2.4. MATLAB was applied to simulate in the experiments for determining the algorithms and performed the real-time experiments using HALCON to see the accuracy and performance of the system. The cardboard type BC, C and B were used in the tests. As experimental results, it is shown that the system provided the correct counting with tolerance of 2 cm and 3 cm in the case of the cardboard placed away further the camera focus for the slitter and cut-off side respectively.

These researches are useful for studying and understanding some techniques that can be applied to our project. Some techniques of aforementioned research are used to implement and solve in our project. Such as center making or centroid, edge detection, etc.

Chapter 3

Background Knowledge

3.1 Histogram Equalization of Gray-scale.

Histogram is a graphical representation of the intensity distribution of an image. It quantifies the number of pixels for each intensity value considered. Histogram equalization is a method that improves the contrast in an image, in order to stretch out the intensity range. To make it clearer, from Figure 3.1, the pixels seem clustered around the middle of the available range of intensities.What Histogram Equalization does is to stretch out this range. Take a look at the Figure 3.2: The green circles indicate the underpopulated intensities. After applying the equalization, the result is shown in a histogram in the Figure 3.2 in the center. The resulting image is shown in the Figure 3.2 at the right. Equalization implies mapping one distribution (the given histogram) to another distribution (a wider and more uniform distribution of intensity values) so the intensity values are spreaded over the whole range.

To accomplish the equalization effect, the remapping should be the *cumulative distribution function (cdf)*. For the histrogram H(i), its *cumulative distribution H'(i) is:*

$$H'(i) = \sum_{0<=j<i} H(j) \tag{3.1}$$

Where i and j refer to an intensity value in horizontal x-axis.To use this as a remapping function, we have to normalize H'(i) such that the maximum value is 255 (or the maximum value for the intensity of the image). From the Figure 3.2, the cumulative function is shown in Figure 3.3. Finally, using a simple remapping procedure to obtain the intensity values of the equalized image:

$$equalized(x, y) = H'(src(x, y)) \tag{3.2}$$

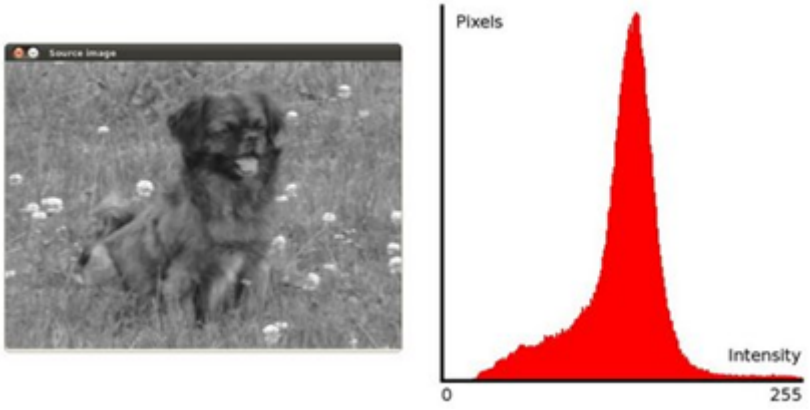


Figure 3.1: Original image and histogram

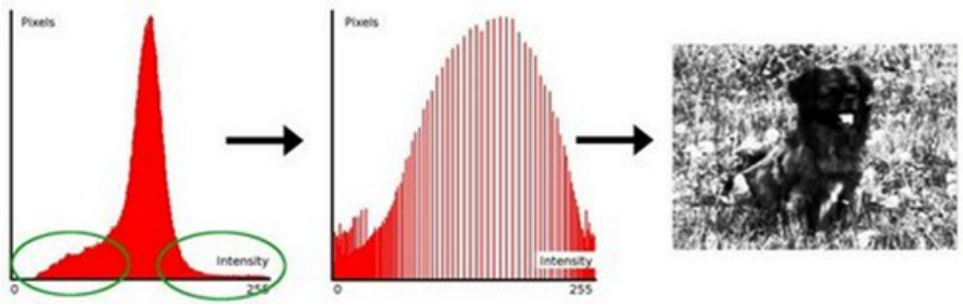


Figure 3.2: Gray-scale image equalization and histogram

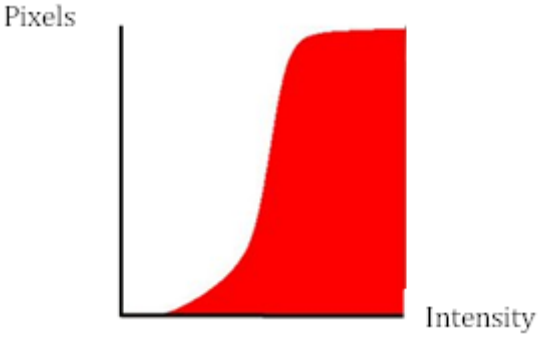


Figure 3.3: Maximum values for the intensity of the image

Where *equalized(x,y)* is a function which performs equalization and *src(x,y)* is a source image, x and y represent intensity values and pixels in sequence.

3.2 Otsu Thresholding

The thresholding function is typically used to get a binary image out of a gray-scale image or for removing a noise. There are several types of thresholding some of them are shown as in Figure 3.4.

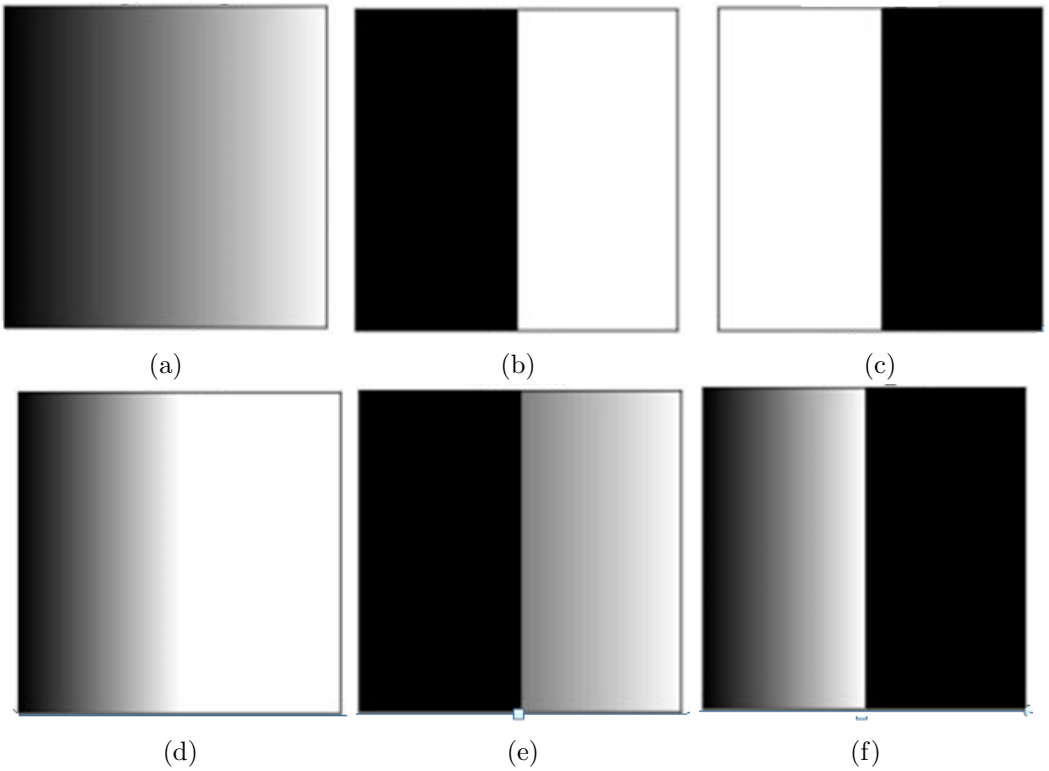


Figure 3.4: (a) An original image (b) Binary image (c) Binary inverted image (d) Threshold truncated image (e) Threshold to zero image (f) Threshold to zero inverted image

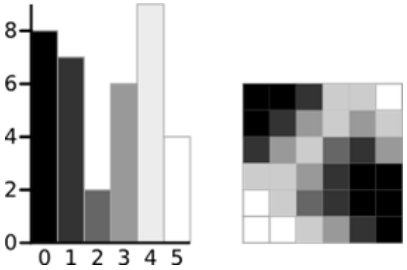


Figure 3.5: Histogram

Normally, threshold value is set manually. However, there is an algorithm that can compute the threshold value automatically, such as Otsu's algorithm.

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either falls in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

The algorithm will be demonstrated using the simple 6x6 pixels shown in Figure 3.5. The histogram for this image is shown on its right. To simplify the explanation, only 6 grey-scale levels are used.

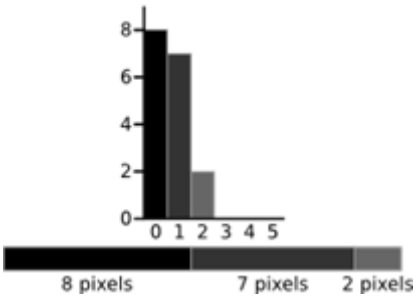


Figure 3.6: Background

The calculations for finding the foreground and background variances for a single threshold as it set to 3.

Finding variances of background, it start from finding means value of background. The mean (μ_b) is an average of multiplications between intensity values and its number pixels. Consider Figure 3.6, horizontal x-axis represents intensity values and vertical y-axis represents number pixels. It shows intensity values which less than the threshold value. After the mean is calculated, the variance will be calculated. It is an average of multiplications between square of difference mean and the number pixels. Where the difference mean are difference between an intensity values, relate with the number pixels, and the mean value.

$$W_b = \frac{8+7+2}{36} = 0.4722 \quad (3.3)$$

$$\mu_b = \frac{(0*8)+(1*7)+(2*2)}{17} = 0.6471 \quad (3.4)$$

$$\sigma_b^2 = \frac{((0-0.6471)^2*8)*((1-0.6471)^2*7)+((2-0.6471)^2*2)}{17} = 0.4637 \quad (3.5)$$

Finding variances of foreground, it start from finding means value of foreground. The mean (μ_f) is an average of multiplications between intensity values and its number pixels. Consider Figure 3.7 horizontal x-axis represents intensity values and vertical y-axis represents number pixels. It shows intensity values which more than the threshold value. After the mean is calculated, the variance will be calculated. It is an average of multiplications between square of difference mean and the number pixels. Where the difference mean are difference between an intensity values, relate with the number pixels, and the mean value.

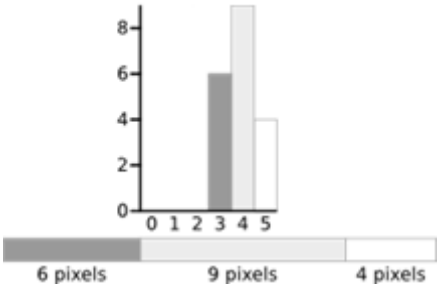


Figure 3.7: Foreground

$$W_f = \frac{6+9+4}{36} = 0.5278 \quad (3.6)$$

$$\mu_f = \frac{(3*6)+(4*9)+(5*4)}{19} = 3.8947 \quad (3.7)$$

$$\sigma_f^2 = \frac{((3-3.8947)^2*6)*((4-3.8947)^2*9)+((5-3.8947)^2*4)}{19} = 0.5152 \quad (3.8)$$

The next step is to calculate the Within-Class Variance (σ_w^2). This is simply the sum of the two variances, i.e. background variance (3.5) , foreground variance (3.8), multiplied by their associated weights. Note that, the weights are average of intensity, i.e. background weight equation (3.3), foreground weight equation (3.6).

$$\sigma_w^2 = W_b\sigma_b^2 + W_f\sigma_f^2 \quad (3.9)$$

$$= 0.4722 * 0.4637 + 0.5278 * 0.5152$$

$$= 0.4909 \quad (3.10)$$

This final value is the sum of weighted variances, equation (3.10), for the threshold value 3. This same calculation needs to be performed for all the possible threshold values 0 to 5. Figure 3.8 shows the results for these calculations. The highlighted column shows the values for the threshold calculated at equation (3.10). It can be seen that for the threshold equal to 3, as well as being used for the example, also has the lowest sum of weighted variances. Therefore, this is the final selected threshold. All pixels with a level less than 3 are background, all those with a level equal to or greater than 3 are foreground. The result of Otsu's method with threshold equals to 3 is shown in Figure 3.9. It can be seen that this method shows a good result in separation of foreground and background.

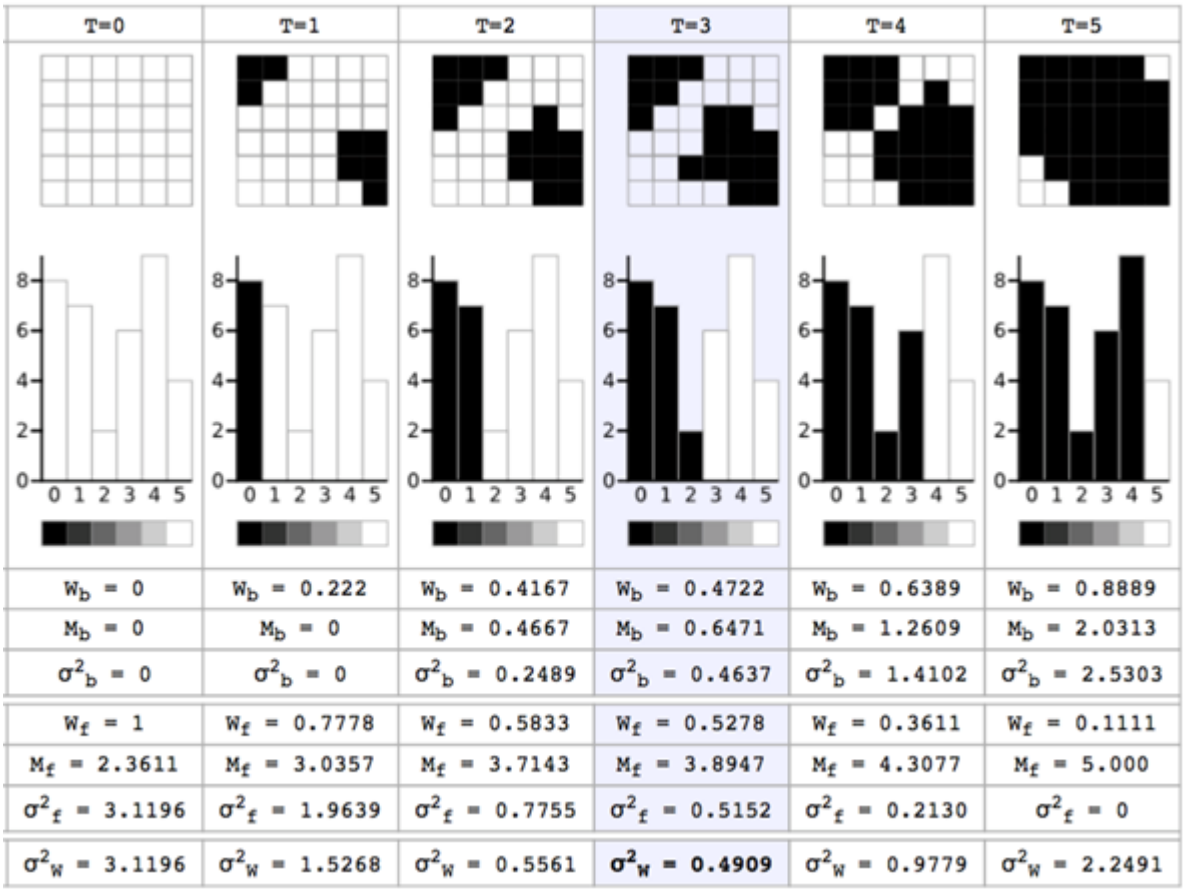


Figure 3.8: The results

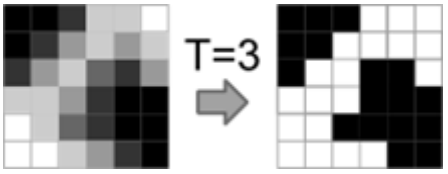


Figure 3.9: The threshold value

3.3 RGB to Grey-Scale Image Conversion

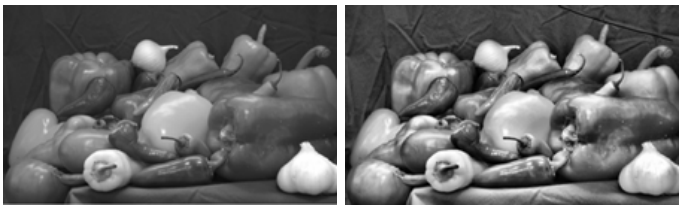
Normally, an input image is in RGB format. However, in order to perform image processing techniques, the RGB image is required to be converted into a gray-scale image. There are several methods to perform this conversion. In MATLAB the following conversion is used by forming a weighted sum of the R, G, and B components:

$$0.2989 * R + 0.5870 * G + 0.1140 * B \tag{3.11}$$

Consider an RGB image is shown in Figure 3.10a. When apply the previous formula to convert it to the gray-scale image, the result is shown in Figure 3.10b.

(a) (b)

Figure 3.10: (a) An original image (b) RGB to gray-scale image conversion result



(a) (b)

Figure 3.11: (a) An original image (b) RGB to gray-scale image conversion result

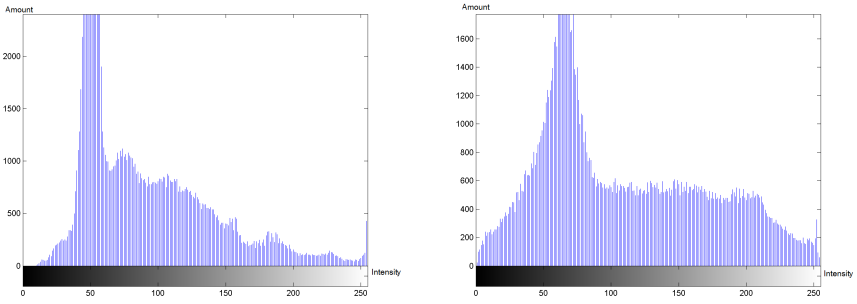
3.4 Adaptive Histogram Equalization

This technique is used to improve contrast in images. It differs from ordinary histogram equalization in the respect that the adaptive method computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image. It is therefore suitable for improving the local contrast of an image and bringing out more detail. Consider Figure 3.11 illustrate the differentiation of gray-scale image. An input image is shown in Figure 3.11a. Figure 3.11b, enhanced the contrast of the gray-scale image by transforming the values using contrast-limited adaptive histogram.

More differentiation detail of enhancing the contrast of the gray-scale image is shown in Figure 3.12. It illustrate comparing with the histogram differentiation between the gray-scale image and the enhanced contrast of gray-scale image. Figure 3.12a represents to the gray-scale image before uses the contrast-limited adaptive histogram. Figure 3.12b represents to the gray-scale image after uses the contrast-limited adaptive histogram.

3.5 Binary Image Conversion

This technique uses to convert gray-scale image to binary image. The binary image is a digital image which has only two possible values, i.e. a 1 or 0 for each pixel. Consider Figure 3.13a, an input image will be converted to binary image. It has intensity values



(a) (b)

Figure 3.12: (a) Histogram of gray-scale image (b) Histogram of enhancing the contrast of the gray-scale image

(a) (b)

Figure 3.13: (a) Enhancing the contrast of the gray-scale image (b) Converting the binary image

in rang [0, 255]. Next step, a threshold value is assigned in range [0, 1] which is decimal number. The output image then replaces all pixels with luminance greater than threshold value with the value 1, white color, and replaces all other pixels with the value 0, black color,. Specify level in the range [0, 1]. The converted image is shown in Figure 3.13b.

3.6 Edge Detection

Edge detection method is a method to find edges in the gray-scale image by taking a gray-scale or a binary image as its input, and returns a binary image of the same size as one, with 1's where the function finds edges in gray-scale and 0's elsewhere. It is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. Note that, edge is a set of connected curves which indicate the boundaries of objects. Typically, there is corner detection algorithms follow:

- **Sobel Method:** This method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of gray-scale image is maximum. The gradient is computed with 3x3 kernels in horizontal $G(y)$ and

vertical $G(x)$.

$$G(y) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} G(x) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- **Prewitt Method:** This method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of gray-scale image is maximum. The gradient is computed with 3x3 kernels in horizontal $G(y)$ and vertical $G(x)$.

$$G(y) = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} G(x) = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Roberts Method:** This method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of gray-scale image is maximum.The gradient is computed with 2x2 kernels in horizontal $G(y)$ and vertical $G(x)$.

$$G(y) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} G(x) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- **Laplacian Method:** This is Gaussian method finds edges by looking for zero crossings after filtering gray-scale image with a Laplacian of Gaussian filter.

- **Zero-cross Method:** This method finds edges by looking for zero crossings after filtering gray-scale image with a filter you specify.

- **Canny Method:** This method finds edges by looking for local maxima of the gradient of gray-scale image. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

For example, Figure 3.14a is an input gray-scale image. After using methods of edge detection algorithms, i.e. sobel, prewitt, roberts, laplacian of Gaussian, zero-cross, and canny. The output images are shown in Figure 3.14b - 3.14g respectively.

All the basic knowledge presented in this chapter will be applied to the work of sheet of cloths counting in the later chapter.

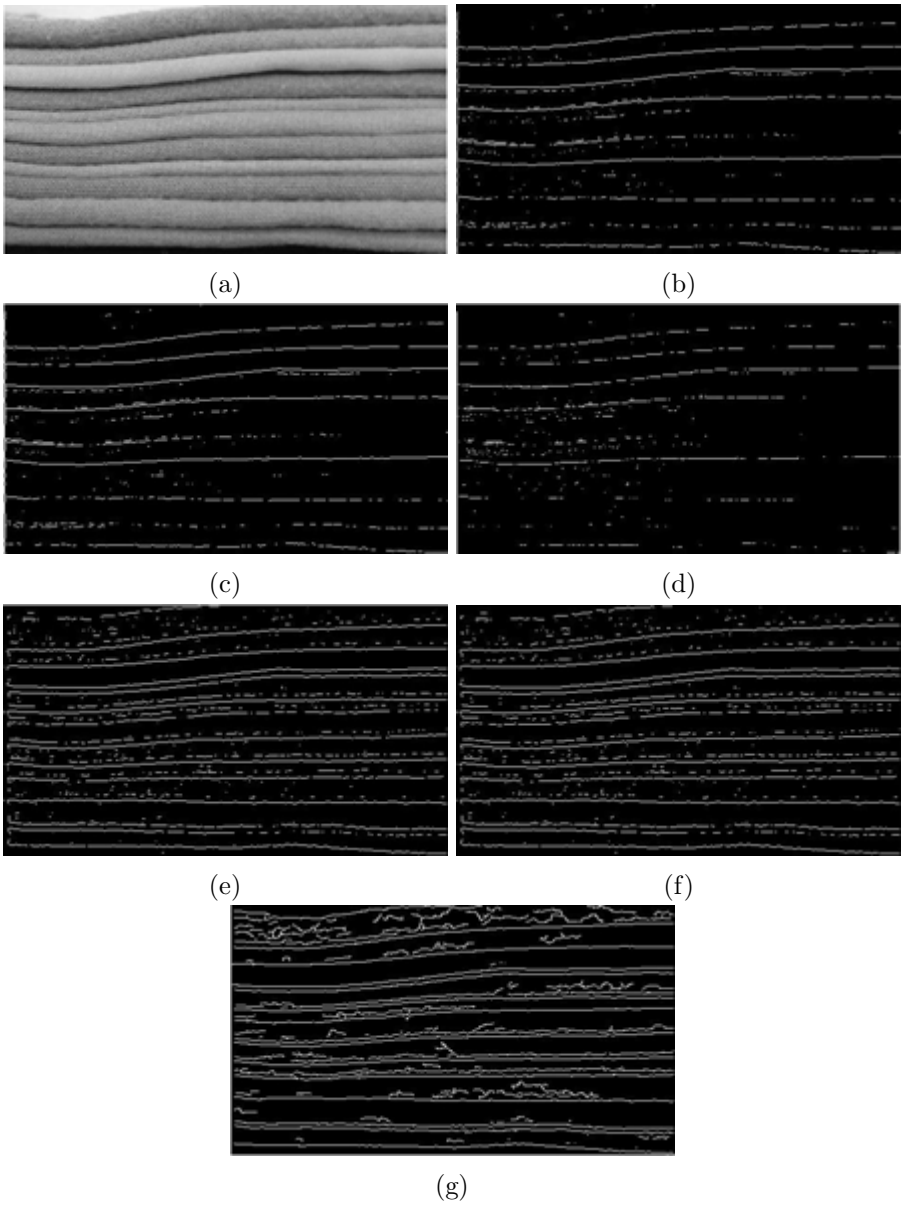


Figure 3.14: (a) Gray-scale image (b) Sobel image(c) Prewitt image (d) Roberts image(e) Laplacian of Gaussian image (f) Zero-cross image (g) Canny image

Chapter 4

Requirements and Analysis

4.1 Requirements

Application requirements describe a particular design of this project. The requirements include functional requirements and non-functional requirements.

4.1.1 Functional Requirements

Functional requirements consist of user requirement and system requirement. This is the information that explains the all the functions performed by user and system. For this application the user and the system can perform the following functions:
User:
- Can use images from gallery to counting.
- Can use images taken by iPhone camera to counting.
- Can see the result of number sheets of cloth.

System:
- Can count sheets of cloth that it is placed overlay with acceptable accuracy.

4.1.2 Non-Functional Requirements

Non-functional requirements include user requirement and system requirement. This is the requirements that emphasize on the scope of work of the application. For this application the user and the system can perform the following functions:
- The system can count sheets of cloth faster than manual counting.
- User interface of application is simple and easy to use.

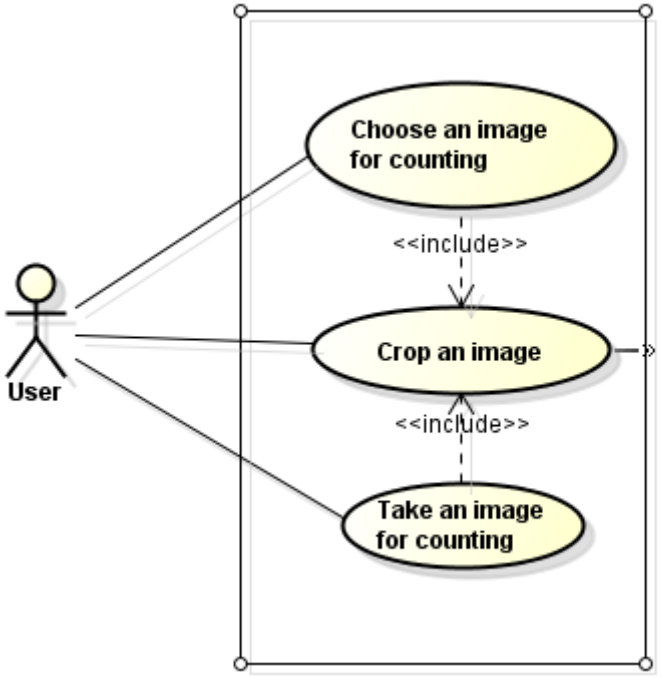


Figure 4.1: System's Use Case Diagram

- The system can be developed further features.

4.2 Use Case

A use case is a list of steps defining in Unified Modeling Language (UML). It consists of use case diagram and use case description.

4.2.1 Use Case Diagram

Use case diagram is a representation of a user's interaction with the system and depicting the specifications of a use case. It can portray the different types of users of a system and the various ways that they interact with the system as shown in Figure 4.1.

Users can interact with the application in three different tasks. First, user can choose an image from gallery to counting. Second, users can take an image from camera application to counting. Both tasks are input of application. Final task user can crop an image with particular area in an image to receiving the acceptable result. Note that, the users have to start from the application's input tasks.

4.2.2 Use Case Description

Use case description describes user and system both are interaction in details. The details of all tasks are explained in tables 4.1 - 4.3.

Table 4.1 describes about the task of choosing an image. This task allows the user to choose an image from the image gallery. The step of the tasks can be shown as in the following. First, the user presses a gallery button. The system responds by showing the screen of image gallery. Then the user chooses an image. The image will be shown on the screen. In case of alternative flow, user dose not choose the image from gallery, the system will display a message. Then user gets an image from photography.

Table 4.2 describes about the task of taking an image. This task allows the user to photograph the piled of sheets of cloth. The step of the tasks can be shown as in the following. First, the user presses the camera button. The iPhone's camera is opened. Then the user takes an image. The image is shown on the iPhone's screen. In case of alternative flow, user does not photograph, the system will display a message. Then the user chooses an image from gallery.

Table 4.3 describes about the task of cropping an image. This task allows the user to crop an image that is shown on iPhone's screen. The step of the tasks can be shown as in the following. First, user crops the desired area of the image. Then presses use image button. The cropped image and the message of counting result are shown on the screen of iPhone.

4.3 Activity Diagram

Activity diagram are a graphical that represents a work-flow of stepwise activities and actions with support for concurrency.

Figure 4.2 describes flows of the application. Starting from a black circle at the top of the diagram, a user can select one of the two possible ways in order to obtain an input image: a) get an image from the gallery or b) get a new image from photographing. Then the user has to crop the image. The result will be an image with only the stack of cloths without background. Note that before the step of submitting the image, the user can cancel the task. This action will take the user back to the previous step.

This chapter presents all of requirements, use case diagram and activity diagram. It will be applied for software design in next chapter.

Use case	Choose an image from gallery for counting	
Actor	User	
Pre-condition	The image stored in image galley on the iPhone.	
Post-condition	The image is shown on the screen of the iPhone.	
Flow of event	Actor input 1) User presses the gallery button. 3) User chooses an image from the gallery.	System response 2) The iPhone's screen shows the image gallery. 4) The image is shown on the screen of iPhone.
Alternative flow	1) User does not press the gallery button. 3) User gets an image from photography.	 2) System display message "Please choose an image"

Table 4.1: Use case diagram of choosing an image

Use case	Take an image for counting	
Actor	User	
Pre-condition	The image does not store in image gallery on the iPhone.	
Post-condition	The image is shown on the screen of the iPhone.	
Flow of event	Actor input 1) User presses the camera button. 3) User photographs an image.	System response 2) The iPhone's camera is opened. 4) The image is shown on the screen of iPhone.
Alternative flow	1) User does not press the camera button. 3) User chooses an image from gallery.	 2) System display message "Please choose an image"

Table 4.2: Use case diagram of taking an image

Use case	Crop an image	
Actor	User	
Pre-condition	The image does not store in image gallery on the iPhone.	
Post-condition	The image is shown on the screen of the iPhone.	
Flow of event	Actor input 1) User crops the desired area on image. 2) User presses use button.	System response 3) The system cropped and shows the cropped image. 4) The message of counting result is shown on the screen of iPhone.

Table 4.3: Use case diagram of cropping an image

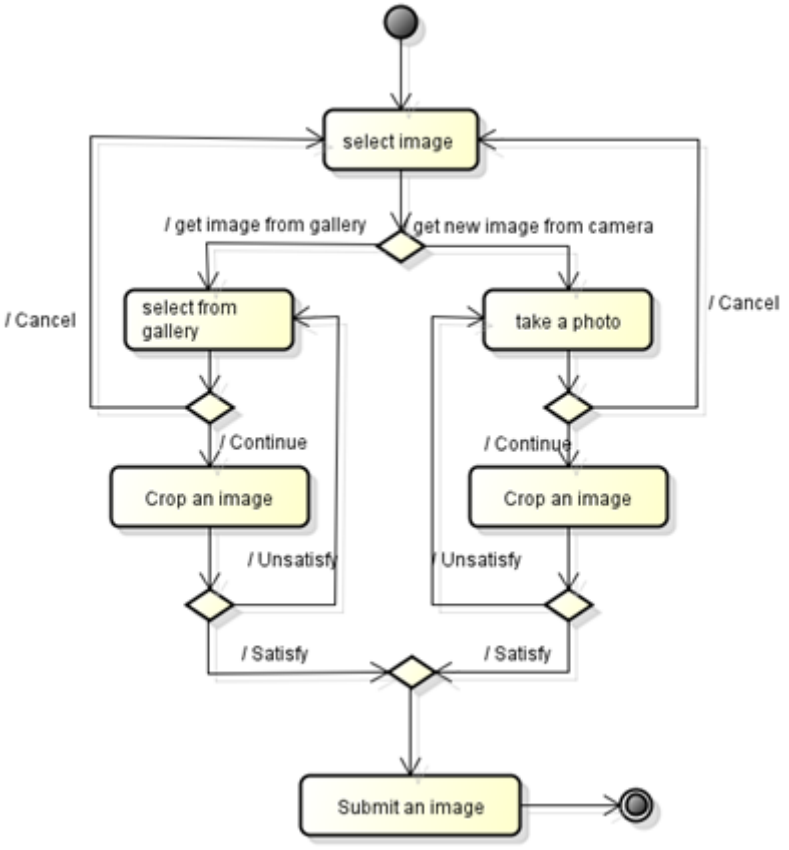


Figure 4.2: Activity diagram of application

Chapter 5

Software Design

5.1 Software Architecture

Software architecture serves as a blueprint for both system and project developing it, defining work assignments that must be carried out by design and implementation teams. The architecture is presented in terms of classes diagram.

5.1.1 Overall Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. All of system's structure is shown in Figure 5.1

The structure of an application consist three packages including *Framework package*, *ImageUpdate package*, and *Counting package*. The framework package functions like an interface class. The package gets an input image and shows final result. An input image will be sent into the ImageUpdate package. The ImageUpdate package is package that preprocessing an image such as image rotation, image conversion, etc. The last package is Counting package. It counts the number of sheets of cloth.

5.1.2 Framework Package

Framework is a package served as user interface part of the application. It gets an image input from user and displays a counting result. The input image can be obtained from image gallery or photographing. The counting result is returned from *MattoArray* class in Counting package. This package has four classes including *Control class*, *Gallery class*,

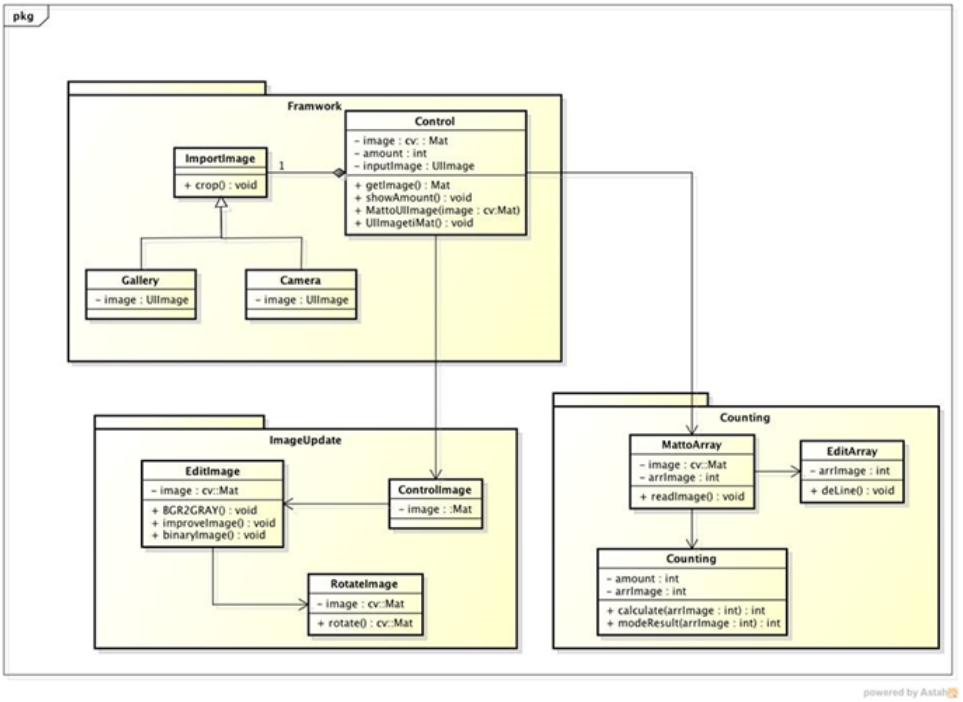


Figure 5.1: Overall class diagram of the application

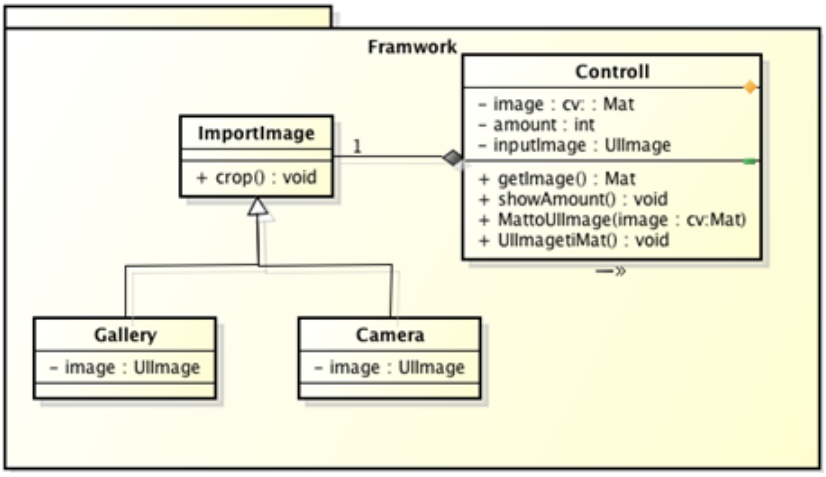


Figure 5.2: Framework package

Camera class and *ImportImage class*. Control class is a main of application. Gallery class is an input image class from the image gallery and Camera class is an input image class from photography both inherit from ImportImage class. The Framework package is shown in Figure 5.2.

5.1.3 ImageUpdate Package

ImageUpdate is a package that pre-processes an image before sending to Counting package. It contains three classes, i.e. *ControllImage class* gets and returns an image to

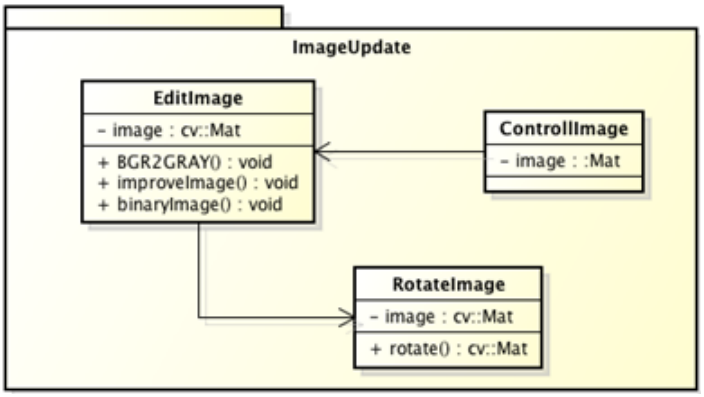


Figure 5.3: ImageUpdate package

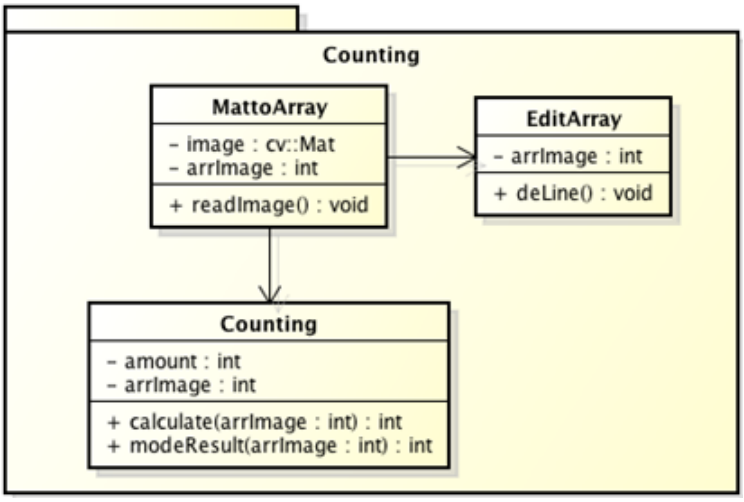


Figure 5.4: Counting package

Framework package, *EditImage class* converts an original image to binary image with OpenCV library and *RotateImage class* checks orientation of the sheets of cloth in the image and rotate the image such that the layers of the sheets of cloth align vertically for counting. The ImageUpdate package is shown in Figure 5.3.

5.1.4 Counting Package

There are three classes in the Counting package, i.e. MattoArray class which received an image and stored intensity value of each pixel to the array, *EditArray class*, which improved the number of intensity value in the array, and Counting class that count number of intensity value in the array count length of the series of 0 or 1. it will be discussed later chapter. The Counting package is shown in Figure 5.4.

All of these presents about software designing of the application thorough. Explanations, each functions in classes is performed.

Chapter 6

Developments

6.1 Development Tools

Development tools used to create an application of counting sheets of cloth are shown in Figure 6.1 The tools include MATLAB, Xcode, and OpenCV library. The proposed algorithms are first developed under MATLAB and then implemented in the iPhone using Xcode and OpenCV.

6.1.1 MATLAB

MATLAB is a tool used to develop and test algorithms. There is an image processing toolbox provided in MATLAB which can be used to achieve the tasks.

6.1.2 Xcode

Xcode is a tool used for developing application for OS X and iOS platform. This project uses Xcode to implement the algorithms developed by using MATLAB together with OpenCV library. The tool can then be used to create an application on iPhone.

6.1.3 OpenCV

OpenCV library is image processing in C++ language library. It is function that similar image processing in MATLAB program. In this project use OpenCV library to apply the algorithm that provide by MATLAB program.

(a) (b) (c)

Figure 6.1: (a) MATLAB (b) Xcode (c) OpenCV

6.2 Project Implementation

The implementation of this project is separated into two major parts, i.e. application part and algorithm part. Application part is implemented using Objective-C language on Xcode. The aim of this part is to develop a friendly user interface and image pre-processing. The application development part is defined to be the part concerning user interface development and iOS based device implementation. This part uses Xcode to create an application based on iOS platform with OpenCV library. The algorithms development part develops and test the algorithm using MATLAB as a tool. The developed algorithm is then transferred to be used in the application development part.

6.3 Application Development Part

Application development part develops user interface of the application for counting sheets of cloth. The user interface is developed by using Objective-C in Xcode. The user interface is designed as follows Figure 6.2. Consider Figure 6.2 at the left, the application name is displayed on top of frame. At center, it is an input image area which displays the input image from gallery or camera. There are two lower left buttons, i.e. gallery button and camera button. Both buttons refer to an input image buttons.

- **Gallery Button:** this button specifies the device's photo library.
- **Camera Button:** this button specifies the device's built-in camera.

After obtained an input image, user crops an input image in cropped area which is area fixed. But the input image can be resized as shown in Figure 6.2 at the middle. There are two lower left and right buttons, i.e. unused button and use button.

- **Unuse Button:** the input image will be retaken.

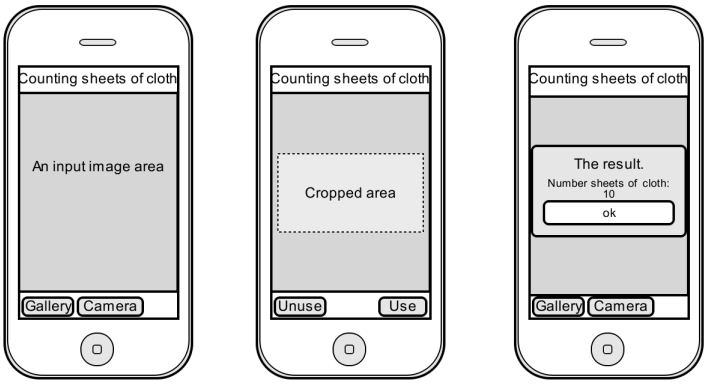


Figure 6.2: Designed Moqup

- **Use Button:** the input image will be cropped in cropped area.

After the input image is cropped, a system will alert a result message as shown in Figure 6.2 at the right.

In order to apply the application algorithm from MATLAB, Objective-C will be used OpenCV library to creates algorithm with image processing technique.

6.4 Algorithm Development Part

The algorithm development aims to develop algorithm that can count the sheets of cloth from an input image. The overview algorithms developed in this project is shown in Figure 6.3.

There are five steps to achieve the counting result: receive an input image (Original image box), preprocessing the image (Gray image box, Enhance quality of image box, Binary image box) and display the number of sheets of cloth (Counting box).

6.4.1 RGB to Gray-scale Image Conversion

This part of the algorithm converts a RGB image into a gray-scale image. The process starts by selecting an original input image from the image gallery or the one taken by camera. An example of the input image is shown in Figure 6.4a. The original input image is then converted into the gray-scale image as shown in Figure 6.4b.

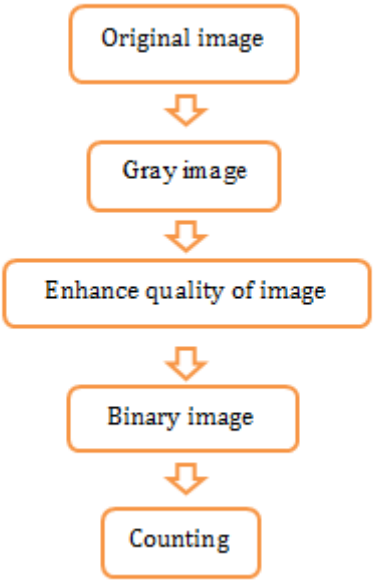
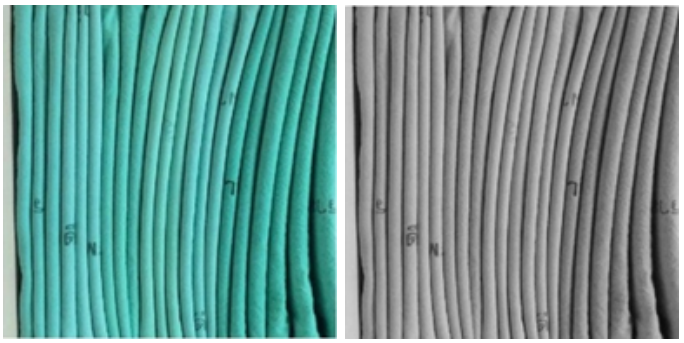


Figure 6.3: Flow of algorithm



(a) (b)

Figure 6.4: (a) An original image (b) Gray-scale image

6.4.2 Image Quality Enhancement

This step is used to improve contrast of an image Figure 6.6 illustrates the result after applying the enhancement algorithm. In MATLAB, the function used for contrast adjustment is called adpthisteg. The detail of this method is discussed previously in section 3.4. In application part, the contrast of the gray-scale image is improved by using the function from OpenCV library, i.e. histogram equalization (discuss in section 3.1).

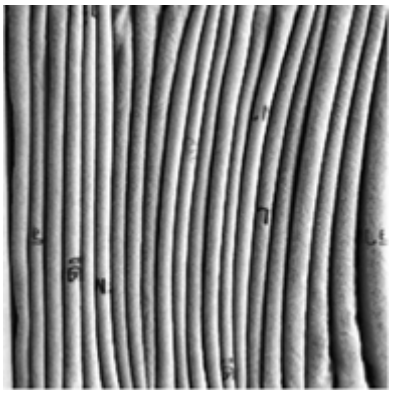


Figure 6.5: Result of image enhancement

Figure 6.6: Binary image

6.4.3 Binary Image Conversion

The gray-scales image that already enhanced the contrast in image is converted into binary image as shown in Figure 6.6. After conversion, the result binary image possesses two intensity values 1 or 0. In MATLAB, the conversion of gray-scales image to binary image used im2bw function. The im2bw function converts an RGB image to binary image (discus in section 3.5). In Objective-C, gray-scales image is converted to binary image using Otsu thresholding with functionally *cv::threshold()* (discus in section 3.2).

6.4.4 Counting

This section discusses about counting algorithm that counts the sheets of cloth from binary image. The algorithm divides the binary image into ten parts using ten horizontal straight lines Figure 6.9a. Each black straight line contains intensity values which are stored in an array. An example of intensity values contained in the array corresponding to each straight line is shown in Figure 6.7.

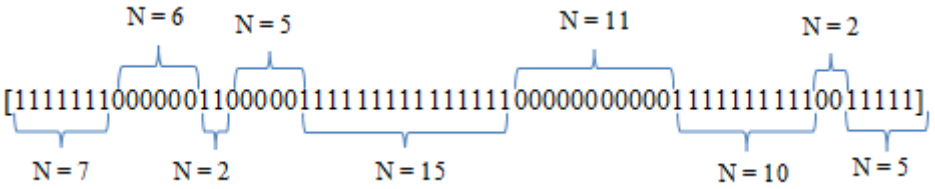
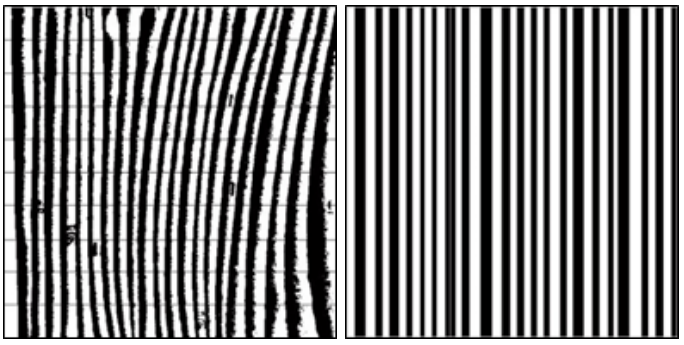


Figure 6.7: Intensity value in array

[1111111000000000000011111111111111000000000000111111111111111111]

Figure 6.8: Improved intensity value in array



(a) (b)

Figure 6.9: (a) Binary image with ten (b) New binary image resulted from the first horizontal straight line

Elements in array flip from one to zero and zero to one where the number of N less than five where N refers to the length of the series of 0 or 1. The intensity values are altered in two locations marked as underline in Figure 6.8. From the array of 0 and 1 shown in Figure 6.8, the result of counting sheets of cloth is three sheets. This counting result will be kept and used in the later processing.

By using this method each horizontal straight line produces series of black and white strips. Figure 6.9a illustrates the binary image with ten horizontal straight lines superimpose on it while Figure 6.9b shows the result after applying the algorithm to the first line.

This chapter presents all development tools and implementation application. After that an evaluation will be explained in next chapter.

Chapter 7

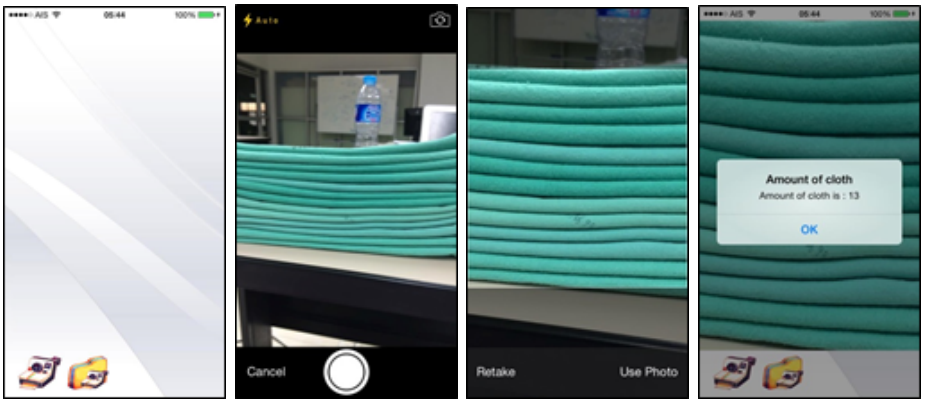
Evaluations and Discussions

7.1 Evaluations

The proposed system counts the sheets of cloth by using several image processing techniques. It starts by obtaining an input image from two possible methods, i.e. photographing a new image or choosing from image gallery in the device. It calculates the number of sheets of cloth in the image. The acceptable result has shown on the screen of iPhone device. The screen-shot of the application in case of photographing a new image is shown in Figure 7.1 and case of selection from image gallery is shown in Figure 7.2.

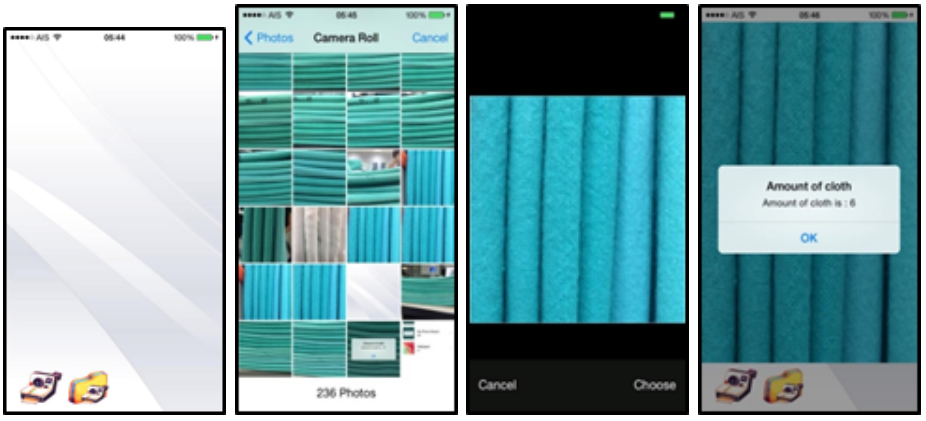
As shown in Figure 7.1, application calculates sheets of cloth from an image taken by device's built-in camera. The result is shown when user interacts with starting from left to right. Figure 7.1a illustrates the starting screen. There are two icons on the lower left of the screen. The camera icon on the left indicates that the input image is obtained by photographing. The corresponding screen-shot is shown in Figure 7.1b. After obtaining the image, the application asks the user to crop it. This is confirmed when the user presses the "Use Photo" button on the lower right of the screen. This is shown in Figure 7.1c. The application then starts the counting process. The result is reported on a pop-up window as shown in Figure 7.1d. The user can exit from this screen by pressing the "OK" button.

As shown in Figure 7.2, application calculates sheets of cloth from an image that is chosen from an image gallery in an iPhone device. The result is shown when user interacts with starting from left to right. Figure 7.2a illustrates the starting screen. There are two icons on the lower right of the screen. The gallery icon on the right indicates that the input image is obtained by choosing from the image gallery. The corresponding screen-shot is shown in Figure 7.2b. After obtaining the image, the application asks the user to



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

Figure 7.1: Screen-shot of sheets of cloth counting application (selecting image file from the gallery)



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

Figure 7.2: Screen-shot of sheets of cloth counting application (selecting image file from the gallery)

crop it. This is confirmed when the user presses the "Choose" button on the lower right of the screen. This is shown in Figure 7.2c. The application then starts the counting process. The result is reported on a pop-up window as shown in Figure 7.2d. The user can exit from this screen by pressing the "OK" button.

7.2 Discussions

The proposed application completes all requirements. And fulfill the aims of counting sheets of cloth with acceptable accuracy comparing to manual count. Thus the application reduces fatigue and time. This application spent a few time to counting sheets of cloth. The result of this project count sheets of cloth is acceptable.

Chapter 8

Results

8.1 Experimental Setting and Results

The application is tested for the following experimental setup. Since the proposed application aim to be used in the hospital, thus there are only specific sheets of cloth to be tested. Each experiment uses twenty tested image. The experiment setups are classified as shown in Figure 8.1.

Conditions testing of sheets of cloth are separated following color. There are green color, white color and both colors. In each color of sheets of cloth includes vertical axis and horizontal axis. In each vertical axis and horizontal axis includes four cases. The cases is shown in Table 8.1. After the conditions are specified, the application experimental will be begun.

Green color testing
test 1: number of sheets of cloth is fixed at ten in vertical axis.
test 2: number of sheets of cloth is fixed at ten in horizontal axis.
test 3: number of sheets of cloth is not fixed at ten in vertical axis.
test 4: number of sheets of cloth is not fixed at ten in horizontal axis.

White color testing
test 5: number of sheets of cloth is fixed at ten in vertical axis.
test 6: number of sheets of cloth is fixed at ten in horizontal axis.
test 7: number of sheets of cloth is not fixed at ten in vertical axis.
test 8: number of sheets of cloth is not fixed at ten in horizontal axis.

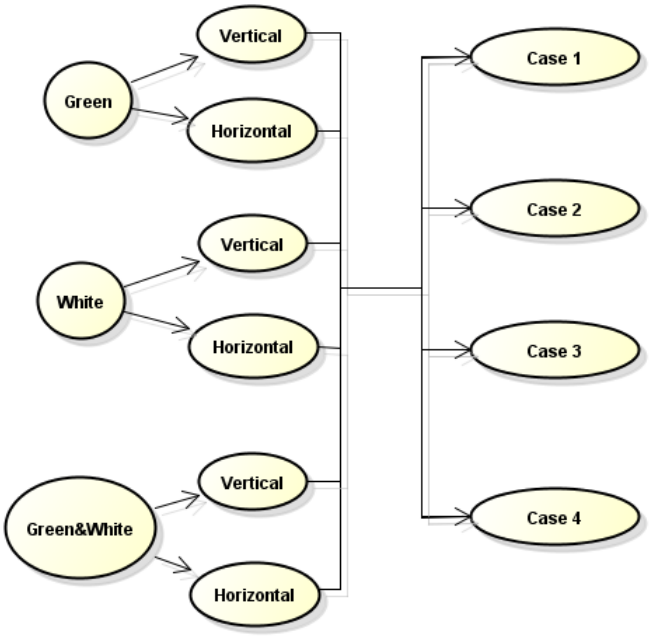


Figure 8.1: Conditions testing

Case	Direction	Top-Boarder	Lower-Boarder	Direction	Left-Boarder	Right- Boarder
1	Horizontal	Cloth	Cloth	Vertical	Cloth	Cloth
2	Horizontal	Cloth	Not Cloth	Vertical	Cloth	Not Cloth
3	Horizontal	Not Cloth	Cloth	Vertical	Not Cloth	Cloth
4	Horizontal	Not Cloth	Not Cloth	Vertical	Not Cloth	Not Cloth

Table 8.1: Cases testing

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
10	10	8	8	9
10	8	9	8	9
10	8	8	10	9
10	9	9	8	8
10	9	9	9	9
total(%)	88	86	86	88

Table 8.2: Test 1

Both color testing

test 9: number of sheets of cloth is fixed at ten in horizontal axis.

test 10: number of sheets of cloth is not fixed at ten in horizontal axis.

From the Experimental, the best case of green sheets of cloth is case number one. It means that the best result will be obtained when users take an image in horizontal axis sheets of clothas shown in Table 8.2 - 8.5. the best case of white sheets of cloth is also

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
10	10	9	7	10
10	10	9	9	10
10	9	9	7	9
10	10	9	6	9
10	9	9	7	8
total(%)	96	90	72	92

Table 8.3: Test 2

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
5	5	5	5	5
10	10	10	9	9
15	13	12	14	12
20	19	16	17	18
25	23	21	22	20
total(%)	93.33	85.33	89.33	85.33

Table 8.4: Test 3

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
5	5	5	5	5
10	10	10	11	7
15	14	12	14	10
20	17	15	12	14
25	22	15	15	17
total(%)	90.67	76	76	70.67

Table 8.5: Test 4

case number one as shown in Table 8.6 - 8.9. So sheets of cloth in horizontal axis is the best way to obtain the best result. From above summarize, both color testing will be tested in only horizontal axis. So the result is shown in Table 8.10 - 8.11.

Conclusion, a good result will be obtained when color sheets of cloth was same color and directed in horizontal axis. An accurate of case number one more than 80 percent.

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
10	8	7	7	8
10	8	7	7	8
10	9	5	8	6
10	8	8	8	7
10	8	9	7	9
total(%)	82	72	74	76

Table 8.6: Test 5

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
10	10	10	7	10
10	10	10	9	10
10	10	9	7	9
10	9	9	6	9
10	9	9	7	8
total(%)	96	94	72	92

Table 8.7: Test 6

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
8	6	7	5	7
11	9	9	9	9
12	11	11	11	12
15	11	11	12	12
20	13	13	13	12
total(%)	75.75	77.72	75.75	78.78

Table 8.8: Test 7

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
8	7	8	7	8
12	10	10	10	10
15	11	12	9	13
16	13	12	10	11
20	17	16	16	9
total(%)	87.87	80.30	78.78	77.27

Table 8.9: Test 8

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
10	2	1	1	1
10	2	2	9	5
10	4	5	7	5
10	4	6	5	6
10	3	4	4	4
total(%)	30	36	52	42

Table 8.10: Test 9

	Case 1	Case 2	Case 3	Case 4
Actual Number	Counting Number	Counting Number	Counting Number	Counting Number
7	2	2	3	3
9	2	2	6	4
10	4	4	4	4
10	4	2	2	2
11	6	6	6	6
total(%)	38.30	34.04	44.68	40.43

Table 8.11: Test 10

Chapter 9

Conclusion

9.1 Summary

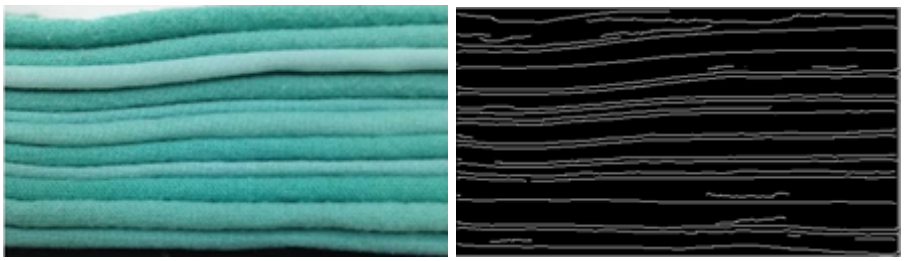
In summary, this project aims to develop an application that can count sheets of cloth with the acceptable accuracy result. It is expected to reduce hospital staff's fatigue in manual sheets counting. An input image should have enough contrast and there should be no pattern on the cloth. Furthermore, the sheets of cloth should be of the same color. Each layer the width of sheets of cloth should cover more than ten pixels.

9.2 Lessons learned

Throughout the semester, two of the most important lessons we all have learned are teamwork and communication skills. When we are teamwork the good communication skill is important.

Beside what is previous mention, we have learned lesson as follows:

- MATLAB programming
- iOS application development
- Image processing techniques
- Machine Learning (for data cluttering)



(a) (b)

Figure 9.1: (a) An original image (b) Edge detection

9.3 Problems and obstacles

During this project, the team encountered several problems. These problems can be separated into two parts, application part and algorithm part.

9.3.1 Problems Concerning Application Part

The problems concerning the application part are merely programing problems. Most of the problems found are about syntax error. The team spent lots of time to solve such problems, for example, the program cannot convert image to *cvMat* type, the program cannot pass value from OpenCV class to Objective-C, etc.

9.3.2 Problems Concerning Algorithm Part

There are two algorithm developed in this project. The first algorithm is already implemented into an iOS device. However, the second algorithm is still in process and possesses several problems. The second algorithm contains machine learning to classify data by using K-means. The proposed of this algorithm is to classify the area which is shadow of cloth and areas that are not for increasing counting accuracy. The team applied two features of image to classify the image, i.e. an average of intensity value in each area and size of area. The area's intensity and size can be obtained as shown in Figure 9.1

Starting from an original input image is shown in Figure 9.1a. The edge detection is applied and the result image is shown in Figure 9.1b. In order to get feature of area and intensity the image is rotated 90 degree clockwise and the straight lines are drawn. The areas composed of the overlay straight lines and the edge produced edge detection can be obtained. The center of each area are found and marked with "o". This is shown in Figure 9.2.

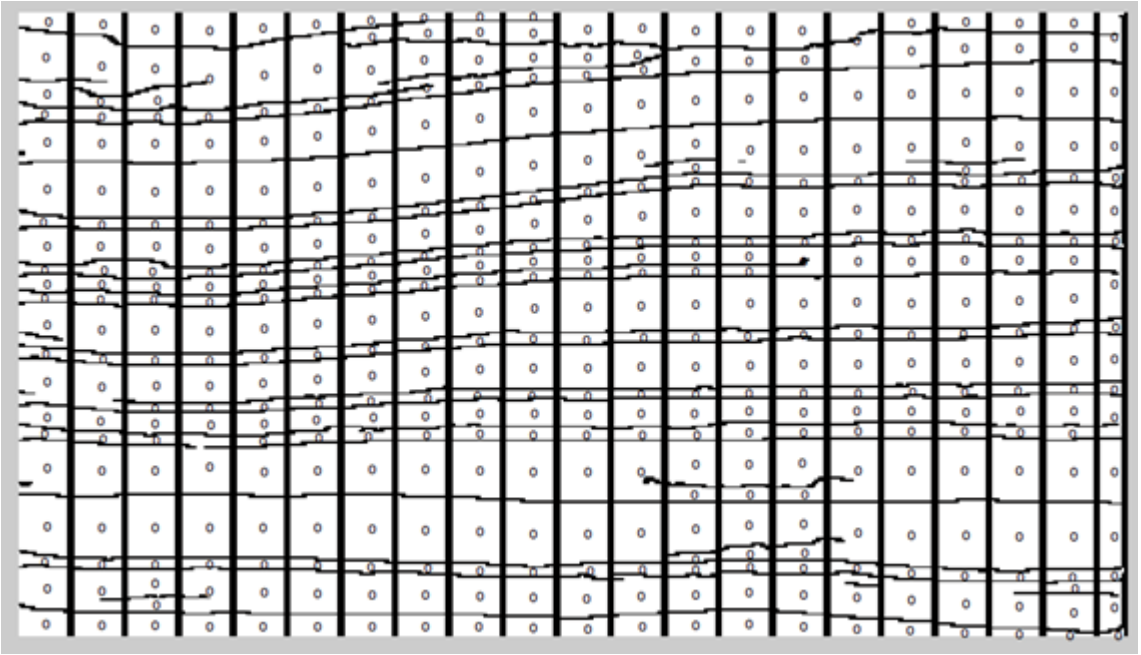


Figure 9.2: Area for calculating image features

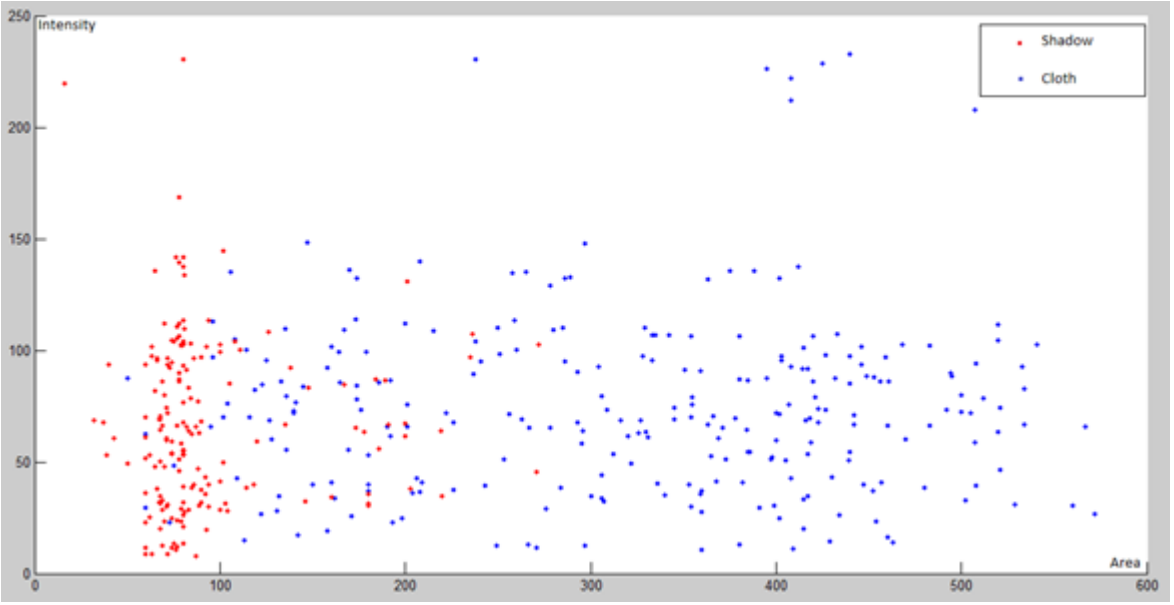


Figure 9.3: Features extraction of an input image

Figure 9.3 plots the features obtained from each area. The horizontal axis represents the size of the area while the vertical axis represented the average intensity of each area. It is found that small size areas tens to be that area of cloth. However, the intensity of shadow and cloth area is not very well distinguishable.

Thus, the result suggests that the classification using these two features and K-mean techniques cannot accomplish the classification task. This is because the features of shadow and sheets of cloth are not explicitly distinguishable. This means that the chosen

features may be not proper for the task and the better features should be applied. By applying the good features, it should separate the data into different groups more clearly.

In summary, the feature of area intensity is not proper to use for classification in this project.

9.4 Future work

Nowadays, application counting sheets of cloth calculates number sheets of cloth with accuracy less than 80 percent because the application uses basic algorithm to calculate. So, first of all the algorithm should be improved for increasing accuracy. Addition, finding better features can classify data such as different color space of input image.

Bibliography

[1] Jingting L.,Ying W.,Qiang Z., and Wei C, *Method of Counting Thin Steel Plates Based on Digital Image Processing*. School of Electronics Information, Wuhan University, 2011.

[2] Chatchai S.and Meena R., *Machine Vision System for Counting the Number of Corrugated Cardboard*. Electrical Engineering, Faculty of Engineering Rajamangala University of Technology Thanyaburi, 2014.

[3] Instructables, *Image Processing and Counting using MATLAB*. http://www.instructables.com/id/Image-Processing-and-Counting-using- MATLAB, May 22, 2014.

[4] MathWorksM, *Image processing*. http://www.mathworks.com/help/images/functionlist.html, May 22, 2014.