

Elankumaran Arulliah

47 Jackson Road,

Karawara, WA 6152

5th November 2004

Professor Syed Islam

Head of Department

Department of Electrical and Computer Engineering

Curtin University of Technology

Bentley, WA 6102

Dear Sir

Re: Computer Systems Engineering Project

As part of the requirement for the Bachelor of Engineering (Computer Systems), I hereby present my project thesis titled "Image Acquisition and Processing for Low Vision Image Enhancer" for your perusal.

I hereby state that this thesis is entirely my own work outside of where the acknowledgement is given

Thank you.

Yours sincerely

(Elankumaran Arulliah)

TITLE: Image Acquisition and Processing for Low Vision Image Enhancer

AUTHOR:

FAMILY NAME: Arulliah

GIVEN NAME: Elankumaran

DATE : 5th November 2004

SUPERVISOR : Mr. Iain Murray

DEGREE: Bachelor of Engineering

OPTION: Computer Systems

ABSTRACT:

An image enhancer has been designed and implemented for visually impaired students. This thesis discusses the design, image acquisition and image processing techniques of a Low Vision Image Enhancement System (LOVIES). LOVIES is a portable low cost CCTV based device which can be used in a class room environment to display the far away objects, diagrams and texts on a screen next to the user. This enhancer accommodates most of the image enhancement techniques with a user input interface, which can be used to combine two or more of these techniques.

INDEXING TERMS

Image processing, Low vision, Image enhancer, Enhancement techniques

TECHNICAL WORK

REPORT PRESENTATION

GOOD

AVERAGE

POOR

EXAMINER

CO-EXAMINER



Image Acquisition and Processing for Low Vision Image Enhancer

By

Elankumaran Arulliah

Supervisor: Mr. Iain Murray

A thesis submitted for the degree of Bachelor of Engineering

in

Computer Systems Engineering

November 2004

ACKNOWLEDGEMENTS

I would like to thank my project supervisor Iain Murray for his help and support through out the course of this project and Andrew Pasquale for his assistance with the technical and information he provided. I would also like to thank Prof. R. T. Rajeswaran for sponsoring me and helping me to follow my undergraduate studies in Curtin University of Technology, Australia.

Synopsis

An image enhancer has been designed and implemented for visually impaired students. This thesis discusses the design, image acquisition and image processing techniques of a Low Vision Image Enhancement System (LOVIES). LOVIES is a portable low cost CCTV based device which can be used in a class room environment to display the far away objects, diagrams and texts on a screen next to the user. This enhancer accommodates most of the image enhancement techniques with a user input interface, which can be used to combine two or more of these techniques.

Index

| | |
|---|----|
| Synopsis..... | ii |
| List of Figures | v |
| List of Tables..... | vi |
| 1.0 Introduction..... | 1 |
| 1.1 The problem of vision impaired..... | 1 |
| 1.2 Problem explained..... | 3 |
| 1.3 Solution | 3 |
| 1.4 Thesis structure | 4 |
| 2.0 Vision Impairment and Existing assistive technology..... | 6 |
| 2.1 Visual impairment..... | 6 |
| 2.2 The aids available for the visually impaired students | 9 |
| 2.2.1 Low vision devices | 9 |
| 2.2.2 Optically magnifying devices | 10 |
| 2.2.3 CCTV..... | 11 |
| 2.3 Latest available technology..... | 12 |
| 2.3.1 Optelec ClearView 317..... | 12 |
| 2.3.2 Aladdin Rainbow Pro | 13 |
| 2.3.3 LVES | 14 |
| 3.0 Design Specifications..... | 16 |
| 3.1 Expected features of new system | 16 |
| 3.2 The overview of the CCTV system..... | 16 |
| 3.3 Why the newly built system is attractive | 17 |
| 4.0 Selection of DSP Platform | 19 |
| 4.1 Requirements | 19 |
| 4.2 TMS320C6711 | 20 |
| 4.2.1 Properties of TMS320C6711 | 20 |
| 4.2.2 Enhanced Direct Memory Access (EDMA) | 22 |
| 4.2.3 External Memory Interface (EMIF) | 22 |
| 4.2.4 Multi-Channel Serial Port Interface (McBSP)..... | 23 |
| 4.2.5 Interrupt Selector..... | 23 |
| 4.2.6 Host Port Interface (HPI) | 24 |
| 5.0 Selection of Image capturing unit | 25 |
| 5.1 Requirements | 25 |
| 5.2 Camera M3188A..... | 28 |
| 5.3 OV7120 | 28 |
| 5.3.1 Image Sensor properties..... | 28 |
| 5.3.2 SRAM mode | 30 |
| 6.0 Selection of the image processing techniques..... | 33 |
| 6.1 Vision Impairment in Australia..... | 33 |
| 6.2 Blurred vision..... | 33 |
| 6.2.1 Sharpening..... | 33 |
| 6.2.2 Spatial filters..... | 35 |
| 6.2.3 Focus lines..... | 35 |
| 6.3 Localised vision degradation..... | 36 |
| 6.3.1 Zoom..... | 36 |
| 6.4 Reduction in contrast and colour perception..... | 38 |

| | |
|---|-----|
| 7.0 User Input Subsystem | 40 |
| 7.1 Design considerations | 40 |
| 7.2 Design | 41 |
| 8.0 Overall system and its functionality | 44 |
| 8.1 System block diagram | 44 |
| 8.2 Camera – DSP interface | 45 |
| 8.3 Memory allocation | 45 |
| 8.4 User Input – DSP interface | 46 |
| 8.5 Output Format for Display Subsystem..... | 46 |
| 9.0 Implementation | 48 |
| 9.1 Interfacing Camera to TMS320C6711 | 48 |
| 9.1.1 Interfacing concept | 48 |
| 9.1.2 Input image synchronized with the VSYNC of the camera | 48 |
| 9.1.3 Using SRAM mode of the Camera..... | 53 |
| 9.2 Image processing techniques implementation and results | 56 |
| 9.2.1 Implementation tools | 56 |
| 9.2.2 Sharpening using FFT | 56 |
| 9.2.3 Sharpening using spatial filters | 59 |
| 9.2.4 Zoom..... | 61 |
| 9.2.5 Focus lines | 64 |
| 9.2.6 Log contrast | 65 |
| 9.2.7 Threshold..... | 66 |
| 9.2.8 Changing the foreground and background colour..... | 67 |
| 9.3 Testing on actual environment and Performance..... | 70 |
| 9.4 Performance | 70 |
| 10.0 Conclusion and Future directions | 72 |
| 12.0 Bibliography..... | 76 |
| Appendix A – Function listings in C..... | 80 |
| Appendix B – Function Listings in MatLab..... | 100 |

List of Figures

| | |
|--|----|
| Figure 2.1 (a) Normal vision | 7 |
| Figure 2.1 (b) Vision encompassing Macular degeneration | 7 |
| Figure 2.2 (a) Normal vision | 8 |
| Figure 2.2 (b) Vision encompassing Retinitis Pigmentosa | 8 |
| Figure 2.3 (a) Normal vision | 9 |
| Figure 2.3 (b) Vision encompassing cataract | 9 |
| Figure 2.4 The Optelec ClearView 317 | 12 |
| Figure 2.5 The Aladdin Rainbow Pro RBP-1 | 13 |
| Figure 2.6 Low Vision Enhancement System (LVES) | 14 |
| Figure 5.1 Timing diagram of Image Sensor M3188A | 29 |
| Figure 5.2 Interface between OV7120 and Microcontroller | 31 |
| Figure 5.3 Timing diagram for SRAM mode of OV7120 | 31 |
| Figure 6.1 Sharpening algorithm using FFT-method | 34 |
| Figure 6.2 Spatial high pass filter matrices | 35 |
| Figure 7.1 Interface between the User Input keypad and DSP C6711 | 43 |
| Figure 8.1 Low Vision Image Enhancer system block diagram | 44 |
| Figure 8.2 Output pixel format of Image Enhancer | 47 |
| Figure 9.1 Sample input image 1 using Vsync synchronized image capturing method | 51 |
| Figure 9.2 Sample input image 2 using Vsync synchronized image capturing method | 52 |
| Figure 9.3 Sample input image using SRAM mode method | 55 |
| Figure 9.4 (a) Image before sharpening | 57 |
| Figure 9.4 (b) Image after sharpening – amplifying factor > 1 – using FFT method | 57 |
| Figure 9.4c Image after sharpening – amplifying factor 0.2 – using FFT method ... | 58 |
| Figure 9.5 Image before convolving with edge enhancement matrix | 60 |
| Figure 9.6 Sharpened image using spatial filter matrix | 60 |
| Figure 9.7 Modified input image before applying zoom technique | 62 |
| Figure 9.8 Modified input image after applying zoom technique | 62 |
| Figure 9.9 Gaussian smoothening matrix 5x5 | 63 |
| Figure 9.10 Image with the focus line | 65 |
| Figure 9.11 Black and white threshold image | 69 |
| Figure 9.12 Threshold image after applying two colours | 69 |

List of Tables

| | |
|--|----|
| Table 5.1: CCD vs CMOS Feature comparison | 27 |
| Table 5.2: CCD vs CMOS performance comparison | 27 |
| Table 7.1: Controllable parameters of the Image Enhancement techniques | 41 |
| Table 7.2: User input interface state assignment table | 42 |
| Table 7.3: User input interface state transition table | 42 |
| Table 9.1: Spatial edge enhancement matrix | 59 |

1.0 Introduction

1.1 The problem of vision impaired

Visual aids are used extensively in all areas of education. Verbal communication is not sufficient, when it comes to expressing the idea. Hence traditional teaching methods involve the use of blackboards, whiteboards, overhead projectors and other types of visual demonstration to supplement verbal explanations. Books and computers are also important learning tools for they facilitate easy storage and retrieval of large amounts of information in a form that can be easily distributed.

The major challenge facing visually impaired students in the science/educational environment is the overwhelming mass of visual material to which they are continually exposed such as textbooks, class outlines, class schedules, and chalkboards writing. Students with visual impairments encounter many difficulties in exploring interesting articles and attending seminars and, thus, may miss opportunities to learn. The result can be a far less effective learning experience and thus a hindrance to their education.

Certain types of visual impairment, such as macular degeneration, result in a loss of contrast sensitivity. This can make it difficult to identify low contrast shapes such as text and diagrams, for example if written on a whiteboard with an old pen. A parallel aspect is the time-variance of the sufferer's colour sensitivity. The result can be a person who has difficulty with low-contrast images, and in addition who may be less sensitive to certain ranges of colour. Some assistive devices can be used to make a sufferer's interaction with the teaching environment more productive.

The extent of visual disability depends upon the physical sensory impairment of the student's eyes, the age of the student at the onset of vision impairment, and the way in which that impairment occurred. Vision in the classroom environments may also be influenced by factors such as inappropriate lighting, light glare, or fatigue. These students need help in using their residual vision more efficiently and in working with special aids and materials.

To assist in overcoming a student's visual limitation requires unique and individual strategies based on that student's particular visual impairment and his/her skill of communication. Magnification and contrast are the most important tools for low vision rehabilitation. Magnifiers, strong bifocals, telescopic systems, microscopic systems, closed circuit television systems, large print, and computer screen magnification programs may all provide effective magnification for low vision people

Magnifiers are the simplest tools selected by the vision-impaired people. Small magnifiers are used to read books, price tags or menus and larger illuminated stand magnifiers can be used at home or work. They use the optical zoom for magnification.

Telescopic systems are used to magnify distant objects. These may be handheld or mounted in eyewear. These systems can be worn full time. But this also mainly uses the optical zoom, which makes the system heavier to get better performance.

The CCTV is a highly customizable versatile magnification system designed to enlarge and enhance images and print material using video camera and digital signal processing technology.

1.2 Problem explained

Most of the currently available visual aid systems are either head mounted displays or systems that are used to read close up materials like reading books, sewing. Although these devices are useful in day-to-day activity, they have plenty of shortcomings. Since they use a computer monitor or television as the display unit, they become non-portable systems and a general-purpose low vision enhancement system costs more than 3000 US dollars. Hence there arises a need for a low cost, portable CCTV system which assists to read not only the close up materials but also far away objects in a class room environment. This system also needs facility to combine different techniques depending on their personal needs.

1.3 Solution

This thesis outlines a design and implementation process of a low vision image enhancer CCTV system that can be used in a classroom situation as well as in a home environment. Unlike the CCTV systems that are available in the market, this is an easily portable system, weighs less than 2kg, with a cost around AUS \$2000, which is affordable to most of the people in the community. It was designed as a stand-alone system so they do not require any computer monitor or television as a display unit, which make a system non-portable.

This Image Enhancer provides most of the image enhancements techniques that a high cost CCTV system offers with some additional features included. It also includes a keypad acting as a user interface to control and combine the different types of enhancements available. The image sensor used in this system is capable of picking up infrared lights too. This would increase its usage under lowlight situations.

1.4 Thesis structure

Chapter 2 describes the low vision symptoms and effects, and it explores the current practices to overcome these low vision impairments

Chapter 3 outlines the system requirements and the design specifications of the suggested Image Enhancer

Chapter 4 – justifies the selection of the DSP platform and describes the features and peripherals available in TMS320C6711

Chapter 5 – compares the differences between the CCD image sensor and the CMOS and describes the image sensor (M1388A) and the CMOS chip (OV7120) that was used in the system

Chapter 6 discusses the common division of the vision impairments and selects the appropriate image enhancement techniques that can be used to overcome each of those impairments

Chapter 7 proposes a simple design of user input interface and describes accessible features, and justifies the selection.

Chapter 8 brings all parts of this system together while chapter 9 discusses the hardware implementation of the system and its issues

Chapter 10 is the conclusion and Chapter 11 explains the possible developments that can be done in the future.

2.0 Vision Impairment and Existing assistive technology

2.1 Visual impairment

Human eye is made up of many different parts like the cornea, iris, lens, and retina. The eye uses special optic nerves to send all the images captured to the brain where, the image is processed and recognized what is seen. This is almost an instantaneous process.

Visual impairment happens when there is a problem with one or more parts of the eyes or the parts of the brain needed to process the images sent from the eyes. Most people have some type of visual problem at some point in their lives. Some can no longer see objects far away; others have problems reading small print. These types of conditions are often easily treated with eyeglasses or contact lenses. But when one or more parts of the eye or brain that are needed to process images become diseased or damaged, severe or total loss of vision can occur. In these cases, vision cannot be restored with medical treatment, surgery, or corrective lenses like glasses or contacts.

Visual impairment also can be caused by heredity or genes. If a pregnant mother gets certain type of disease, then the chances of the unborn child developing a visual impairment is high. Accidents that hurt the eyes, infections, and some diseases also can cause visual impairment [27].

The most common diseases that cause visual impairment are macular degeneration, cataracts, glaucoma, and diabetic retinopathy. Age-related Macular Degeneration is a degenerative retinal eye disease that causes the progressive loss of central vision. The Macula is the centre of the retina, which allows a person to see clearly and appreciate

colours. When a person has macular degeneration the macula becomes scared. The eye may still have good side vision, but blank spots appear in the centre. This makes reading, sewing, or seeing distance objects difficult. This disease does not lead to total blindness [26]. The following figures figure 2.1 conveys an example of how the macular degeneration affects [13]. The figure (a) shows how a sighted person see landscape which is perfectly in focus and the figure (b) shows how a person affected by the macular degeneration would see the same landscape whereby the background is in focus yet the trees in the foreground appear blurred.



(a)



(b)

Figure 2.1: (a) Normal vision
Figure 2.1(b) Vision encompassing Macular degeneration

In Macular Degeneration or MD, the macula, which is the most important area of the eye, gets distorted or diseased. The cones, the major retinal cell in the macula, also are damaged so that the ability to read and distinguish colours is severely affected.



(a)



(b)

Figure 2.2 (a) Normal vision
Figure 2.2 (b) Vision encompassing Retinitis Pigmentosa

In contrast to MD, the patients with Retinitis Pigmentosa or RP lose their peripheral vision and gradually they might go total blind. RP is a degenerative eye disorder that afflicts approximately one in 21,000 people; it is known that this is typically a hereditary disease where a recessive gene carries the defect. The symptoms vary from patient to patient, but in general terms begins with a loss of night vision and gradually a loss of peripheral vision. RP Patients experience a form of persistent tunnel vision that progressively closes inward encroaching on the region of central acuity [19] [18].

Cataract is a clouding of the lens inside the eye and makes a person feel that he / she is looking through a frosty window. It may vary in its severity from a small amount of clouding to dense areas of haziness. This is usually an age related condition, which disturbs the passage of light and prevents the eye from focusing correctly. A cataract is caused by a disturbance of nutrition to the lens, resulting from a lack of oxygen. It may also be caused by injuries, radiation or exposure to toxic chemicals. Cataracts can cause blindness if it is not diagnosed and treated at its early stage. The

figure 2.2 compares the image seen by a well-sighted person (Figure 2.3 a) and the image viewed by the person with the cataract (figure 2.3 b) [13].

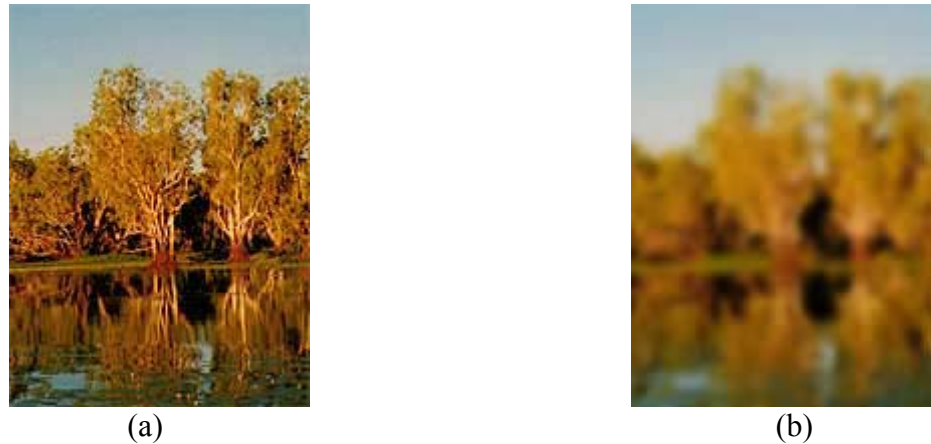


Figure 2.3: (a) Normal vision
Figure 2.3(b) Vision encompassing cataract

Glaucoma is an eye disease that slowly damages the fine nerves that connect the eyes to the brain. This is usually caused by a build-up of pressure in the eye. The eye is normally filled with 'intra ocular' fluid, which constantly drains away and is then replaced. In the case of glaucoma, intra ocular fluid is not drained away properly, or it may be produced in large amounts. If this causes too much pressure in the eye, the optic nerve is damaged and the blind areas in the field of vision develop. Glaucoma mostly affects the side vision. The edge of the field of vision starts to fade, causing vision to narrow.

2.2 The aids available for the visually impaired students

2.2.1 Low vision devices

There are numerous organisations, research centres and private companies working on creating devices for visual impaired people. This gives the opportunity to the affected of selecting a suitable aid from a vast number of available low vision

devices. Some of the devices are designed for people with normal vision but with changes to the contrast, colour and size. Some only do optical magnification of the real world objects. The others are software based text readers for personal computers.

2.2.2 Optically magnifying devices

There are various optically magnifying assistive devices available for vision impaired. Telescopes, hand-held and spectacle-mounted monocular devices, and magnifying lenses of varying size, magnification and functional application are some of the common devices.

Magnifier is one of the simplest tools selected by the vision-impaired people for their day-to-day activity. Small magnifiers are used to read books, price tags or menus and larger illuminated stand magnifiers can be used at home or work. These devices use the optical zoom for magnification. Hence they become heavier with the increasing efficiency of the optical lens. Strong bifocals are dedicated glasses with microscopic eyewear. But this cannot be used to view far distance writing such as in classroom environment [11].

Telescopic systems are used to magnify far distance objects. These may be handheld or mounted in eyewear. These systems can be worn full time. But this also mainly uses the optical zoom, which makes the system heavier to get better performance.

The main advantages of these are that they are relatively inexpensive, commonly lightweight and portable. But they only magnify the real world systems and they fail to enhance the images. The performance of these devices is severely affected by the

surroundings such as lighting or glare. One main disadvantage of these devices is that magnification does not assist all vision-impaired people.

2.2.3 CCTV

CCTVs are highly customizable versatile magnification systems specially designed to enlarge printed material for people who have low vision and can no longer comfortably use glasses or special lenses to read regular size print. The concept of a CCTV system is very simple. A basic system consists of a video camera, a digital signal processor and a monitor. The basic component of the CCTV system is the camera. This is used to capture the images of the reading materials or object and transmit to the DSP processor. The DSP processes the image as required by the user and display on the monitor [24].

Magnification from telescopes, magnifiers and microscopic lenses is limited by their design. On the other hand, Closed circuit television systems are capable of higher levels of magnification and can manipulate the brightness and contrast of the image. Both optical and digital zoom can be employed in these systems, which makes CCTV preferable over the other systems.

These systems can be fixed or portable and used not only to view the far distance objects such as in class room environment but also to provide the magnification necessary to complete many other tasks including drawing, writing and some other craft activities as well as viewing labels, maps and other small technologies such as hand held calculators.

2.3 Latest available technology

2.3.1 Optelec ClearView 317



Figure 2.4: The Optelec ClearView 317

The Optelec ClearView 317 is an integrated video magnifier with a custom designed 17-inch monitor that sits securely and tilts for comfortable reading. The table is large enough to accommodate a large book or magazine, and includes wrist rests to reduce fatigue and two pencil holders. The front controls are activated with the thumbs and fingers so the user does not have to take his hands off the table when reading. The table has a brake that can lock the table in place to make writing easier. The control includes instant focus, fingertip zoom, location marker (to help users find their place on the page) and photo, reading, and reverse reading modes. This system can magnify text and pictures from 3 times up to 30 times their original size with a zoom range of 10:1. Those enlarged images enable many people who are legally blind or severely visually impaired to read and write independently [25]. The figure 2.4 gives an idea of how this device can be used. ClearView 317 Black & White with 17" monitor Price: \$1,895

2.3.2 Aladdin Rainbow Pro



Figure 2.5: The Aladdin Rainbow Pro RBP-1

The Aladdin Rainbow Pro is a TeleSensory product built to assist the vision-impaired students. The Aladdin Rainbow Pro is a full-colour low cost system with the versatility of the other professional products. This offers six settings including full colour, black and white, and several selectable foreground and background colours allowing the user to choose which setting best suits his or her individual eye condition. This supports up to 50x magnification.

The Rainbow Pro has the added functionality of tint and colour saturation control enabling the user to adjust the picture to meet their viewing needs along with offering vertical and horizontal line markers and shadow mask to help track text. This system also features an extremely large depth of field for viewing three-dimensional objects and a smooth, non-glare reading table which permits reading heavy books. This system includes a colour monitor of 14" with a 13.3" diagonal viewing area. The whole system weighs around 20kg with a cost of US \$ 2845.00 [20].

2.3.3 LVES



Figure 2.6: Low Vision Enhancement System (LVES)

LVES was introduced to the commercial marketplace in 1994 after almost a decade of development by a multi-organizational team that included NASA, the Wilmer Eye Institute of The Johns Hopkins Medical Institutions, the Department of Veteran Affairs, and Visionics Corporation, Minneapolis, Minnesota, which manufactures the system under license. The device was invented and patented by a trio of researchers: Dr. Robert W. Massof, director of Wilmer Eye Institute's Lions Vision Center; Dr. Thomas Raasch, also of Wilmer; and Dr. Donald O'Shea of Georgia Tech.

The LVES consists of a head-mounted video display worn like goggles, a set of three video cameras, and a control unit that allows the user to select and control the cameras, and to adjust contrast and image polarity to suit the user's needs. The cameras feed the images to a computer that corrects for the particular vision problem of the user, and then sends the images to the video display in the goggles.

LVES is powered by a battery carried in a nylon belt pack along with a control unit that allows the wearer to adjust contrast and magnification. The belt pack, including the battery, weighs 2 1/2 pounds and is about the size of a book. The headset, worn like aviator's goggles, weighs 34 ounces.

These patients may have macular degeneration associated with aging or diabetic retinopathy, in which diabetes causes swelling and leakage of fluid in the centre of the retina, the macula. The system also has benefited people who have lost peripheral or side field vision, a problem associated with glaucoma, and people suffering from retinitis pigmentosa, a progressive degeneration of the retina that results in tunnel vision and extreme sensitivity to light. Persons with optic nerve disease and congenital damage to the retina also use the LVES successfully [15]. The device and a half-day training cost US \$5600.00 [9].

3.0 Design Specifications

3.1 Expected features of new system

1. Portable – to use in any place at any time
2. Variable magnification
3. Variable contrast enhancement
4. Variable threshold value
5. A variable combination of background and foreground colour.
6. Light weight and easy to manoeuvre
7. Low cost
8. User Inputs – allows the user to select his / her preferred techniques
9. Wide range - enough to cover the whole white board

3.2 The overview of the CCTV system

The CCTV system designed in this project is a portable device such that it can be carried to where it is needed (classrooms, lecture halls). The main components of this system are a camera, a DSP board, a LCD screen and a six-button keyboard. The image source is a CMOS camera M3188A. The M3188A is a 1/3" B/W camera module with digital output. The digital video port supplies a continuous 8 bit-wide image data stream. All sensor functions, such as exposure, gamma, gain, white balance, colour matrix, windowing, is programmable through I²C interface. . The camera captures the images to be enhanced from the white board (or from the text book on the table) at a sufficiently high frame rate to avoid jerkiness in the final video stream. These captured images are sent to the DSP TMS320C6711 continuously. The DSP processes these images as a sequence of independent stills,

performing filtering, contrast enhancement, and then de-blurring operations on each still, before the images are displayed on the screen.

The device have digital zoom implemented to allow the user to deal with objects both close by, as in the case of a book on their desk, and further away, such as a lecturer's writing on a whiteboard. By making the device portable, it could also be used in a library to read the spine of books for example. This also performs a reassignment of colour on a given image to allow use of colours, which are easier to see for the vision-impaired user. All the functionalities of this device such as zoom, contrast, threshold and foreground, background colours are adjustable. The input keyboard allows the user to change the settings according to his/her needs.

3.3 Why the newly built system is attractive

Most of the CCTV image enhancement systems that exist work on the principles of optical magnification. Zooming by optical means does not reduce the image quality and zooming is continuous. But this requires a camera that shall accommodate variable zoom lenses. Also, the selection of the lenses affects the efficiency of the camera. This makes the system not only heavy and physically large but also costly.

Some enhancement systems for vision impaired used to be head mounted. These systems suffer from disadvantages like restricted field of view because of the use of head mounted displays, the virtual world being presented in the display having less resemblance to the real world, inefficient methods of zooming which makes the system heavy and high power consumption.

Current low vision enhancement systems available suffer from various disadvantages. Most of the devices are heavy, thus not portable. Portable devices with similar functionality are priced over US \$2000, far out of reach to a middle class family. The devices that are priced most reasonably are only capable of reading only closed materials such as textbooks, not distance objects like a white board in a classroom environment [15] [20].

The image enhancement system that is suggested here was designed after considering the above factors and was made as a low cost, portable CCTV system accommodating most of the image enhancement techniques that are provided by the existing CCTV systems.

4.0 Selection of DSP Platform

4.1 Requirements

In the field of Signal Processing and Embedded Systems, there is a vast choice of processor platforms today, and one of the first steps for any new hardware development is, choosing the right processor platform to build the application product/solution. Signal processing systems market is extremely cost sensitive for the portable gadgets that are in use. In order to enable the reduction in cost, single chips are used to load with more and more of functionality and computations. Programmable digital signal processors provide the opportunity for the application developers to cope up with the heterogeneity of standards (protocols) to be mapped with fast turn around times.

One of the basic requirements of this Image Enhancer is to display clear enhanced images fast enough to avoid jittering. The enhancements techniques include digital zoom, sharpening / edge enhancement, threshold and histogram. This implies that the processor shall be able to execute quite a number of loops, which might go through each pixel of the image ($480 \times 640 = 307200$ pixels). This is a lot of work and requires quite a bit of processing power to finish within the short period available for processing.

The selected DSP platform shall have DMA functionality so that the transfer of image from the image sensor occurs without interrupting the CPU of the DSP. There may a situation where the different instances of processed images have to be stored for later use. Also the output frame needs to be constantly accessible. Hence the DSP shall have sufficient memory to store these images.

This DSP shall have built in interfaces such as EMIF (External Memory Interface), HPI (Host Port Interface), EDMA (Enhanced DMA), McBSP (Multi-channel Serial Port), external interrupts to ease the interfacing between DSP and image sensor and DSP and display unit. The technical support and cost also influence the selection of the DSP.

The DSP shall also have the following general features that ease the development process and technical support [23].

1. High-level language support (C compiler)
2. Addressable memory space
3. Floating point / fixed point arithmetic
4. Multi-tasking and multi-processing support
5. Low power consumption
6. Code optimization support in the compiler
7. Integrated development environment (IDE) availability

4.2 TMS320C6711

4.2.1 Properties of TMS320C6711

TMS320C6711 is optimized for highest performance and ease-of-use in high-level language programming. C6711 has 32-bit instruction set architecture with 150MHz operating power and 16MByte external memory connected through the EMIF. High performance, ease of use, and affordable pricing properties of C6711 make this platform the ideal solution for multi-channel, multifunction applications, such as: pooled modems, wireless local loop base stations, remote access servers (RAS),

digital subscriber loop (DSL) systems, Multi-channel telephony systems, virtual reality 3-D graphics, speech recognition, audio, imaging (examples: fingerprint recognition, ultrasound, and MRI). At the time of implementation the development board cost AUD\$ 550.00, but the actual chip for the Image Enhancer would cost much less than that. Since this device is being developed as a university student project, there is not much money spent for the development board.

The DSP C6711 includes the following features [23].

1. Industry's most efficient C compiler on DSP benchmark suite
2. 8/16/32-bit data support, providing efficient memory support for a variety of applications
3. Executes up to eight instructions per cycle for up to ten times the performance of typical DSPs
4. Allows designers to develop highly effective RISC-like code for fast development time
5. Gives code size equivalence for eight instructions executed serially or in parallel
6. Reduces code size, program fetches, and power consumption
7. Reduces costly branching
8. Code executes as programmed on independent functional units

There are also a variety of memory and peripheral options available for the DSP C6711. Some of the important parts are listed below [23].

1. Large on-chip RAM for fast algorithm execution

2. 32-bit external memory interface (EMIF) supports SDRAM, SBSRAM, SRAM, and other asynchronous memories, for a broad range of external memory requirements and maximum system performance
3. Multi-channel Extended-DMA controller
4. Multi-channel serial ports
5. External interrupts

4.2.2 Enhanced Direct Memory Access (EDMA)

The EDMA controller transfers data between regions in the memory map without intervention by the CPU. It allows the movement of data between internal memory, internal peripherals, or external devices to occur in the background of CPU operation. EDMA on the C6711 has sixteen independently programmable channels allowing sixteen different contexts for operation. One of the important features of the EDMA is “event synchronization”. This functionality allows each channel to be initiated by a specific event. In video processing, the transfers may be either synchronized by element (pixel) or by frame [23].

4.2.3 External Memory Interface (EMIF)

The external memory interface supports interface to several external devices, allowing additional data and program memory space beyond that which is included on-chip. This supports synchronous burst SRAM (SBSRAM), synchronous DRAM (SDRAM), asynchronous devices and external shared memory devices [23].

4.2.4 Multi-Channel Serial Port Interface (McBSP)

TMSC6711 has two multi-channel serial port interfaces. The standard serial port interface provides the following [22].

1. Full-duplex communication
2. Double-buffered data registers, which allow a continuous data stream
3. Independent framing and clocking for reception and transmission
4. Direct interface to industry-standard codecs, analogue interface chips (AICs), and other serially connected A/D and D/A devices
5. External shift clock generation or an internal programmable frequency shift clock
6. 8-bit data transfers with LSB or MSB first
7. Programmable polarity for both frame synchronization and data clocks
8. Highly programmable internal clock and frame generation

The most useful feature of the McBSP in this project is that it can be arranged as general purpose input output for parallel data transfer [23].

4.2.5 Interrupt Selector

The C6711 peripheral set produces sixteen interrupt sources. The CPU has 12 interrupts available for use. The interrupt selector allows the program to choose 12 of the 16 interrupts available and to effectively change the polarity of external interrupt inputs [23].

4.2.6 Host Port Interface (HPI)

The Host-Port Interface (HPI) is a 16-bit wide parallel port through which a host processor can directly access the CPU's memory space. The host device functions as a master to the interface, which increases ease of access. The host and CPU can exchange information via internal or external memory. The host also has direct access to memory-mapped peripherals. This interface can be used to interface the image capturing unit in the stand alone Image Enhancer [23].

Different versions of the Texas Instruments' DSP platforms including TMS320C6711 are used within the Electrical and Computer Engineering department at this university. Hence it was easier to hold onto one of these processors for the sole purpose of this project. Although there was no one in particular who mastered the handling of C6711 processor, this processor has all the properties that suit the requirements of the Image Enhancer and there were postgraduate students who actually were working on the C5000 series platform. So it was easier to get help from them. Hence it was decided to use one of those C6000 series as the Image Enhancer's platform.

5.0 Selection of Image capturing unit

5.1 Requirements

People need different types of enhancement techniques to overcome different types of impairments. Sharpening may help people with blur vision but it does not assist them who have Retinitis Pigmentosa. Most of the people prefer a combination of these image enhancements to maximize their vision. To accommodate various types of techniques, the image-capturing unit needs to retain 1:1 representation of the real world. In a classroom situation, an image-capturing unit shall be zoomed and locked on to a certain field of display such as white board. When using cameras speed of operation and resolution must be taken into consideration as well.

The image-capturing unit shall perform the following functions:

- 1) Digitize the raw output from the image sensor and send one pixel at a time
- 2) Normalize frame size to 640x480
- 3) Normalize the frame rate to configurable values
- 4) Prepare output image data in grey scale
- 5) Send a single frame upon request
- 6) Control the following parameters through I2C interface
 - a. Brightness
 - b. Frame rate
 - c. Exposure

The image sensor shall produce reasonably large (640x480) clear images, since there is some image information loss while using digital zoom technique. One of the aims

of this Image Enhancer is to provide the information contained in the frame in simple and clearest way possible. Using a colour image sensor would necessitate about three times as much storage space for each frame as well as three-time processing time. This might reduce the number of enhancement techniques that could be employed in the Image Enhancer. Hence it was decided to use a grey scale image sensor.

CCD, CMOS cameras were the two different cameras that were considered during the selection of the image-capturing unit. The operation of a Complementary Metal Oxide Semiconductor (CMOS) camera is similar to a Charged Coupled (CCD) camera. Both types of camera operate by converting light into electric charge and process it as an electronic signal, however the main difference between the two camera types is that the CCD cameras have an analogue output whereas the output of CMOS cameras are in digital bits [6].

The following tables indicate the main features and performance for both CCD and CMOS cameras [6].

| Feature | CCD | CMOS |
|----------------------|-----------------------------|------------------------|
| Signal out of pixel | Electron packet | Voltage |
| Signal out of chip | Voltage (analog) | Bits (digital) |
| Signal out of camera | Bits (digital) | Bits (digital) |
| Fill factor | High | Moderate |
| Amplifier mismatch | N/A | Moderate |
| System Noise | Low | Moderate to High |
| System Complexity | High | Low |
| Sensor Complexity | Low | High |
| Camera components | PCB + multiple chips + lens | Chip + lens |
| Relative R&D cost | Depends on Application | Depends on Application |
| Relative system cost | Depends on Application | Depends on Application |

Table 5.1: CCD vs CMOS Feature comparison

| Performance | CCD | CMOS |
|----------------------|--------------------------|---------------------|
| Responsivity | Moderate | Slightly better |
| Dynamic Range | High | Moderate |
| Uniformity | High | Low to Moderate |
| Uniform Shuttering | Fast, common | Poor |
| Uniformity | High | Low to Moderate |
| Speed | Moderate to High | Higher |
| Windowing | Limited | Extensive |
| Antiblooming | High to none | High |
| Biasing and Clocking | Multiple, higher voltage | Single, low-voltage |

Table 5.2: CCD vs CMOS performance comparison

After considering the speed, outputs, reliability, availability, weight, cost and variety of the image enhancement techniques that were used in the Image Enhancer, it was decided to select a CMOS camera with grey scale outputs.

5.2 Camera M3188A

The M3188A is a 1/3” monochrome CMOS camera module with digital output of 40x28 mm size. It uses Omni Vision’s CMOS image sensor) OV7120. The combination of the CMOS technology and the easy to use digital interface makes this camera a low cost solution for higher quality video image application.

The digital video port supplies a continuous 8 bit-wide data stream. All camera functions, such as exposure, gamma, gain, windowing are programmable through I2C interface. The camera provides both progressive and interlace output modes and accommodates a video analogue output pin [17].

5.3 OV7120

5.3.1 Image Sensor properties

The specification sheets describe OV7120 as follows:

“The OV7120 (black and white) CMOS Image sensor is single-chip video/imaging camera device designed to provide a high level of functionality in a single, small-footprint package. The device incorporates a 640 x 480 image array capable of operating at up to 30 frames per second. Proprietary sensor technology utilizes advanced algorithms to cancel Fixed Pattern Noise (FPN), eliminate smearing, and drastically reduce blooming. All required camera functions including exposure control, gamma, gain, white balance, colour matrix, colour saturation, hue control,

windowing, and more, are programmable through the serial SCCB interface. The device can be programmed to provide image output in different 16-bit and 8-bit formats.”[17]

OV7120 is a 1/3” CMOS imaging device. It contains approximately 326,688 pixels producing 307,200 (640x480) valid data pixels. It is based on field integration read-out system with line-by-line transfer and an electronic shutter with synchronous pixel readout scheme. The digital video output offers 8bit data format, which is synchronized with the pixel clock (pclk). Other than the 8bit data bus (Y0-Y7), OV7120 also supplies standard video timing signals such as VSYNC, HREF, PCLK. The following figure clearly explains the timing of this sensor [17].

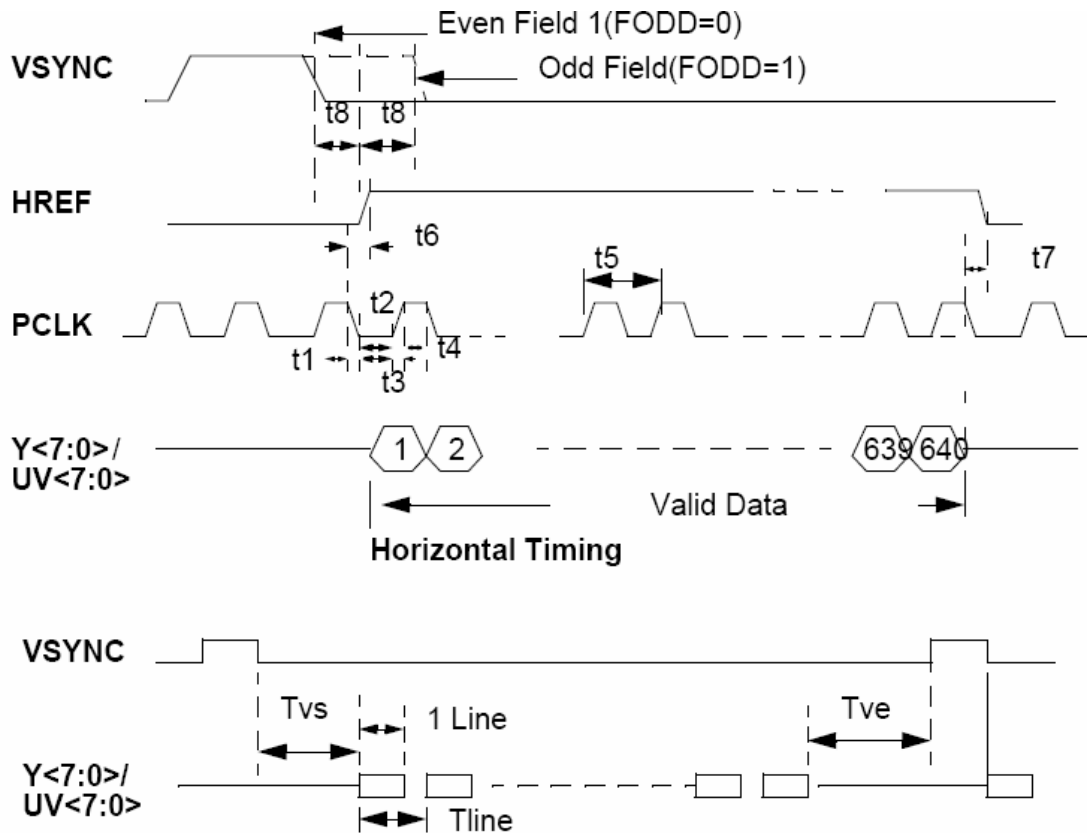


Figure 5.1: Timing diagram of Image Sensor M3188A

A pulse in VSYNC pin indicates the start of the new frame. Immediately after the pulse was created on the VSYNC pin, the image is available on the data bus, line by line. Each line is indicated by the logic '1' in the horizontal reference (Href) output pin. The data is valid only when the HREF signal is driven high. The pixel is latched one by one from the data bus at the positive transition of the each pixel clock (PCLK).

The default operation of this image sensor is time critical. After setting the frame rate of this sensor, the image is available on its data bus continuously. Hence the VSYNC has to be monitored, which tells the start of a frame, to grab a whole frame without any defection. This is time critical and absorbs quite a number of processor cycles of the interfacing Micro-Controller. The SRAM mode functionality of this sensor allows the interfacing microcontroller to request and receive a single image frame at any time. This makes the interfacing relatively simple.

5.3.2 SRAM mode

OV7120 can be programmed to output single frame data to external RAM. The following block diagram and timing diagram explains the structure of this mode [17].

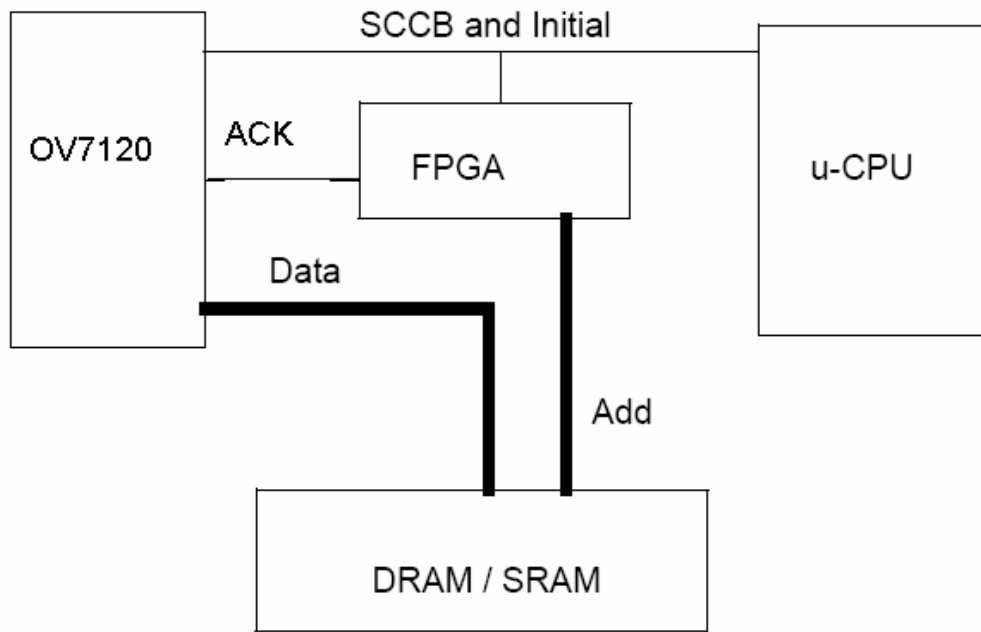


Figure 5.2: Interface between OV7120 and Microcontroller

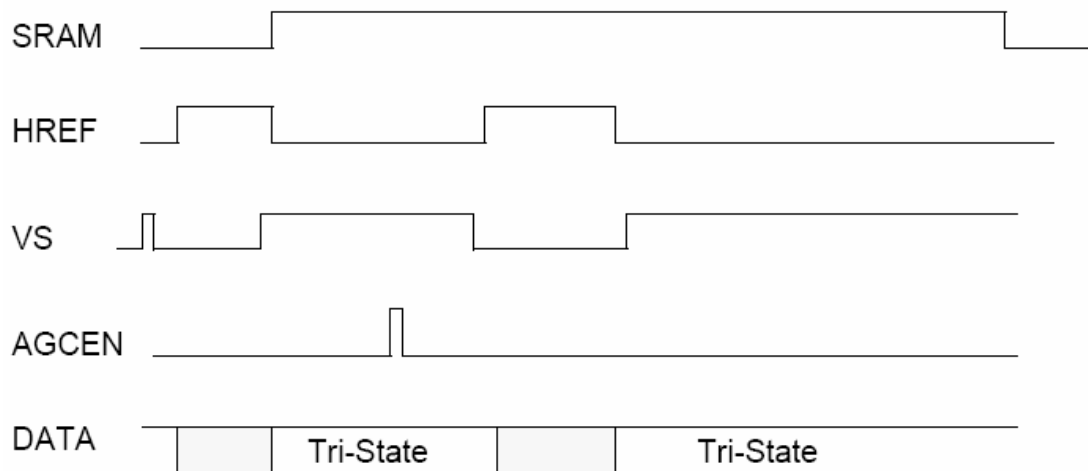


Figure 5.3: Timing diagram for SRAM mode of OV7120

SRAM is internal control bit which is high indicates that the OV7120 enters into external RAM status. This is programmed by SCCB (register 27) or power-on read-in. When SRAM = 1, the VSYNC is pulled high, the data bus is tri-stated and ready to send the data. After this a single frame can be grabbed using two methods.

1. Micro-controller sends an initial pulse to OV7120 AGCEN input pin
2. Micro-controller send a SCCB command (I2C interface) to program OV7120 to send single frame data (writing 0x03 on register 27 would do the job)

Immediately after the request of a frame, the VSYNC is pulled low and stays low until the whole frame is transferred. Hence the micro-controller can receive the frame using the HREF and PCLK signals.

6.0 Selection of the image processing techniques

6.1 Vision Impairment in Australia

According to the survey conducted during 2002/2003, it is estimated that there are currently approximately 380,000 people living in Australia with legal blindness or low vision. This number is expected to double in the next 20 years, as a result of the aging population. More than 80 per cent of vision loss in Australia is caused by just five conditions: refractive error (53%), age-related macular degeneration (13%), Cataract (9%), glaucoma (5%) and diabetic retinopathy (3%). There are also other types of conditions that cause visual impairments and each has different effects on a person's ability to see [13]. Most of these impairments can be categorized into three main divisions. They are

1. Blurred vision
2. Localized vision degeneration
3. Reduction in colour/contrast perception.

6.2 Blurred vision

6.2.1 Sharpening

Blurred vision is the loss of visual acuity or sharpness of vision causing difficulties to see small details. In other words, blurred vision can be described as the loss of the high frequency components of the image. This loss of clear vision could be a symptom of underlying eye problems, such as macular degeneration, cataracts, eye infection, diabetes mellitus, or high blood pressure.

Since blurred vision is considered as the loss of high frequency components of an image, sharpening is the most likely technique to over come this impairment. The

goal of this sharpening technique was to enhance the high frequency components of the image so that the impaired person with the blurred vision can see the images clearly with the improved acuity. Here the high frequencies were extracted from the image and amplified before it was added back with the low frequency components. The following diagram explains the simple algorithm that was used to generate an enhanced image.

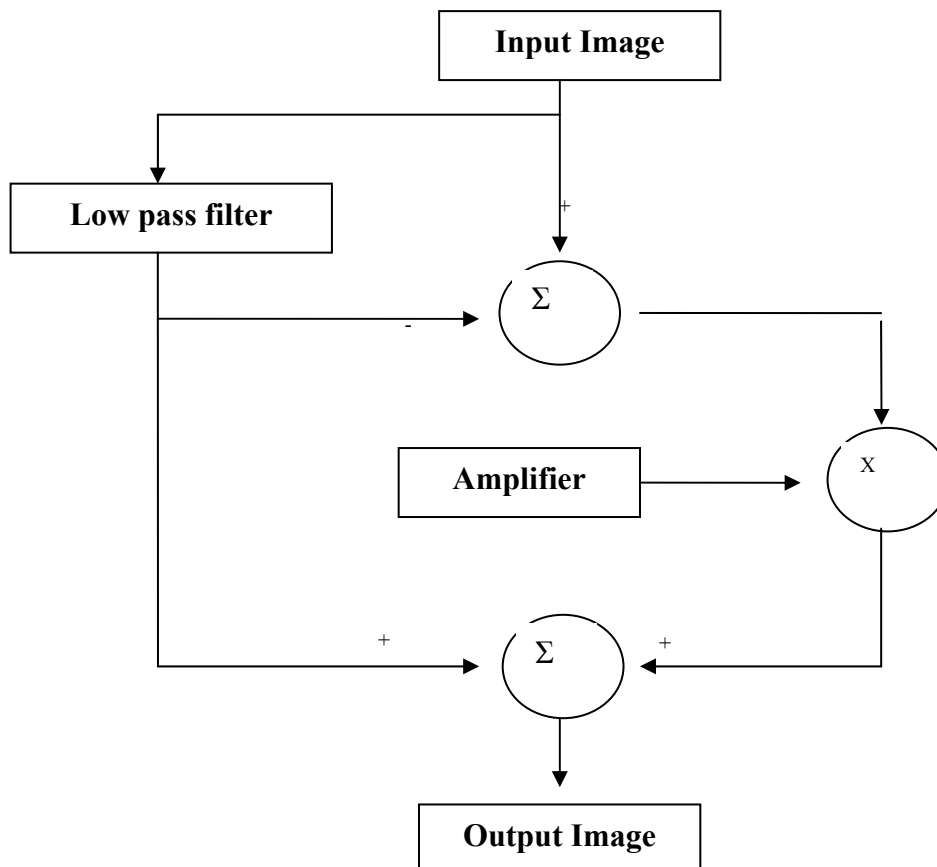


Figure 6.1: Sharpening algorithm using FFT-method

Here the input image was passed through a low pass filter with a fixed bandwidth, which gives the low frequency components of the image as an output. The high frequencies are extracted from the image by simply subtracting the low frequencies from the original image. These high frequencies then passed through an amplifier, the amplitude of which could be controlled through the user input interface

(chapter7). The high and low frequency components were added back together before the image had been displayed on the screen.

The above algorithm was tedious and took a considerable time to process a single frame.

6.2.2 Spatial filters

To overcome the timing constraints in executing FFT and IFFT of an image, a class of spatial high pass filters were used to generate the similar output that the previously discussed algorithm did. Their application was also based on the discrete convolution over the original image with a special mask.

| | | |
|----|----|----|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

| | | |
|----|----|----|
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |

| | | |
|----|----|----|
| 1 | -2 | 1 |
| -2 | 5 | -2 |
| 1 | -2 | 1 |

Figure 6.2: Spatial high pass filter matrices

The above three masks generated almost the same outputs and the first mask was the easiest to optimize. So the first mask was selected to perform the sharpening for the image enhancer.

6.2.3 Focus lines

Another relatively simple technique that was used to overcome the blurred vision was focus lines. Patients with low vision often have difficulty following a single line of text if it is closely surrounded by other text or diagrams. This can be a result of blurred vision, lack of eye-movement control or a combination of both. In order to

compensate for this difficulty, the Low Vision Image Enhancer provides a filter that enables the user to highlight a region of the display to help keep track of areas of interest. Focus lines highlight the area of interest on the image. These focus lines can be moved up and down by selecting the mid point of the focus lines and widen by selecting the width parameter (the distance between the focus lines). This focus lines make the text lines easier to follow and helps the user to concentrate on certain area.

6.3 Localised vision degradation

6.3.1 Zoom

Localized vision degradation is described as the loss of visual field. The person with vision degradation sees some blank spots in his / her vision. The patients with macular degeneration normally have dull or blank spot in the middle part of their vision.

Various image processing techniques can be associated together to help the patients with this eye defection. The image enhancer includes zoom technique with user controllable starting point of zoomed area and zooming factor. This assists the user to zoom and view the area of interest of the image. The image enhancer also accommodates other techniques like sharpening, threshold, generating focus lines, and contrasting which can be selected together with the zoom to make things easier to the patient. These never will cure the problem but help the patient to get the maximum use of the available vision.

Zoom magnification is one of the essential techniques used in the low vision enhancement system. Students with low vision often require text materials in large print and/or high contrast, which makes the text easier to read. This enforced the inclusion of the zoom functionalities within the enhancement system. Zoom can be categorized as optical zoom and digital zoom. It is a common knowledge that the optical zoom is superior to digital zoom.

An optical zoom brings the subject closer to the user without the user actually moving towards it. Thus it increases the amount of image information. On the other hand, digital zoom does not increase the amount of image information. Instead, it crops a portion of the image and then enlarges it back to the normal size of the original image using various interpolation techniques. In so doing, image quality is lost.

Optical zoom can be achieved by attaching additional lens to the camera. But this is bulky and power intensive, thus it hinders the portability of the image enhancer. For these reasons digital zoom dominates in the low vision enhancement system industry. Hence, it was decided to implement the digital zoom in the enhancer.

Digital zoom

Digital zoom increases the apparent image size by various interpolation methods. The image enhancer uses half of the pixels from the point that is nominated by the user and selects the number of pixels according to the zoom factor which is also user controllable and ignores all the other pixels. Here, the number of pixels used by the

enhancer is going to be $(640/\text{zoom factor} \times 480/\text{zoom factor})$. Then it will use interpolation techniques to add detail to the blank area.

First, it was decided to use the simple interpolation methods like nearest neighbour interpolation and bilinear interpolation. The smoothening functions might be introduced later depending on the image quality and the processing power required.

Nearest neighbour interpolation

This is the simplest method and basically makes the pixels bigger. The colour of a pixel in the new image is the colour of the nearest pixel of the original image.

Bilinear interpolation

Here the pixel of the new image is based on the weighted average of the 4 nearest 2x2 neighbourhood of the pixel in the original image.

6.4 Reduction in contrast and colour perception

Colour and contrast perception start at the retina of the human. The retina has different types of cones, light receptors that have sensitivity to different ranges of light frequencies [8].

When the retina and its neural circuitry are affected by infection or high blood pressure the viewer loses the ability to recognize the small changes in contrast and the patient experience headache and dizziness. This also causes troubles to people who have the disease that are associated with aging.

Colour vision fades or changes with age. The yellowing lens tends to absorb and scatter blue light, rendering blues darker and less intense. The unimpeded red and

yellow light are allowed to pass through and cast a warm, reddish glow onto objects [12].

To assist the students with this disease, the contrast between the texts, diagrams and the background can be increased. Considering the processing power and simplicity, contrast and brightness are controlled through built in controls in the display unit of the image enhancer.

Transferring the image into two colours also eases this problem and allows the viewer to read the texts clearly and comfortably. First the threshold value is detected using fuzzy methods. Then the enhancer allows the user to adjust the value until they are happy in viewing the image. This setup also allows the user to select the preferred foreground and background colours as well.

7.0 User Input Subsystem

7.1 Design considerations

One other important stage in designing this Image Enhancer was to generalize its usage. Not every impaired person is going to have the same vision impairment. Also people with the same visual impairment may prefer different combinations of the enhancements techniques. Hence in the real world, the CCTV system must be able to satisfy the users by having various combinations of enhancement techniques. So there arises a need for a user input subsystem that has the control over the selection and combination of these image-processing techniques.

Another major reason for having a user input interface is the subjective user satisfaction. Most of the users want the systems to perform what they wanted them to do. So having a user input interface, which allows the user to vary the parameters to adjust the enhancements to suit their condition, satisfies the user the most than having predefined enhancements even though the latter might offer high quality output.

Since this Image Enhancer was designed for the vision-impaired students, the user inputs have to be simple and clear controls. Also it shall have the access to most of the image processing techniques available. The user interface that was designed was a basic design illustrating the possibility of having simple user inputs.

The following table shows the major available image processing techniques that were used in this system and the parameters that are controllable.

| Image processing Technique | Parameters |
|---------------------------------------|---|
| | |
| Threshold | Background colour, foreground colour |
| Edge enhancement | Strength of the amplifier |
| Focus lines | Center point, width |
| Digital zoom | Zoom factor, starting point of the zooming area |

Table 7.1: Controllable parameters of the Image Enhancement techniques

7.2 Design

The user input interface is made of six push buttons and some combinational logic gates structure. Four of the buttons are dedicated to the four main enhancement techniques, threshold, zoom, edge enhancement, and focus lines. The other two acts as the enable or increment and disable or decrement buttons.

Pressing a dedicated button once allows the user to enable or disable that enhancement technique. Pressing consecutively allows changing the parameters in the order that is specified in table 7.1.

| Inputs | | | Technique |
|--------|----|----|----------------------|
| I1 | I2 | I3 | |
| | | | |
| 0 | 0 | 1 | Threshold (A) |
| 0 | 1 | 0 | Edge enhancement (B) |
| 1 | 0 | 0 | Zoom (C) |
| 1 | 0 | 1 | Focus (D) |
| | | | |

Table 7.2: User input interface state assignment table

| Buttons | | | | | Inputs | | |
|---------|---|---|---|--|--------|----|----|
| A | B | C | D | | I1 | I2 | I3 |
| 1 | 0 | 0 | 0 | | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | | 1 | 0 | 1 |
| | | | | | | | |

Table 7.3: User input interface state transition table

$$I1 = A'B'CD' + A'B'C'D$$

$$I2 = A'BC'D'$$

$$I3 = AB'C'D' + A'B'C'D$$

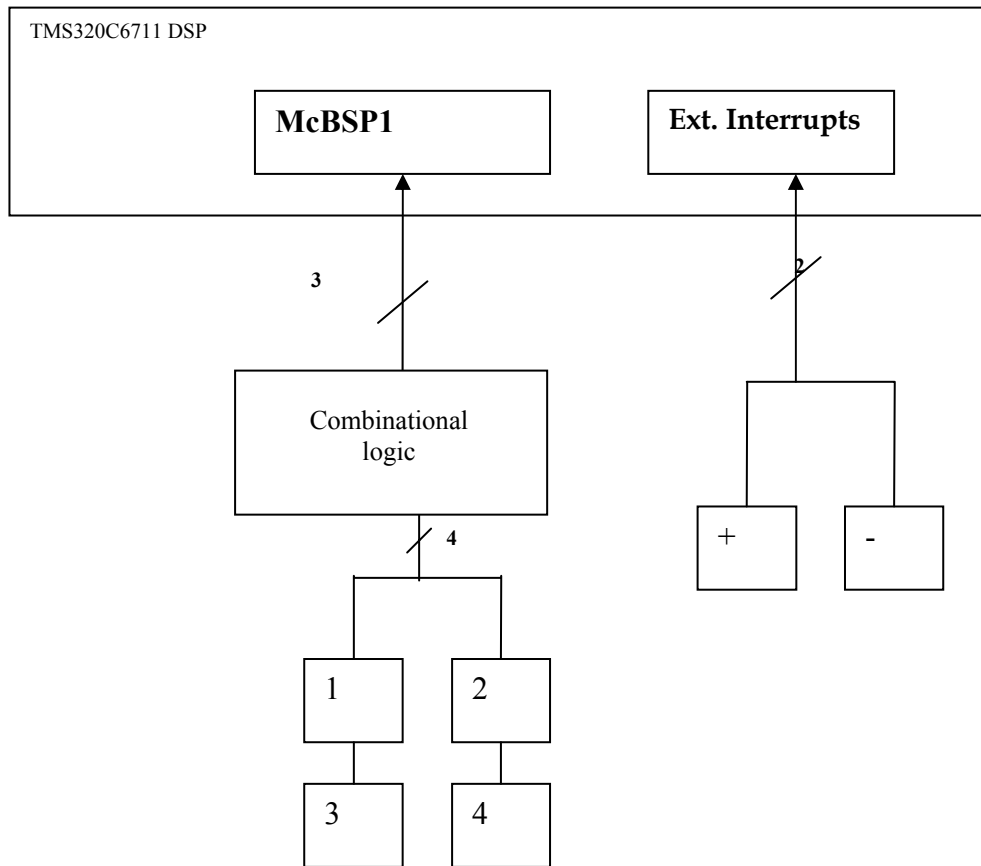


Figure 7.1: Interface between the User Input keypad and DSP C6711

8.0 Overall system and its functionality

8.1 System block diagram

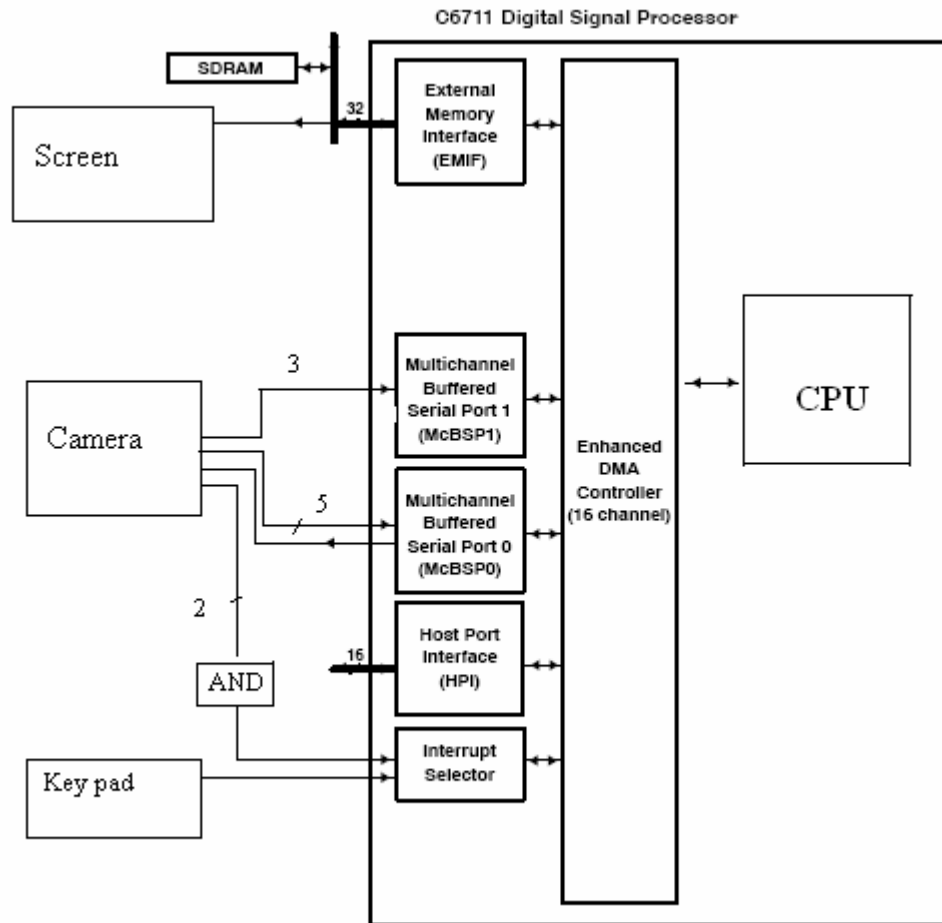


Figure 8.1: Low Vision Image Enhancer system block diagram

8.2 Camera – DSP interface

The camera module was connected to the DSP platform through the two McBSP channels; lowest five bits was transferred through McBSP0 and the highest three bits were transferred through McBSP1. The pixel clock and horizontal reference signals were ANDed together and connected to the external interrupt pins 4 and 5, which synchronized the EDMA element transfer. The DSP uses the connection between the McBSP1 and AGCEN to request single frame at a time.

8.3 Memory allocation

The only major data storage facility available in this Image Enhancer was the 16MB 100MHz SDRAM that is connected to the DSP board through the EMIF interface. This was imaginarily divided into two major blocks, one for the program storage and the other for the temporary image storage. The data storage was further virtually divided into sub memory blocks each was at least 0.6MB of size, so that it was able to accommodate multiple images at a time. Two of the sub blocks were dedicated to the input images, two for the output images and the two more for storing the temporary images during the process. This partition allowed the CPU and EDMA to work concurrently.

The minimum element size that an EDMA supports is 8 bits. According to the image transfer protocol that was created to this particular device (See Chapter9), one pixel took two bytes of memory. So a single image required 614,400 ($=307200*2$) bytes of memory. This is almost 0.6MB of memory. That is why the memory was divided into sub blocks each of size 0.6MB.

8.4 User Input – DSP interface

The user input interface was connected to the DSP via the unused pins in the McBSP channels and the external interrupts. The three control lines of the keypad interface were connected to the McBSP channels and the increment/decrement buttons were directly connected to the external interrupts six and seven. The user can select the variable to be modified by pressing the four control buttons available and modify the variables through the increment / decrement buttons. Each time the CPU processes a new frame it accesses the modified variables in the RAM and then starts processing. This allows the user to view the changes almost instantly.

8.5 Output Format for Display Subsystem

In order for the display system to function correctly and fast, the output image is constantly available in a pre-defined format. The output image is stored depending on whether the output required is colour or greyscale. The colour output format is as yet only used whenever the threshold filter (see section 9.2) is applied and configured to display the foreground and background in colours other than black and white. Figure 8.2 shows the pixel formats for colour and greyscale pixels.

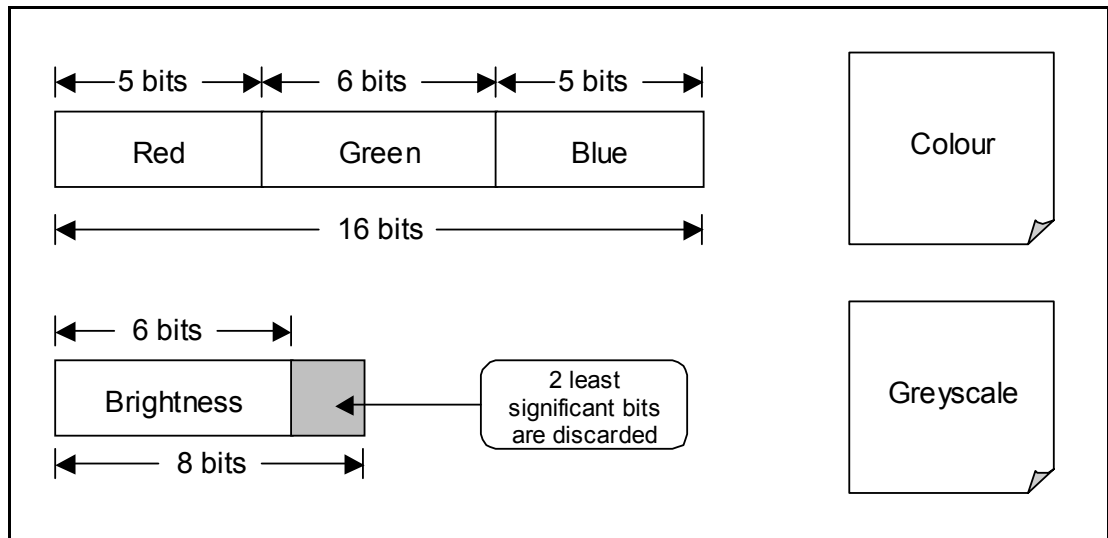


Figure 8.2: Output pixel format of Image Enhancer

Greyscale output is stored identically throughout the system, using 8-bits per pixel. Thus the resultant image need only be transferred to the appropriate location before being output to the display subsystem. Currently, as the display uses only 6-bits per greyscale pixel, the 2 least significant bits are discarded before being passed to the display. In the future it is recommended that a dithering algorithm be implemented to optimise image quality.

Colour output is handled in a slightly different manner. Once the threshold image is ready for output it is processed in the following manner. Replicas of the desired foreground and background colour pixels are stored in memory. For each pixel in the image, a colour foreground or background pixel is transferred to the equivalent location in the output buffer. While the greyscale output could be processed in the same manner, this would require a 16-bit transfer per pixel where only 8 bits are needed. Thus the differing methods are utilised to minimise the bandwidth requirement of the display subsystem [2].

9.0 Implementation

9.1 Interfacing Camera to TMS320C6711

9.1.1 Interfacing concept

The camera sends the image through the 8 bit parallel output synchronized with the pixel clock (pclk). A pulse in the vertical synchronization (Vsync) indicates the starting point of a new frame. The DSP (C6711) does not have any dedicated general purpose input / output port to support the parallel transfer of the image. The HPI was used to interface the host computer and it has its own dedicated circuit. So there were only two options left to interface the camera to the DSP. One was using the McBSP as GPIOs and the other was interfacing the camera through the EMIF.

9.1.2 Input image synchronized with the VSYNC of the camera

McBSP configuration

The C6711 has two McBSP channels, one is dedicated for audio transfer and the other is used to transfer serial data. These McBSPs can be reconfigured as GPIOs. Within the 7 pins that are available for GPIO, pin DX can only be used as an output and pin DR can only be used as an input leaving 5 pins as input/output. Hence transferring a pixel (8 bits of information) at a time required both McBSPs configured as GPIOs. Therefore the DSP chip was modified to disconnect the audio input sockets from the McBSP0.

Pin description McBSP

DX FSX CLKX CLKS FSR CLKR DR

In order to reduce the calculation time in rearranging the data to get the proper pixel value, the least significant pins (Y0-Y4) were connected to the least five significant pins of the McBSP0 and the rest of the three pins were connected to the least significant pins of the McBSP1.

Regaining the proper value pixel value needed the following operation.

$$\text{Pixel} = (8 \text{ least significant bits of McBSP0} \& 0x1F) | \\ (8 \text{ least significant bits of McBSP1} \& 0x07) \ll 5$$

EDMA configuration

Two EDMA channels (channel 4 and 5) were employed to transfer the data (8bit) from both McBSPs synchronized to the pixel clock. These EDMA channels 4 and 5 are synchronized to external interrupt 4 and 5 respectively, to which the pixel clock was connected.

The EDMA can transfer 0xFFFF (65535) elements at a time when it is configured to element synchronized transfer. But a frame contains (640x48) 307,200 pixels. So it was decided to transfer one line (0x0280 pixels) of the frame at a time and after each transfer the EDMA was configured again to transfer the next line. This process continued until one whole frame was transferred. The EDMA transfer complete interrupt (TCINT) was used to detect the transfer completion of each line.

The transferred data was stored in different memory locations and the pixel was regained during the image processing. The pixel clock was logically ANDed with the Href signal to get the clear transition in its positive edge.

Vsync of the camera was connected to the external interrupt 7. This interrupt service routine was used to synchronize the frame transfer.

Algorithm

1. The camera generates a pulse in the Vsync pin to inform the start of a new frame.
2. The interrupt routine for Vsync initializes the two EDMAs and the memory address and starts the element synchronized transfer process.
3. After one line is transferred, EDMA transfer finished interrupt is generated
4. Within this interrupt routine, EDMA element count and the destination address are updated and start the transfer for the next line. This interrupt routine should be executed before the next positive edge in the Href signal of the camera.
5. Thus the transfer operation continues until the EDMAs finish transferring the whole frame.
6. While the EDMAs are transferring a frame, the CPU processes the frame that was transferred before, applying the image processing techniques that are required by the user.

7. both EDMA and CPU waits for each other to finish and switch the working memory location and EDMA starts transferring the new frame while the CPU starts processing on the newly arrived frame

Results

The following figures were generated using the above configuration



Figure 9.1: Sample input image 1 using Vsync synchronized image capturing method



Figure 9.2: Sample input image 2 using Vsync synchronized image capturing method

The images obtained were clear for normal display. But they were not clear enough to apply the various types of image processing techniques. It was obvious that most part of the image had distorted pixel, which would make the image even worse after applying the enhancement techniques.

Another draw back here is that the algorithm was time critical. The CPU was interrupted every time a new frame arrived and also every time a new line was transferred. Hence, if the camera captures images at 3 frames per second, the CPU is interrupted 1443 ($3 \times 480 + 3$) times in a second. This was large enough to affect the image processing speed of the system.

The other issue was the waiting period of the processor. The CPU and the Camera had to wait for each other until they finish their processing. This really is time consuming and the frame rate from the camera was independent. So this would not be a better solution for image capturing.

9.1.3 Using SRAM mode of the Camera

In the SRAM mode of the camera, the Vsync is pulled high and the data bus is tri-stated. A single frame is sent to the DSP memory when a pulse is sent to the AGCEN pin of the camera or 0x03 value is written to the register 0x27 in the camera. Although the camera has an I²C interface through which the camera registers can be read or written, there were no ports available in the DSP C6711 to communicate to that interface.

Hence it was decided to request a frame by sending a pulse to the AGCEN of the camera through any of the available pin of the McBSP1. The extension of pin AGCEN from the CMOS image sensor was not provided with the image-capturing package. This needed a thin wire to be soldered to the AGCEN pin of the camera module.

The pin (DX) in McBSP0 was selected to send the frame requesting pulse to the image sensor. Writing a 0 then 1 and then 0 would generate a pulse.

McBSP setting

The McBSP settings remained the same as in the previous method except an extra wire was used to connect the pin DX to the pin AGCEN.

EDMA setting

The EDMA was allowed to run through out the transfer without any configuration changes since it was known that only one frame would be transferred as a result of a request. The element count and element count reload were set to 0x280. After the EDMA transferred each element the element count register is decremented by one. When the element count register reaches zero it is reloaded with the value indicated in the element count reload register.

Algorithm

1. The camera module is initialized to SRAM mode
2. CPU initializes the EDMA and memory block, and requests a frame by sending a pulse to the AGCEN pin
3. The EDMA starts the transfer process synchronized to the ANDed signal of pixel clock (pclk) and Href
4. During the transfer the CPU applies the enhancement techniques to the frame that was already transferred.
5. After CPU finishes its work on the previous frame it requests the new frame and starts working on the frame that has just arrived

The following figure show the image that was generated using the above algorithm. The image is clear and does not have any distorted pixels or has little distorted pixels, which hardly affect the enhancement techniques.

In this algorithm the CPU is not interrupted at all during the frame transfer and the CPU controlled the input frame rate. This is a way better solution than the previous one.



Figure 9.3: Sample input image using SRAM mode method

9.2 Image processing techniques implementation and results

9.2.1 Implementation tools

Image processing techniques were implemented in two steps. First step contained the implementation and validation of the techniques in MatLab software using image processing toolbox. The second was porting the MatLab code into C, testing, validating and optimizing the code in the TMS320C6711 platform using Code Composer Studio 6.0.

9.2.2 Sharpening using FFT

Algorithm

1. Perform FFT on the input image
2. Convolve the results with the low pass filter matrix
3. Separate the high frequency components and the low frequency components of the image by subtracting the low frequency components with the high frequency components.
4. Multiply the high frequency components with the sharpening factor (an integer)
5. Add the sharpened high frequency components with the low frequency components to get the output image

Analysis

The results were convincing. The edges were enhanced and even the darker area edges were also noticeable. But the problem was it took so long to execute. In MatLab it took about one to two seconds to finish the task. But with the DSP it took about two to three seconds to finish the sharpening. But the aim of this Image

Enhancer was to process at least three frames per second (which was the maximum frame transfer rate the EDMA and McBSP combination provided).



Figure 9.4a: Image before sharpening

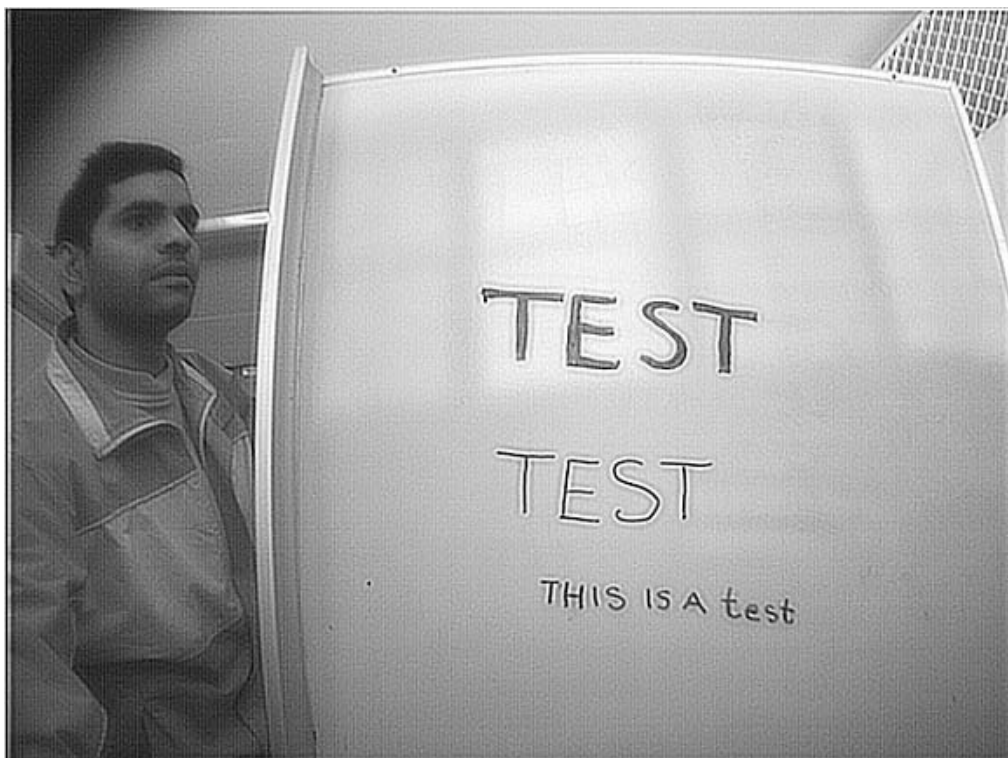


Figure 9.4b: Image after sharpening – amplifying factor > 1 – using FFT method

A second feature of the Edge Enhancement filter is the ability to reduce the significance of low frequency components. In other words, any soft transitions such as reflections or shadows across the whiteboard can be reduced or even eliminated altogether. Figure 9.4c shows the output of the Edge Enhancement filter with the background amplification at 0.2 and the normalised cut-off frequency at 0.08 [2].

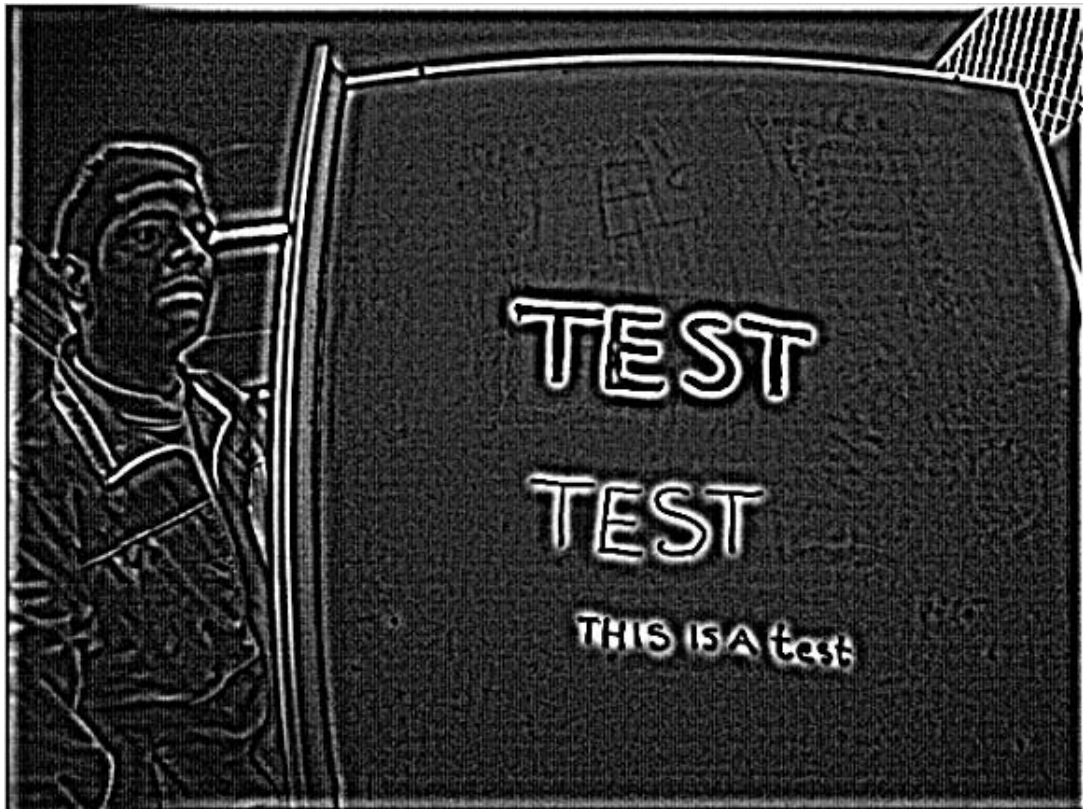


Figure 9.4c: Image after sharpening – amplifying factor 0.2 – using FFT method

9.2.3 Sharpening using spatial filters

This technique was discussed in previous chapter ‘Selection of Image Processing techniques’. Although all the spatial matrices that were suggested in that chapter would give almost the same effect on the output image, the following matrix was selected for the Image Enhancer. This is because the programming code could be optimized and the number of arithmetic operations could be reduced.

| | | |
|----|----|----|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

Table 9.1: Spatial edge enhancement matrix

Optimising

The image consists of 480x640 (307 200) pixels. While generating the output image, each pixel needed nine multiplications and nine additions. Thus the processor would perform at least $307200 \times (9 \text{ multiplications} + 9 \text{ additions})$ operations per frame.

In the matrix shown above, four elements are zeros. So there is no need to do a multiplication operation for them. Another four elements are -1 . So instead of multiplying by -1 those values can be subtracted. Thus it leaves with only one multiplication. Hence the total operations are reduced to $307200 \times (5 \text{ additions} + 1 \text{ multiplication})$. Since the multiplication takes almost double the amount of clock cycles that is required for the addition, the time taken for sharpening using spatial matrix was reduced a lot from its conventional time.



Figure 9.5: Image before convolving with edge enhancement matrix

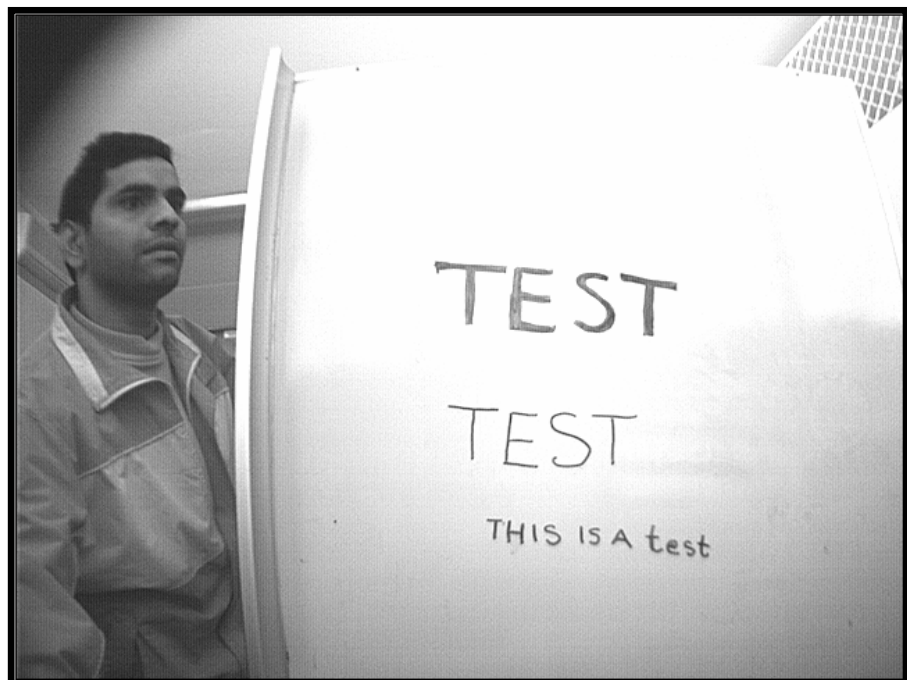


Figure 9.6: Sharpened image using spatial filter matrix

Every single part of the input image was sharpened in the output image. The only draw back of using this algorithm is that if a brighter area already exists, this algorithm makes it even brighter and not every one would prefer to see that. Giving a user input option to select a desired area for sharpening (as it was done in zoom) can eliminate this effect. Combining this technique with other techniques like threshold, zoom is another option.

9.2.4 Zoom

During the implementation period, many different algorithms for zooming were tested. The following details the algorithm, digital zoom using nearest neighbour interpolation that was selected after considering the speed of execution and clearness of the output image.

Algorithm

1. Get the starting coordinates of the zooming area and the zoom factor from the allocated registers or variables
2. Decide the area that would be zoomed
 - a. If left hand top corner coordinates is (x,y)
 - b. Then the right hand bottom coordinates will be

$$(x + \frac{\text{image width}}{\text{zoom factor}}, y + \frac{\text{image length}}{\text{zoom factor}})$$

3. Spread the pixels that were selected in equal spacing on the output image
4. Fill the blanks with the closest pixel value in the output



Figure 9.7: Modified input image before applying zoom technique

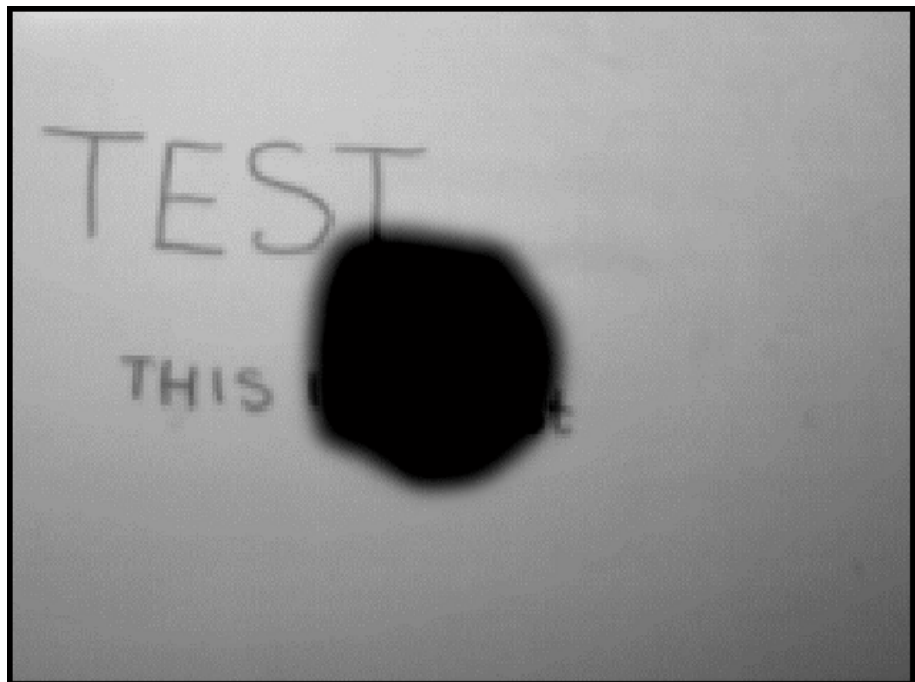


Figure 9.8: Modified input image after applying zoom technique

Analysis

The above algorithm provided clear zoomed images for zoom factors less than four. But when the zoom factor was increased over four, the image was distorted with the squaring effects. This could have been eliminated using some of the smoothing filters available widely. But when a Gaussian smoothing filter was tried, it took more than two seconds to perform the zoom. Hence it was eliminated from the Enhancer that was designed. If a better-optimized zoom algorithm is found then all those smoothing functions can be employed without disturbing the frame rate.

The following shows the Gaussian smoother that was used during the implementation period [4].

$$\frac{1}{273}$$

| | | | | |
|---|----|----|----|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

Figure 9.9: Gaussian smoothening matrix 5x5

The above two pictures explain how the zooming technique helps the students to utilize their vision in the maximum possible way. The original image contains a big blank spot in the middle of the vision. Hence some of the text or diagram may be hidden behind. Sometimes if the size of the blank spot is big compared to the text, then it may cause uneasiness and gives headache to students. Zooming made the text size bigger compared to the blank spot and since the starting point and the zoom

factor could be changed, the user has the option to select those parameters to suit his needs.

9.2.5 Focus lines

Algorithm

1. Input the centre line coordinate and the width of the focus line. Let y_c is the y-coordinate of the centre line of the focus line and w is the width of focus line (all measurements in pixels)

2. Determine the two defining lines of the focus line. Let the y-coordinates of these lines are y_{top} and y_{bottom} then

$$y_{top} = y_c - \text{int}(w/2)$$

$$y_{bottom} = y_c + \text{int}(w/2)$$

3. Subtract c ($= 0x40$) from all the pixels that have y-coordinates less than y_{top} or greater than y_{bottom}

- a. If ((input pixel's y-coordinate $< y_{top}$) OR (input pixel's y-coordinate $> y_{bottom}$)) then

- i. Output pixel = input pixel $- c$

- b. Else

- i. Output pixel = input pixel

4. The value c can be changed

Analysis

The following figure (figure 9.10) shows the output of this algorithm where c is $0x40$. The difference between the focused area and the area surround it can be increased by increasing the value for c . The lowest value the darker pixels can take is $0x00$ (colour black)



Figure 9.10: Image with the focus line

9.2.6 Log contrast

This was not used as a stand-alone function within the DSP; instead this was used as an assistive function for performing threshold on the input image. The normal contrasting was done either through the display unit control buttons or sending a command to the camera through the I²C interface.

This function increases the contrast between the foreground and background. At the same time it lightens the image [5].

Algorithm

1. Generate the histogram for the input image

The histogram array contained 256 elements representing all the possible grey levels a pixel can take (0 to 255). The array values indicate the number of occurrence of each pixel in the input image.

2. Find the pixel value that occurred the most – $\max(\text{histogram})$
3. Apply the following to each pixel of the image

$$\text{output pixel value} = c * \log(\text{abs}(\text{original pixel value}))$$

Where

$$c = \frac{255}{\log(1 + \text{abs}(\max(\text{histogram})))}$$

When this function was implemented in the MatLab, it worked without an error. But implementing on the DSP platform required great caution. In MatLab the variables do not have to be declared. The compiler automatically finds the type of the variable. But in the compiler provided with the DSP does not do that. So the variables had to be declared before its usage just like a normal C programming environment. The memory and the processing power of the DSP are small compared to the desktop PC that is available in the market today. Hence not all the variables could not be declared as double. But when the variables were declared as integers a certain part of the valuable information was lost in decimal calculation. Hence the variables were defined as float (not double) only in places that dealt with the decimal numbers, otherwise they were declared as integers.

9.2.7 Threshold

This was the simplest function that was built during the implementation, but this became the function that consumed the longest time during the implementation period.

Algorithm

The concept of this function was very simple. It took an integer variable – threshold value – as its input. It looped through all the pixels available (307 200) in the input image and set their value to 0x0 or 0xFF if they were less or greater than the threshold value respectively.

Analysis

This was executed in MatLab software first, and did not have any problem in producing the desired output. But when it was implemented in the C6711 it did not give the proper output. So a lot of time was spent on this function to figure out what went wrong.

Actually this was what happened to cause such a problem. The pointer to the start of the input image was declared as a global variable. Whenever the program entered into the threshold function, the pointer did not have the value that was allocated earlier which was the start of the input image; instead it had some other arbitrary value. This might have happened because of the self-optimizing functionality of the software (Code Composer Studio 6.0) that was used in the development. Hence to correct this problem a local pointer was declared within the threshold function and it was assigned to the start of the input image. After that modification the function executed without any error.

9.2.8 Changing the foreground and background colour

This functionality allows the user to select the foreground and background combination of the output image. This was an added functionality of the threshold technique, because threshold is the only technique that gives an output with two colours (foreground and background).

Algorithm

1. Two arrays of colour were defined.
2. Foreground colour [4] = {'Black', 'Red', 'White', 'Blue'}
3. Background colour [4] = {'white', 'Green', 'Black', 'Yellow'}
4. The 'Black' actually represented by the hex value 0x00
5. If the user selects the combination number 3, then the foreground colour and background colour will be foreground colour [3] and background colour [3] respectively.

The following pictures are the actual outputs of the functions threshold and threshold with fore ground and background selection respectively. Getting more details of the background area requires some adjustment in the threshold value. This is done through the user input interface. It also can be used to change the colour combination of the foreground and the background colour.

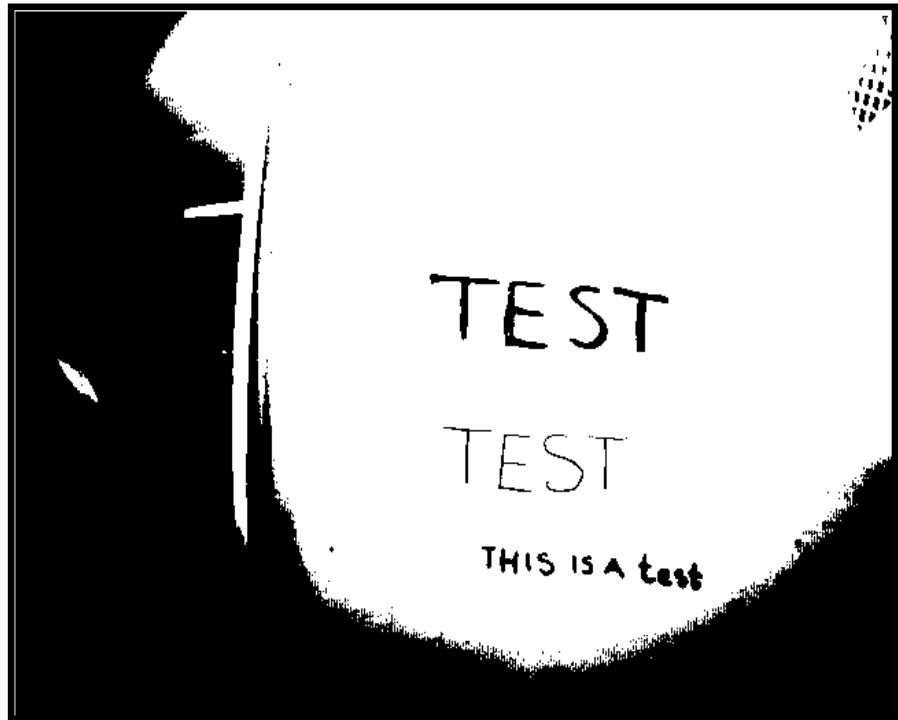


Figure 9.11: Black and white threshold image

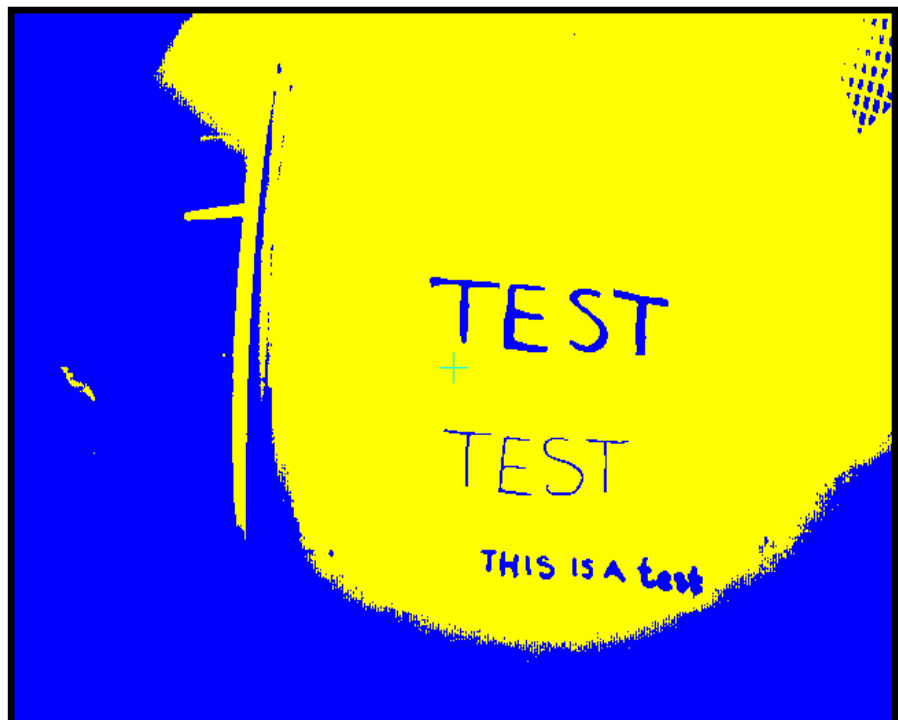


Figure 9.12: Threshold image after applying two colours

9.3 Testing on actual environment and Performance

One of the important parts of testing these implemented algorithms was to select the appropriate testing environment. Testing on some sample images that were already exists does not help in deriving any conclusions on the performance of the Image Enhancer.

Hence at the start of the project a digital camera was used to capture the still images of real time handwriting from a white board. These were used to test the validity of the algorithms that were generated both in MatLab and DSP. Then the camera was connected to the DSP and the still images were extracted from the DSP memory. This helped not only to validate the image processing techniques under the real time environment but also helped to analyse the quality of the images from the camera. This was one of the main reasons to switch the image capturing technique from Vsync synchronized to SRAM mode technique.

The distance between the camera and the white board that was used for this testing was around one meter. This was because the lens that was provided with the camera was not powerful enough to capture the clear images of the far away objects. But this lens can be replaced later according to the classroom requirements.

9.4 Performance

The Image Enhancer was supposed to display the images continuously and without any jitter. But at the moment the device is not connected to the display unit. This display unit is supposed to get the images from the memory blocks and display on the screen [7]. So it was difficult to find the over all performance of the device. So

within the program, the number of frames captured was varied and an estimate was taken for the frame rate. In this Enhancer the image processing and image transferring occurs at the same time. Hence whichever takes the longest time is going to decide the frame rate of the device. At this point the maximum number of frames that the EDMA can support is three. Hence the maximum transfer rate for this device is three frames per second. When the image processing part was included in the execution, the rate dropped to two frames per second and might drop to one frame per second after the screen is connected. But this was only because the EDMA supports only three frames per second. This might have happened because the pixel was transferred through two EDMA channels and it was element synchronized. Had the pixel been transferred through one port (HPI or EMIF), it would have made to support may be fifteen to twenty frames per second.

10.0 Conclusion and Future directions

In this thesis, a prototype of an Image Enhancer for the visually impaired students was designed and implemented. Although there are many devices available for the purpose of assisting the visually impaired, there are not many devices that help the students in all possible ways. Hence this device was designed in a way so that it includes the solution for the drawbacks the other devices carry. This is a low cost portable image enhancer with a simple user input facility to be used not only to read the texts and diagrams which are closer but also to read from the far away sources such as white board or the theatre screen.

DSP platform

The main part of this design process was to select suitable platform and other components such as camera and display screen. It was decided to use TMS320C6711, one of the Texas Instruments' C6000 series, as the platform because it is a floating point DSP designed for high-precision applications. The only problem faced using C6711 platform was that it does not have any dedicated GPIO port. So the image had to be input into the DSP memory using the two McBSP channels, which were reconfigured as GPIO. This required two EDMA channels transferring the input image and at least requires another EDMA channel to transfer the output image (at the time of writing this conclusion the screen has not been connected to the DSP to check and validate the device's performance). This not only slowed down input frame rate of the enhancer, but also consumed some of the valuable CPU cycles, which might have used for the other calculations.

So one of the important developments suggested to be done in the future is generating a new method to interface the image-capturing unit to the DSP through HPI or EMIF. The easiest solution is to select another C6000 series, which have a dedicated GPIO. Now there is a DSP evaluation board available in the Electrical and Computer engineering department, which has a dedicated camera interface port and I2C port. These properties might make the interfacing lot easier and simpler.

Image capturing unit

The camera used in this device has a focusing lens with 3.5mm focal length. This lens is not suitable for focusing far away objects. That is why the device was tested on the closer object rather on the objects that were far away. So the first thing to do with the image-capturing unit is to change the lens. An optical lens with variable focal length is the best solution because it can then capture the objects that situated closer or far away. Availability also was the factor that heavily influenced the selection of this camera. There are lot of cameras that would have suit this application. But none of them were (or at least could not find any of them) available for retail selling in Australia at the time of development.

The SRAM mode operation is the big advantage that this selected camera has. This allows the DSP to control the input frame rate of the Image Enhancer.

Image Processing

The device includes the image processing techniques that is sufficient to assist most of the visual impairment. It also gives the choice of changing some of the input factors to have a different level of processing. All of the techniques were executed

fast except the smoothening function (this was the reason that the smoothening function was removed from the device). Edge detection technique was not employed because the latency in executing the Fast Fourier Transform (FFT) function was high enough to slow down the processing power drastically. It was a set back not knowing that the moving average calculation can replace the need of the FFT.

Another way to increase the processing speed is to use the optimized function libraries written in assembly language. This not only increases the frame rate, but also increases the variety of image processing techniques that can be employed within the device. This was not used because of the unavailability of licence to use that function.

So far the device only captures the images, processes it and sends it to the display; it does not care about the contents of the images. This may result in some unwanted or corrupted outputs being displayed. In the future a hybrid intelligent system of artificial neural networks, fuzzy logic can be implemented within the device so it intelligently recognize the text and diagrams and display them in a proper and clearer manner. Using image segmentation techniques also makes the detection easier.

User Input Interface

The user input interface that was designed was a very basic design. The main purpose of this design was to show that a simple interface could be developed so that the image enhancement techniques can be controlled and combined together to suit different users in different environments. This interface allows the user to change the image processing parameters in the easiest manner.

The main draw back of this simple design is that if the user keeps pressing the keypads continuously then the CPU will be busy managing the input combinations since the inputs are connected to the interrupts. This slows down the image-processing rate and provides corrupted outputs. In the future the input buttons can be connected to the DSP through some flip-flops to overcome this problem. So the input variables can be stored in the flip-flops and the CPU can access it whenever it starts to work on a new image. Since the keypad is not connected to the interrupts, the CPU will not be interrupted even the keypads are pressed continuously. Providing audio or on screen feed backs about the inputs also increase the quality of this device.

12.0 Bibliography

- 1) Arthur, R. Myler, Harley, R. 1993, The pocket handbook of image processing algorithms in C. Prentice Hall, New Jersey.
- 2) Beaumont-Field, J. 2004, Image Acquisition and Processing on the Low Vision Image Enhancer, Australia.
- 3) Ferris, D. 2004, Retinitis Pigmentosa and Retinal Degeneration. Downloaded from 'http://www.netserv.net.au/doonbank/rd.html' in October 2004.
- 4) Fisher, R. et al. 2003, Gaussian Smoothing, Digital Filters. Downloaded from 'http://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm' in October 2004.
- 5) Fisher, B. et al. 1994. Logarithm Operator, Point Operation. Downloaded from 'http://www.cee.hw.ac.uk/hipr/html/pixlog.html'
- 6) Litwiller, D. 2001. CCD vs CMOS: Facts and Fiction, DALSA technology with vision. Downloaded from 'http://www.dalsa.com/shared/content/Photonics_Spectra_CCDvsCMOS_Litwiller.pdf' in September 2004.
- 7) Lowe, J. 2004. Display Subsystem Development for the Low Vision Image Enhancer, Australia.

- 8) McIrvin, M. 2000, Colour perception in Human eyes and brains. Downloaded from 'http://world.std.com/~mmcirvin/bluesky/eyes.html' in July 2004.
- 9) Perez, C. New vision-enhancement system could work miracles for patients. Downloaded from 'http://www.stp.uh.edu/vol61/951012/1d.html' in September 2004.
- 10) Sid-Ahmed, & Maher, A. 1995. Image processing : theory, algorithms, and architectures. McGraw-Hill, New York.
- 11) Windsor, R. L. & Windsor L. K. 2001. Low Vision Rehabilitation: An Introduction Downloaded from 'http://www.eyessociates.com/images/an_introduction_to_low_vision_re.htm'
- 12) Age Related Changes 2004, downloaded from 'http://www.innvista.com/health/ailments/eyeail/agerelch.htm'
- 13) Australian Bureau of Statistics 2003. Statistics in eye health in Australia, Downloaded from 'http://www.visionaustralia.org.au/index.asp' in October 2004.
- 14) Australian Bureau of Statistics 2003. Statistics in eye health in Australia, Downloaded from 'http://www.visionaustralia.org.au/index.asp' in October 2004.
- 15) Help for the visually Impaired 1995. downloaded from 'http://vesuvius.jsc.nasa.gov/er/seh/pg66s95.html'

- 16) Low Vision Help in Retinitis Pigmentosa 2003. Downloaded from
'http://www.eyecassociates.com/low_vision_help_in_retinitis_pig.htm'
- 17) M3188A – 1/3” Digital Output Monochrome Camera Module 2004. QUASAR
electronics. Downloaded from '<http://www.quasarelectronics.com/m3188a.htm>'
- 18) Nature of Macular Degeneration, Fighting Blindness, Western Australian
Retinitis Pigmentosa Foundation. Downloaded from
'<http://members.iinet.net.au/~warpf/md.html>' in July 2004.
- 19) RP Primer, RP/Visual Field Augmentation, theworkshop.ca. Downloaded from
'<http://www.theworkshop.ca>' in August 2004.
- 20) Telesensory 2002. Aladdin Rainbow Pro. Products – Product Catalog,
Telesensory Corporation. Downloaded from '<http://www.telesensory.com/products2-1-7.html>' in August 2004.
- 21) Telesensory 2002. Retinitis Pigmentosa. Vision Resources, Telesensory
Corporation. Downloaded from '<http://www.telesensory.com/vision2.html>' in
August 2004.
- 22) Texas Instruments 2004. Multichannel Buffered Serial Port (McBSP).
TMS320C6000 DSP Peripherals Overview, Texax Instruments Inc.

23) Texas Instruments 1999.Peripherals. TMS320C6000 Technical Brief, Texas Instruments Inc.

24) The Low Vision 1995. Closed Circuit Video Magnification Systems.

Downloaded from ‘http://www.lowvision.org/closed_circuit_television_system.htm’

25) The Optelec Clear View 317, Optelec Low Vision Products, NanoPac, Inc.

Downloaded from ‘<http://www.nanopac.com/Optelec.htm>’ in September 2004.

26) Vision Simulations 2003. Downloaded from

‘<http://www.thechicagolighthouse.org/ser-eduAMD.htm>’

27) What is visual Impairment 2004.Download from

http://kidshealth.org/kid/health_problems/sight/visual_impaired.html

Appendix A – Function listings in C

B.1 EDMA and McBSP configuration Listing

```
/* Include Header File */
#include "pingpongcfg.h"

/* Config Structures */
EDMA_Config edmaCfg0 = {
    0x50310000,    /* Option */
    0x018C0024,    /* Source Address - Numeric */
    0x01DF0280,    /* Transfer Counter */
    0x800A9000,    /* Destination Address - Numeric */
    0x00000000,    /* Transfer Index */
    0x02800000     /* Element Count Reload and Link Address */
};

EDMA_Config edmaCfg1 = {
    0x50200000,    /* Option */
    0x01900024,    /* Source Address - Numeric */
    0x01DF0280,    /* Transfer Counter */
    0x800F5000,    /* Destination Address - Numeric */
    0x00000000,    /* Transfer Index */
    0x02800000     /* Element Count Reload and Link Address */
};

MCBSP_Config mcbSPCfg0 = {
    0x00000000,    /* Serial Port Control Reg. (SPCR) */
    0x000000A0,    /* Receiver Control Reg. (RCR) */
    0x000000A0,    /* Transmitter Control Reg. (XCR) */
    0x203F1F0F,    /* Sample-Rate Generator Reg. (SRGR) */
    0x00000000,    /* Multichannel Control Reg. (MCR) */
    0x00000000,    /* Receiver Channel Enable(RCER) */
    0x00000000,    /* Transmitter Channel Enable(XCER) */
    0x00003000     /* Pin Control Reg. (PCR) */
};

/* Handles */
EDMA_Handle hEdmaExtint4;
EDMA_Handle hEdmaExtint5;
MCBSP_Handle hMcbSP0;
MCBSP_Handle hMcbSP1;
/* ===== CSL_cfgInit() =====
*/
void CSL_cfgInit()
{
    hEdmaExtint4 = EDMA_open(EDMA_CHA_EXTINT4, EDMA_OPEN_RESET);
    hEdmaExtint5 = EDMA_open(EDMA_CHA_EXTINT5, EDMA_OPEN_RESET);
    hMcbSP0 = MCBSP_open(MCBSP_DEV0, MCBSP_OPEN_RESET);
    hMcbSP1 = MCBSP_open(MCBSP_DEV1, MCBSP_OPEN_RESET);
    EDMA_config(hEdmaExtint4, &edmaCfg0);
    EDMA_config(hEdmaExtint5, &edmaCfg1);
    MCBSP_config(hMcbSP0, &mcbSPCfg0);
    MCBSP_config(hMcbSP1, &mcbSPCfg0);
}
```

B.2 Input Image Capturing Listing

```
#define CHIP_6711
#define _TI_ENHANCED_MATH_H 1
#include <math.h>
#include <std.h>
#include <stdio.h>
#include <log.h>
#include <hwi.h>
#include <csl.h>
#include <csl_irq.h>
#include <csl_mcbasp.h>
#include <csl_edma.h>
#include "pingpongcfg.h"

//first define memory locations
#define FRAME_ONE_A 0x800A9000 //Memory location of frame
#define FRAME_ONE_B 0x800F5000 //buffers

#define FRAME_TWO_A 0x80141000 //Have left 0x1000 between
#define FRAME_TWO_B 0x8018D000 //each data segment

#define PROCESSED_FRAME_START 0x801D9000 //start of actual frame
#define FINAL_FRAME          0x80225000    // start of processed frame

//list of globals needed
int EDMA_finished,
    new_frame,
    count,
    EDMA_number,
    CPU_number,
    CPU_finished;

int swap_times; //needed if we only want it to swap a certain number of times
unsigned char *processed_start = (unsigned char *)PROCESSED_FRAME_START,
    *PCR0 = (unsigned char *)0x018C0024;
unsigned int *PCR1 = (unsigned int *)0x01900024;

unsigned char *EDMA_frame_a, //pointer to data segment A for EDMA
    *EDMA_frame_b, //pointer to data segment B for EDMA

int enable;

/*function declarations*/
void initialise(void); //setup variables/interrupts. run once at start
void EDMA_frame_finished(void); //called when EDMA transfer completed
void CPU_process(void); //post processing of frame

void main(void)
{
    initialise();
    for (count=0; count<0x4d000; count++)
```

```

        {
            *((unsigned char *)FRAME_ONE_A+count)=0x00;
            *((unsigned char *)FRAME_ONE_B+count)=0x00;
            *((unsigned char *)FRAME_TWO_A+count)=0x00;
            *((unsigned char *)FRAME_TWO_B+count)=0x00;
            *((unsigned char *)PROCESSED_FRAME_START+count) =
0x00;
        }
        count=0;

        while(EDMA_finished<15) //transferring 15 frames
        {
            if (new_frame==0)
            {
                EDMA_enableChannel(hEdmaExtint4); //first set up
                EDMA_enableChannel(hEdmaExtint5); //EDMA transfer
                new_frame=1;
                *PCR1 = 0x3010; //then send a pulse to the M3188A
                *PCR1 = (*PCR1 | 0x3030);
            }
            CPU_process(); //once the frame is received, post-process it
        }
        EDMA_close(hEdmaExtint4); //once all frames are received
        EDMA_close(hEdmaExtint5); //tidy up and finish
        MCBSP_close(hMcbasp1);
        MCBSP_close(hMcbasp0);

        printf("finished and all over \n");
    }

    void initialise(void)
    {
        EDMA_finished=0;
        new_frame=0;
        EDMA_number = 1;
        CPU_number = 0;
        CPU_finished = 1;

        IRQ_resetAll(); //clear all interrupts
        EDMA_intClear(1);
        EDMA_intEnable(1); //enable TCINT 1

        IRQ_enable(IRQ_EVT_EDMAINT);
        IRQ_globalEnable();
    }

    void EDMA_frame_finished(void)
    {
        //this function is called by the edma when it is finished a frame

        EDMA_disableChannel(hEdmaExtint4);
        EDMA_disableChannel(hEdmaExtint5);
    }

```

```

printf("frame finished %d\n",EDMA_finished);
EDMA_intClear(1);//clear the cipr register so this
//function will trigger again

if(CPU_finished) //once the frame finishes, switch buffer
{
    if(EDMA_number == 1)
    {
        EDMA_number = 2;
        CPU_number = 1;
        edmaCfg0.dst = FRAME_TWO_A;
        edmaCfg1.dst = FRAME_TWO_B;
    }
    else
    {
        EDMA_number = 1;
        CPU_number = 2;
        edmaCfg0.dst = FRAME_ONE_A;
        edmaCfg1.dst = FRAME_ONE_B;
    }
    CPU_finished = 0;
}
EDMA_config(hEdmaExtint4,&edmaCfg0); //reset the EDMA channels
EDMA_config(hEdmaExtint5,&edmaCfg1); //for each frame

EDMA_finished++;
new_frame = 0;
}

void CPU_process(void)
{
    long int read_counter;
    int i;
    unsigned char *CPU_frame_a,
                  *CPU_frame_b;
    if(CPU_number != 0)
    {
        if(CPU_number==1)
        {
            CPU_frame_a = (unsigned char *)FRAME_ONE_A;
            CPU_frame_b = (unsigned char *)FRAME_ONE_B;
        }
        else
        {
            CPU_frame_a = (unsigned char *)FRAME_TWO_A;
            CPU_frame_b = (unsigned char *)FRAME_TWO_B;
        }
        //this for loop reads both half pixels at once
        //combines them into a single pixel and then
        //writes the output to processed_start
        for(i=0;i<256;i++)
        histo[i] = 0;
        for(read_counter=0;read_counter<307200;read_counter++)
        {
            //entire process done in one line to save time

```

```

*(processed_start+read_counter) = ((*CPU_frame_b+read_counter) & 0x0F)<<4)
                                   |(*CPU_frame_a+read_counter) & 0x0F);
    // histogram
}

CPU_finished = 1;
}

```

B.3 Focus Line generation Listing

```
void focusLine()
{
    int const rows=480;
    int const cols=640;
    int i,line1,line2,centre,width;
    unsigned char * input = (unsigned char *) input_location;
    unsigned char * output = (unsigned char *) output_location;

    centre=302;
    width=60;
    line1=centre-round(0.5*width);
    line2=centre+round(0.5*width);

    for (i=0;i<rows*cols;i++)
    {
        if ((i<(line1-1)*640)|(i>=(line2+1)*640)) //if outside guidelines
        {
            if (*(input+i)<0x40)
                *(output+i)=0x00; //cant get any darker than
black...
            else
                *(output+i)=*(input+i)-0x40;//make image darker

        }
        else if ((i>(line1+2)*640)&(i<(line2-2)*640)) //if inside guidelines
        {
            *(output+i)=*(input+i);
        }
        else //otherwise, there should be a line there
        {
            *(output+i)=0x00; //a black line
        }
    }

    printf("completed\n");
}
```

B.4 Threshold Listing

```
#define ROWS 480
#define COLS 640
#define IMAGE_IN 0x800A9000
#define IMAGE_OUT 0x80240000
```

```
#define TEMP 0x800F5000
#define HISTO 0x80140002
```

```
int histo[256]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    histo_max,
    histo_max_location;

//unsigned char *out = (unsigned char *)IMAGE_OUT;
//unsigned char *inp = (unsigned char *)IMAGE_IN;
float s_factor;

void logcontrast(void);
void histogram(void);
int assumption_threshold(void);
int last_sig_histo();
int sig_factor(void);
void threshold(int);
unsigned char red[]={0xFF,0x00,0x00,0xFF,0x00,0xFF},
               blue[]={0x00,0xFF,0x00,0x00,0x00,0xFF},
               green[]={0x00,0x00,0xFF,0xFF,0x00,0xFF};

unsigned int foreground;
unsigned int background;

/*****
 * this function finds the distribution of the pixel values between
 * 0 and 255
 *****/
void histogram()
{
```

```

unsigned char *tem = (unsigned char *)TEMP;

int i;
unsigned char ch;

for(i=0;i<307200;i++)
{
    ch = *(tem+i);
    histo[ch]++ ;
}
}

int assumption_threshold()
{

    unsigned int thresh_value;
    int i;

    histo_max=0;
    for (i=0;i<256;i++)
    {
        if (histo[i]>histo_max)
        {
            histo_max=histo[i];
            histo_max_location=i;
        }

        if (histo_max_location<250)
        {
            thresh_value=histo_max_location-(255-histo_max_location);
        }
        else
        {
            thresh_value=histo_max_location-(255-histo_max_location+5);
        }
    }
    return thresh_value;
}

void threshold(int thresh_value) //this function generates the thresholded image
{
    int i;
    unsigned char *tem = (unsigned char *)IMAGE_IN;
    unsigned char *output = (unsigned char *)IMAGE_OUT;

    for(i=0;i<302700;i++)
    {
        if( *(tem+i)<thresh_value)
        {

```



```

        *(output+(i)*3) = red[foreground];
        *(output+(i)*3+1) = green[foreground];
        *(output+(i)*3+2) = blue[foreground];
    }
    else
    {
        *(output+(i*3)) = red[background];
        *(output+i*3+1) = green[background];
        *(output+i*3+2) = blue[background];
    }
}
}

int last_sig_histo()
{
    int flag;
    int i;
    unsigned int p_end;

    /* the following finds the last significant pixel of the histo array
    */

    flag = 0;
    i=255;
    do
    {
        if((histo[i]>s_factor) & (histo[i-1]>s_factor))
        {
            flag = 1;
            p_end = i;
        }
        i--;
    }
    while(i>=0 && flag==0);

    return p_end;
}

int sig_factor()
{
    float factor;
    unsigned int max;
    int i;

    /* the following loop finds the maximum value of the distribution array
    histo: see histogram for the values in histo
    */
    max = 0;
    for(i=0;i<256;i++)
    {
        if(histo[i]>max)
            max = histo[i];
    }
    /* a pixel value is significant if it occurs more than 0.05 times the
    occurance of the most occuring pixel

```

```

    */
    factor = round(0.05*max);//rounding the value

    return factor;
}

int first_sig_histo()
{
    int flag;
    int i;
    unsigned int p_start;

    /* the following finds the last significant pixel of the histo array
    */

    flag = 0;
    i=0;
    do
    {
        if((histo[i]>s_factor)// & (histo[i+1]>s_factor))
        {
            flag = 1;
            p_start = i;
        }
        i++;
    }
    while(i<256 && flag==0);

    return p_start;
}

```

B.5 Zoom Listing

```
void zoom()
{
int const rows=480;
int const cols=640;
float const factor=2.0;
int y_offset=300,x_offset=270;
    float
average[3][3]={ {1.0/15,2.0/15,1.0/15},{2.0/15,3.0/15,2.0/15},{1.0/15,2.0/15,1.0/15}};

    int row_zoomed,col_zoomed,i_in,j_in;
    int equiv_0_0,
        equiv_0_1,
        equiv_0_2,
        equiv_1_0,
        equiv_1_2,
        equiv_2_0,
        equiv_2_1,
        equiv_2_2;
    int i,j,out_pointer_offset,equiv_input_offset;
    unsigned char * input = (unsigned char *) input_location;
    unsigned char * output = (unsigned char *) output_location;


    row_zoomed=rows/factor;
    col_zoomed=cols/factor;


    for (i=0;i<640*480;i++){
        *(output+i)=0x00;
    }

    if (x_offset>(480-(480/factor)))
    {
        x_offset=480-(480/factor)-1;
    }

    if (y_offset>(640-(640/factor)))
    {
        y_offset=640-(640/factor)-1;
    }


    for (i=0;i<row_zoomed;i++)
    {
        for (j=0;j<col_zoomed;j++)
        {
            /*spread the pixels out*/

            *(output+((i*(int)factor*cols)+(j*(int)factor)))=*(input+(((i+x_offset)*cols)+(j+y_
offset)));
        }
    }
}
```

```

//now to interpolate
for (i=0;i<480;i++)
{
    for (j=0;j<640;j++)
    {
        //first we need to know what pixel we are talking about
        out_pointer_offset=i*640+j;

        if (*(output+out_pointer_offset)==0x00)
        {

            //calculate the closest equivalent pixel in input frame
            i_in=round((i/factor))+x_offset;
            j_in=round((j/factor))+y_offset;
            equiv_input_offset=(i_in)*640+(j_in);
            equiv_0_0=equiv_input_offset-641;
            equiv_0_1=equiv_input_offset-640;
            equiv_0_2=equiv_input_offset-639;
            equiv_1_0=equiv_input_offset-1;
            //note that 1_1 is equiv_input_offset
            equiv_1_2=equiv_input_offset+1;
            equiv_2_0=equiv_input_offset+639;
            equiv_2_1=equiv_input_offset+640;
            equiv_2_2=equiv_input_offset+641;

            *(output+out_pointer_offset)=
                average[0][0] * *(input+equiv_0_0)+
                average[0][1] * *(input+equiv_0_1)+
                average[0][2] * *(input+equiv_0_2)+
                average[1][0] * *(input+equiv_1_0)+
                average[1][1] * *(input+equiv_input_offset)+
                average[1][2] * *(input+equiv_1_2)+
                average[2][0] * *(input+equiv_2_0)+
                average[2][1] * *(input+equiv_2_1)+
                average[2][2] * *(input+equiv_2_2);

        }
    }
}

printf("completed\n");
}

```

B.6 Edge Enhancement Listing

```
void edge()
{
    double edge_strength=1;
    double gx[3][3]={1, 0, -1},{2, 0, -2},{1, 0, -1}};
    double gy[3][3]={-1, -2, -1},{0, 0, 0},{1, 2, 1}};
    unsigned char * line_ref;
    long int i,j,row_pos;
    double horiz_edge,vert_edge;
    float pixel_value,max_edge=0.0;
    unsigned char * input = (unsigned char *) input_location;
    unsigned char * output = (unsigned char *) output_location;
    unsigned char * edge = (unsigned char *) edge_location;

    for (i=0;i<479;i++)
    {
        //read position line_ref over the rows we need
        row_pos=i*640;
        line_ref = (unsigned char *)(input+(row_pos));
        for (j=1;j<639;j++)
        {
            //now perform convolution
            horiz_edge=edge_strength*gx[0][0]*(int)*(line_ref+(j-1))+
                edge_strength*gx[0][1]*(int)*(line_ref+j)+
                edge_strength*gx[0][2]*(int)*(line_ref+(j+1))+
                edge_strength*gx[1][0]*(int)*(line_ref+640+(j-1))+
                edge_strength*gx[1][1]*(int)*(line_ref+640+j)+
                edge_strength*gx[1][2]*(int)*(line_ref+640+(j+1))+
                edge_strength*gx[2][0]*(int)*(line_ref+1280+(j-1))+
                edge_strength*gx[2][1]*(int)*(line_ref+1280+j)+
                edge_strength*gx[2][2]*(int)*(line_ref+1280+(j+1));

            vert_edge=edge_strength*gy[0][0]*(int)*(line_ref+(j-1))+
                edge_strength*gy[0][1]*(int)*(line_ref+j)+
                edge_strength*gy[0][2]*(int)*(line_ref+(j+1))+
                edge_strength*gy[1][0]*(int)*(line_ref+640+(j-1))+
                edge_strength*gy[1][1]*(int)*(line_ref+640+j)+
                edge_strength*gy[1][2]*(int)*(line_ref+640+(j+1))+
                edge_strength*gy[2][0]*(int)*(line_ref+1280+(j-1))+
                edge_strength*gy[2][1]*(int)*(line_ref+1280+j)+
                edge_strength*gy[2][2]*(int)*(line_ref+1280+(j+1));

            *(edge+(i*640)+j)=(int)sqrt((vert_edge*vert_edge+horiz_edge*horiz_edge));
            if (*(edge+(i*640)+j)>max_edge)
                max_edge=*(edge+(i*640)+j);
        }
    }

    for (i=0;i<640*480;i++){
        pixel_value=round(((input+i) * (edge+i)/max_edge))+*(input+i);
    }
}
```

```
        if(pixel_value>255)
            *(output+i)=0xFF;
        else
            *(output+i)=pixel_value;
    }

    printf("completed\n");
}
```

B.7 User Input Interface Listing

```
#define PCR0_ADDR 0x018C0024
#define PCR1_ADDR 0x01900024

void initialise(); //set camera/interrupts/edma/default params
void check_UI();
void adjust_threshold();
void adjust_edge();
void adjust_zoom();
void adjust_focus();

typedef struct
{
    unsigned int enable;
    signed int offset;
    unsigned char foreground;
    unsigned char background;
} thresh;

typedef struct
{
    unsigned int enable;
    float strength;
} edge;

typedef struct
{
    unsigned int factor;
} zoom;

typedef struct
{
    unsigned int enable;
    int position;
    int width;
} focus;

int    UI_prev,UI_comb;
int    current_parameter=0;

unsigned char *pcr0 = (unsigned char *)PCR0_ADDR;

unsigned char *pcr1 = (unsigned char *)PCR1_ADDR;
thresh thresh;
edge edge_enhance;
zoom zoom_in;
focus focus_lines;

void main()
{
    //step 1 : set everything up
    initialise();
```

```

//step 2 : begin main loop
while (1)
{
    //check user inputs
    check_UI();
    printf("The current UI number is : %d\n",current_parameter);
    //grab frame
    //process frame
    //output frame
}

//step 3 : shutdown
//power_down();
}

void initialise()
{
    threshold.enable=0;
    threshold.offset=0;
    threshold.foreground=0x00;
    threshold.background=0xFF;

    edge_enhance.enable=0;
    edge_enhance.strength=1;

    zoom_in.factor=1;

    focus_lines.enable=0;
    focus_lines.position=240;
    focus_lines.width=20;
    UI_prev=0;
}

void check_UI()
{
    UI_comb=(*pcr0&0x10)>>4|(*pcr0&0x40)>>5|(*pcr1&0x40)>>4;
    if (UI_comb==UI_prev)
    {
        UI_comb=0;
    }
    else
    {
        UI_prev=UI_comb;
    }

    switch(UI_comb)
    {
        case 4 : if (current_parameter==1)
                    current_parameter=2;
                else if (current_parameter==2)
                    current_parameter=3;
                else

```



```

        current_parameter=1;
        break;

case 2 : if (current_parameter==4)
        current_parameter=5;
        else
        current_parameter=4;
        break;

case 1 : current_parameter=6;
        break;

case 5 : if (current_parameter==7)
        current_parameter=8;
        else if (current_parameter==8)
        current_parameter=9;
        else
        current_parameter=7;
        break;
default : printf("default\n");
    }
}

void positive_interrupt()
{
    switch(current_parameter)
    {
        case 1:
            threshold.enable = 1;
            break;
        case 2:
            threshold.offset++;
            break;
        case 3:
            threshold.foreground=(threshold.foreground+1) % 3;
            break;
        case 4:
            edge_enhance.enable = 1;
            break;
        case 5:
            edge_enhance.strength+=0.2;
            break;
        case 6:
            zoom_in.factor=(zoom_in.factor + 1 )%3;
            break;
        case 7:
            focus_lines.enable = 1;
            break;
        case 8:
            focus_lines.position+=2;
            if((focus_lines.position + focus_lines.width/2)>480)
                focus_lines.position = 480-focus_lines.width/2;
            break;
        case 9:
            focus_lines.width +=2;

```

```

        if(focus_lines.position>240)
        { if((focus_lines.position + focus_lines.width/2)>480)
            focus_lines.width-=2;
        }
        else
        {
            if((focus_lines.position - focus_lines.width/2)<0)
            focus_lines.width-=2;
        }
        break;
    //default;
}
}

```

```

void negative_interrupt()
{
    switch(current_parameter)
    {
        case 1:
            threshold.enable = 0;
            break;
        case 2:
            threshold.offset--;
            break;
        case 3:
            threshold.background=(threshold.background+1) % 3;
            break;
        case 4:
            edge_enhance.enable = 0;
            break;
        case 5:
            edge_enhance.strength-=0.2;
            break;
        case 6:
            if(zoom_in.factor != 1)
                zoom_in.factor=(zoom_in.factor - 1)%3;
            break;
        case 7:
            focus_lines.enable = 0;
            break;
        case 8:
            focus_lines.position-=2;
            if((focus_lines.position - focus_lines.width/2)<0)
                focus_lines.position = focus_lines.width/2;
            break;
        case 9:
            focus_lines.width -=2;
            if(focus_lines.width<5)
                focus_lines.width = 5;
            break;
    //default;
    }
}
}

```

B.8 Fuzzy threshold

```
int fuzzy_threshold()
{
    float y[256];
    int x[256];
    unsigned int p_start, p_end, p_range;
    unsigned int thresh_value;
    int i;
    unsigned int sum;
    unsigned int histo0;
    unsigned char *tem = (unsigned char *)PROCESSED_FRAME_START;

    p_start = first_sig_histo(s_factor);
    p_end = last_sig_histo(s_factor);

    p_range = p_end - p_start;

    for(i=0;i<256;i++)
    {
        x[i]=i;
        y[i] = 0;
    }

    zmf(x,y,round(p_start+0.1*p_range), round(p_start+0.4*p_range));

    sum = 0;

    for(i=0;i<307200;i++)
        sum = sum + (y[(*(tem+i)+1)]);

    //sum = sum/1000;
    thresh_value = p_start;

    histo0 = histo[0];

    while(histo0<sum)
    {
        thresh_value = thresh_value + 1;
        histo0 = histo0 + histo[thresh_value];
    }

    return thresh_value;
}

void zmf(int x[],float y[],int start, int end)
{
    int i;
    int range;
    float mid;

    range = end - start;
    mid = (end+start)/2;
```

```

for(i=0;i<256;i++)
{
    if(x[i]<start)
        y[i] = 1;
    else if(start<=x[i] && x[i]<=mid)
    {
        y[i] = (((float)x[i]-start)/range);/*^2;*/
        y[i] = 1-2*y[i] * y[i];
    }
    else if(mid<x[i] && x[i]<=end)
    {
        y[i] = (((float)end-x[i])/range); /*^2;*/
        y[i] = 2*y[i] * y[i];
    }
    else
        y[i]=0;

    //printf("\ny %d = %f",i,y[i]);

}

}

```

Appendix B – Function Listings in MatLab

C.1 Fuzzy threshold

```
function [output,l_list]=fuzzy_threshold(input)

input=double(input);
histo=histogram(input);
[rows,cols]=size(input);
output=zeros([rows,cols]);

sig_factor=round(0.05*max(histo));
%a pixel value is significant if it occurs more than 0.05 times the occurrence
%of the most occuring pixel

flag=0;
p_start=0;
for i=1:length(histo) %for each pixel value
    if (histo(i)>sig_factor)&&(flag==0) %check if its significant
        flag=1; %if yes, define it as the start
        p_start=i;
    end;
end;

flag=0;
p_end=0;
for i=length(histo):-1:1 %for each pixel value
    if (histo(i)>sig_factor)&&(flag==0) %check if its significant
        flag=1; %if yes, define it as the end
        p_end=i;
    end;
end;

p_range=p_end-p_start;
x=0:1:255;
y=zmf(x,[round(p_start+0.1*p_range),round(p_start+0.4*p_range)]);
sum=0;

for i=1:rows
    for j=1:cols
        sum=sum+y(input(i,j)+1);
    end
end

thresh_value=p_start;
i=histo(1);
while (i<sum)
    thresh_value=thresh_value+1;
    i=i+histo(thresh_value);
end;

output=uint8(threshold(input,thresh_value));
```

C.2 Log-contrast

```
function output=log_contrast(input)
input=double(input);
[rows,cols]=size(input);
output=zeros([rows,cols]);
histo=histogram(input);
flag=0;
for i=256:-1:1
    if (histo(i)>0.1*max(histo))&(flag==0)
        r=i-1;
        flag=1;
    end
end

c=255/log(1+abs(r));
for i=1:1:rows
    for j=1:1:cols
        output(i,j)=c*log(abs(input(i,j)));
    end
end
output=uint8(output);
```

C.3 threshold

```
function output=threshold(input,level)
input=double(input);
[rows,cols]=size(input);
output=zeros([rows,cols]);

for i=1:1:rows
    for j=1:1:cols

        if (input(i,j)<level)
            output(i,j)=0;

            else
                output(i,j)=255;
            end
        end;
    end;
end;
output=uint8(output);
```

C.4 Histogram

```
function [histo]=histogram(input)

%needed for matlab
input=double(input);
[rows,cols]=size(input);
histo=zeros(1,256);
for i=1:1:rows
    for j=1:1:cols
        histo(input(i,j)+1)=histo(input(i,j)+1)+1;
    end;
end;
```

C.5 Gaussian Smoothing

```
function [output]=gauss_smth(input)
input=double(input);
[cols,rows]=size(input);
output=zeros([cols,rows]);
gauss=(1/115)*[2 4 5 4 2;4 9 12 9 4;5 12 15 12 5;4 9 12 9 4;2 4 5 4 2];
%for j=3:1:(rows-2)      %find output for gaussian smoothing filter
%    for i=3:1:(cols-2)
%        output=conv2(input,gauss);
output=uint8(output(1:480,1:640));
```