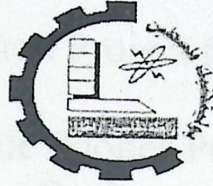


بسم الله الرحمن الرحيم

Palestine Polytechnic University



College of Engineering & Technology
Electrical and Computer Department

Graduation Project

Pre-paid Cards For Mobile Phones PC Vending Machine Using
Banknote

Project Team

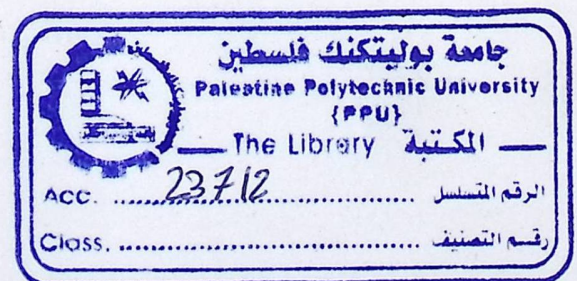
Ady Abed Allah

Mahmoud Motor

Project Supervisor:
Dr. Alaa Al-Halawani

Hebron – Palestine
January, 2008

I



Palestine Polytechnic University

Hebron-Palestine

Graduation Project Evaluation

Pre-paid Cards For Mobile Phones PC Vending Machine Using
Banknote

Project Team

Ady Abed Allah

Mahmoud Motor

According to the project supervisor and according to the agreement of the testing committee members, this project is submitted to the Department of Electrical and Computer Engineering at college of engineering and technology in partial fulfillment of the requirements of (B.Eng.) degree. In computer Engineering

Supervisor Signature

.....

Testing Committee Members Signature

.....

Department Head Signature

.....

جامعة بوليتكنك فلسطين

الخليل - فلسطين

عنوان مشروع التخرج

Pre-paid Cards for Mobile Phones PC Vending Machine Using
Banknote

فريق العمل

محمود إبراهيم مطور

عدي فائق عبد الله

بناء على طلب مشرف المشروع و بموافقة أعضاء اللجنة المناقشة, تم تقديم هذا المشروع إلى دائرة الهندسة
الكهربائية و الحاسوب في كلية الهندسة و التكنولوجيا للوفاء بمتطلبات الحصول على درجة البكالوريوس في هندسة
أنظمة الحاسوب

توقيع المشرف

توقيع أعضاء اللجنة المناقشة

توقيع رئيس الدائرة

Dedication

To our great home Palestine ...

To our Families ...

To our teachers ...

To whom help us ...

To our friends ...

To our supervisor Dr. Alaa Al-Halawani

To whom their guidance and supports made this work possible
We dedicate our humble effort

Project team

Acknowledgments

We would like to thanks every one who helped us to complete this project, and to appreciate all the encouragements of our families and friends

We would like specially to thank:

Dr. Alaa Al-Halawani, project supervisor, for his efforts.

Abstract

This project aims to recognizing an entered banknote, implement of a practical circuit to receive the entered banknote and printing a thirteen number (serial number of mobile card) having a costal value equal to that of the entered banknote.

The problem in this project is that how to recognize among different multiple banknotes (20, 50, 100 & 200NIS), also to decide whether the entered paper is really a banknote or any other paper.

After image processing of banknotes, if it is true the printer prints the mobile card number, else; the machine returns it back to the customer.

This project simplifies the selling process that is it performs the selling process by itself, and can be found in cafes, hospitals, streets, sale points, etc.

هذا المشروع يهدف إلى تمييز العملات النقدية المدخلة، بناء دائرة عملية لتقوم بدورها باستقبال تلك العملات، و من ثم تقوم بطباعة ثلاثة عشر رقما (الرقم التسلسلي لبطاقة شحن الهاتف الخليوي) قيمته النقدية تساوي قيمة العملة النقدية المدخلة.

المشكلة في هذا المشروع هي كيفية التمييز بين عملات نقدية ورقية مختلفة من حيث القيمة الشرائية لها (20, 50, 100 و 200 شيقل جديد)، أيضا لها القدرة على تحديد فيما إذا كانت الورقة المدخلة هي عملة نقدية أساسا أم أنها عبارة عن أي ورقة عادية أخرى.

بعد إجراء عملية معالجة صورية للعملات النقدية، إذا كانت العملة المدخلة صحيحة تقوم الطابعة بطباعة الرقم التسلسلي لبطاقة اعتماد الهاتف الخليوي، غير ذلك، فإن الآلة تقوم بإرجاع الورقة المدخلة إلى الزبون. هذا المشروع يسهل عملية البيع لأن الآلة التي تم بناؤها كي تأخذ على عاتقها عملية البيع بنفسها و بصورة أوتوماتيكية. و هذه الآلة يمكن وجودها في كل مكان، كالمقاهي، و المستشفيات، و الشوارع و نقاط البيع و غيرها.

Table of contents

Title	Page
Dedication	IV
Acknowledgment	V
Abstract	VI
Table of content	VII
List of figures	XII
List of table	XIII
 CHAPTER ONE	
Introduction	
1.1 Introduction	2
1.2 Proposed project	2
1.3 project importance	3
1.4 Estimated cost	4
1.5 Project Scheduling	5
1.6 project Risks	8
1.7 projects scope	8
1.8 documentation outline	9
 CHAPTER TWO	
Theoretical Background	
2.1 Introduction	11
2.2 System requirement	11
2.2.1 User requirements	11

2.2.1.1 Functional requirements	11
2.2.1.2 Non-functional requirements	11
2.2.2 System requirements	12
2.2.2.1 Functional requirement	12
2.2.2.2 Non-functional requirement	12
2.3 project component	13
2.3.1 Printer	13
2.3.2 Banknote Depositing	14
2.3.2.1 Camera	14
2.3.2.2 LDR	14
2.3.2.3 Relay	15
2.3.3 Cabinet Box	15
2.3.3.1 Stepper Motors	16
2.4 Image Processing	17
2.4.1 Introduction	17
2.4.2 Digital image processing	17
2.4.2.1 Coordinate Conventions	18
2.4.2.2 Image as Matrices	18
2.4.3 Color Space	19
2.4.3.1 RGB color model	19
2.4.3.1.1 Grayscale	21
2.4.3.1.2 Convert RGB to Grayscale	21
2.4.3.1.3 Histogram	21
2.4.3.1.3.1 Implementation	24
2.4.3.1.3.2 Possible problems in Conventional Histogram	24
2.4.3.1.3.3 Comparing Histograms	26
2.4.3.2 HSV Color Model (Hue, Saturation, Value)	27

2.4.4 Color conversion	28
2.4.4.1 RGB to HSV conversion	28
2.4.4.2 Edge Detection	29
2.4.4.2.1 Edges can reflect:	30
2.4.4.3 Sobel steps:	30
2.4.4.4 Sobel explanation	31

CHAPTER THREE

Architectural Design

3.1 Introduction	34
3.2 objectives	34
3.3 Block diagram	34
3.4 How the system works	36
3.5 System Model	36
3.5.1 Flow chart Model	36

CHAPTER FOUR

Hardware Design

4.1 Introduction	39
4.2 Banknote Depositing	39
4.3 Circuit Design	40

Chapter Five

Software Design

5.1 Overview	42
5.2 Functional Requirement	42
5.3 Algorithms	42
5.3.1 Banknote Recognizer	42
5.3.2 Motor	43
5.3.3 Camera	43
5.3.4 Printer	43
5.4 MATLAB Code	43
5.5 Visual C++ software code	44

Chapter six

Testing

6.1 introduction	46
6.2 Software test	46
6.2.1 Mat lab:	46
6.2.2 Visual C++	50
6.2.2.1 RGB Histogram	50
6.2.2.2 Histogram for orient edge detection	51
6.2.2.3 RGB Histogram and Orient Histogram	51
6.3 Hardware test	52
6.3.1 LDR	52
6.3.2 Stepper motor	52
6.3.3 The cabinet box	53
6.3.4 Camera	53

Chapter Seven

Conclusion and Future Work

7.1 Conclusions

55

7.2 Future Work

55

REFERENCES

56

APPENDIX A

18

APPENDIX B

20

List of Figure

Figure	Page
Figure (2.1): project component	13
Figure (2.2): LDR	15
Figure (2.3): Relay	15
Figure (2.4): stepper motor	16
Figure (2.5): image coordinate convention	18
Figure (2.6): RGB coordinates system	20
Figure (2.7): RGB color model	20
Figure (2.8): a) gray image, b) histogram	22
Figure (2.9): image histogram	22
Figure (2.10): histogram bins	23
Figure (2.11), a) The original image, b) and Intensity shifted image	25
Figure (2.12): Original histogram and Intensity histograms of images with intensity shift.	25
Figure (2.13): Two images with similar color histograms	26
Figure (2.14) HSV coordinates	27
Figure (2.15) Obtainable HSV color from RGB color space	28
Figure (2.16) the two kernels for Dx & Dy.	29
Figure (2.17) edge detection	30
Figure (2.18): Image convolution	32
Figure (3.1): general block diagram	35
Figure (3.2): complete system block diagram	35
Figure (3.3): flow chart	37
Figure (4.1): banknote depositing	40

Figure (6.1) the nearest 4 images by using HSV_ histogram	47
Figure (6.2): Figure (6.2): a) input image, b) grayscale for image a.	48
Figure (6.3): the correct 4 images by using HSV_ histogram	49
Figure (6. 4) : a) input image , b) Sobel edge for input image a.	50

List of tables

Table	Page
Table (1.1): Hard Ware Estimated cost	4
Table (1.2): Soft Ware Estimated cost	5
Table (1.3) Time Plan	7
Table (6.1): HSV testing	46
Table (6.2): gray histogram testing	47
Table (6.3): RGB Histogram testing	48
Table (6.4): Sobel edge detection testing	49
Table (6.5):VC++ codes testing	52

Chapter one

Introduction

1.1 Introduction

1.2 Proposed project

1.3 project importance

1.4 Estimated cost

1.5 Project Scheduling

1.6 project Risks

1.7 projects scope

1.8 documentation outlines

1.1 Introduction

A vending machine is a machine that sells some merchandise when a customer deposits money, validated by a currency detector, sufficient to purchase the desired item such as soft drinks, chocolate, cigarettes...etc. The project is aimed to design pre-paid cards serial numbers for mobile phones vending machine using PC that has the ability to accept a banknote value type such as (20 New Israel shekel (NIS), 50NIS, 100NIS, and 200NIS) from the user and deliver the selected item to the user.

The project will give a simple practical design that can be implemented in the future in our region. This design of the machine includes PC, banknote recognizer unit including a camera, and printer.

1.2 Proposed project

There are many applications used in daily life that treat and react with images, pictures, banknotes, coins, many in general. All the previous need what is known as image processing. Let us give a practical example "the vending machine ".This machine sells hot or cold drinks, and needs providing money by the user.

The main problem of that machine is how to recognize the entered banknote, with respect to its value, and whether it is real or not.

The suggested solution is to capture more than one image for each banknote value type (many images of a 20 NIS, 50 NIS, etc.). Also those images should be for both sides of the banknote.

The captured image is to be stored in the database of the system, then to be processed and analyzed using visual C++ software and using a certain circuit that compares among a new entered banknote and the old ones stored in the database.

1.3 project importance

The following points explain the impact of the PC vending machine on many aspects of life and clarify the importance of the vending machine by giving many examples.

1- We can now purchase all kinds of drinks, chocolate, newspapers, and pre-paid cards for mobile phones. The list is inexhaustible.

2- We are now living in a fast-paced, automated world whereby we will like everything to be fast, effective and convenient. We have benefited from the usage of vending machines around us that we are not even consciously aware of their important in our everyday life.

1.4 Estimated cost

The following table specifies the estimated total for the different components in the project.

Table (1.1): Hardware estimated cost

	Object	Number	cost
1	Personal Computer(PC) Pentium III (800 Mega Hertz (MHz),512 RAM,20 Giga Bytes(GB), Hard disk)	1	\$200
2	Camera	1	\$10
3	Stepper Motors	3	\$3
4	Power supplies (12 V,15 V,5V)	2	\$50
5	Cable and connectors	2	\$10
6	Light Dependent Resistors (LDR)	1	\$10
7	Printer HP3000	1	\$50
8	Parallel Port Cable	1	\$5
9	Relay	2	\$5
10	Lamp	2	\$10
11	Cabinet Box	1	\$30
	Total		\$383

Table (1.2): Software estimated cost

	Object	Number	cost
1	Visual C++ 2003	1	\$100
2	Windows XP	1	\$50
	Total		\$150

1.5 Project Scheduling

The timing management will divide the system hierarchy according to the dependent and independent actions. This process will explain the critical path of the systems.

The project activities here depend on each other, so the task durations and dependencies are the following:-

T1: Preparation of the project: here we introduce to start in the project and discuss with the advisor to initialize the project, and preparing the group and evaluate the project tasks and levels. At this period we explain the system requirements.

The project consists of four major high level parts:

- Software: design program software using MATLAB.
- Convert MATLAB code to C++ programming language.
- Hardware: build the Banknote Depositing.
- Interface: interface between the code and the Banknote Depositing to work properly.

Table (1.2): Software estimated cost

	Object	Number	cost
1	Visual C++ 2003	1	\$100
2	Windows XP	1	\$50
	Total		\$150

1.5 Project Scheduling

The timing management will divide the system hierarchy according to the dependent and independent actions. This process will explain the critical path of the systems.

The project activities here depend on each other, so the task durations and dependencies are the following:-

T1: Preparation of the project: here we introduce to start in the project and discuss with the advisor to initialize the project, and preparing the group and evaluate the project tasks and levels. At this period we explain the system requirements.

The project consists of four major high level parts:

- Software: design program software using MATLAB.
- Convert MATLAB code to C++ programming language.
- Hardware: build the Banknote Depositing.
- Interface: interface between the code and the Banknote Depositing to work properly.

Table (1.2): Software estimated cost

	Object	Number	cost
1	Visual C++ 2003	1	\$100
2	Windows XP	1	\$50
	Total		\$150

1.5 Project Scheduling

The timing management will divide the system hierarchy according to the dependent and independent actions. This process will explain the critical path of the systems.

The project activities here depend on each other, so the task durations and dependencies are the following:-

T1: Preparation of the project: here we introduce to start in the project and discuss with the advisor to initialize the project, and preparing the group and evaluate the project tasks and levels. At this period we explain the system requirements.

The project consists of four major high level parts:

- Software: design program software using MATLAB.
- Convert MATLAB code to C++ programming language.
- Hardware: build the Banknote Depositing.
- Interface: interface between the code and the Banknote Depositing to work properly.

Table (1.2): Software estimated cost

	Object	Number	cost
1	Visual C++ 2003	1	\$100
2	Windows XP	1	\$50
	Total		\$150

1.5 Project Scheduling

The timing management will divide the system hierarchy according to the dependent and independent actions. This process will explain the critical path of the systems.

The project activities here depend on each other, so the task durations and dependencies are the following:-

T1: Preparation of the project: here we introduce to start in the project and discuss with the advisor to initialize the project, and preparing the group and evaluate the project tasks and levels. At this period we explain the system requirements.

The project consists of four major high level parts:

- Software: design program software using MATLAB.
- Convert MATLAB code to C++ programming language.
- Hardware: build the Banknote Depositing.
- Interface: interface between the code and the Banknote Depositing to work properly.

T2: The project searching and analysis: at this period we start the first step to search and analyze the project, collect information and data about the project levels and sublevels, tasks and subtasks, there are many resources to searching and analyzing the concepts.

T3: Write the MATLAB code.

T4: The project requirement analysis: the project has many tasks and equipments that must be provided and explained to implement the final project to achieve the system requirements and tasks. The system has hardware and software requirements which must be achieved through the prototype and final presentation.

T5: Convert the MATLAB program to Visual C++ Programming language.

T6: Build the Banknote Depositing and connect the printer.

T7: Interface between the software and hardware.

T8: Test the project: we will test the project and implement it to insure that the system and user requirements levels are achieved or not, to adjust the problems and errors in the project and can maintain it, and then try to test and execute it again.

T9: Documentation writing: the writing begins with us from the first step to the last one in parallel.

Table (1.3) shows the time plane in our project

Table (1.3) Time Plan

Task Name	Feb 07	Mar 07	Apr 07	May 07	Sep 07	Oct 07	Nov 07	Dec 07
Preparing Team and Project								
Searching and Analysis								
Write MATLAB code and Testing								
System Requirements								
Convert MATLAB to VC++ 2003 Code								
System Design								
Interfacing								
Testing								
Documentation								

1.6 Project Risks

The project must reach to high performance and more security to work perfect, but some risks may be occur during the design, manipulation, and testing. Some of these risks are:

- The devices may be not found here, this leads to delaying the collection of the machine.
- The recognized banknote will not work perfect any way.
- Not reaching to top security when recognizing the banknote.

1.7 Projects Scope

After completing the main requirements of our project, it can be used in a range widely in our life:

- Communication company.
- Point of sale.
- University.
- Hospital.

1.8 Documentation Outlines

Chapter two introduces and illustrates the idea of the project. It describes the components which are used in the project.

Chapter three introduces the design in details, shows the general block diagram of the system and explains how the system works.

Chapter four "hardware System Design" shows the system hardware, flow charts, connections and circuits.

Chapter five "Software System Design" shows the system software flow charts, and subroutines.

Chapter six "Implementation and Testing" discusses the steps followed to implement and test our own design.

Chapter seven "Conclusion and Future Work" provides many suggestions to expand this project and summarizes the project conclusions.

Chapter two

Theoretical Background

2.1 Introduction

2.2 system requirements

2.3 project components

2.4 Image Processing

2.5 Recognize Image

2.1 Introduction

This chapter focuses on the theoretical subjects related to the main idea of the project, and information about the components used in the project and also some shape designs about the commercial vending machine.

2.2 system requirements

This section is a listing of the main requirements that must be met in our project, in order to set the main services that will be provided.

2.2.1 User requirements

These are our system requirements for the PC vending machine users.

2.2.1.1 Functional requirements

- Depositing the banknote.
- Receiving the item.

2.2.1.2 Non-functional requirements

- Speed: The user can use the PC vending machine at an appropriate speed.

- Flexibility: The PC vending machine will provide the user with a higher flexibility of usage.
- Ease of use: User can easily use the vending machine.
- Accuracy: The vending machine takes user events accurately without any error.
- Efficiency: The vending machine can take the place of traditional selling methods.

2.2.2 System requirements

2.2.2.1 Functional requirements

- Allow the user to deposit a banknote.
- Recognize the banknote.
- Reject the request, if no item.
- Reject the request, if incorrect banknote.

2.2.2.2 Non-functional requirements

- Low power consumption.
- Fast response and speed: The vending machine must respond to the user's action within a limited period of time.
- Efficiency: The vending machine will be accurate, and maintainable.
- Performance: Vending machine gives user greater performance, and high-speed interaction with the users.

2.3 project components

The project has many components which are listed in the following and shown in figure (2.1). These components are:

- Banknote Depositing (camera, and banknote recognizer).
- Printer.
- Cabinet Box.
- PC "Personal Computer".

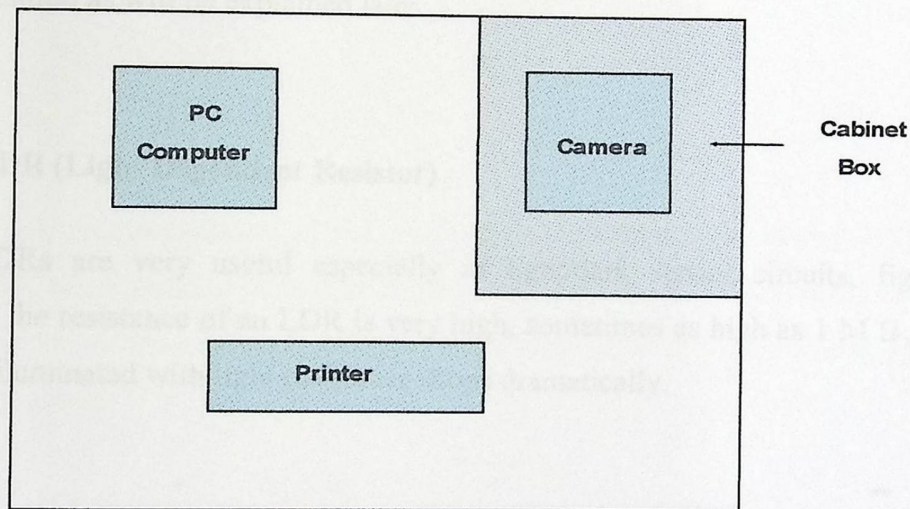


Figure (2.1) project component

2.3.1 Printer

The printer is used to copy the mobile card serial number with any object of banknote.

2.3.2 Banknote Depositing

This part responsible for accepting the banknote deposited from the user and transferring a banknote to the camera to capture it. This part contains a camera, and stepper motor to move a banknote to the correct position.

2.3.2.1 Camera

This part is responsible for capturing a banknote deposited from the user, to start the recognition as will be explained later.

2.3.2.2 LDR (Light Dependent Resistor)

LDRs are very useful especially in light/dark sensor circuits, figure (2.2). Normally the resistance of an LDR is very high, sometimes as high as $1\text{ M}\Omega$, but when they are illuminated with light resistance drops dramatically.

This part is responsible for the beginning of the banknote capturing process it uses the led and the inverter.

LIGHT DEPENDENT RESISTOR

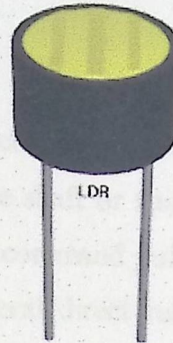


Figure (2.2): LDR [13]

2.3.2.3 Relay

A relay is an electrical switch that uses an electromagnet to move the switch from the off to on position instead of a person moving the switch. It takes a relatively small amount of power to turn on a relay but the relay can control something that draws much more power.

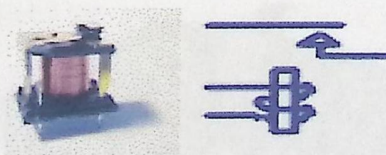


Figure (2.3): relay [12]

2.3.3 Cabinet Box

This part consists of stepper motor, its driving circuit, belt, camera, LDRs and lamps (our project hardware).

2.3.3.1 Stepper Motors

A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motor's rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shaft's rotation. The speed of the motor shaft rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied.

A stepper motor can be a good choice whenever controlled movement is required. They can be used to advantage in applications where you need to control rotation angle, speed, position and synchronism. Because of the inherent advantages listed previously, stepper motors have found their place in many different applications. Some of these include printers, plotters, high end office equipment, hard disk drives, medical equipment, fax machines, automotive and many more.[7]



Figure (2.4) stepper motor [7]

2.4 Image Processing

2.4.1 Introduction

This section discusses how to find the features of an image, recognition image and comparisons between input images with many images in database using many methods.

2.4.2 Digital image processing

An image may be defined as a two-dimensional light intensity function $f(x,y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x,y) is proportional to the brightness or (gray level) of the image at that point. When x , y , and the amplitude values of f are all finite, discrete quantities, we call the image a digital image. The field of digital image processing refers to processing digital image by means of digital computer. Note that a digital image is composed of finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pixel, or pels. Pixels is the term most widely used to denote the elements of a digital image. A digital image can be considered a matrix whose row and column defines a point in the image and the corresponding matrix element value identifies the gray level at that point.

2.4.2.1 Coordinate Conventions

The result of sampling and quantization is a matrix or real numbers. An image $f(x,y)$ is sampled so that the resulting image has M row and N columns, we say the image is of size $M \times N$. the value of the coordinates (x,y) are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates. In many image processing books, the image origin is defined to be at $(x,y) = (0,0)$. The next coordinate value along to the first row of the image are $(x,y) = (0,1)$. The notation $(0,1)$ is used to signify the second sample along the first row. It does not mean that these are actual values of physical coordinates when image was sampled. Figure (2.6) shows this coordinate convention.

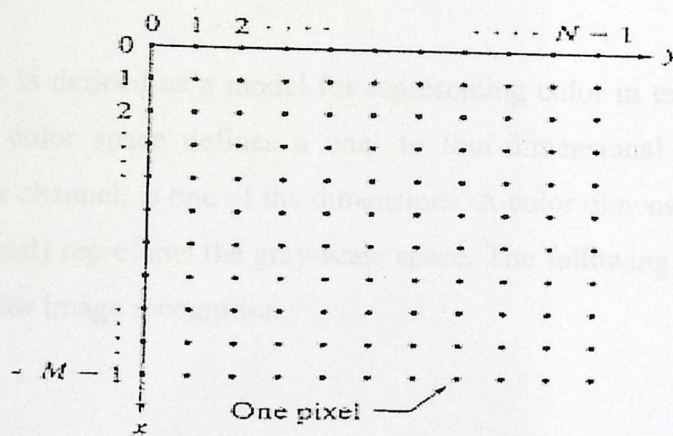


Figure (2.5): image coordinate convention [1].

2.4.2.2 Image as Matrices

The coordinate system in figure (2.6) and the preceding lead to the following representation for a digitized image function:

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N - 1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N - 1) \\ \vdots & \vdots & & \vdots \\ f(M - 1, 0) & f(M - 1, 1) & \cdots & f(M - 1, N - 1) \end{bmatrix}$$

The right side of this equation is a digital image by definition, each position of this array is called pixel, the terms image and pixel are used throughout the rest of our discussion to denote a digital image and its elements.

2.4.3 Color Space

A color space is defined as a model for representing color in terms of intensity values. Typically, a color space defines a one- to four-dimensional space. A color component, or a color channel, is one of the dimensions. A color dimensional space (i.e. one dimension per pixel) represents the gray-scale space. The following two models are commonly used in color image recognition.

2.4.3.1 RGB color model

The RGB color model is composed of the primary colors Red, Green, and Blue. This system defines the color model that is used in most color raster graphics. They are considered the "additive primaries" since the colors are added together to produce the desired color. The RGB model uses the Cartesian coordinate system as shown in Figure

(2.7). Notice the diagonal from (0, 0, 0) black to (1, 1, 1) white which represents the grey-scale.

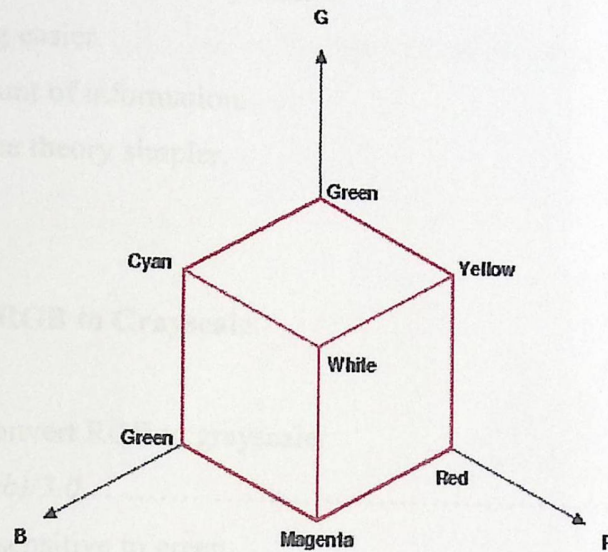


Figure (2.6): RGB coordinates system [10].

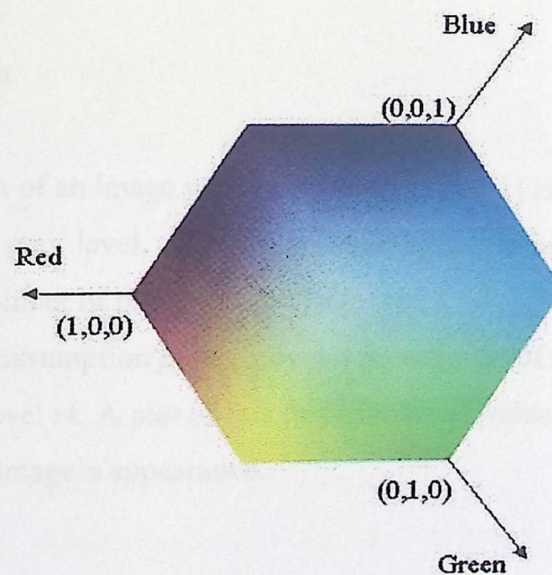


Figure (2.7): RGB color model [10].

2.4.3.1.1 Gray Scale

Convert an image from RGB to Gray that:

- Makes processing easier.
- Reduces the amount of information.
- Makes some of the theory simpler.

2.4.3.1.2 Convert RGB to Grayscale

This cases use to convert RGB to grayscale.

- Case (1) $i = (r+g+b)/3.0$(2.1)

Our eyes are more sensitive to green

- Case (2) $i = 0.3r + 0.59g + 0.11b$(2.2)

2.4.3.1.3 Histogram

The histogram of an image with gray levels in $[0, L-1]$ is a discrete function

Where rk is the k th gray level, nk is the number of pixels in the image With tat gray level, n is the total number of pixels in the image, and $k = 0, 1, 2 \dots L-1$.

Under the ergodicity assumption $p(rk)$ provides an estimate of the probability of Occurrence of gray level rk . A plot of this function for all values of k can provide a Crude description of image's appearance.

It's obvious that the spread of values and the shape of the gray level distribution provide useful information about the content of an image. In order to quantify this information, several features can be extracted.

Image histogram is a chart that shows the distribution of intensities in an image, figure (2.9) shows the histogram for gray image.

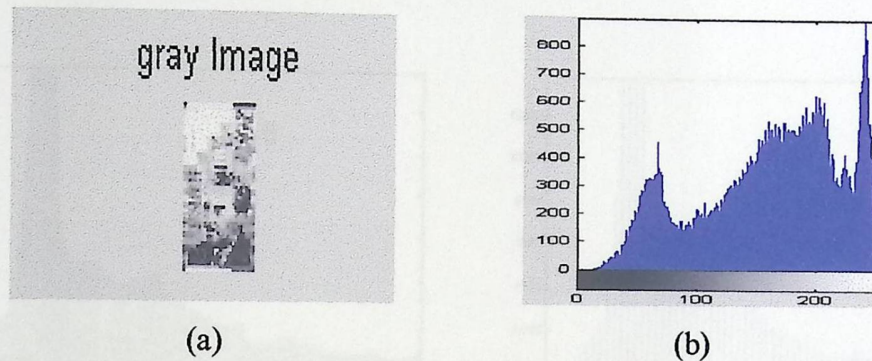


Figure (2.8): a) gray image, b) histogram



Figure (2.9): image histogram [9]

A histogram is a method used to describe the frequency distribution of a digital signal. Histograms are composed of multiple bins, with each bin corresponding to a range of values. The value of each bin is obtained by summing the occurrences of digital signal samples whose values fall in the domain of that corresponding bin as in figure (2.10). We assume that all the images are scaled as a discrete color space which contains the same pixels of n . The histograms are the feature vectors to be stored as the index of the image database.

Histogram can have different number of bins, it done by divide the maximum value of pixel over 4, 8, 16, and 32, figure (2.11) shows the histogram for different bins.

If the maximum value equals 255, and we want 16 bins histogram, we divide each pixel of the image over 16.

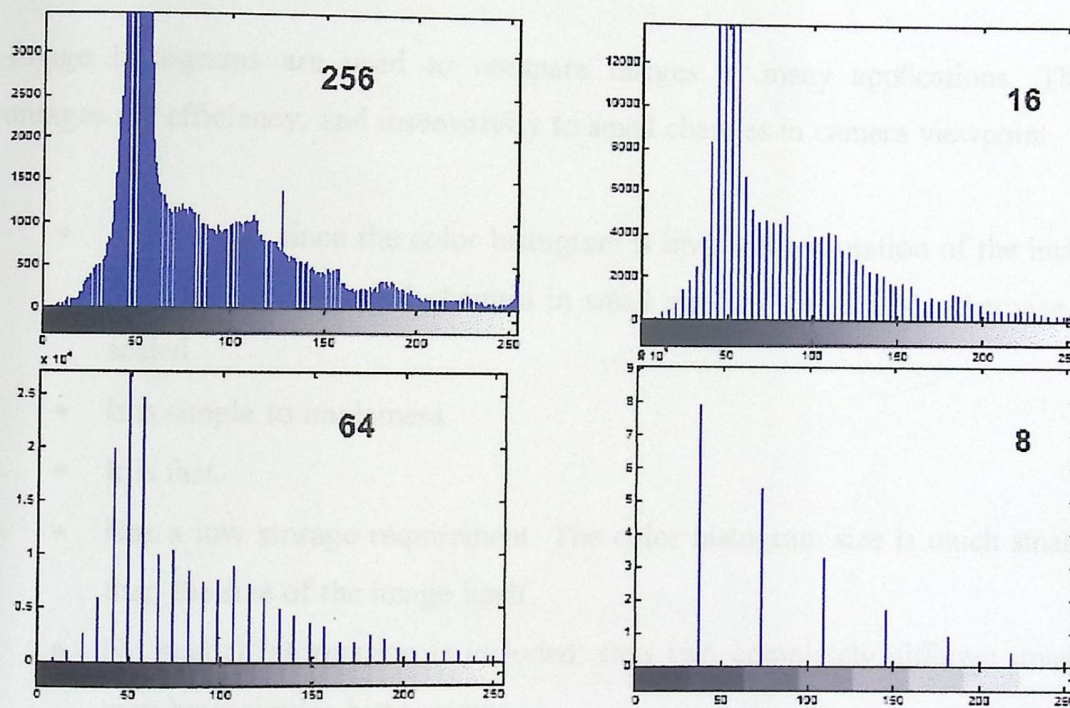


Figure (2.10): histogram bins

2.4.3.1.3.1 Implementation:

- For intensity images: One dimensional array of integer's [0...Bins].
- For colored images: 3-dimensional array of integer's [0...Bins for Red, 0...Bins for Green, 0...Bins for Blue].

The characteristics of histogram are as follows:

Image histograms are used to compare images in many applications. Their advantages are efficiency, and insensitivity to small changes in camera viewpoint.

- It is robust, since the color histogram is invariant to rotation of the image on the view axis, and changes in small steps when rotated otherwise or scaled.
- It is simple to implement.
- It is fast.
- Has a low storage requirement. The color histogram size is much smaller than the size of the image itself.
- No spatial information is included: thus two completely different images may have similar histograms.
- Lighting conditions may alter the histogram of an image.

2.4.3.1.3.2 Possible problems in Conventional Histogram

Images with same visual information but with shifted light intensity may degrade the similarity level significantly if conventional histogram method is used.

Histogram is very sensitive to lighting conditions due to the nature of color space. Perceptually, similar colors are interpreted as completely different thus giving no appealing results. In figure (2.11), (a) is original image, and (b) is intensity shifted image that has different brightness condition. Figure (2.12). Is original histogram and Intensity histogram of images with intensity shift.

In figure (2.11), and figure (2.12), the similar histograms in different brightness conditions of the two pepper images are shown. The intersection of the two histograms holds a result of smaller value, although the two images are highly visually similar [2].



Figure (2.11), a) The original image, b) and Intensity shifted image [2].

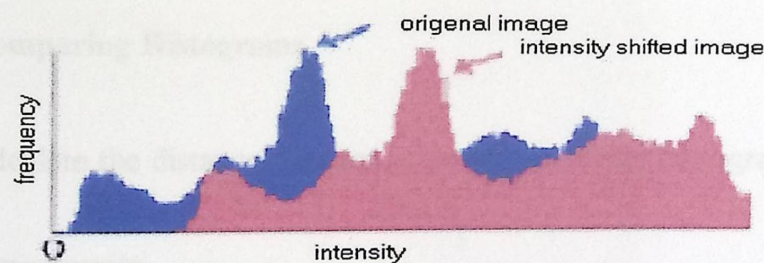


Figure (2.12): Original histogram and Intensity histograms of images with intensity shift [2].

The histogram method computes 64 bin gray level histograms of the two images and Euclidean distance measure is used to find the histogram difference.



Figure (2.13): Two images with similar color histograms [8].

The images shown in figure (2.13) have similar color histograms, despite their rather different appearances. The color red appears in both images in approximately the same quantities. In the left image the red pixels (from the flowers) are widely scattered, while in the right image the red pixels (from the golfer's shirt) form a single coherent [8].

2.4.3.1.3.3 Comparing Histograms

Is done by calculate the distance between points in the image Histograms equivalent metrics.

$$\text{Distance} = \sqrt{\sum (A_i - B_i)^2} \dots\dots\dots(2.3)$$

Where:

A: the histogram of image one.

B: the histogram of image two.

i: the ith entity of image histogram.

2.4.3.2 HSV Color Model (Hue, Saturation, Value)

The HSV stands for the Hue, Saturation, and Value based on the artists (Tint, Shade, and Tone).

H: what color is it?

S: How strongly colored is it.

V: How bright or dark is it.

The coordinate system in a hexatone in Figure (2.14) a view of the HSV color model. The Value represents intensity of a color, which is decoupled from the color information in the represented image. The hue and saturation components are intimately related to the way human eye perceives color resulting in image processing algorithms with physiological basis. As hue varies from 0 to 1.0, the corresponding colors vary from red, through yellow, green, cyan, blue, and magenta, back to red, so that there are actually red values both at 0 and 1.0. As saturation varies from 0 to 1.0, the corresponding colors (hues) vary from unsaturated (shades of gray) to fully saturated (no white component). As value, or brightness, varies from 0 to 1.0, the corresponding colors become increasingly brighter [10].

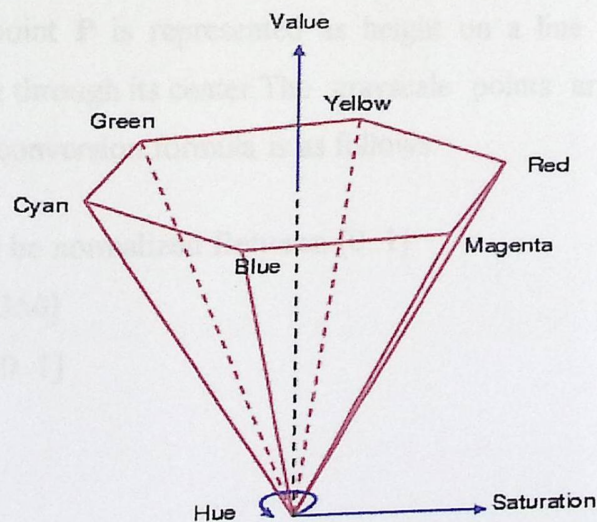


Figure (2.14): HSV coordinates [10].

2.4.4 Color conversion

2.4.4.1 RGB to HSV conversion

In Figure (2.15) the obtainable HSV colors lie within a triangle whose vertices are defined by the three primary colors in RGB space:

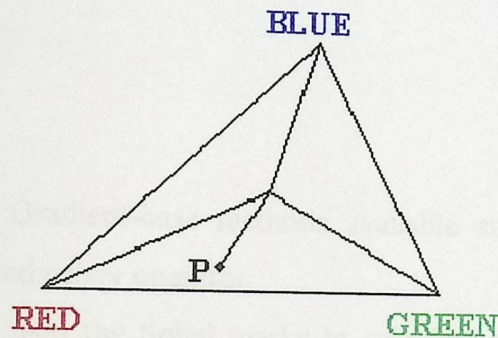


Figure (2.15) Obtainable HSV color from RGB color space[10].

The hue of the point **P** is the measured angle between the line connecting **P** to the triangle center and line connecting RED point to the triangle center. The saturation of the point **P** is the distance between **P** and triangle center. The value (intensity) of the point **P** is represented as height on a line perpendicular to the triangle and passing through its center. The grayscale points are situated onto the same line. And the conversion formula is as follows:

- R, G and B should be normalized Between [0..1]
- H in the range [0..360]
- S and V between [0..1]

$$H = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R-G) + (R-B)]}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \right\} \dots\dots\dots(2.4)$$

$$S = 1 - \frac{3}{R+G+B} [\min(R, G, B)] \dots\dots\dots(2.5)$$

$$V = \frac{1}{3}(R+G+B) \dots\dots\dots(2.6)$$

2.4.4.2 Edge Detection

There are many Gradient-base methods available such as Sobel operator, Roberts Cross operator and canny operator.

Sobel edge detection uses the Sobel masks to calculate the gradient magnitude. The Sobel masks are shown in Figure (2.16). With Sobel edge , we could easily get the distinct edges of an image in both horizontal and vertical directions. For fast computation, we use the approximated gradient magnitude calculation $G = |G_x| + |G_y|$.

-1	0	+1		+1	+2	+1
-2	0	+2		0	0	0
-1	0	+1		-1	-2	-1
Gx				Gy		

Figure (2.16) the two kernels for Dx & Dy.

Edge detection is a problem of fundamental importance in image processing. The goal of edge detection is to mark the points in images at which the intensity changes sharply.

2.4.4.2.1 Edges can reflect:

- Discontinuities in depth, or surface orientation,
- Changes in material properties.
- Variations in scene illumination.

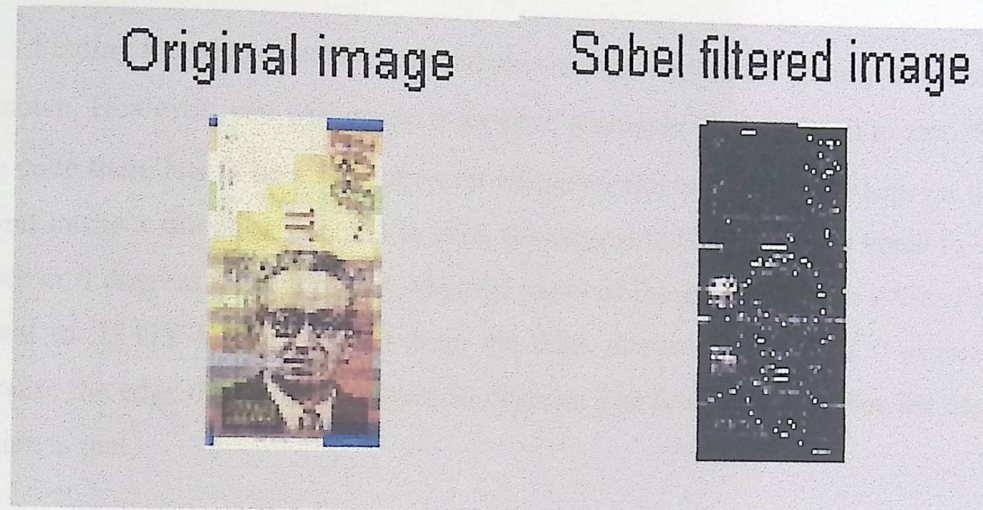


Figure (2.17) edge detection

2.4.4.3 Sobel steps:

There is one step to find sobel edge and the other steps are for calculating the orientation histogram.

Step1: Find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows).

The magnitude, or Edge Strength, of the gradient is then approximated using the formula:

$$|G| = \sqrt{G_x^2 + G_y^2} \dots\dots\dots(2.7)$$

An approximate magnitude can be calculated using:

$$|G| = |G_x| + |G_y| \dots\dots\dots(2.8)$$

Step 2:

Finding the edge direction is trivial once the gradient in the x and y directions are known. However, you will generate an error whenever G_x is equal to zero. So in the code there has to be a restriction set whenever this takes place. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to. If G_y has a value of zero, the edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. The formula for finding the edge direction is just.

$$\theta = \tan^{-1} (G_y / G_x) \dots\dots\dots(2.9)$$

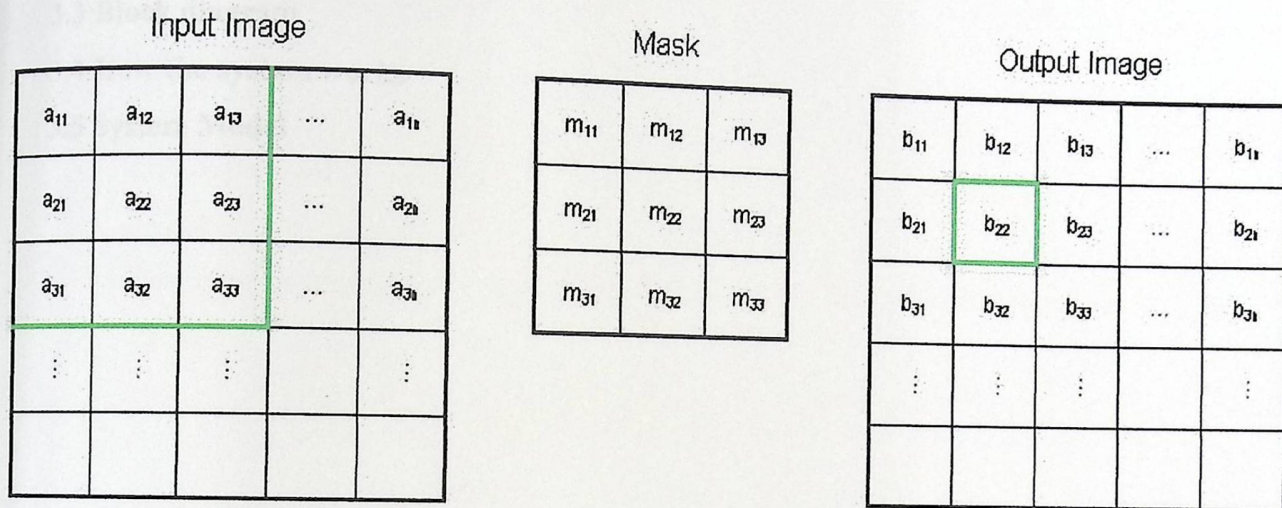
Step 3:

Make histogram for the orientation or theta.

2.4.4.4 Sobel Explanation

The mask is slid over an area of the input image, changes that pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of a row. It then starts at the beginning of the next row. The example below shows the mask being slid over the top left portion of the input image represented by the green outline. The formula shows how a particular pixel in the output image would be calculated. The center of the mask is placed over the pixel you are manipulating in the image, and the I & J values are used to move the file pointer so you can multiply,

for example, pixel (a₂₂) by the corresponding mask value (m₂₂). It is important to notice that pixels in the first and last rows, as well as the first and last columns cannot be manipulated by a 3x3 mask. This is because when placing the center of the mask over a pixel in the first row (for example), the mask will be outside the image boundaries.



$$b_{22} = (a_{11} * m_{11}) + (a_{12} * m_{12}) + (a_{13} * m_{13}) + (a_{21} * m_{21}) + (a_{22} * m_{22}) + (a_{23} * m_{23}) + (a_{31} * m_{31}) + (a_{32} * m_{32}) + (a_{33} * m_{33})$$

Figure (2.18): image convolution [11].

Chapter 3

Architectural Design

3.1 Introduction

3.2 objectives

3.3 Block diagram

3.4 How the system works

3.5 System Model

- Designing an independent PC program for mobile phone vending machine.
- Designing a barcode recognizer that has the ability to recognize the barcode.

3.3 Block diagram

Figure (3.1) shows the general block diagram for a vending machine. It contains three parts:

- Input part: This part is responsible for all inputs to the PC, this contains the barcode, allowing to enter the barcode, and the system - input barcode.
- PC Function III: This part is responsible for all operations such as recognition the barcode, activation of mobile phone camera, and conversion between the value of the barcode and price of product.
- Output part: This part is responsible for providing the serial number of product to the user.

Figure (3.1) shows the general block diagram, and Figure (3.2) shows the complete system block diagram.

3.1 Introduction

This chapter discusses the design concepts of the project. It includes the project objectives, the system block diagram that helps to understand the system and how it works.

3.2 objectives

- Designing self independent PC pre-paid cards for mobile phones vending machine.
- Designing a banknote recognizer unit which has the ability to recognize the banknote.

3.3 Block diagram

Figure (3.1) shows the general block diagram for a vending machine. It contains three parts:

- Input part: This part is responsible for all inputs to the PC; this contains the banknote depositing to enter the banknote, and the camera to capture banknote.
- PC Pentium III: This part is responsible for all operation such as recognition the banknote, activation of motor, printer, camera, and comparison between the value of the banknote and price of product.
- Output part: This part is responsible for printing the serial number of product to the user.

Figure (3.1) shows the general block diagram, and figure (3.2) shows the complete system block diagram.

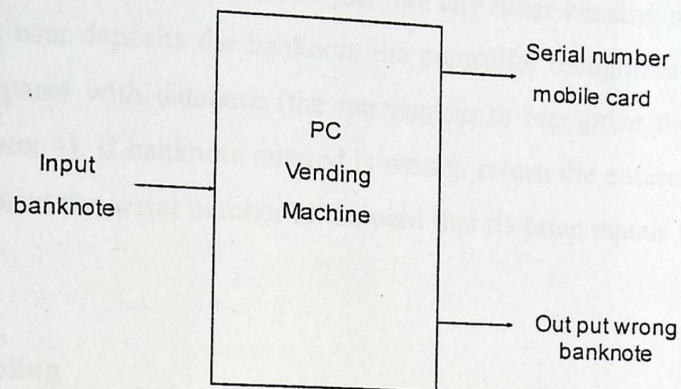


Figure (3.1) general block diagram

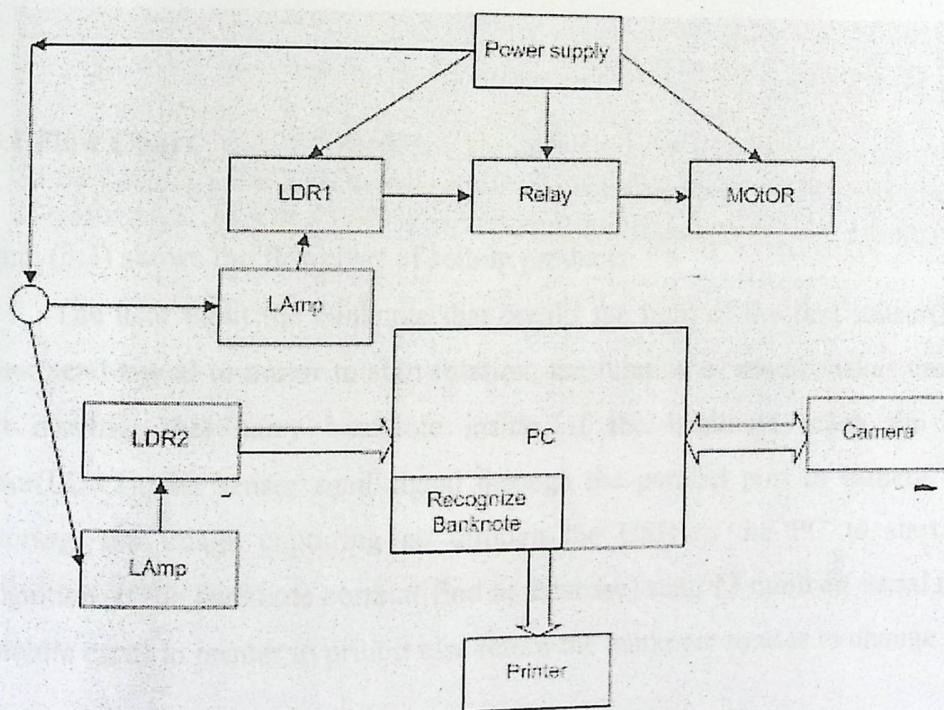


Figure (3.2) complete system block diagram

3.4 How the system works

The PC vending machine operates just like any other vending machine in the world. When the user deposits the banknote the controller recognizes the banknote entered and compares with database (the mechanism to recognize the banknote is described in chapter 4). If banknote entered is wrong, return the entered banknote to the user, if true, print the serial number of the card that its price equals the banknote.

3.5 System Modeling

In this section we will view the model that describes the functionality of our system components in both hardware and software.

3.5.1 Flow Chart

Figure (3.3) shows the flowchart of selling products

The user input the banknote that breaks the light of the first sensor(LDR1), sensor send signal to motor to start rotation, the rotation of motor makes the belt to start moving, that carry banknote inside, if the banknote reach the second sensor(LDR2), the sensor send signal through the parallel port to camera to start capturing, the image capturing go through the USB to the PC to start image recognition, if the banknote correct(find in database) send 13 number(serial number of mobile card) to printer to print it else return the banknote to user to change it.

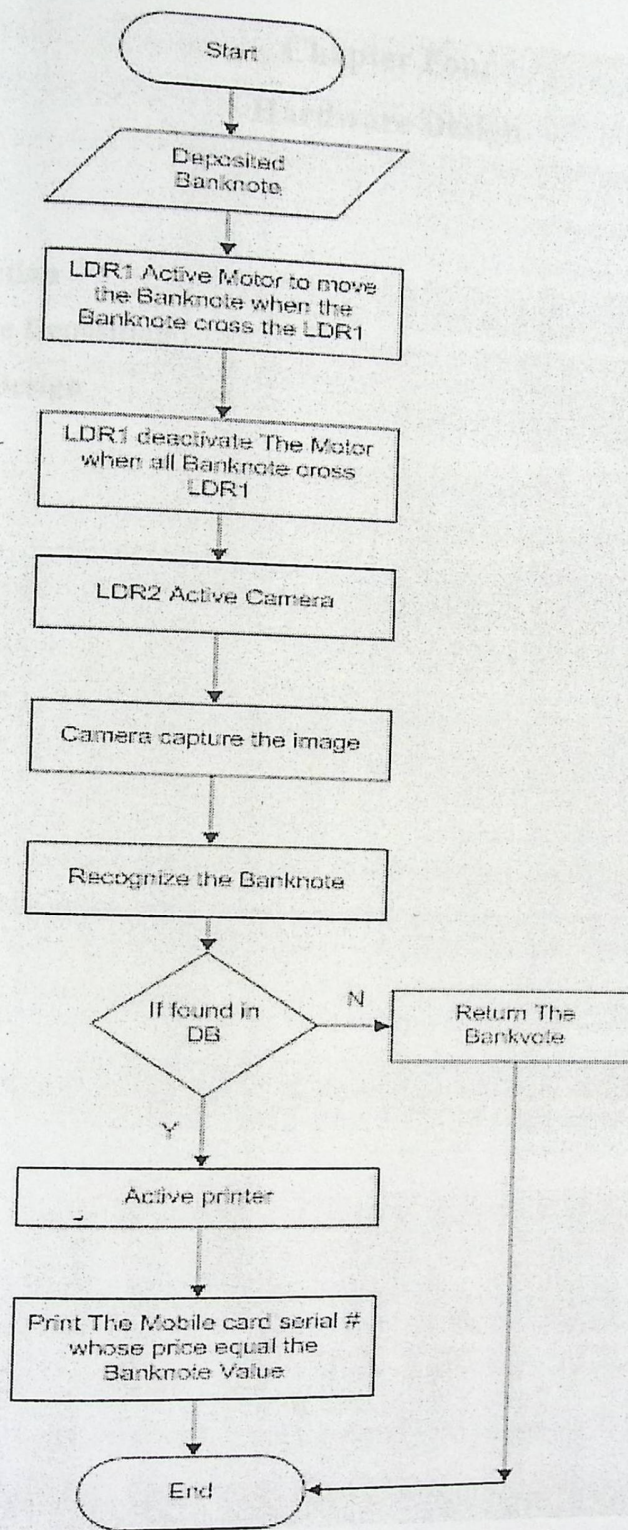


Figure (3.3) flow chart

Chapter Four

Hardware Design

4.1 Introduction

4.2 Banknote Depositing

4.3 Circuit Design

Contents of following components:

1. Two LCDs

2. Two laser lamps

3. Two transistors (amplifiers)

4. Sensor circuit

5. Two relays

6. PCB

When the user entered a new banknote:

1. The banknote breaks the light of laser lamp

2. Sensor sends signal to micro to start sequence

3. The start rotation of motor makes belt to start moving that carry banknote

4. The rotation of motor can't stop until breaking light of laser lamp

5. If the banknote reach the second sensor, the second sensor send signal through the parallel port to the camera to start capture the banknote

6. The picture goes to the software algorithm to start recognizing

7. If the banknote is true (found in database), inform the printer to print the serial

8. serial number with price equal the entered banknote

9. If the banknote had not found in database, return the banknote to the user

4.1 Introduction

The system Hardware design contains: the electrical part, which includes the banknote recognizer.

4.2 Banknote Depositing

Consists of following components:

- Two LDR.
- Two laser lamps.
- Two transistors (amplifiers).
- Stepper motor.
- Two relay.
- Belt.

When the user entered a new banknote:

- 1- The banknote breaks the light of laser lamp.
- 2- Sensor sends signal to motor to start rotation.
- 3- The start rotation of motor makes belt to start moving that carry banknote inside.
- 4- The rotation of motor can't stop until braking light of laser lamp.
- 5- If the banknote reach the second sensor, the second sensor send signal through the parallel port to the camera to start capture the banknote.
- 6- The picture goes to the software algorithms to start recognizing.
- 7- If the banknote is true (found in database), enable the printer to print the mobile card serial number with price equal the entered banknote.
- 8- If the banknote bad (not found in database), return the banknote to the user.

9- The relays used to control the sensors voltage and current.

4.3 Circuit Design

Figure (4.1) show the banknote depositing. This circuit show how connect all the components together, to work well.

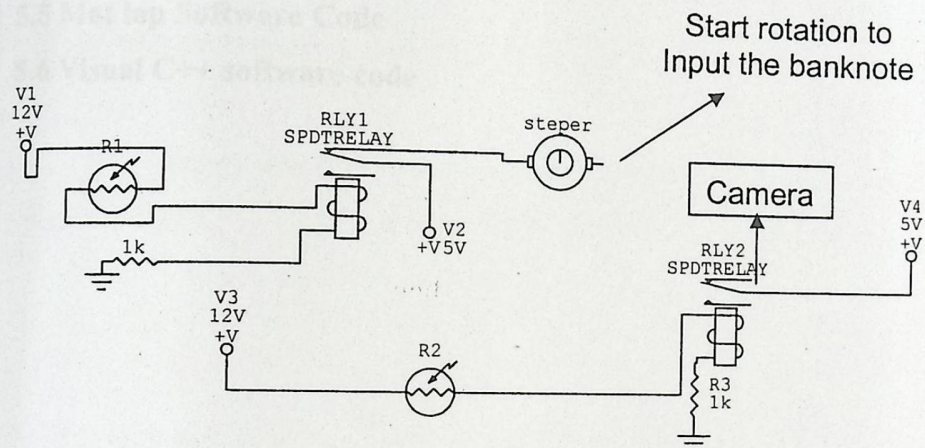


Figure (4.1): banknote depositing

Chapter Five

Software Design

5.1 Overview

5.2 Functional Requirement

5.3 Non Functional Requirement

5.4 Algorithms

5.5 Mat lap Software Code

5.6 Visual C++ software code

5.1 Overview

In Chapter Six: Hardware Design the hardware equipments and their interconnection have been discussed .In this chapter the software subroutines and their functionality are going to be explained in the next sections. The Software functions include writing a program to recognize banknote, print the mobile card serial number.

5.2 Functional Requirements

The functional requirements of the project are:

- Recognizing a Banknote deposited by user.
- Print the Mobile Card Serial Number.

5.3 Algorithms

In this section the main parts of the program are going to be explained. These parts are banknote recognizer; print the mobile card serial number.

5.3.1 Banknote Recognizer

Banknote recognizer consist of two Lamps, two sensors and motor, the check will depends if there found same banknote from data base that like the input banknote if found print the serial mobile card serial number that its price equal the banknote if not found return the banknote to the user to change it.

5.3.2 Motor

This motor will be connected after LDR , if the light of laser lamp break ,that mean there is anew banknote well be entered then the LDR send signal to the motor to start rotation ,the belt start moving to carry banknote to the second LDR, Then second LDR send signal to camera to start capturing.

5.3.3 Camera

If the banknote reaches the second LDR, the second LDR send signal to camera to take picture for the banknote, the captured image goes to PC to start recognition.

5.3.4 Printer

After recognizing the banknote through the software and find equal one in database that mean the banknote is true then the printer print 13 numbers (the mobile card serial number) with equal price of the interred banknote.

5.4 MATLAB Code

We using MATLAB program initially in our project testing, because some image processing classes were built, that easy to use and understand the output of all step in image recognition methods that we used in our project see appendix.

5.5 Visual C++ software code

After MATLAB testing see chapter seven, we choosing the RGB Histogram and Orient Edge detection methods are the best methods for image recognition, then implement these methods in VC++ see appendix.

Chapter Six

Testing

6.1 Software

6.2 Hardware

6.3 IMATLAB

We use IMATLAB to check which method of image recognition method is the best to use in our project.

We put in database 100 pictures for background. 25 for background 10, 25 for background 50 and 25 for background 200. Then we take new picture as input that we put in to test if it is from database or not. Then we take new picture as input that we put in to test if it is from database or not.

If we compare images by subtraction of the matrices for those images but the result is not good to use it in our project because any shift for the image parts will result different subtraction.

We compare images using HSV for recognition. If we take 100 pictures as input for each background the result, see Table 6.1, show these results.

Table 6.1: HSV testing

object	HSV result	Percentage
Background 25	56/100	56%
Background 50	70/100	70%
Background 100	49/100	49%
Background 200	63/100	63%

6.1 Introduction

This chapter talks about the test for our project that includes software and hardware

6.2 Software test:

In our project the testing operation divided to two part software testing and hardware testing.

6.2.1 MATLAB:

We use MATLAB to check which method of image recognition methods is the best to use it in our project.

We put in database 100 pictures(25 for banknote 20,25 for banknote 50,25 for banknote 100 and 25 for banknote 200) then we take new picture as input then run our codes to return four image from data base that's have minimum different of input image.

1- we compare images by subtraction of the matrices for these images but the result is not good to use it in our project because any shift for the image pixels well return different subtraction.

2- we compare images using HSV the recognition, if we take 100 different input images for each banknote the result, table(6.1) show these result testing.

Table (6.1): HSV testing

	object	HSV result	Percentage
1	Banknote 20	56/100	56%
2	Banknote 50	70/100	70%
3	Banknote 100	40/100	40%
4	Banknote 200	65/100	65%

Figure (6.1) show the input banknote, its HSV, and the nearest four images from database.

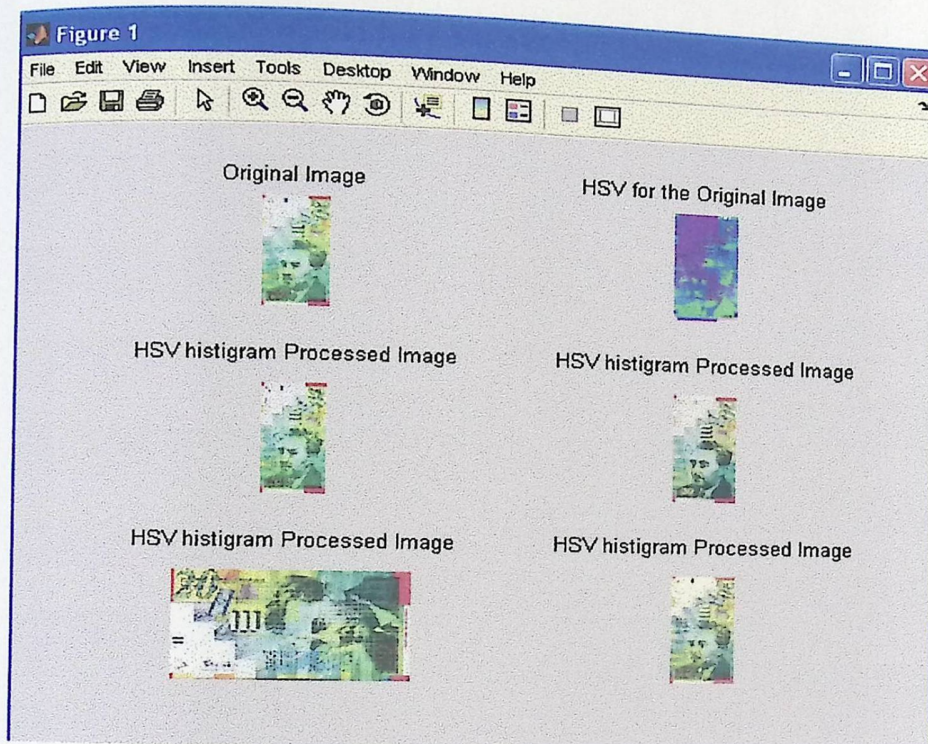


Figure (6.1): the nearest 4 images by using HSV_ histogram

But we need more efficient results.

3 we use gray histogram, figure(6.2) show the gray scale for input image a, table (6.2) show the result testing for the input banknote and the database.

Table (6.2): gray histogram testing

	object	Gray histogram result	percentage
1	Banknote 20	74/100	74%
2	Banknote 50	78/100	78%
3	Banknote 100	80/100	80%
4	Banknote 200	64/100	64%

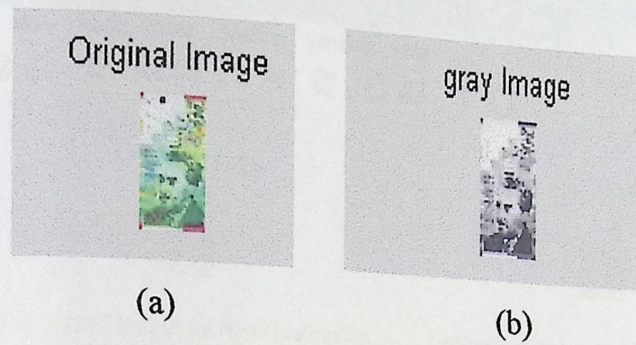


Figure (6.2): a) input image, b) grayscale for image a.

The gray histogram will change for the same image if we take the pictures in different lightness.

4 we use RGB Histogram:

Table (6.3): RGB Histogram testing

	object	RGB histogram result	percentage
1	Banknote 20	93/100	93%
2	Banknote 50	90/100	90%
3	Banknote 100	92/100	92%
4	Banknote 200	89/100	89%

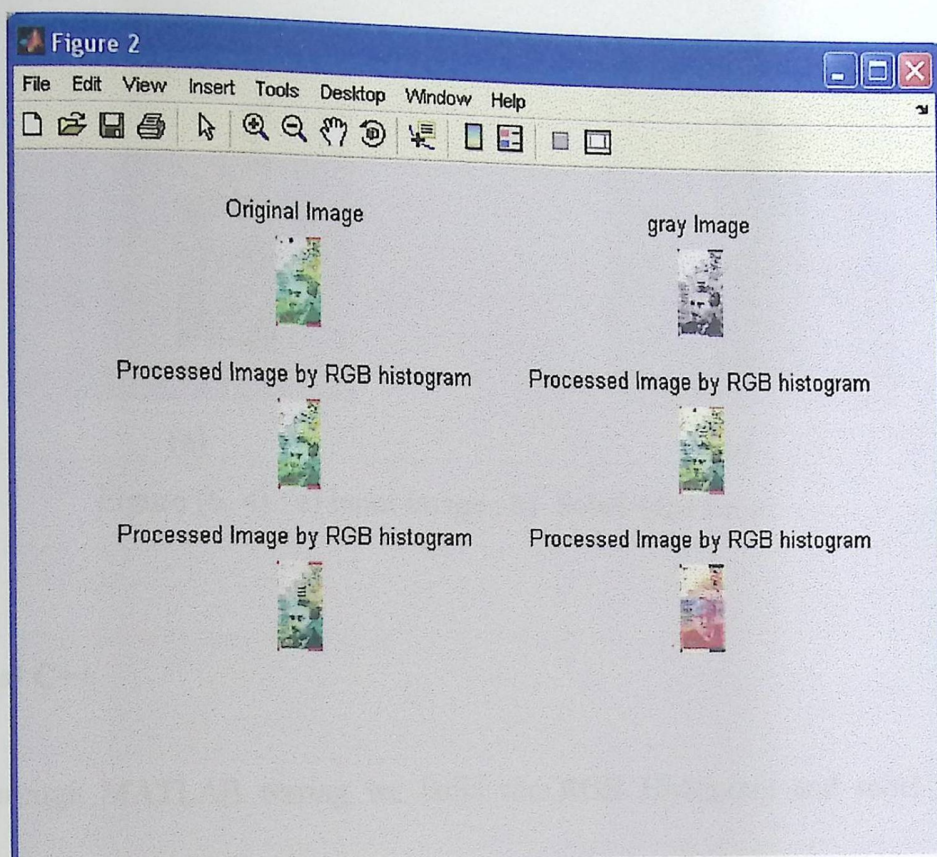


Figure (6.3): the correct 4 image using histogram.

5th we use Sobel edge detection

Table (6.4): Sobel edge detection testing

	object	RGB histogram result	percentage
1	Banknote 20	90/100	90%
2	Banknote 50	89/100	89%
3	Banknote 100	82/100	82%
4	Banknote 200	88/100	88%

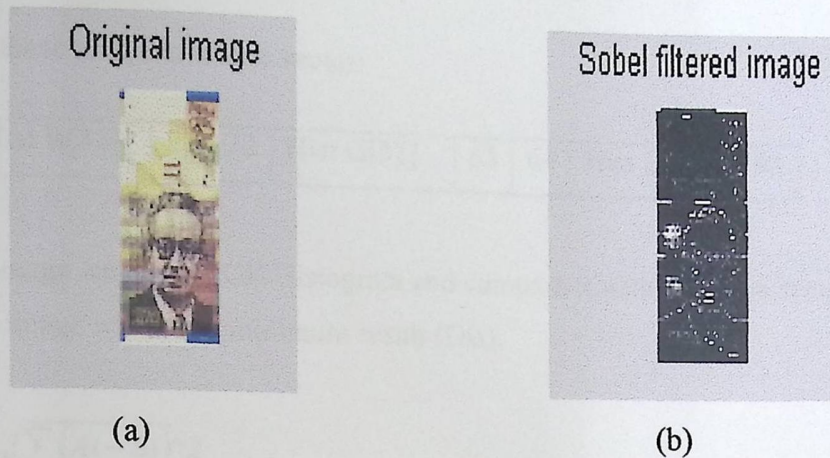


Figure (6. 4) : a) input image , b) Sobel edge for a.

6.2.2 Visual C++

Through MATLAB testing we build the RGB Histogram and sobel edge detection.

The database contain 16 images (4 for banknote 20, 4 for banknote50, 4 for banknote 100 and 4 for banknote 200).

6.2.2.1 RGB Histogram

There are 3 arrays

The maximum color in the image is 255 so the array length is 256 but we convert it to 32 to make processing faster by divided the max color with 8. $[\text{max color} / 8]$.

0.....31 for red histogram.

0.....31 for green histogram.

0.....31 for blue histogram.

We make these arrays in to one vector:

0	Hist R[32]	31	32	Hist G[32]	63	64	Hist B[32]	95
---	------------	----	----	------------	----	----	------------	----

For input image we make RGB Histogram and compare it with database histogram to return one image which has minimum result (Dis).

$$\text{Distance} = \sqrt{\sum (A_i - B_i)^2}$$

6.2.2.2 edge orientation Histogram

We take histogram for theta (orient) that done in Sobel edge detection.

The orientation between 0.....360 degree, but we take it from 0 to 35 by [orient/10]

And do the same in RGB histogram for input image to return equal image from DB.

6.2.2.3 RGB Histogram and orientation Histogram

We take RGB histogram and Orient histogram in one vector:

0	RGB[96]	95	96	Hist orient[36]	131
---	---------	----	----	-----------------	-----

And do the same in RGB histogram for input image to return equal image from DB.

If we take 100 different pictures for the same kind (100 for bank note 20,100 for banknote 50,100 for banknote 100 and 100 for banknote 200) and compare them with database, the testing result:

Table (6.5): VC++ codes testing

	method	Banknote 20	Banknote 50	Banknote 100	Banknote 200
1	RGB Histogram	94/100 (94%)	93/100 (93%)	91/100 (91%)	96/100 (90%)
2	Orient Histogram	92/100 (92%)	90/100 (90%)	91/100 (91%)	92/100 (92%)
3	RGB & orientation histogram	96/100 (96%)	92/100 (92%)	91/100 (91%)	95/100 (95%)

6.3 Hardware Test

This section discusses the steps we followed to test our design the circuits we implement during developing this project.

6.3.1 LDR

LDR are being used to know is there a new input banknote and to start camera capturing.

6.3.2 Stepper motor

The motor was chosen to implement the banknote depositing were stepper motor which is very easy to use and control, the stepper motor used, provides enough torque which has ability to belt that carry banknote inside.

The stepper motor was tested by connecting 12V.

6.3.3 The cabinet box

The cabinet box was implemented with the dimensions and shape in the chapter of hardware design; it was constructed in aluminum shop.

6.3.4 Camera

The camera used to take a picture of input banknote and send it to software algorithms by parallel port.

When the light of the second photo diode was broken, the second LDR send signal to camera to start capturing

Chapter Seven

Conclusion and Future Work

7.1 Conclusions

7.2 Future Work

7.1 Conclusions:

- The main difficulty to implement this project is to detect the incorrect banknote.
- It was decided to depend on the kind of paper which banknote made by, size and drawing on the banknote but because of not getting the sensor we depend on the size, drawing and colors.
- Vending machine is a full automated machine facilitates the life of the human in different aspects.

7.2 Future Work:

- Because we haven't found a sensor for sensing the kind of paper which banknote made the banknote recognizer unit has been implemented depending on the size, colors and drawing which banknote have by using software processing algorithms.
- Return reminder to user if the banknote large that card price.
- Change the PC by using Microcontrollers.
- Sell the products cards rather than printing the serial number of mobile cards.

References

- [1] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins, Digital image using MATLAB.
- [2] Rafael C. Gonzalez and Richard E. Woods, Digital image Processing.
- [3] <http://www.railway-technical.com/relay.gif>.
- [4] Bilal Tamimi & Motaz Tamimi, "Candy Bar Vending Machine ", Palestine polytechnic university.
- [5] <http://www.mikroe.com/en/books/8051book/02.htm>.
- [6] Russ J.K. [1999]. "The Image Processing Handbook".
- [7] Ion Boldea And S. A. Nasar, Electric Drives
- [8] <http://www.cs.cornell.edu/home/rdz/ccv.html>.
- [9] <http://www.sphoto.com>.
- [10] <http://wang1.ist.psu.edu>.
- [11] http://www.pages.drexel.edu/~weg22/can_tut.html.
- [12] http://www.makingthings.com/resources/datasheets/acc_pht_001.pdf
- [13] <http://www.technologystudent.com/elec1/ldr1.htm///LDR>

APPENDIX A

Mat-lab codes

RGB and Gray Histogram

```
%clear all;
function [x] = images2hist(file_name)
load('all_img_hist5.mat')

img = imread('f:\p.jpg');
figure(1)
subplot(1,2,1)
imshow(img)
title('Original Image');

r_hist = imhist(img(:,:,1));
g_hist = imhist(img(:,:,2));
b_hist = imhist(img(:,:,3));
hist_img = [r_hist;g_hist;b_hist];

for j=1:13,
    img_def = s(j).hist_img - hist_img;
    img_def1 = sqrt(sum(img_def.^2));

    a(j)=img_def1;

end
y=sort(a)
%w= a(:,13);
whos y;

for k=1:4,
    for j=1:13,
        if (y(k)==a(j))
            p=(s(j).location);
            figure(2)
            subplot(2,2,k)
            figure(2), imshow(p);
            title('Processed Image')

        end
    end
end
end
```


database histogram

```
function [x] = save_hist ()
s = struct( 'img_index', {} , 'location' , {} , 'hist_img', {} )
av_files = dir('c:\pics\*.jpg')
for j=1:13,
    s(j).img_index = j;
    s(j).location = ['c:\pics\' av_files(j).name] ;
    img1 = imread(s(j).location);
    r_hist = imhist(img1(:, :, 1));
    g_hist = imhist(img1(:, :, 2));
    b_hist = imhist(img1(:, :, 3));
    s(j).hist_img = [r_hist;g_hist;b_hist];
end
save all_img_hist5.mat s;
```

HSV

```
%clear all;
function [x] = images2HSV(file_name)
load('all_IMG_hist6.mat')

IMG = imread ('c:\f.jpg');
RGB = IMG;
HSV=rgb2hsv(RGB);
H=HSV(:, :, 1);
S=HSV(:, :, 2);
V=HSV(:, :, 3);
H_hist = imhist(H);
S_hist = imhist(S);
V_hist = imhist(V);
hist_HSV = [H_hist;S_hist;V_hist];
%imshow(hist_HSV)
%figure, imshow(HSV);
figure, imshow(RGB);
%HSV_logo = [H;S;V];
%figure, imshow(HSV_logo);
% HSV FOR ALL DB & NEW INPUT

% make histogram for the hsvc image
% and the distance for the histograms
for j=1:13,
    IMG_def = [s(j).hist_HSV - hist_HSV];
    IMG_def1 = sqrt(sum(IMG_def.^2));
    %-----
    a(j)=IMG_def1;
end
```



```

y=sort(a)
%w= a(:,13);
whos y;

for k=1:4,
for j=1:13,
    if (y(k)==a(j))

figure,imshow(s(j).location);

        end
    end
end

%-----
% to display four pics that is around the input pic

```

Database HSV

```

function [x] = save_HSV ()
s = struct( 'IMG_index',{ } , 'location' , { } , 'hist_HSV' , { } , 'def_hist',{ } )
av_files = dir('c:\pics\*.jpg')
for j=1:13,
    s(j).IMG_index = j;
    s(j).location = ['c:\pics\' av_files(j).name] ;
    % s(j).location1 = ['c:\pics\' av_files(j).name] ;
    RGB = imread(s(j).location);
    HSV = rgb2hsv(RGB);
    HSV=rgb2hsv(RGB);
    H=HSV(:,:,1);
    S=HSV(:,:,2);
    V=HSV(:,:,3);
    H_hist = imhist(H);
    S_hist = imhist(S);
    V_hist = imhist(V);
    s(j).hist_HSV = [H_hist;S_hist;V_hist];
end
save all_IMG_hist6.mat s;

```

Sobel edge detection and orient histogram

```

function [imnew1,edge]=sobcas(a)
load('all_IMG_edg.mat')
a=imread('c:\e.jpg');
figure(1)

```



```

subplot(1,2,1)
imshow(a)
title('Original Image');
[m1,n1,p1]=size(a);
img1=a;
imnew1=zeros(m1,n1);
edge=zeros(m1,n1);

s1=[1 2 1;0 0 0;-1 -2 -1];
s2=[ -1 0 1;-2 0 2;-1 0 1];
for i=2:m1-1,
    for j=2:n1-1,
        mat=[img1(i-1,j-1) img1(i-1,j) img1(i-1,j+1);
              img1(i,j-1) img1(i,j) img1(i,j+1);
              img1(i+1,j-1) img1(i+1,j) img1(i+1,j+1)];

        sum1=double(mat).*s1;
        sumx=sum(sum(sum1));
        sum2=double(mat).*s2;
        sumy=sum(sum(sum2));
        %orient calculation
        if sumx == 0
            if sumy == 0
                orient=0.0;
            elseif sumy < 0
                sumy = -sumy;
                orient = 90.0;
            else
                orient=90.0;
            end
        elseif sumx<0 & sumy>0
            sumx = -sumx;
            orient = 180 - ((atan(sumy/sumx)) * (180/pi));
        elseif sumx>0 & sumy<0
            sumy = -sumy;
            orient = 180 - ((atan(sumy/sumx)) * (180/pi));
        else
            orient = (atan(sumy/sumx)) * (180/pi);
        end

        %edge direction
        if(orient < 22.5) edge(i,j) = 0;
        elseif(orient < 67.5) edge(i,j) = 45;

    elseif(orient < 112.5) edge(i,j) = 90;

```



```

elseif(orient < 157.5) edge(i,j) = 135;
else edge(i,j) = 0;
end
sum1=abs(sumx)+abs(sumy);
if sum1>255
    sum1=255;
elseif sum1<0
    sum1=0;
end
imnew1(i,j)=sum1;
j=j+1;
end
i=i+1;
end
for j=1:13,
    IMG_def = [s(j).EDG - orient];
    IMG_def1 = sqrt(sum(IMG_def.^2));
    a(j)=IMG_def1;

    end
y=sort (a)
%w= a(:,13);
whos y;

    for k=1:4,
        for j=1:13,
            if (y(k)==a(j))

                p=(s(j).location);
                figure(2)
                subplot(2,2,k)
                figure(2), imshow(p);
                title('Processed Image')
            end

        end

    end
end
End

```

Sobel for database

```

f=imread('c:\e.jpg');
mask = roipoly(f, :, :);
red = immultiply(mask, f(:, :, 1));
green = immultiply(mask, f(:, :, 2));

blue = immultiply(mask, f(:, :, 3));

```



```

g = cat(3, red, green, blue);
figure, imshow(g)
figure, imshow(f)
[m, n, k] = size(g);
i = reshape(g, m*n, 3);
idx = find(mask);
i = double(i(idx, 1:3));
[c, m] = covmatrix(i);
d=diag(c);
sd=sqrt(d);
E25 = colorseg('euclidean',f,25,m);

```

VC++ CODES

Camera codes

```

#include <vfw.h>
#include <winuser.h>
#include <Windows.h>
#include <stdio.h>
hWndC = capCreateCaptureWindow("Capture Window",
WS_CHILD /* WS_VISIBLE
/*WS_THICKFRAME*/WS_DLGFAME,
m_rectFrame.TopLeft().x, m_rectFrame.TopLeft().y,
m_rectFrame.Width(), m_rectFrame.Height(),
GetSafeHwnd()/*hwndParent*/, 1);

capDriverConnect (hWndC, 0);//connect to cammera's driver.
capDlgVideoSource(hWndC);//pick the source camera.
capDlgVideoFormat(hWndC); //to change the resolution.
capDriverDisconnect(hWndC);CString FileName;
FileName="C:/pic/A.bmp";
//getting user's string.
char file_str[30]={". "};
for(int i=0;i<FileName.GetLength();i++)
{
file_str[i]=FileName[i];
}
file_str[i]=NULL;

bmp2ppm(file_str,m_hBmp);

```



```
SaveToFile(file_str,m_hBmp,dcMem);
```

RGB Histogram

```
int* histf(pgm* image6)
{
    int colorr;
    int colorg;
    int colorb;
    int histr[32];
    int histg[32];
    int histb[32];
    //int histrgb[768];
    image6->histrgb = (int*)malloc(sizeof(int) * 96);
    for(int i=0;i<32;i++)
    {
        histr[i]=0;
        histg[i]=0;
        histb[i]=0;
    }
    for(int i=0;i<image6->w * image6->h;i++)
    {

        colorr= image6->R[i];
        colorg= image6->G[i];
        colorb= image6->B[i];
        histr[colorr / ( image6->max / 32 )]++;
        histg[colorg / ( image6->max / 32 )]++;
        histb[colorb / ( image6->max / 32 )]++;
    }
}
```

Sobel Edge orient histogram

```
#include "stdafx.h"
#include <mscorlib.dll>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#include <math.h>
```



```

#include <cmath>
# define kW 3
# define kH 3
# define factor 1
#define MAX(a,b) a < b ? b : a
#define MIN(a,b) a > b ? b : a
#define t 5
// #define w xxxxx
// #define h xxxxx
#include <fstream>
#include <iostream>
using namespace std;
using namespace System;
typedef struct pgm {
    int w;
    int h;
    int max;
    int* R;
    int* G;
    int* B;
    int* pData2;
    //int* pData; ///distance between dx & dy
    double* pData1;/// oriantation
    int* Hisgram;///histogram fore oriantation
    int* histrgb;///histogram for RGB
} pgm;

typedef struct diff {
    int r[6];
    int r1[6];
    int name[6];
} diff;

void Loadppm( char* filename, pgm* pPpm ) {
    FILE* ifp;
    int word;
    int nRead = 0;
    char readChars[256];

    //open the file, check if successful
    ifp = fopen( filename, "r" );

    if (!ifp) {

```



```

printf("Error: Unable to open file %s.\n\n", filename);
exit(1);
}

//read headers from file
printf("Reading PPM file: %s...\n", filename);
fscanf(ifp, "%s", readChars);

if(strcmp(readChars, "P3") == 0) {
    //valid file type
    //get a word from the file
    fscanf(ifp, "%s", readChars);
    /* while (readChars[0] == '#') {
        //if a comment, get the rest of the line and a new word
        fgets(readChars, 255, ifp);
        fscanf(ifp, "%s", readChars);
    }*/

    //ok, comments are gone
    //get width, height, color depth
    sscanf(readChars, "%d", &pPpm->w);
    fscanf(ifp, "%d", &pPpm->h);
    fscanf(ifp, "%d", &pPpm->max);
    // allocate some memory, note that on the HandyBoard you want to
    // use constant memory and NOT use calloc/malloc
    pPpm->pData2 = (int*)malloc(sizeof(int) * pPpm->w * pPpm->h);
    pPpm->R = (int*)malloc(sizeof(float) * pPpm->w * pPpm->h);
    pPpm->B = (int*)malloc(sizeof(float) * pPpm->w * pPpm->h);
    pPpm->G = (int*)malloc(sizeof(float) * pPpm->w * pPpm->h);

    // now read in the image data itself
    for (nRead = 0; nRead < pPpm->w * pPpm->h; nRead++)
    {
        fscanf(ifp, "%d", &word);
        pPpm->R[nRead] = word;

        fscanf(ifp, "%d", &word);
        pPpm->G[nRead] = word;

        fscanf(ifp, "%d", &word);
        pPpm->B[nRead] = word;

        pPpm->pData2[nRead] = 0.299*pPpm->R[nRead] + 0.587*pPpm->G[nRead] + 0.114*pPpm->B[nRead];
    }
}

```



```

//pPpm->pData[nRead]=pPpm->R[nRead]+pPpm->G[nRead]+pPpm->B[nRead]/3;
}
printf("Loaded PPM. Size: %dx%d, Greyscale: %d\n\n",pPpm->w, pPpm->h,
pPpm->max + 1);
}
else {
printf("Error: Read file type %s. Format unsupported.\n\n", readChars);
exit(1);
}

fclose(ifp);
}

```

```

pgm* Convolution(pgm *image,int kernel[3][3])
{
int ix, iy, kx, ky;
int imageX=0, imageY=0, imageW, imageH;
float sum=0.0;

```

```

pgm *result = (pgm *) malloc (sizeof(pgm));
result->pData2 = (int*)malloc(sizeof(int) * image->w * image->h);

```

```

imageH=result->h = image->h;
imageW=result->w = image->w;
result->max = image->max;

```

```

for (ix=0; ix< imageH; ix++)
for (iy=0; iy< imageW; iy++)
{
sum = 0.0;
for (kx=0; kx<kH; kx++)
for (ky=0; ky<kW; ky++)
{
imageX = (ix - kH / 2 + kx + imageH) % imageH;
imageY = (iy - kW / 2 + ky + imageW) % imageW;
sum = sum + image->pData2[(imageX * imageW) + imageY] *
kernel[kx][ky];
}
result->pData2[ix*imageW + iy] = MIN(MAX( ((int) sum*factor) ,0),255);
// MIN and MAX are defined at the start of this program
}
return result;
}

```



```

pgm* Edgedetection( pgm* pPgm1,pgm* pPgm2)
{
    pgm* pPbm1;
    //int lowThreshold = 20;
    edgeDirection;    int
    highThreshold=180;    int
    lowThreshold=70;    int
    // Edge orientation    ORIENT=0;    int
    double distance;
    pPbm1 = (pgm *) malloc (sizeof(pgm));
    pPbm1->h=pPgm1->h;
    pPbm1->w=pPgm1->w;
    //pPbm1->pData = (int*)malloc(sizeof(int) * pPgm1->w * pPgm1->h);
    pPbm1->pData1 = (double*)malloc(sizeof(double) * pPgm1->w * pPgm1->h);
    for (int n = 0; n < pPgm1->w * pPgm1->h; n++)
    {
        distance = sqrt( (double)(pPgm1->pData2[n])*(pPgm1->pData2[n])+(pPgm2->pData2[n])*(pPgm2->pData2[n]) );
        if(distance >= 60 )
            ORIENT = (atan((double)(pPgm2->pData2[n])/(double)(pPgm1->pData2[n]))) * (180/3.14);
        else
            ORIENT=0;

        /*ORIENT = 0.0;

        else if(pPgm1->pData2[n] == 0 && pPgm2->pData2[n]<0)
            ORIENT = 90.0;
        else if(pPgm1->pData2[n] == 0 && pPgm2->pData2[n]>0)
            ORIENT = 90.0;
        else if(pPgm1->pData2[n]<0 && pPgm2->pData2[n]>0)
            ORIENT = 180 - ((atan((double)(pPgm2->pData2[n])/(double)(pPgm1->pData2[n]))) * (180/3.14));
        else if(pPgm1->pData2[n]>0 && pPgm2->pData2[n]<0)
            ORIENT = 180 - ((atan((double)(pPgm2->pData2[n])/(double)(pPgm1->pData2[n]))) * (180/3.14));
        else
            ORIENT = (atan((double)(pPgm2->pData2[n])/(double)(pPgm1->pData2[n]))) * (180/3.14);
        */

        pPbm1->pData1[n]=ORIENT;

    }
}

```



```

    return pPbm1;
}
int* Hgram( pgm *img1 )
{

    int indx ,i , mse =0 ;
    img1->Hisgram = (int*)malloc(sizeof(int) * 32);

    for( i=0 ; i< 32; i++ )
        img1->Hisgram[i] = 0;// to set zero
    for( i=0 ; i< img1->h*img1->w; i++ )
    {
        indx = img1->pData1[i];
        img1->Hisgram[ indx/( img1->max / 32 ) ] += 1;
    }
    return img1->Hisgram;
}

```

```

int* histf(pgm* image6)
{
    int colorr;
    int colorg;
    int colorb;
    int histr[32];
    int histg[32];
    int histb[32];
    //int histrgb[768];
    image6->histrgb = (int*)malloc(sizeof(int) * 96);
    for(int i=0;i<32;i++)
    {
        histr[i]=0;
        histg[i]=0;
        histb[i]=0;
    }
    for(int i=0;i<image6->w * image6->h;i++)
    {

        colorr= image6->R[i];
        colorg= image6->G[i];
        colorb= image6->B[i];
        histr[colorr / ( image6->max / 32 )]++;

        histg[colorg / ( image6->max / 32 )]++;
    }
}

```



```
histb[colorb / ( image6->max / 32 )]++;
```

```
}  
for(int i=0;i<32;i++)  
image6->histrgb[i]=histr[i];  
for(int i=32;i<64;i++)  
image6->histrgb[i]=histg[i-32];  
for(int i=64;i<96;i++)  
image6->histrgb[i]=histb[i-64];
```

```
return image6->histrgb;  
}
```

```
int _tmain()
```

```
{  
    // TODO: Please replace the sample code below with your own.
```

```
    diff* diff1;
```

```
    diff1 = (diff *) malloc (sizeof(diff));
```

```
    pgm* image1, *image2,*image3;
```

```
    pgm* image4 ;
```

```
    int* image5;
```

```
    int* image6;
```

```
    int a = 65;
```

```
    int num=0,num1 = 0;
```

```
    int kernel1[ kW][ kH] ={-1,0,1,-2,0,2,-1,0,1};
```

```
    int kernel2[ kW][ kH] ={1, 2, 1, 0, 0, 0,-1,-2,-1};
```

```
    char filename1[]="c:/pic/A.ppm";
```

```
    int i2=0;
```

```
    for(int i=1;i<=6;++i)
```

```
{  
    filename1[7]=a++; -
```

```
    //allocate memory for the pgm struct
```

```
    image1 = (pgm *) malloc (sizeof(pgm));
```

```
    //read the file
```

```
        Loadppm(filename1, image1);
```

```
    //dx //Convolution
```

```
    image2=Convolution(image1,kernel1);
```

```
    //dy //Convolution
```

```
    image3=Convolution(image1,kernel2);
```

```
    //Edgedetection
```

```
    image4=Edgedetection( image2,image3);
```

```
    //HistoGram
```

```
    image5=Hgram(image4);
```



```

image6=histf(image1);
//////////write to file //////////
/*ofstream Histo("ady.txt", ios::app);
Histo << ++i2;
for (int as=0;as<32;++as)
Histo << "\t"<< image5[as];
for (int as=32;as<128;++as)
Histo << "\t"<< image6[as-32];
Histo << "\n";*/

```

```

//////////
//free(image1);           //to free memory
//////////
int xxx;
int img[32];
int img1[96];
int result1;
int result11;

ifstream Histo("ady.txt");
//ifstream HistoRGB("ady1.txt");
for(int aqq=0;aqq<=5;aqq++)
{
Histo >> xxx;           //write PIC # from DB
for (int as1=0;as1<32;++as1)
Histo >> img[as1];      //write data from DB
for (int as1=0;as1<96;++as1)
Histo >> img1[as1];     //write data from DB

```

```

result1=0;result11=0;
for(int za=0;za<32;za++)
{
result1+= ( (image5[za]-img[za])*(image5[za]-img[za])
); // find the distance between input PIC & all PIC in DB
}
for(int za=0;za<96;za++)
{
result11+= ( (image6[za]-img1[za])*(image6[za]-
img1[za]) ); // find the distance between input PIC & all PIC in DB
}
result1=abs(result1); //find absolute value
result1=sqrt((double)result1); // find square root

```



```

result11=abs(result11); //find absolute value
result11=sqrt((double)result11); // find square root
diff1->r[aaq] =result1;
diff1->r1[aaq] =result11;
diff1->name[aaq] =xxx;
}
int min=100000,names=0,min1=100000;

////////////////////////////////////

printf("\nEdge PIC# %d\n",++num);

for(int minn=0;minn<6;minn++)
printf("%d ",diff1->r[minn]);

for(int aa=0;aa<6 ;++aa)
{
if (diff1->r[aa]< min)
{
min = diff1->r[aa];
names = diff1->name[aa];
}
}

if (names >= 1 && names <= 2 )
Console::WriteLine(S"\nthe banknote is 20!!\n");
else if (names>=3 && names<=4)
Console::WriteLine(S"\nthe banknote is 50!!\n");
else if (names>=5 && names<=6)
Console::WriteLine(S"\nthe banknote is 100!!\n");
else
Console::WriteLine(S"bad banknote!!");
////////////////////////////////////
printf("\nHist RGB for PIC %d\n",++num1);
for(int minn=0;minn<6;minn++)
printf("%d ",diff1->r1[minn]);
for(int aa=0;aa<6 ;++aa)
{
if (diff1->r1[aa]< min1)
{
min1 = diff1->r1[aa];
names = diff1->name[aa];
}
}

}

```



```

if (names >= 1 && names <= 2 )
    Console::WriteLine(S"\nthe banknote is 20!!\n");
else if (names>=3 && names<=4)
    Console::WriteLine(S"\nthe banknote is 50!!\n");
else if (names>=5 && names<=6)
    Console::WriteLine(S"\nthe banknote is 100!!\n");
else
    Console::WriteLine(S"bad banknote!!");
}
//////////*/

```

```

Console::WriteLine(S"DONE..!!");
for(int af=0 ;af<5;) {}

```

```

//////////
return 0;

```

```

}

```

RGB and Orient Histogram

```

#include "stdafx.h"
#include <mscorlib.dll>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <cmath>
# define kW 3
# define kH 3
# define factor 1
#define MAX(a,b) a < b ? b : a
#define MIN(a,b) a > b ? b : a
#define t 5
// #define w xxxxx
// #define h xxxxx
#include <fstream>
#include <iostream>
using namespace std;
using namespace System;

typedef struct pgm {

```

```

int w;

```



```

int h;
int max;
int* R;
int* G;
int* B;
int* pData2;
//int* pData; ///distance between dx & dy
double* pData1;/// oriantation
int* Hisgram;///histogram fore oriantation
int* histrgb;///histogram for RGB
} pgm;

```

```

typedef struct diff {
int r[6];
int r1[6];
int name[6];
} diff;

```

```

void Loadppm( char* filename, pgm* pPpm ) {
FILE* ifp;
int word;
int nRead = 0;
char readChars[256];

```

```

//open the file, check if successful
ifp = fopen( filename, "r" );
if (!ifp) {
printf("Error: Unable to open file %s.\n\n", filename);
exit(1);
}

```

```

//read headers from file
printf ("Reading PPM file: %s...\n", filename);
fscanf (ifp, "%s", readChars);

```

```

if (strcmp(readChars, "P3") == 0) {
//valid file type
//get a word from the file
fscanf (ifp, "%s", readChars);
/* while (readChars[0] == '#') {
//if a comment, get the rest of the line and a new word
fgets (readChars, 255, ifp);

```

```

fscanf (ifp, "%s", readChars);

```



```

    */

    //ok, comments are gone
    //get width, height, color depth
    sscanf(readChars, "%d", &pPpm->w);
    fscanf(ifp, "%d", &pPpm->h);
    fscanf(ifp, "%d", &pPpm->max);
    // allocate some memory, note that on the HandyBoard you want to
    // use constant memory and NOT use calloc/malloc
    pPpm->pData2 = (int*)malloc(sizeof(int) * pPpm->w * pPpm->h);
    pPpm->R = (int*)malloc(sizeof(float) * pPpm->w * pPpm->h);
    pPpm->B = (int*)malloc(sizeof(float) * pPpm->w * pPpm->h);
    pPpm->G = (int*)malloc(sizeof(float) * pPpm->w * pPpm->h);

    // now read in the image data itself
    for (nRead = 0; nRead < pPpm->w * pPpm->h; nRead++)
    {
        fscanf(ifp, "%d", &word);
        pPpm->R[nRead] = word;

        fscanf(ifp, "%d", &word);
        pPpm->G[nRead] = word;

        fscanf(ifp, "%d", &word);
        pPpm->B[nRead] = word;

        pPpm->pData2[nRead] = 0.299*pPpm->R[nRead] + 0.587*pPpm->G[nRead] + 0.114*pPpm->B[nRead];
        //pPpm->pData[nRead] = pPpm->R[nRead] + pPpm->G[nRead] + pPpm->B[nRead]/3;
    }
    printf("Loaded PPM. Size: %dx%d, Greyscale: %d \n\n", pPpm->w, pPpm->h,
    pPpm->max + 1);
    }
    else {
        printf("Error: Read file type %s. Format unsupported.\n\n", readChars);
        exit(1);
    }

    fclose(ifp);
}

pgm* Convolution(pgm *image, int kernel[3][3])
{

```



```

int ix, iy, kx, ky;
int imageX=0, imageY=0, imageW, imageH;
float sum=0.0;

```

```

pgm *result = (pgm *) malloc (sizeof(pgm));
result->pData2 = (int*)malloc(sizeof(int) * image->w * image->h);

```

```

imageH=result->h = image->h;
imageW=result->w = image->w;
result->max = image->max;

```

```

for (ix=0; ix< imageH; ix++)
for (iy=0; iy< imageW; iy++)
{
    sum = 0.0;
    for (kx=0; kx<kH; kx++)
        for (ky=0; ky<kW; ky++)
        {
            imageX = (ix - kH / 2 + kx + imageH) % imageH;
            imageY = (iy - kW / 2 + ky + imageW) % imageW;
            sum = sum + image->pData2[(imageX * imageW) + imageY] *
            kernel[kx][ky];
        }
    result->pData2[ix*imageW + iy] = MIN(MAX( ((int) sum*factor) ,0),255);
    // MIN and MAX are defined at the start of this program
}
return result;
}

```

```

pgm* Edgedetection( pgm* pPgm1,pgm* pPgm2)
{
    pgm* pPbm1;
    //int lowThreshold = 20;
    edgeDirection;    int
    highThreshold=180;    int
    lowThreshold=70;    int
    // Edge orientation    ORIENT=0;    int
    double distance;
    pPbm1 = (pgm *) malloc (sizeof(pgm));
    pPbm1->h=pPgm1->h;
    pPbm1->w=pPgm1->w;
    //pPbm1->pData = (int*)malloc(sizeof(int) * pPgm1->w * pPgm1->h);
    pPbm1->pData1 = (double*)malloc(sizeof(double) * pPgm1->w * pPgm1->h);
}

```



```

for (int n = 0; n < pPgm1->w * pPgm1->h; n++)
{
    distance = sqrt( (double)(pPgm1->pData2[n])*(pPgm1->pData2[n])+(pPgm2-
>pData2[n])*(pPgm2->pData2[n]) );
    if(distance >= 60 )
    ORIENT = (atan((double)(pPgm2->pData2[n])/(double)(pPgm1-
>pData2[n]))) * (180/3.14);
    else
    ORIENT=0;

```

```

/*ORIENT = 0.0;

```

```

else if(pPgm1->pData2[n] == 0 && pPgm2->pData2[n]<0)
ORIENT = 90.0;
else if(pPgm1->pData2[n] == 0 && pPgm2->pData2[n]>0)
ORIENT = 90.0;
else if(pPgm1->pData2[n]<0 && pPgm2->pData2[n]>0)
ORIENT = 180 - ((atan((double)(pPgm2-
>pData2[n])/(double)(pPgm1->pData2[n]))) * (180/3.14));
else if(pPgm1->pData2[n]>0 && pPgm2->pData2[n]<0)
ORIENT = 180 - ((atan((double)(pPgm2-
>pData2[n])/(double)(pPgm1->pData2[n]))) * (180/3.14));
else
    ORIENT = (atan((double)(pPgm2->pData2[n])/(double)(pPgm1-
>pData2[n]))) * (180/3.14);
*/

```

```

pPbm1->pData1[n]=ORIENT;

```

```

}

```

```

return pPbm1;

```

```

}

```

```

int* Hgram( pgm *img1 )

```

```

{

```

```

int indx ,i , mse =0 ;

```

```

img1->Hisgram = (int*)malloc(sizeof(int) * 32);

```

```

for( i=0 ; i< 32; i++ )

```

```

img1->Hisgram[i] = 0;// to set zero

```

```

for( i=0 ; i< img1->h*img1->w; i++ )

```

```

{

```

```

indx = img1->pData1[i];

```



```
img1->Hisgram[ indx/( img1->max / 32 ) ] += 1;
```

```
}  
return img1->Hisgram;  
}
```

```
int* histf(pgm* image6)
```

```
{  
    int colorr;  
    int colorg;  
    int colorb;  
    int histr[32];  
    int histg[32];  
    int histb[32];  
    //int histrgb[768];  
    image6->histrgb = (int*)malloc(sizeof(int) * 96);  
    for(int i=0;i<32;i++)  
    {  
        histr[i]=0;  
        histg[i]=0;  
        histb[i]=0;  
    }  
    for(int i=0;i<image6->w * image6->h;i++)  
    {
```

```
        colorr= image6->R[i];  
        colorg= image6->G[i];  
        colorb= image6->B[i];  
        histr[colorr / ( image6->max / 32 )]++;  
        histg[colorg / ( image6->max / 32 )]++;  
        histb[colorb / ( image6->max / 32 )]++;
```

```
    }  
    for(int i=0;i<32;i++)  
        image6->histrgb[i]=histr[i];  
    for(int i=32;i<64;i++)  
        image6->histrgb[i]=histg[i-32];  
    for(int i=64;i<96;i++)  
        image6->histrgb[i]=histb[i-64];
```

```
    return image6->histrgb;  
}
```



```

int _tmain()
{
    // TODO: Please replace the sample code below with your own.
    diff* diff1;
    diff1 = (diff *) malloc (sizeof(diff));
    pgm* image1, *image2,*image3;
    pgm* image4 ;
    int* image5;
    int* image6;
    int a = 65;
    int num=0,num1 = 0;
    int kernel1[ kW][ kH] ={-1,0,1,-2,0,2,-1,0,1};
    int kernel2[ kW][ kH] ={1, 2, 1, 0, 0, 0,-1,-2,-1};
    char filename1[]="c:/pic/A.ppm";
    int i2=0;
    for(int i=1;i<=18;++i)
    {
        filename1[7]=a++;
        //allocate memory for the pgm struct
        image1 = (pgm *) malloc (sizeof(pgm));
        //read the file
                                Loadppm(filename1, image1);

        //dx //Convolution
        image2=Convolution(image1,kernel1);
        //dy //Convolution
        image3=Convolution(image1,kernel2);
        //Edgedetection
        image4=Edgedetection( image2,image3);
        //HistoGram
        image5=Hgram(image4);
        image6=histf(image1);
        ////////////write to file //////////
        /*ofstream Histo("ady.txt", ios::app);
        Histo << ++i2;
        for (int as=0;as<32;++as)
        Histo << "\t"<< image5[as];
        for (int as=32;as<128;++as)
        Histo << "\t"<< image6[as-32];
        Histo << "\n";*/
    }
}

```

```

//////////

```



```
//free(image1); //to free memory
////////////////////
```

```
int xxx;
int img[128];
int img1[96];
int result1;
int result11;
int im[128];
for(int x1=0;x1<32;x1++)
im[x1]=image5[x1];
for(int x2=32;x2<128;++x2)\
im[x2]=image6[x2-32];
```

```
ifstream Histo("ady.txt");
//ifstream HistoRGB("ady1.txt");
for(int aqq=0;aqq<=5;aqq++)
{
Histo >> xxx; //write PIC # from DB
for (int as1=0;as1<128;++as1)
Histo >> img[as1]; //write data from DB
//for (int as1=0;as1<96;++as1)
//Histo >> img1[as1]; //write data from DB
```

```
result1=0;
//result11=0;
for(int za=0;za<128;za++)
{
result1+= ( (im[za]-img[za])*(im[za]-img[za]) ); // find
the distance between input PIC & all PIC in DB
}
//for(int za=0;za<96;za++)
//{
result11+= ( (image6[za]-img1[za])*(image6[za]-
img1[za]) ); // find the distance between input PIC & all PIC in DB
//}
result1=abs(result1); //find absolute value
result1=sqrt((double)result1); // find square root
//result11=abs(result11); //find absolute value
//result11=sqrt((double)result11); // find square root
diff1->r[aqq] =result1;
```



```
//diff1->r1[aqq] =result11;  
diff1->name[aqq] =xxx;
```

```
}  
int min=100000,names=0,min1=100000;
```

```
////////////////////////////////////
```

```
printf("\nEdge PIC# %d\n",++num);
```

```
for(int minn=0;minn<6;minn++)  
printf("%d ",diff1->r[minn]);
```

```
for(int aa=0;aa<6 ;++aa)  
{  
if (diff1->r[aa]< min)  
{  
min = diff1->r[aa];  
names = diff1->name[aa];  
}  
}
```

```
if (names >= 1 && names <= 2 )  
Console::WriteLine(S"\nthe banknote is 20!!\n");  
else if (names>=3 && names<=4)  
Console::WriteLine(S"\nthe banknote is 50!!\n");  
else if (names>=5 && names<=6)  
Console::WriteLine(S"\nthe banknote is 100!!\n");  
else  
Console::WriteLine(S"bad banknote!!");
```

```
}  
////////////////////////////////*/
```

```
Console::WriteLine(S"DONE..!!");  
for(int af=0 ;af<5;){}  
////////////////////////////////  
return 0;
```

```
}
```


APPENDIX B

NIS: New Israel shekel

LDR: Light Dependent Resistors

DB: Database

RGB: Red, Green, Blue.

HSV: Hue, Saturation, Value.

RAM: Read Access Memory.

PC: Personal Computer.

MHz: Mega Hertz.

GB: Giga Bytes.

Dis: Distance.

VC++: Visual C++.