DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FPGA BASED ADVANCED VIDEO ENHANCEMENT ALGORITHMS

by
Erdem ADAK

September, 2011
İZMİR

FPGA BASED ADVANCED VIDEO ENHANCEMENT ALGORITHMS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Electronics Engineering

by
Erdem ADAK

September, 2011
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **“FPGA BASED ADVANCED VIDEO ENHANCEMENT ALGORITHMS”** completed by **ERDEM ADAK** under supervision of **PROF. DR. MUSTAFA GÜNDÜZALP** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis fort he degree of Master of Science.

[illegible]

Prof. Dr. Mustafa GÜNDÜZALP

Supervisor

[illegible]	[illegible]
Yard. DOÇ. Dr. Tuncay Ercan	Yrd. Doç. Dc. Yavuz ŞENOL
(Jury Member)	(Jury Member)

[illegible]

Prof.Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I express my deepest gratitude to my advisor Prof. Dr. Mustafa GÜNDÜZALP for his valuable guidance and support in every stage of my research. The experience I have gained by his supervisory is a valuable asset for me.

Finally, I would like to thank to my parents, whose constant love and support have made my achievements possible.

Erdem ADAK

FPGA BASED ADVANCED VIDEO ENHANCEMENT ALGORITHMS

ABSTRACT

The principal goal of enhancement techniques is to process an image so that the result is more suitable than the original image for a specific (Contrast enhancement, edge enhancement, etc.) application.

In this thesis work, it is aimed to design a low cost, portable and reconfigurable system to capture composite video data, decode and make image enhancement on this video data.

This thesis is composed of two main parts; an analog composite video input is converted to digital VGA format and applied image enhancement algorithms on this video signal. Noise reducing, sharpening, contrast enhancement and color enhancement techniques are used for video improving.

Field Programmable Gate Arrays (FPGA) are integrated circuits that can be programmable in the field by the customer after manufacturing. In this study, it is aimed to develop a system by using ALTERA DE2-70 FPGA board. Design is done using hardware design language Verilog. Analog NTSC composite video input is decoded on a video decoder module ; then in FPGA, the video signal is de-interlaced. Image enhancement algorithms are applied on this video signal and it is converted from YCbCr format to RGB format which is suitable for driving a VGA monitor.

Development of image processing techniques in FPGA is the main subject of this thesis. FPGAs have many advantages that can be used in image processing areas. Therefore, the system can be improved more to use in the areas like LCD Television market by some enhancements which will be done in the future.

Keywords: Image enhancement, video processing, sharpness, noise reduction, color enhancement, contrast enhancement, FPGA.

SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ TABANLI İLERİ GÖRÜNTÜ İYİLEŞTİRME ALGORİTMALARI

ÖZ

Görüntü iyileştirme tekniklerinin temel amacı, bir görüntüyü belli bir amaca yönelik özel bir uygulama için orijinal halinden daha uygun bir hale getirmektir.

Bu tez çalışmasında bir kompozit video işaret bilgisinin yakalanıp, çözüleceği ve bu video bilgisinde görüntü iyileştirmesinin yapılacağı düşük maliyetli, başka sistemlere taşınabilen, tekrar konfigüre edilebilir bir sistemin tasarlanması hedeflenmiştir.

Bu tez iki ana parçadan oluşmaktadır; FPGA üzerinde, analog kompozit video görüntüsünün sayısal VGA formatına dönüştürülerek ekrana basılması ve bu video işaretinde görüntü iyileştirme tekniklerinin uygulanması. Video iyileştirmesi için gürültü azaltma, keskinleştirme, kontrast iyileştirme ve renk iyileştirme teknikleri kullanılmıştır.

Programlanabilir Kapı Dizileri (FPGA), üretildikten sonra, müşteri tarafından sahada programlanabilir tümleşik devrelerdir. Bu çalışmada ALTERA DE2-70 FPGA (Sahada Programlanabilir Kapı Dizileri) kartı kullanarak bir sistem geliştirme amaçlanmıştır. Tasarım, Verilog donanım tasarım dili ile yapılmıştır. Analog NTSC kompozit video işareti video çözücüsü üzerinden FPGA donanımına aktarılarak video işaretine binişimsizleştirme işlemi uygulanır. Bu video işaretine görüntü iyileştirme algoritmaları uygulanır ve VGA monitörü sürmeye uygun olan RGB formatına YCbCr formatından dönüştürülür.

Sahada Programlanabilir Kapı Dizilerinde veri işleme teknikleri geliştirmek, bu tezin ana amacıdır. Sahada Programlanabilir Kapı Dizileri, görüntü işleme alanında kullanılabilecek çok sayıda avantaja sahiptir. Bu nedenle, sistem, gelecekte yapılacak geliştirmelerle, LCD Televizyon piyasasında kullanmak üzere daha da iyileştirilebilir.

Anahtar Sözcükler: Görüntü iyileştirme, video işleme, keskinlik, gürültü azaltma, renk iyileştirme, kontrast iyileştirme, FPGA.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Image Enhancement With FPGA

Field Programmable Gate Array (FPGA) technology is very important for the implementation of algorithms of image processing applications. Field Programmable Gate Arrays (FPGAs) are reconfigurable technology. Reconfigurable feature is needed for a flexible design.

Today, FPGAs can be developed to implement parallel design methodology, which is not supported in DSP (Digital Signal Processors) designs. ASIC (Application Specific Integrated Circuits) design methods can be used for FPGA design and the design is at gate level. However, engineers use a hardware language, which is similar to software design.

In this thesis, image processing is performed with FPGA. Image enhancement algorithms are applied to FPGA. In order to obtain clear and vivid images, image enhancement techniques, most of them are based on filters to remove noise and unwanted effects of the light, are used. To improve image quality with an efficient way, realization on FPGA is a good choice.

There is no general theory of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works. Visual evaluation of image quality is a highly subjective process, thus making the definition of a "good image" an elusive standard by which to compare algorithm performance (Gonzales, & Woods2002).

Enhancement refers to accentuating or sharpening of image features, such as contrast, boundaries, edges, etc. The process of image enhancement, however, in no way increases the information content of the image data. It increases the dynamic range of the chosen features with the final aim of improving the image quality (Acharya, & Ray, 2005).

In this thesis, enhancement techniques is first performed in MATLAB and useful algorithms are applied to FPGA. ALTERA DE2-70 board is used for hardware design. All the system architecture blocks and algorithms are written in Verilog Hardware Design Description Language. The DVD player which is configured to NTSC 60 Hz refresh rate, 4:3 aspect ratio, non-progressive CVBS output is used as a source of video data.

1.2 Outline Of The Thesis

The thesis has nine chapters. Each chapter (excluding Chapter 8, and Chapter 9) is organized in a form that first few sections are comprised of theoretical information and the last sections (Chapter 8 and Chapter 9) includes detailed information about the design of the related part of the overall system architecture.

Chapter 1 presents an introduction to the project.

Chapter 2 defines what Field Programmable Gate Array (FPGA) is and how it works, what are the advantages. Then used hardware design languages, namely Verilog and VHDL, are discussed.

In Chapter 3 Digital Image Processing techniques; point operations, neighbourhood operations and geometrical operations are explained.

Chapter 4 includes Filters which are used in image proccessing. Spatial domain and frequency domain filters are discussed in this chapter.

Chapter 5 defines image enhancement algorithms which are used in this thesis. Contrast stretching, smoothing operations, edge detection techniques and color enhancement techniques are discussed in this chapter.

In chapter 6, video basics are discussed. Digital video, video timings, interlaced and progressive descriptions, composite video interface, computer signal interface and ITU-R 656 4:2:2 YCbCr video interface are explained in this chapter.

In chapter 7, general description of the hardware of the system, ALTERA DE2-70 FPGA development Board is described.

In Chapter 8, design of the whole system is described from the video input to output. Design is considered as a 3 part process; video receiver part, image enhancement part and encoding part. For the video receiver part, configuration of the decoder, video conversions, de-interlacing, buffering, video synchronization timing and encoding process are explained. Then image enhancement algorithms are outlined. Also results are presented in this chapter.

In chapter 9, conclusion and future works are presented.

CHAPTER TWO
FPGA (FIELD PROGRAMMABLE GATE ARRAY)

With the advent of mobile embedded multimedia devices that are required to perform a range of multimedia tasks, especially image processing tasks, the need to design efficient and high performance image processing systems in a short time to market schedule needs to be addressed. Image processing algorithms implemented in hardware have emerged as the most viable solution for improving the performance of image processing systems. The introduction of reconfigurable devices and system level hardware programming languages has further accelerated the design of image processing in hardware (Rao, Patil, Babu, Muthukumar, 2006).

Implementing such applications on a general purpose computer can be easier, but not very time efficient due to additional constraints on memory and other peripheral devices. Application specific hardware implementation offers much greater speed than a software implementation. With advances in the VLSI (Very Large Scale Integrated) technology hardware implementation has become an attractive alternative. Implementing complex computation tasks on hardware and by exploiting parallelism and pipelining in algorithms yield significant reduction in execution times (Rao, Patil, Babu, Muthukumar, 2006).

There are two types of technologies available for hardware design. One is Application Specific Integrated Circuits (ASIC), other one is programmable devices which are Digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGA's). ASICs are full custom devices and can not be programmed or changed after it designed. DSPs and FPGA's are programmable devices. ASIC design has high performance but design process is very complex and is not cost effective. FPGA designs are very flexible and it can be changed any time. DSP designs can be programmed easily but they are not flexiable as FPGAs. Parallelism tecniques can be used in FPGAs and ASICs, but it is not possible in DSPs.

2.1 Adventage Of FPGA In Image Processing

Many new and exciting innovations, such as HDTV and digital cinema, revolve around video and image processing and this technology's rapid evolution. Leaps forward in image capture and display resolutions, advanced compression techniques, and video intelligence are the driving forces behind the technological innovation. The move from standard definition (SD) to high definition (HD) represents a 6X increase in data that needs to be processed. Video surveillance is also moving from Common Intermediate Format (CIF) (352 x 288) to D1 format (704 x 576) as a standard requirement, with some industrial cameras even moving to HD at 1280 x 720. Military surveillance, medical imaging, and machine vision applications are also moving to very high resolution images (Altera, 2007). FPGAs are getting popular for this reasons.

System architecture can be designed with ASICs, DPSs and FPGA's. Each has advantages and disadvantages. In a system design, high performance, flexibility, easy upgradability and low development cost properties are very important.

2.1.1 High Performance

Performance not only applies to compression, but also pre- and postprocessing functions. In fact, in many cases these functions consume more performance than the compression algorithm itself. Examples of these functions include scaling, de-interlacing, filtering, and color space conversion. For the markets described above, the need for high performance rules out processor-only architectures. They simply cannot meet the performance requirements with a single device. A state-of-the-art DSP running at 1 GHz cannot perform H.264 HD decoding or H.264 HD encoding, which is about ten times more complex than decoding. FPGAs are the only programmable solutions able to tackle this problem. In some cases, the best solution is a combination of an FPGA plus an external DSP processor (Altera, 2007).

2.1.2 Flexibility And Easy Upgradeability

When technology rapidly evolves, architectures must be flexible and easy to upgrade. This rules out standard cell ASICs and ASSPs for those applications. Typically designed for very high volume consumer markets, ASSPs often are quickly obsolete, making them an extremely risky choice for most applications (Altera, 2007).

2.1.3 Low Development Cost

When adding up costs for masks and wafer, software, design verification, and layout, development of a typical 90-nm standard-cell ASIC can cost as much as US$30 million. Only the highest volume consumer markets can justify such pricey development costs (Altera, 2007).

2.2 Design Language

Gate-level design can result in optimized designs but the learning this method is very difficult and is not portable while high-level hardware design languages (HDLs) are easy to learn and portable other FPGA platforms.

2.2.1 VHSIC Hardware Design Language (VHDL)

It is an open IEEE Standard and it is supported by a large variety of design tools. It is a high-level language and it is similar to the computer programming language Ada.

2.2.2 Verilog Hardware Design Language

Verilog also support a large variety of design tools. Many designers favor Verilog over VHDL for hardware design because Verilog is very familiar to C programming language.

CHAPTER THREE
FUNDAMENTALS OF DIGITAL IMAGE PROCESSING

Image processing can be thought as a transformation which takes an image and produces a modified (enhanced) image. On the other hand, digital image analysis is a transformation of an image into something other than an image so it produces some information representing a description or a decision (Vernon,1991).

There are three classes of operations in digital image processing: point operations, neighborhood operations, and geometric operations.

3.1 Point Operations

A point operation is an operation in which each pixel in the output image is a function of the gray-level of the pixel at the corresponding position in the input image and, only of that pixel (Vernon,1991).

Point operations are also referred to as gray scale manipulation operations. They cannot alter the spatial relationships of the image. Typical uses of point operations include photometric decalibration, to remove the effects of spatial variations in the sensitivity of a camera system, contrast stretching and thresholding, in which all pixels having gray levels (Vernon,1991).

3.2 Neighborhood Operations

A neighborhood operation generates an "output" pixel on the basis of the pixel at the corresponding position in the input image and on the basis of its neighboring pixels. The size of neighborhood may vary: several techniques use 3x3 or 5x5 neighborhoods centered at the input pixel, but many of the most advanced and useful techniques now use neighborhoods which may be as large as 63x63 pixels. The neighborhood operations are often referred to as "filtering operations". This is particularly true if they involve the convolution of an image with a filter, kernel or mask. Such filtering often addresses the removal of noise or the enhancement of edges, and is most effectively accomplished using convolver (or filtering) hardware, available as sister boards for most frame-grabbers (Vernon,1991).

3.2.1 Convolution

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution is a way of multiplying together two arrays of numbers of different sizes to produce a third array of numbers. In image processing the convolution is used to implement operators whose output pixel values are simple linear combination of certain input pixels values of the image. Convolution belongs to a class of algorithms called spatial filters. Spatial filters use a wide variety of masks, also known as kernels, to calculate different results, depending on the desired function (Vernon,1991).

3.2.1.1 1D- Convolution

The convolution operation is a mathematical operation which takes two functions f(x) and g(x) and produces a third function h(x). Mathematically, convolution is defined as:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau \qquad (3.1)$$

where g(x) is referred to as the filter.

3.2.1.2 2D-Convolution

2D-Convolution, is most important to modern image processing. The basic idea is that a window of some finite size and shape is scanned over an image. The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the filter assigned to every pixel of the window. The window with its weights is called the convolution mask. Mathematically, convolution on image can be represented by the following equation.

$$y(m,n) = \sum_{i=0}^{Heightof\ image} \sum_{j=0}^{widthof\ image} h(i,j)\, x(m-i, n-j) \tag{3.2}$$

where x is the input image, h is the filter and y is the image.

3x3 convolution masks are most commonly used. For example the derivative operators which are mostly used in edge detection use 3x3 window kernels. They operate only a pixel and its directly adjacent neighbors. Figure 3.1 shows a 3x3 convolution mask operated on an image. The center pixel is replaced with the output of the algorithm; this is carried for the entire image. Similarly larger size convolution masks can be operated on an image.

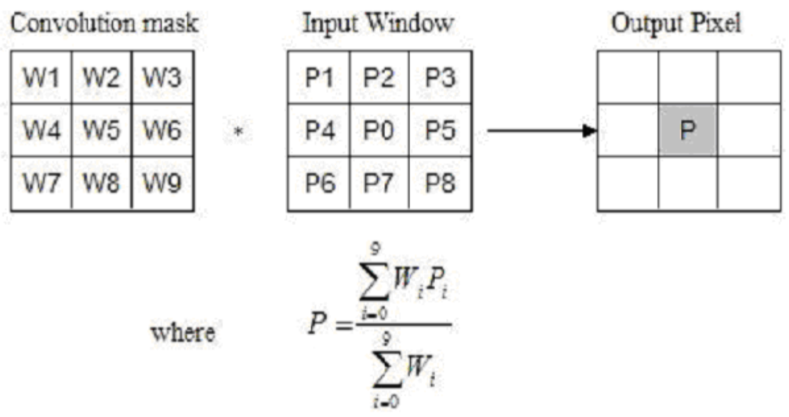


Figure 3.1 Convolution

3.3 Geometric Operations

Geometric operations change the spatial relationships between objects in an image, i.e., relative distances between points a, b and c will typically be different after a geometric operation or “warping”. The application of such warping include geometric decalibration, i.e. the correction of geometric distortion introduced by the imaging system (most people are the familiar with the barrel distortion that arises in photography when using a short focal length “fish-eye” lens), and image registration, i.e. intentional distortion of one image with respect to another so that the objects in each image superimpose on one another (Vernon,1991).

CHAPTER FOUR
FILTERS IN IMAGE PROCESSING

Filtering is also a common concept. Adjusting the bass and treble on a music is a filter example. High-pass filters pass high frequencies and stop low frequencies. Low-pass filters stop high frequencies and pass low frequencies. In image processing, a high-pass filter will pass, amplify or enhance the edge. A low-pass filter will remove the edge.

Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods. The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of an image (Gonzales, & Woods2002).

4.1 Frequency Domain Filters

Frequency filters process an image in the frequency domain. The image is Fourier transformed, multiplied with the filter function and then re-transformed into the spatial domain. Attenuating high frequencies results in a smoother image in the spatial domain, attenuating low frequencies enhances the edges.

All frequency filters can also be implemented in the spatial domain and, if there exists a simple kernel , it is easy to perform the filtering in the spatial domain. If there isn't any straightforward kernel in the spatial domain, frequency filtering is more appropriate.

When an image f(x,y) is convolved with a linear operator h(x,y) , the resultant image g(x,y) is given by

$$g(x,y) = h(x,y) * f(x,y) \tag{4.1 }$$

The convolution theorem states that the convolution in spatial domain is equivalent to multiplication in frequency domain. This implies that

$$G(u,v) = H(u,v)F(u,v) \tag{4.2}$$

where G(u,v), H(u,v), and F(u,v) are the Fourier transforms of g(x,y), h(x,y) and f(x,y) respectively. Taking the inverse Fourier transform of G(u,v), we get

$$g(x,y) = \Im^{-1}[H(u,v)F(u,v)] \tag{4.3}$$

It may be observed that by suitable selection of h(x,y), we get a resultant image g(x,y) which is an enhanced version of the original image f(x,y) (Acharya, & Ray, 2005).

Since the multiplication in the Fourier space is identical to convolution in the spatial domain, all frequency filters can in theory be implemented as a spatial filter. However, in practice, the Fourier domain filter function can only be approximated by the filtering kernel in spatial domain (Fisher, R., Perkins, S., Walker, A., & Wolfart, E. 2004)

4.2 Spatial Domain Filters

The term spatial domain refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression

$$g(x,y) = T[f(x,y)] \tag{4.4}$$

where f(x,y) is the input image, g(x,y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). In addition,T can operate on a set of input images, such as performing the pixel-by-pixel sum of images for noise reduction (Gonzales, & Woods, 2002).

The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular subimage area centered at (x, y). The center of the subimage is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x,y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood. Although other neighborhood shapes, such as approximations to a circle, sometimes are used, square and rectangular arrays are by far the most predominant because of their ease of implementation (Gonzales, & Woods, 2002).

The simplest form of T is when the neighborhood is of size 1*1 (that is, a single pixel). In this case, g depends only on the value of f at (x,y) and T becomes a gray-level (also called an intensity or mapping) transformation function of the form

$$s = T(r) \tag{4.5}$$

where, for simplicity in notation, r and s are variables denoting, respectively, the gray level of f(x, y) and g(x, y) at any point (x, y) (Gonzales, & Woods, 2002).

Larger neighborhoods allow considerably more flexibility. The general approach is to use a function of the values of f in a predefined neighborhood of (x, y) to determine the value of g at (x, y). One of the principal approaches in this formulation is based on the use of so-called masks (also referred to as filters, kernels, templates, or windows). Basically, a mask is a small (say, 3*3) 2-D array, in which the values of the mask coefficients determine the nature of the process, such as image sharpening. Enhancement techniques based on this type of approach often are referred to as mask processing or filtering (Gonzales, & Woods, 2002).

The spatial filtering techniques can be used as follows:

- Spatial low-pass, high-pass and band-Pass filtering
- Unsharp masking and crisping
- Directional smoothing
- Median filtering

Filters can be classified into two groups. Spatial domain filter and Frequency domain filters. Spatial domain filter can be categorized into two types, Linear and non-linear filters. Linear filters are the simplest and most widely available. Non-linear filters are more flexible and powerful, but need to be used with more care.

4.2.1 Spatial Domain Linear Filters

In a linear filter, a pixel is replaced with a linear combination of intensities of neighbouring pixels. The simplest neighbourhood is the 3 by 3 grid centred on the pixel. We may represent the coefficients of the linear combination as a matrix:

$$\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix} \qquad (4.6)$$

If pixel positions in an image are labelled (i,j), where i and j are row and column indices respectively, and the pixel intensity at (i, j) is denoted I_{ij} then the filter output at (i,j) will be (Baldock, & Graham, 2000).

$$\begin{matrix} + aI_{i-1,j-1} & + bI_{i-1,j} & + cI_{i-1,j+1} \\ + dI_{i,j-1} & + eI_{i,j} & + fI_{i,j+1} \\ + gI_{i+1,j-1} & + hI_{i+1,j} & + iI_{i+1,j+1} \end{matrix} \qquad (4.7)$$

A linear filter is an operation where at every pixel $x_{m,n}$ of an image, a linear function is evaluated on the pixel and its neighbors to compute a new pixel value $y_{m,n}$ as in figure 4.1.

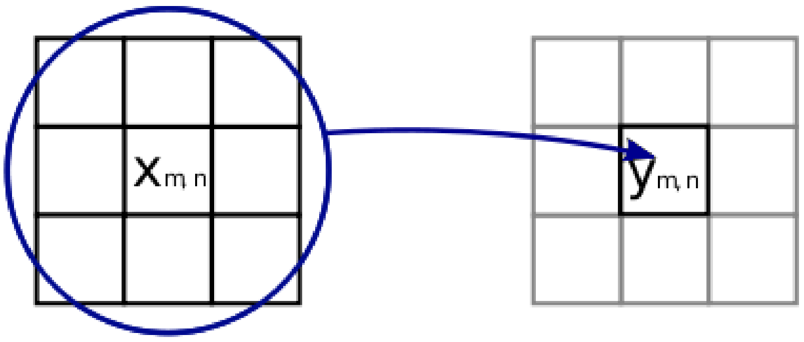


Figure 4.1 Spatial Domain Linear Filter

A linear filter in two dimensions has the general form

$$y_{m,n} = \sum_j \sum_k h_{j,k}\, x_{m-j,n-k} \tag{4.8}$$

where x is the input, y is the output, and h is the filter impulse response. The right-hand side of the above equation is denoted h*x and is called the “convolution of h and x.”

Gaussian, Wiener , Prewitt, Sobel, Robert, Frei-Chen are the examples of linear filters. They are explained in section 5.3.

4.2.2 Spatial Domain Non-linear Filters

Linear filters are easy to use and well understood, but have the drawback of being rather limited. In particular, they are unable to reduce noise levels without simultaneously blurring edges. With non-linear filters, we can define any function of the pixels in a neighbourhood and noise reduction without blurring is possible. However, more care is needed in using them (Baldock, & Graham, 2000).

The most common use of non-linear filters is to smooth without blurring edges. The median filter is the best known. It replaces the pixel with the median of the pixel intensities in a neighbourhood range. It is discussed in section 5.2.2.

CHAPTER FIVE
IMAGE ENHANCEMENT

The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. The word specific is important, because it establishes at the outset that the techniques discussed in this chapter are very much problem oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not necessarily be the best approach for enhancing pictures of Mars transmitted by a space probe. Regardless of the method used, however, image enhancement is one of the most interesting and visually appealing areas of image processing (Gonzales, & Woods, 2002).

There is no general theory of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works. Visual evaluation of image quality is a highly subjective process, thus making the definition of a “good image” an elusive standard by which to compare algorithm performance. When the problem is one of processing images for machine perception, the evaluation task is somewhat easier. For example, in dealing with a character recognition application, and leaving aside other issues such as computational requirements, the best image processing method would be the one yielding the best machine recognition results. However, even in situations when a clear-cut criterion of performance can be imposed on the problem, a certain amount of trial and error usually is required before a particular image enhancement approach is selected (Gonzales, & Woods, 2002).

Enhancement can undo the degradation effects which might have been caused by the imaging system or the channel.

Enhancement refers to accentuation or sharpening of image features, such as contrast, boundaries, edges, etc. The process of image enhancement, however, in no way increases the information content of the image data. It increases the dynamic range of the chosen features with the final aim of improving the image quality (Acharya, & Ray, 2005).

Enhancement techniques are based on combinations of methods from spatial and frequency domains.

5.1 Contrast Enhancement

To improve the contrast of the digital image, it is desirable to utilize the entire brightness range of the display medium, which is generally a video display or hard copy output device. There are linear and nonlinear digital contrast enhancement techniques.

5.1.1 Linear Contrast Enhancement

Contrast enhancement (or contrast stretching) expands the original input brightness values to make use of the total range of the output device. To make a decision about the contrast of the image data its histogram is used. If each pixel in the image is examined and its brightness value noted, a graph of number of pixels with a given brightness versus brightness value can be constructed. This is referred to as histogram of the image (Akbal, 2005).

A 24 bit color image includes 3 components which are blue, green and red. Each of these components uses 8 bits to show different brightness values. This means that 256 levels may be used, from 0 to 255, in each band. A 0 brightness value means darkness and a 255 brightness value means lightness. If only one of these bands is considered, whose range is 255, highest value minus lowest value, and in the original image the range is 100; then a limited portion of the range is used. This limitation can be seen from the histogram of the image. For example if your minimum brightness value is 0 and the highest brightness value is 100, then you have a dark image. If your minimum brightness value is 155 and the highest brightness value is 255, then you have a bright image. But both of them are low contrast images. It is difficult to visually interpret such images. A more useful display can be produced if the range of original brightness values are expanded to use the full dynamic range of the video display (Akbal, 2005).

Linear contrast enhancement is the best applied to remotely sensed images with Gaussian or near-Gaussian histograms, that is when all brightness values fall generally within a single, relatively narrow range of the histogram and only one mode is apparent. Unfortunately, this is rarely the case especially for scenes that contain both land and water bodies. To perform a linear contrast enhancement, the analyst examines the image statistics and determines the minimum and maximum brightness values in the band, min_k and max_k, respectively. The output brightness value BV_{out}, is computed according to the equation

$$BV_{out} = \left(\frac{BV_{in} - min_k}{max_k - min_k}\right) quant_k \tag{5.1}$$

where BV_{in} is the original input brightness value and quant_k is the range of brightness values that can be displayed on the video display. There are many approaches used to select min_k and max_k (Akbal, 2005).

5.1.1.1 Min-max Contrast Stretch

For an 8 bit color system 0 is used as min_k and 255 is used as max_k. If the image has intensity values between 12 to 200, then in the new image intensity value 12 become 0, intensity value 200 become 255. Other values between 12 and 200 are stretched between 0 and 255.

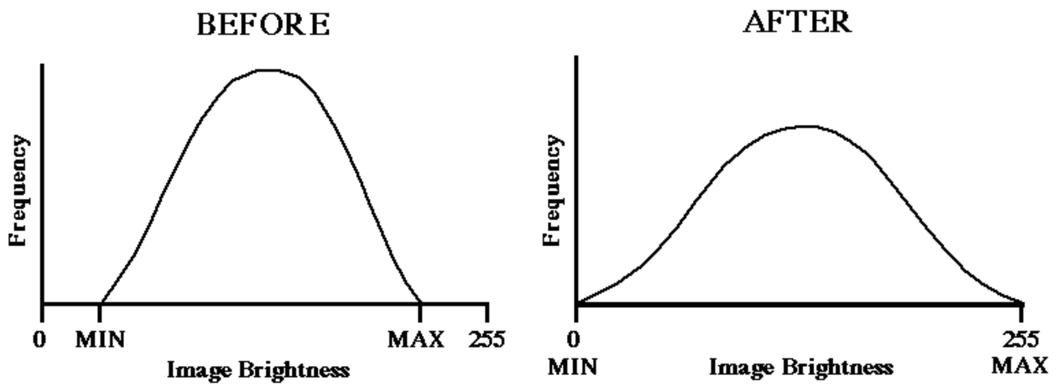


Figure 5.1 Minimum-maximum linear contrast enhancement. The minimum and maximum values of the original image are stretched to produce a range of brightness values that uses the full capabilities of the image display (Campbell, 1987).

5.1.1.2 Percentage Linear Contrast Stretch

Min_k and max_k that lie a certain percentage of pixels from the mean of the histogram may be specified. For example setting the minimum and maximum ±1 standard deviation (±1σ) from the mean. Values smaller than the min_k, which is now mean -1 σ, go to 0, while values larger than the max_k go to 255 (Akbal, 2005).

5.1.1.3 Specific Percentage Linear Contrast Stretch (Saturating Linear Contrast Enhancement

In this method, min_k and max_k values are determined with the user. For example the image has intensity values between 12 to 200. User wants to emphasize values between 50 to 150, then in the new image intensity value 50 become 0, intensity value 150 become 255. Other values between 50 and 150 are stretched between 0 and 255. Figure 5.2 illustrates this method.

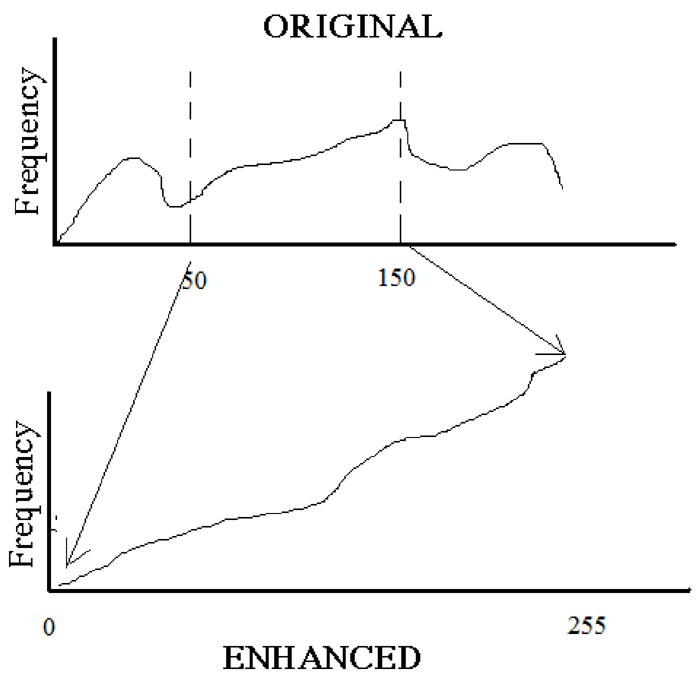


Figure 5.2 Specific Percentage Linear Contrast Stretch

5.1.1.4 Piecewise Linear Contrast Stretch

When the histogram of an image is not Gaussian, it is possible to perform a piecewise linear contrast stretch. In this method min$_k$ and max$_k$ are determined with above equation. The boundaries are selected with the user.

A piecewise linear contrast enhancement involves the identification of a number of linear enhancement steps that expands the brightness ranges in the modes of the histogram. This type can be expressed by:

$$f(x,y)=\begin{cases} ax, & 0 \le x \le x_1 \\ b(x-x_1)+y_{x_1}, x_1 \le x \le x_2 \\ c(x-x_2)+y_{x_1}, x_2 \le x \le B \end{cases} \quad (5.2)$$

where f(x,y) is the Piecewise Linear Contrast Stretch in the image, a, b, and c are appropriate constants, which are the slopes in the respective regions and B is the maximum intensity value (Al-amri, S. S., Kalyankar, N. V., & Khamitkar, S. D. 2010).

5.1.2 Nonlinear Contrast Enhancement

Histogram equalization and logarithmic contrast enhancement are examples to Nonlinear Contrast Enhancement. Histogram equalization emphasizes the brightness values which are most frequent intensity values. It reduces the contrast in the very light and very dark parts of the image. Logarithmic and exponential contrast enhancement modifies images for enhancing dark and light features respectively.

5.2 Smoothing (Noise Reduction)

Smoothing filters are used for blurring and for noise reduction. Noise reduction can be accomplished by blurring with a linear filter and also by non-linear filtering.

In signal processing theory we know that low-pass filtering attenuates the high-frequency components in the signal and is essentially equivalent to integrating the signal. Integration in turn implies summation and averaging the signal. Low-pass filtering of an image is a spatial averaging operation. It produces an output image, which is a smooth version of the original image, devoid of the high spatial frequency components that may be present in the image. In particular, this operation is useful in removing visual noise, which generally appears as sharp bright points in the image. Such high spatial frequencies associated with these spikes are attenuated by the low-pass filter (Acharya, & Ray, 2005).

An image may be effected from noise and interference from several sources like electrical sensor noise, photographic grain noise and channel errors. Effected pixels appear visually to be markedly different from their neighbors. Noise cleaning algorithms can solve this problem.

In this section, several linear and nonlinear techniques that have proved useful for noise reduction are described.

5.2.1 Linear Noise Cleaning

Noise added image generally has a higher-spatial-frequency spectrum than the normal image components. Hence, low-pass filtering can be used for noise cleaning.

A spatially filtered output image can be formed by discrete convolution of an input image with a impulse response array according to the relation

$$G(j,k) = \sum\sum F(m,n)H(m+j+C, n+k+C) \qquad (5.7)$$

where C = (L + 1)/2. Equation 5.7 utilizes the centered convolution notation whereby the input and output arrays are centered with respect to one another, with the outer boundary of width pixels set to zero (Pratt, 2007).

For noise cleaning, H should be of low-pass form, with all positive elements. Several common pixel impulse response arrays of low-pass form are listed below.

Mask 1
$$\mathbf{H} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad (5.8)$$

Mask 2
$$\mathbf{H} = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad (5.9)$$

Mask3
$$\mathbf{H} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad (5.10)$$

These arrays, called noise cleaning masks, are normalized to unit weighting so that the noise-cleaning process does not introduce an amplitude bias in the processed image. Mask 1 and 3 of Eq. 5.8 to 5.10 are special cases of a parametric low-pass filter whose impulse response is defined as: (Pratt, 2007).

$$\mathbf{H} = \left(\frac{1}{b+2}\right)^2 \begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix} \tag{5.11}$$

5.2.2 Nonlinear Noise Cleaning

The linear processing techniques are suitable for images with continuous noise, such as additive uniform or Gaussian distributed noise and they remove the details from the image. Nonlinear techniques often provide a better trade-off between noise smoothing and details in an image.

5.2.2.1 Median Filters

Median filter is a special type of low-pass filter. The median filter takes an area of an image (3x3, 5x5, 7x7, etc.), looks at all the pixel values in that area, and replaces the center pixel with the median value. The median filter does not require convolution. It does, however, require sorting the values in the image area to find the median value (Phillips, 2000).

Median filter has two advantages. First, it is easy to change the size of the median filter. Second, median filters remove noise in images, but change noise-free parts minimally.

For every pixel in an image, the window of neighboring pixels is found. Then the pixel values are sorted in ascending, or rank, order. The middle value is the output of this filter. Figure 5.3 shows an example of this algorithm for a median filter.

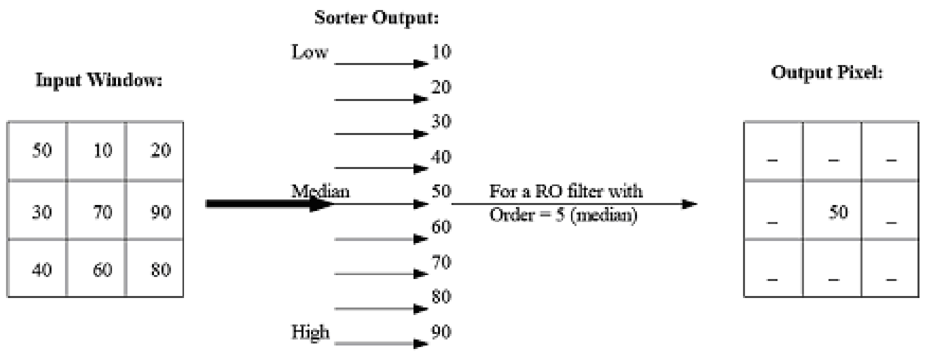


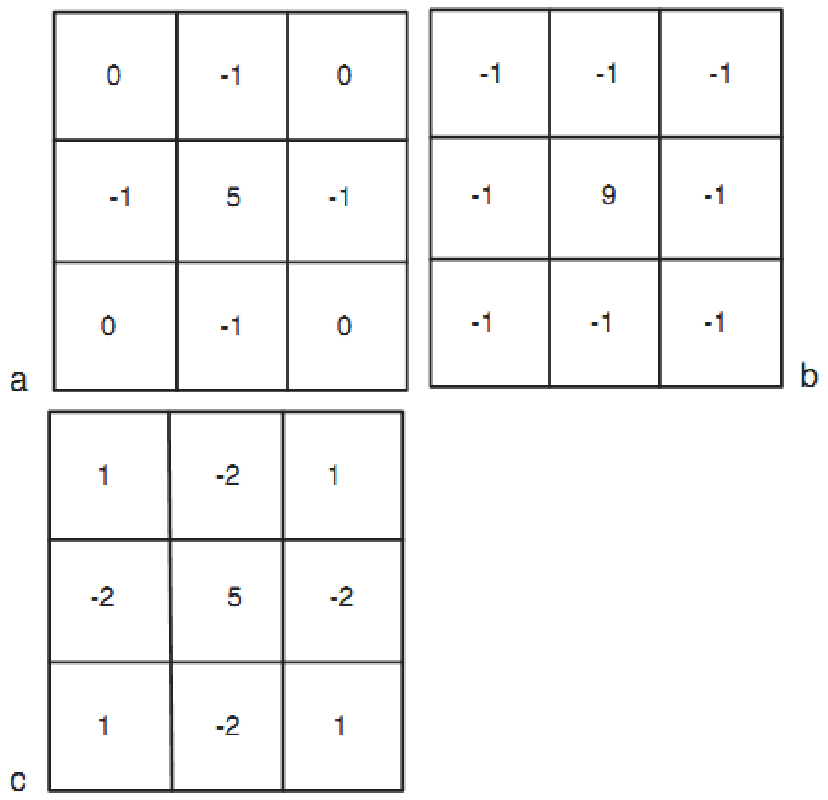
Figure 5.3 Rank Order Filter Operation

5.3 Edge Detection

High-pass filtering of an image produces an output image in which the low frequency components are attenuated. High-pass filtering is used for edge enhancement. The principal objective of sharpening is to highlight fine detail in an image or to enhance detail that has been blurred.

Psychophysical experiments indicate that a photograph or visual signal with accentuated or crispened edges is often more subjectively pleasing than an exact photometric reproduction. Edge crispening can be accomplished in a variety of ways (Pratt, 2007).

Edges in an image can be sharpened by a mask that performs differencing of, instead of averaging pixels. Some useful edge-sharpening masks are listed in Figure 5.4. Of course, the mask weights should sum up to unity, otherwise the output image will be amplified (Thyagarajan, 2006).


a

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

b

-1	-1	-1
-1	9	-1
-1	-1	-1

c

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

Figure 5.4 Spatial mask for sharpening edges in an image; (a) mask with weights in north-south-east-west directions, (b) mask with equal weights in all four directions, and (c) mask with weights emhasizing horizontal and wertical directions more then diagonal directions (Thyagarajan, 2006).

The effects of using edge sharpening by the masks in Figure 5.4 are illustrated in Figure 5.5. Since the sharpened images are too crispy, we can soften them without losing sharpness of the edges by adding a fraction of the edge-sharpened image to the original image and rescaling its intensity to that of the input image. This is shown in Figure 5.5e, where we see that the stripes in the pants are crisper while the face is as smooth as the original. This process is also referred to as unsharp masking (Thyagarajan, 2006).

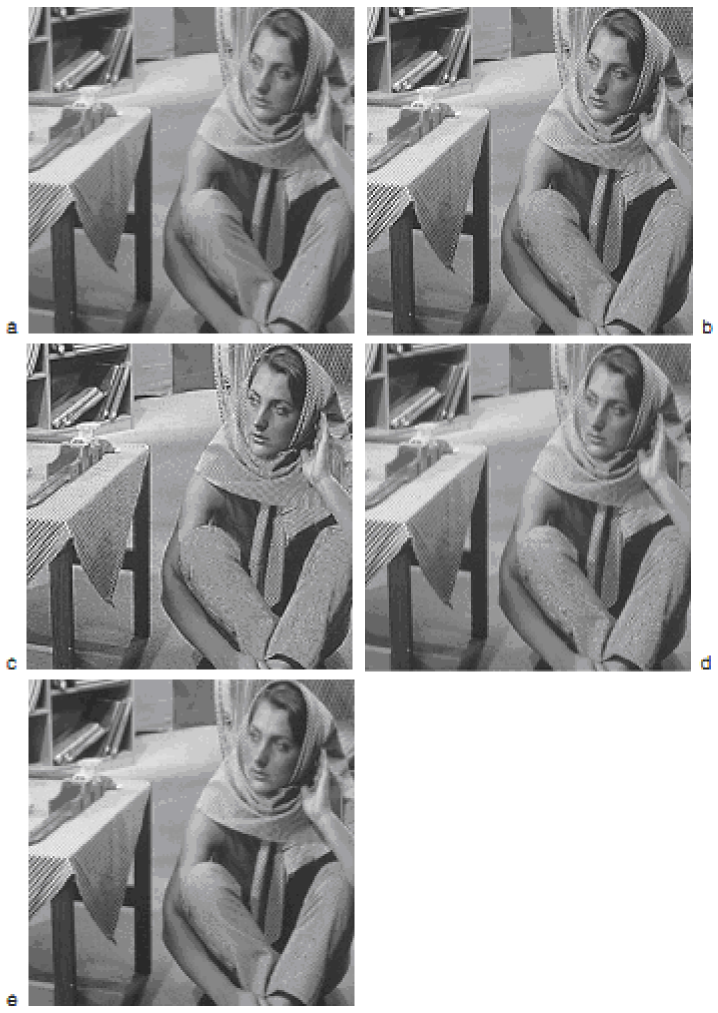


Figure 5.5 An example of edge sharpening using the masks in Figure 5.4: (a) original image, (b) result of using mask in Figure 5.4a, (c) result of using mask in Figure 5.4b, (d) result of using mask in Figure 5.4c, and (e) result of orginal image +0.75 image in d and rescaled (Thyagarajan, 2006).

There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories:

- Gradient: The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.

- Laplacian: The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the onedimensional shape of a ramp and calculating the derivative of the image can highlight its location.

Suppose we have the following signal, with an edge shown by the jump in intensity below.

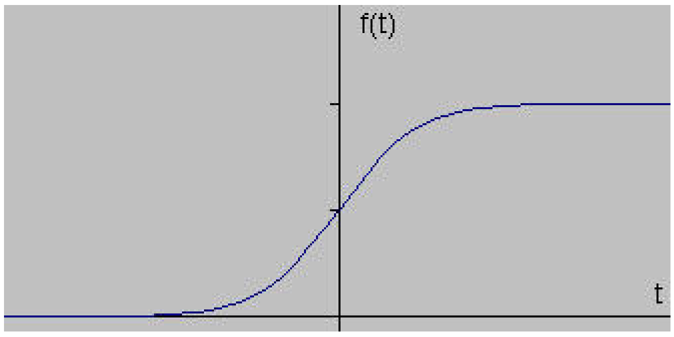


Figure 5.6 An f(t) signal with a sample intensity jump

If we take the gradient of this signal in figure 5.6, we get the following signal in figure 5.7.

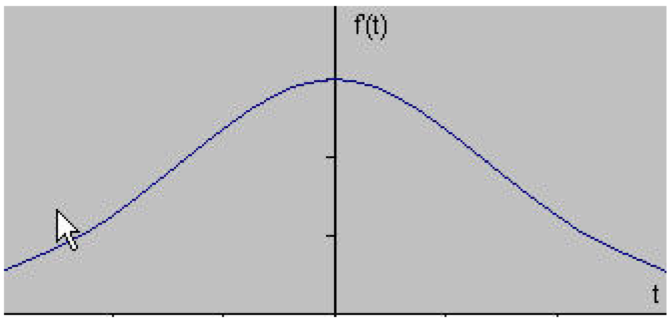


Figure 5.7 First derivative of f(t) signal, f'(t)

Clearly, the derivative shows a maximum located at the center of the edge in the original signal. This method is used in the Sobel method. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian and the second derivative of the signal is shown below figure.

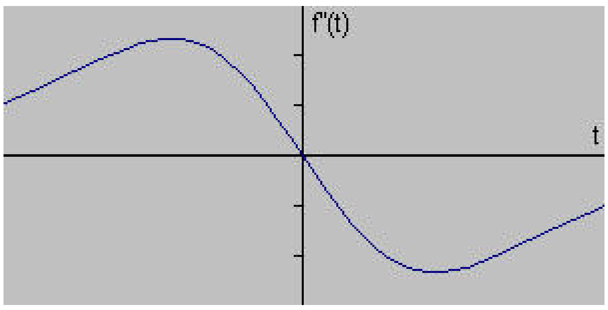


Figure 5.8 Second derivative of f(t) signal, f''(t)

5.3.1 Edge Detection Techniques

5.3.1.1 Sobel Operator

The operator consists of a pair of 3x3 convolution kernels as shown in Figure 5.10. One kernel is simply the other rotated by 90°.

$$Gx = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad Gy = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Figure 5.10 Kernels of Sobel Operator (Li, 2003)

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient (Li, 2003). The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2} \qquad (5.12)$$

Typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy| \tag{5.13}$$

which is much faster to compute. The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \arctan(Gy/Gx) \tag{5.14}$$

5.3.1.2 Robert's Cross Operator:

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point (Li, 2003).

The operator consists of a pair of 2×2 convolution kernels as shown in Figure 5.11. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

$$\text{Gx} = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad \text{Gy} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$$

Figure 5.11 Kernels of Roberts Cross Operator (Li, 2003).

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{5.15}$$

although typically, an approximate magnitude is computed using:

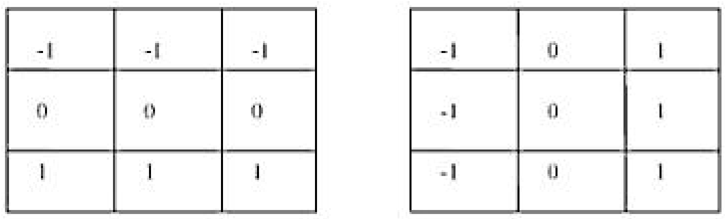
$$|G| = |Gx| + |Gy| \tag{5.16}$$

which is much faster to compute. The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan(Gy/Gx) - 3\pi/4 \tag{5.17}$$

5.3.1.3 Prewitt's Operator:

Prewitt uses the gradient information of an image to detect edges. Here two different types of masks are used, as shown in figure 5.12, one for the vertical edges and one for the horizontal edges. Both of the masks are convolved with the image, producing the derivative in the x-direction and y-direction (Qazi, S. 2008). The coefficients in both of the masks sum to zero, which shows that they give a zero response where the image is constant.



-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure 5.12 Kernels of Prewitt Operator (Qazi, S. 2008).

5.3.1.4 Laplacian of Gaussian:

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian Smoothing filter in order to reduce its sensitivity to noise. The operator normally takes a single graylevel image as input and produces another gray level image as output (Çallı, 2010).

The Laplacian L(x,y) of an image with pixel intensity values I(x,y) is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{5.18}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian (Çallı, 2010).

Three commonly used small kernels are shown in Figure 5.13.

These kernels are very sensitive to noise. To overcome this problem Gaussian filter is applied for noise cleaning, then Laplacian filter is applied for sharpening.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Figure 5.13 Three commonly used discrete approximations to the Laplacian filter. (Baldock, R., & Graham, J. 2000).

5.4 Color Enhancement

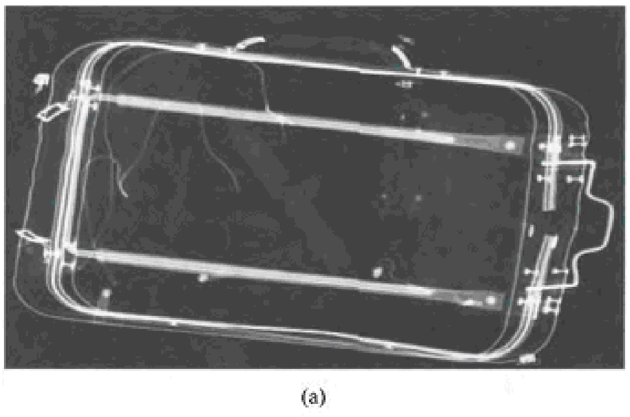
The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray. This second factor is particularly important in manual (i.e., when performed by humans) image analysis (Gonzales, & Woods, 2002).

Color image processing is divided into two major areas: full-color and pseudo-color processing.

5.4.1 Pseudocolor Image Processing

Pseudocolor (also called false color) image processing consists of assigning colors to gray values based on a specified criterion. The term pseudo or false color is used to differentiate the process of assigning colors to monochrome images from the processes associated with true color images. The principal use of pseudocolor is for human visualization and interpretation of gray-scale events in an image or sequence of images (Gonzales, & Woods, 2002).

Pseudocolor coding can be applied, for example, to an x-ray image to aid luggage inspection. Figure 5.14a shows an x-ray image of a bag containing a low-density knife. The knife is not easy to recognize by a screener at an airport. The visibility of the threat object in the image is significantly enhanced by pseudocolor coding (see Figure 5.14b), especially when considering the fatigue of a screener after several hours of duty (Koschan, & Abidi, 2008).



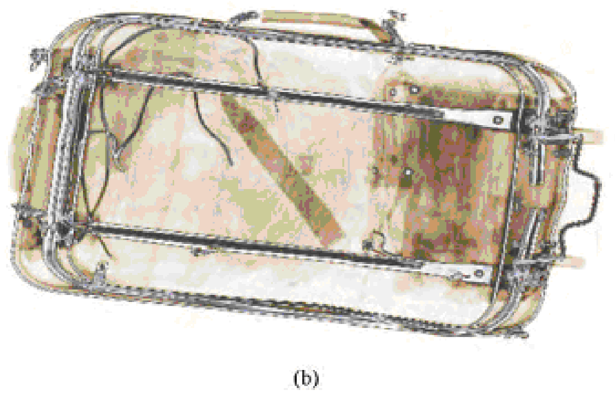


Figure 5.14 Bag contain a low density knife, (a) x-ray image, (b) Enhanced color-coded version of this image (Koschan, & Abidi, 2008).

5.4.2 Full-Color Image Processing

Full-color image processing approaches fall into two major categories. In the first category, we process each component image individually and then form a composite processed color image from the individually processed components. In the second category, we work with color pixels directly. Because full-color images have at least three components, color pixels really are vectors.

5.4.3 Color Spaces And Transformation

A color space is a mathematical representation of a set of colors. The three most popular color models are RGB (used in computer graphics); YIQ, YUV, or YCbCr (used in video systems); and CMYK (used in color printing).

5.4.3.1 RGB Color Space

The red, green, and blue (RGB) color space is widely used for computer graphics and displays. Red, green, and blue are three primary additive colors (individual components are added together to form a desired color) and are represented by a three-dimensional, Cartesian coordinate system (Figure 5.15). The indicated diagonal of the cube, with equal amounts of each primary component, represents various gray levels (Keith, 2007).

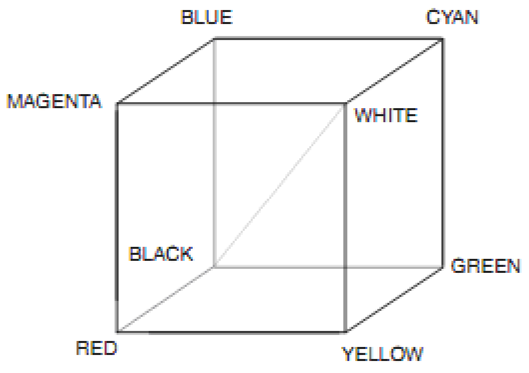


Figure 5.15 The RGB Color Cube (Keith, 2007).

The RGB color space is the most prevalent choice for computer graphics because color displays use red, green, and blue to create the desired color. Therefore, the choice of the RGB color space simplifies the architecture and design of the system. Also, a system that is designed using the RGB color space can take advantage of a large number of existing software routines, since this color space has been around for a number of years (Keith, 2007).

5.4.3.2 CMYK Color Space

Another interesting color model utilizes CMYK (cyan, magenta, yellow, and black) and this model finds utility in color printers. Most of the output devices including color printers or copiers use CMY color model. Just as the primary additive colors are red, green and blue, the primary colors of pigments on the other hand are magenta, cyan and yellow and the corresponding secondary colors are red, green and blue. The conversion from RGB to CMY may be performed as

$$C = 1 - R \tag{5.19}$$
$$M = 1 - G \tag{5.20}$$
$$Y = 1 - B \tag{5.21}$$

where R, G, B represent the normalized color values in the range 0 to 1. It may be easily verified from the above that a cyan coated surface does not contain red, or a surface pigmented by magenta is devoid of green. It may also be noted that equal amounts of pigments primaries (e.g. cyan, magenta, and yellow) produces black. Thus a four-color system cyan (C), magenta (M), yellow (Y), and black (B) forms a four-color model (Acharya, & Ray, 2005).

5.4.3.3 YUV Color Space

The YUV color space is used by the PAL (Phase Alternation Line), NTSC (National Television System Committee), and SECAM (Sequentiel Couleur Avec Mémoire or Sequential Color with Memory) composite color video standards. The black-and-white system used only luma (Y) information; color information (U and V) was added in such a way that a black-and-white receiver would still display a normal black-and-white picture. Color receivers decoded the additional color information to display a color Picture (Keith, 2007).

The basic equations to convert between gamma-corrected RGB (notated as R´G´B´) and YUV are:

$$Y = 0.299R' + 0.587G' + 0.114B' \tag{5.22}$$
$$U = -0.147R' - 0.289G' + 0.436B' = 0.492\,(B' - Y) \tag{5.23}$$
$$V = 0.615R' - 0.515G' - 0.100B' = 0.877(R' - Y) \tag{5.24}$$

$$R' = Y + 1.140V \tag{5.25}$$

$$G' = Y - 0.395U - 0.581V \tag{5.26}$$

$$B' = Y + 2.032U \tag{5.27}$$

For digital R´G´B´ values with a range of 0–255, Y has a range of 0–255, U has a range of 0 to ±112, and V has a range of 0 to ±157. These equations are usually scaled to simplify the implementation in an actual NTSC or PAL digital encoder or decoder (Keith, 2007).

5.4.3.4 YIQ Color Space

The YIQ color space is derived from the YUV color space and is optionally used by the NTSC composite color video standard. (The "I" stands for "inphase" and the "Q" for "quadrature," which is the modulation method used to transmit the color information.) The basic equations to convert between R´G´B´ and YIQ are:

$$Y = 0.299R' + 0.587G' + 0.114B' \tag{5.28}$$

$$I = 0.596R' - 0.275G' - 0.321B' \tag{5.29}$$

$$= V\cos 33^\circ - U\sin 33^\circ = 0.736(R' - Y) - 0.268(B' - Y) \tag{5.30}$$

$$Q = 0.212R' - 0.523G' + 0.311B' \tag{5.31}$$

$$= V\sin 33^\circ + U\cos 33^\circ = 0.478(R' - Y) + 0.413(B' - Y) \tag{5.32}$$

$$R' = Y + 0.956I + 0.621Q \tag{5.33}$$

$$G' = Y - 0.272I - 0.647Q \tag{5.34}$$

$$B' = Y - 1.107I + 1.704Q \tag{5.35}$$

For digital R´G´B´ values with a range of 0–255, Y has a range of 0–255, I has a range of 0 to ±152, and Q has a range of 0 to ±134. I and Q are obtained by rotating the U and V axes 33°. These equations are usually scaled to simplify the implementation in an actual NTSC digital encoder or decoder (Keith, 2007).

5.4.3.5 YCbCr Color Space

The YCbCr color space was developed as part of ITU-R BT.601 during the development of a world-wide digital component video standard YCbCr is a scaled and offset version of the YUV color space. Y is defined to have a nominal 8-bit range of 16–235; Cb and Cr are defined to have a nominal range of 16–240. There are several YCbCr sampling formats, such as 4:4:4, 4:2:2, 4:1:1, and 4:2:0 (Keith, 2007).

To convert 8-bit digital R´G´B´ data with a 16–235 nominal range (Studio R´G´B´) to YCbCr, the analog equations may be simplified to:

$$Y = 0.299R' + 0.587G' + 0.114B' \tag{5.36}$$

$$Cb = -0.172R' - 0.339G' + 0.511B' + 128 \tag{5.37}$$

$$Cr = 0.511R' - 0.428G' - 0.083B' + 128 \tag{5.38}$$

To convert 8-bit YCbCr to R´G´B´ data with a 16–235 nominal range (Studio R´G´B´), the analog equations may be simplified to (Keith, 2007) :

$$R' = Y + 1.371(Cr - 128) \tag{5.39}$$

$$G' = Y - 0.698(Cr - 128) - 0.336(Cb - 128) \tag{5.40}$$

$$B' = Y + 1.732(Cb - 128) \tag{5.41}$$

5.4.4 Color Correction

Color correction is done to change the appearance of the colors in a sequence. Today it is used in movies to make a certain impression on parts of the movie. For example the movie Traffic (it is a movie that awarded with 4 Academy Awards) made the parts that took place in Mexico yellower, and the parts that took part in America bluer. Color correction is also used to correct color faults because of difficult lighting conditions when filming a scene. Some outdoor scenes can have different lighting that makes the color change between shots (Skogmar, K. 2003).

RGB is the most common format for storing images. RGB means red, green and blue, that together can make up all other colors.

There are many ways to make color correction. Usually the Color space axes are changed to some more useful ones. Then the values are changed, and then the colors are transformed back to the previous ones, usually RGB.

Color correction can be implemented to platform with look up tables. This way increases the speed of color enhancement procedure.

In photography and image processing, color balance is the global adjustment of the intensities of the colors (typically red, green, and blue primary colors). An important goal of this adjustment is to render specific colors – particularly neutral colors – correctly; hence, the general method is sometimes called gray balance, neutral balance, or white balance. Color balance changes the overall mixture of colors in an image and is used for color correction; generalized versions of color balance are used to get colors other than neutrals to also appear correct or pleasing (*Color balance,* 2011).

The color balance operations in popular image editing applications usually operate directly on the red, green, and blue channel pixel values. Color balance is normally reserved to refer to correction for differences in the ambient illumination conditions.

Color balancing is sometimes performed on a three-component image (e.g., RGB) using a 3x3 matrix. This type of transformation is appropriate if the image were captured using the wrong white balance setting on a digital camera, or through a color filter (*Color balance,* 2011).

In principle, one wants to scale all relative luminances in an image so that objects which are believed to be neutral appear so. If, say, a surface with R = 240 was believed to be a white object, and if 255 is the count which corresponds to white, one could multiply all red values by 255/240. Doing analogously for green and blue would result, at least in theory, in a color balanced image. In this type of transformation the 3x3 matrix is a diagonal matrix.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255/R'_w & 0 & 0 \\ 0 & 255/G'_w & 0 \\ 0 & 0 & 255/B'_w \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{5.42}$$

where R, G, and B are the color balanced red, green, and blue components of a pixel in the image; R', G', and B' are the red, green, and blue components of the image before color balancing, and R'_w, G'_w, and B'_w are the red, green, and blue components of a pixel which is believed to be a white surface in the image before color balancing. This is a simple scaling of the red, green, and blue channels (*Color balance,* 2011).

An example of a color correction is in figure 5.16

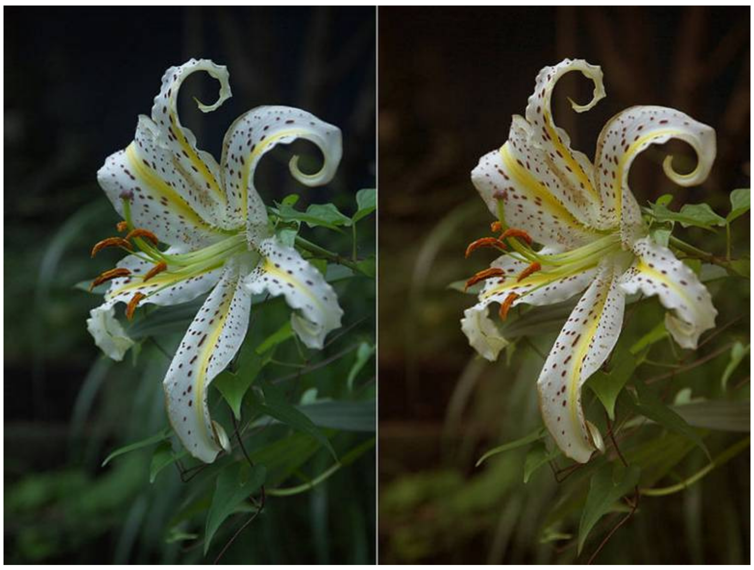


Figure 5.16 Color correction. Left hand side is original, right hand side is color corrected.

Tone correction of three common tonal imbalances are flat, light and dark images, it is illustrated in figure 5.17.

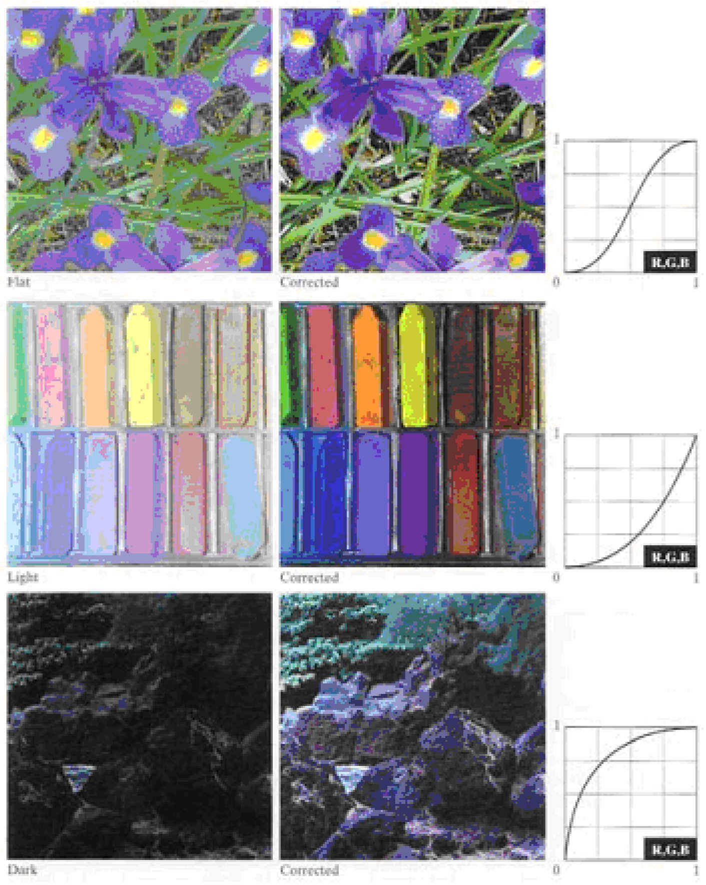


Figure 5.17 Tonal corrections for flat, light (high key), and dark (low key) color images. Adjusting the red, green, and blue components equally does not alter the image hues. (Gonzales, & Woods2002).

CHAPTER SIX
VIDEO BASICS

6.1 Digital Video

Initially, video contained only gray-scale (also called black-and-white) information. While color broadcasts were being developed, attempts were made to transmit color video using analog RGB (red, green, blue) data. However, this technique occupied 3× more bandwidth than the current gray-scale solution, so alternate methods were developed that led to using Y, R–Y, and G–Y data to represent color information. A technique was then developed to transmit this Y, R–Y, and G–Y information using one signal, instead of three separate signals, and in the same bandwidth as the original gray-scale video signal. This composite video signal is what the NTSC, PAL, and SECAM video standards are still based on today (Keith, 2007).

Today, even though there are many ways of representing video, they are still all related mathematically to RGB. YPbPr video signals are used for connecting consumer video products, so it is very popular in television market. RGB and YPbPr signals are analog video signals. YCbCr is the digitized version of the analog YPbPr video signals and it is used in DVDs and digital broadcasting.

6.2 Video Timing

Although it looks like video is continuous motion, it is actually a series of still images, changing fast enough. This typically occurs 50 or 60 times per second for consumer video, and 70–90 times per second for computer displays. Special timing information, called vertical sync, is used to indicate when a new image is starting (Keith, 2007).

Each still image is also composed of scan lines, lines of data that occur sequentially one after another down the display additional timing information, called horizontal sync, is used to indicate when a new scan line is starting. The vertical and horizontal sync information is usually transferred in one of three ways:

- Separate horizontal and vertical sync signals
- 2. Separate composite sync signal
- 3. Composite sync signal embedded within the video signal

The composite sync signal is a combination of both vertical and horizontal sync. Computer and consumer equipment that uses analog RGB video usually uses technique 1 or 2. Consumer equipment that supports composite video or analog YPbPr video usually uses the last technique (3rd one) (Keith, 2007).

6.3 Interlaced Vs. Progressive

Since video is a series of still images, it makes sense to simply display each full image consecutively, one after the another. This is the basic technique of progressive, or non-interlaced, displays. For progressive displays that "paint" an image on the screen, such as a CRT, each image is displayed starting at the top left corner of the display, moving to the right edge of the display. Then scanning then moves down one line, and repeats scanning left-to-right. This process is repeated until the entire screen is refreshed, as seen in Figure 6.1(Keith, 2007).

In the early days of television, a technique called "interlacing" was used to reduce the amount of information sent for each image. By transferring the odd-numbered lines, followed by the even-numbered lines (as shown in Figure 6.2), the amount of information sent for each image was halved. Given this advantage of interlacing, why bother to use progressive? (Keith, 2007).

With interlace, each scan line is refreshed half as often as it would be if it were a progressive display. Therefore, to avoid line flicker on sharp edges due to a too-low frame rate, the line-to-line changes are limited, essentially by vertically lowpass filtering the image. A progressive display has no limit on the line-to-line changes, so is capable of providing a higher resolution image (vertically) without flicker. Today, most broadcasts (including HDTV) are still transmitted as interlaced. Most CRT-based displays are still interlaced while LCD, plasma, and computer displays are progressive (Keith, 2007).

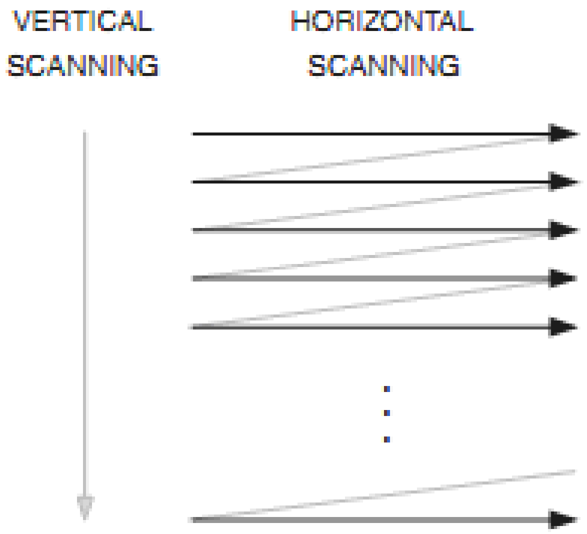


Figure 6.1 Progressive display "paint" the lines of an image consecutivitely, one after another (Keith, 2007).

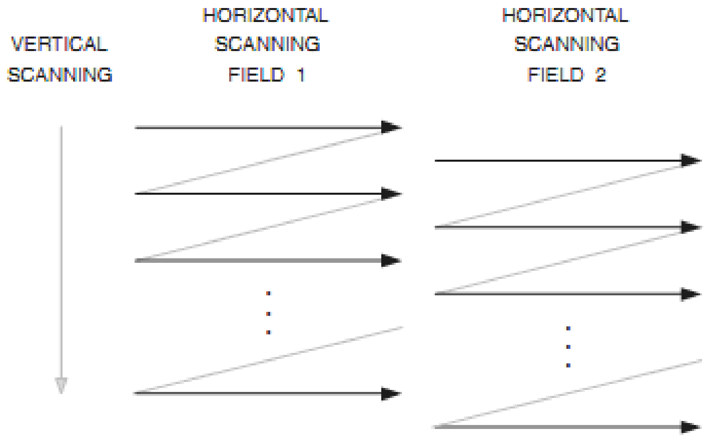


Figure 6.2 Interlaced display "paint" first one-half of the image (odd lines), then the other half (even lines) (Keith, 2007).

6.4 PAL Composite Video Interface

The Phase Alternate Line (PAL) is one of the worldwide used analog television encoding systems. Luminance, synchronization signals, chrominance signal are carried by PAL signals. It supports 4.43 MHz color carrier frequency. The vertical synchronization is 50 Hz and the horizontal synchronization is 16.625 kHz. The resolution of the PAL video signal is 864 x 625.

6.5 NTSC Composite Video Interface

The National Television System Committee (NTSC) is the analog television system. Luminance, synchronization signals, chrominance signal are carried by NTSC signals. It supports 3.579545 MHz color carrier frequency. The vertical synchronization is 59.94 Hz and the horizontal synchronization is 15.734 kHz. The resolution of the NTSC video signal is 858 x 525.

6.6 Composite/CVBS Interface

The most basic video signal is composite video signal which is called CVBS (Composite Video Baseband Signal) . It combines lumination, color, blanking, and sync signals in one cable. A typical waveform of an NTSC composite video signal is shown in Figure 6.3.

Color information is added on top of the luma signal and is a sine wave with the colors identified by a specific phase difference between it and the color-burst reference phase. This can be seen in Figure 6.4, which shows a horizontal scan line of color bars (Maxim, 2001).

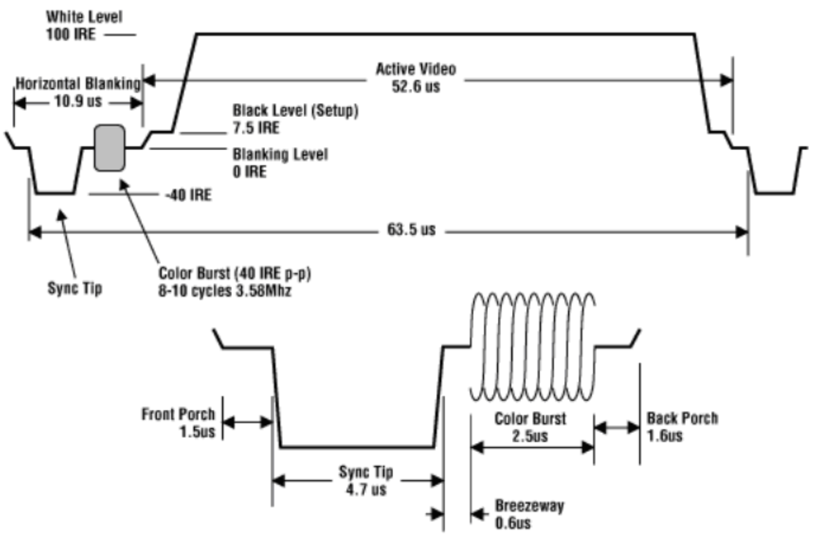


Figure 6.3 NTSC Composite Video Waveform (Maxim, 2001).

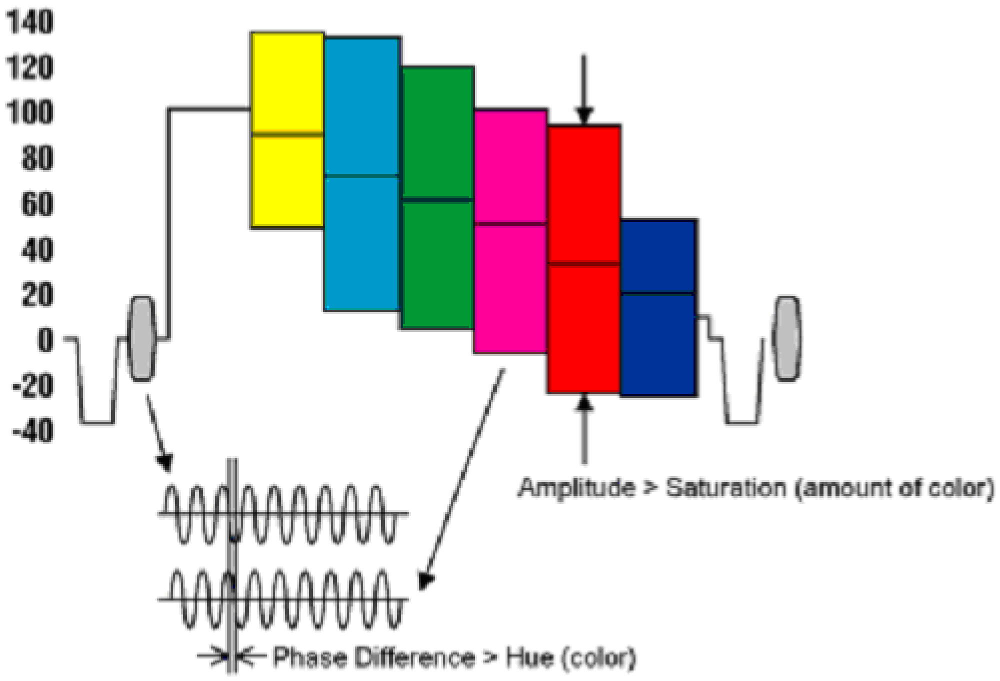

140
120
100
80
60
40
20
0
-20
-40
Amplitude > Saturation (amount of color)
Phase Difference > Hue (color)

Figure 6.4 NTSC Composite Video Waveform : color bars (Maxim, 2001).

The amplitude of the modulation is proportional to the amount of color (or saturation), and the phase information denotes the tint (or hue) of the color.

6.7 Computer Signal Interfaces

Computer interfaces are consist of red (R), green (G), blue (B), horizontal synchronization signal (H) and vertical synchronization signal (V).

The five signals, R, G, B, H, and V, are carried on one cable consisting of a shielded bundle of wires. The connector is almost always a 15-pin D-type connector. Sometimes the H and V sync information is merged with one of the RGB signals, typically the green component, but this is becoming less common. This is referred to as "sync on green." In rarer cases, the sync information is on the red or the blue signal (Maxim, 2001).

6.8 ITU-R BT 656 4:2:2 YCrCb Video Format

The BT.656 parallel interface uses 8 bits or 10 bits of multiplexed YCbCr data and a 27MHz clock. Conventional video timing signals (HSYNC, VSYNC, and BLANK) also being transmitted by video decoder however BT.656 uses unique timing codes embedded within the video stream. This reduces the number of wires and IC pins required for a BT.656 video interface. The 4:2:2 YCbCr data is multiplexed into an 8-bit stream: Cb0Y0Cr0Y1Cb2Y2Cr2, etc (Günay, 2010).

Figure 6.5 and Figure 6.6 illustrate the format for 525/60 and 625/50 video systems, respectively.

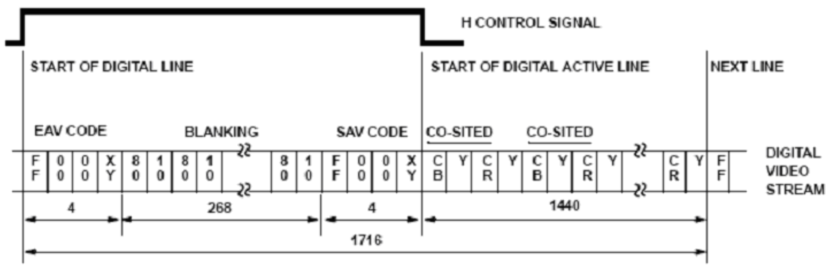


Figure 6.5 BT.656 8-bit parallel interface data format for NTSC (525/60) video stream (Intersil, 2002).

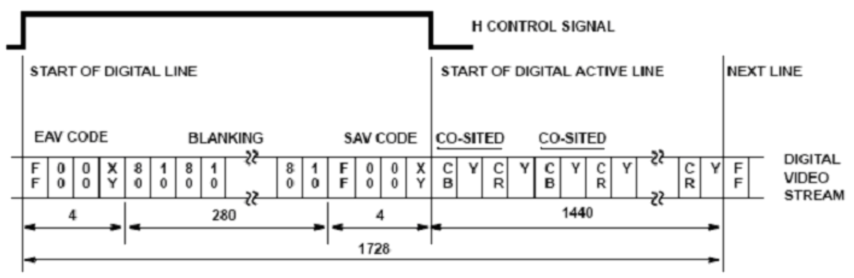


Figure 6.6 BT.656 8-bit parallel interface data format for PAL (625/50) video stream (Intersil, 2002).

CHAPTER SEVEN
ALTERA DE2-70 EVALUATION BOARD SYSTEM ARCHITECTURE

7.1 Introduction

This chapter presents the features and design characteristics of the DE2-70 board.

A photograph of the DE2-70 board is shown in Figure 7.1. It depicts the layout of the board and indicates the location of the connectors and key components.

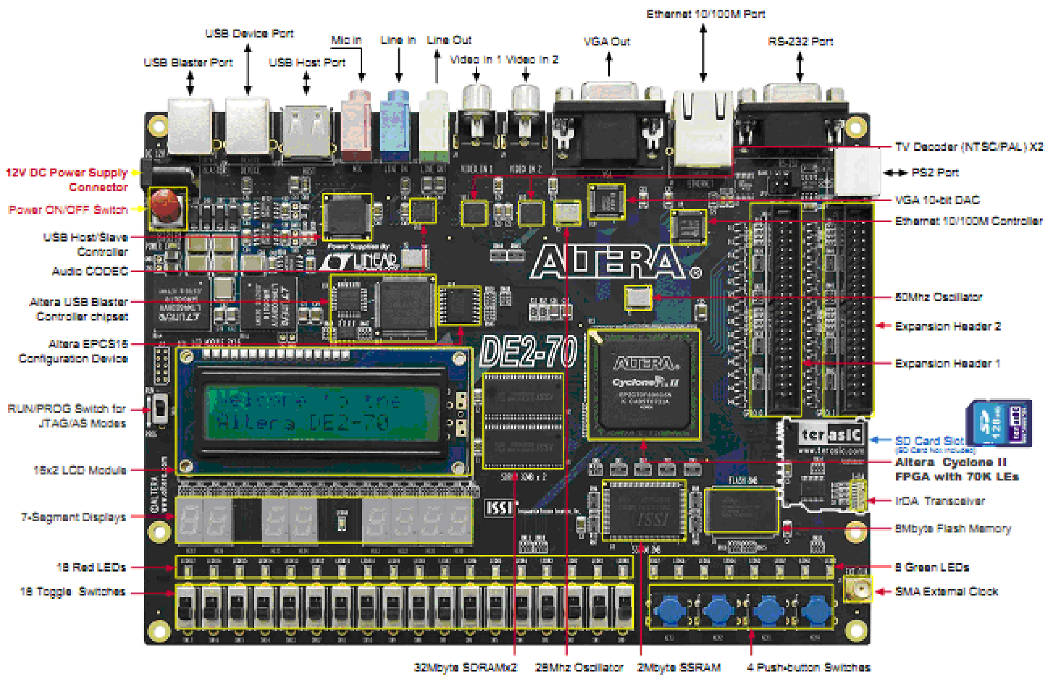


Figure 7.1 The DE2-70 Board (Terasic 2009).

The DE2-70 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the DE2-70 board:

- Altera Cyclone® II 2C70 FPGA device
- Altera Serial Configuration device - EPCS16

- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 2-Mbyte SSRAM
- Two 32-Mbyte SDRAM
- 8-Mbyte Flash memory
- SD Card socket
- pushbutton switches
- 18 toggle switches
- 18 red user LEDs
- green user LEDs
- 50-MHz oscillator and 28.63-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- 10/100 Ethernet Controller with a connector
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- 1 SMA connector
- Two 40-pin Expansion Headers with diode protection

In addition to these hardware features, the DE2-70 board has software support for standard I/O interfaces and a control panel facility for accessing various components.

7.2 Block Diagram Of The DE2-70 Board

Figure 7.2 gives the block diagram of the DE2-70 board. To provide maximum flexibility for the user, all connections are made through the Cyclone II FPGA device. Thus, the user can configure the FPGA to implement any system design.

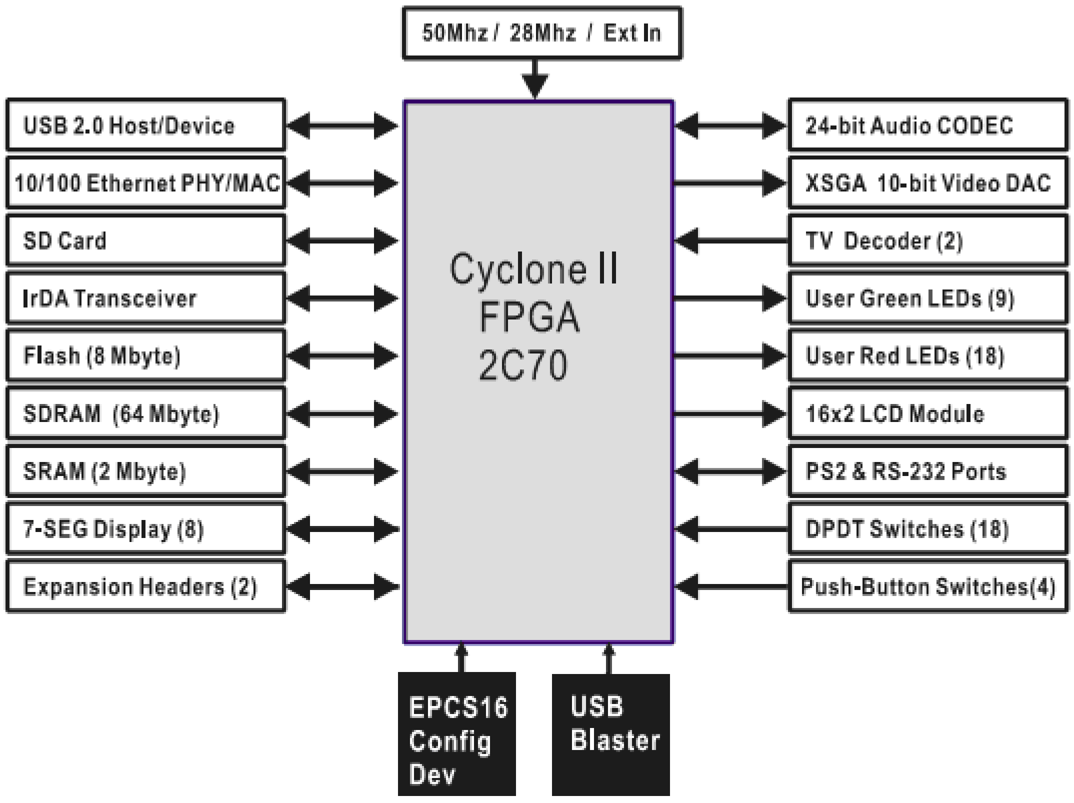


Figure 7.2 Block Diagram of the DE2-70 Board (Terasic 2009).

Following is more detailed information about the blocks in Figure 7.2:

Cyclone II 2C70 FPGA

- 68,416 LEs
- 250 M4K RAM blocks
- 1,152,000 total RAM bits
- 150 embedded multipliers
- PLLs
- 622 user I/O pins
- FineLine BGA 896-pin package

Serial Configuration device and USB Blaster circuit

- Altera's EPCS16 Serial Configuration device
- On-board USB Blaster for programming and user API control
- JTAG and AS programming modes are supported

SSRAM

- 2-Mbyte standard synchronous SRAM
- Organized as 512K x 36 bits
- Accessible as memory for the Nios II processor and by the DE2-70 Control Panel

SDRAM

- Two 32-Mbyte Single Data Rate Synchronous Dynamic RAM memory chips
- Organized as 4M x 16 bits x 4 banks
- Accessible as memory for the Nios II processor and by the DE2-70 Control Panel

Flash Memory

- 8-Mbyte NOR Flash memory
- Support both byte and word mode access
- Accessible as memory for the Nios II processor and by the DE2-70 Control Panel

SD Card Socket

- Provides SPI and 1-bit SD mode for SD Card access
- Accessible as memory for the Nios II processor with the DE2-70 SD Card Driver

Pushbutton Switches

- pushbutton switches
- Debounced by a Schmitt trigger circuit
- Normally high; generates one active-low pulse when the switch is pressed

Toggle Switches

- 18 toggle switches for user inputs

- A switch causes logic 0 when in the DOWN (closest to the edge of the DE2-70 board) position and logic 1 when in the UP position

Clock Inputs

- 50-MHz oscillator
- 28.63-MHz oscillator
- SMA external clock input

Audio CODEC

- Wolfson WM8731 24-bit sigma-delta audio CODEC
- Line-level input, line-level output, and microphone input jacks
- Sampling frequency: 8 to 96 KHz
- Applications for MP3 players and recorders, PDAs, smart phones, voice recorders, etc.

VGA Output

- Uses the ADV7123 140-MHz triple 10-bit high-speed video DAC
- With 15-pin high-density D-sub connector
- Supports up to 1600 x 1200 at 100-Hz refresh rate
- Can be used with the Cyclone II FPGA to implement a high-performance TV Encoder

NTSC/PAL/ SECAM TV Decoder Circuit

- Uses two ADV7180 Multi-format SDTV Video Decoders
- Supports worldwide NTSC/PAL/SECAM color demodulation
- One 10-bit ADC, 4X over-sampling for CVBS
- Supports Composite Video (CVBS) RCA jack input
- Supports digital output formats : 8-bit ITU-R BT.656 YCrCb 4:2:2 output + HS, VS, and FIELD
- Applications: DVD recorders, LCD TV, Set-top boxes, Digital TV, Portable video devices, and TV PIP (picture in picture) display.

10/100 Ethernet Controller

- Integrated MAC and PHY with a general processor interface
- Supports 100Base-T and 10Base-T applications
- Supports full-duplex operation at 10 Mb/s and 100 Mb/s, with auto-MDIX
- Fully compliant with the IEEE 802.3u Specification
- Supports IP/TCP/UDP checksum generation and checking
- Supports back-pressure mode for half-duplex mode flow control

USB Host/Slave Controller

- Complies fully with Universal Serial Bus Specification Rev. 2.0
- Supports data transfer at full-speed and low-speed
- Supports both USB host and device
- Two USB ports (one type A for a host and one type B for a device)
- Provides a high-speed parallel interface to most available processors; supports Nios II with a Terasic driver
- Supports Programmed I/O (PIO) and Direct Memory Access (DMA)

Serial Ports

- One RS-232 port
- One PS/2 port
- DB-9 serial connector for the RS-232 port
- PS/2 connector for connecting a PS2 mouse or keyboard to the DE2-70 board

IrDA Transceiver

- Contains a 115.2-kb/s infrared transceiver
- 32 mA LED drive current
- Integrated EMI shield
- IEC825-1 Class 1 eye safe
- Edge detection input

Two 40-Pin expansion Headers

- 72 Cyclone II I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
- 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives
- Diode and resistor protection is provided

CHAPTER EIGHT
FPGA IMPLEMENTATION

8.1 Main Blocks Of Implementation

The design and simulation platform of this thesis is Quartus II 9.1 Web Edition. The design language is Verilog. After writing codes in verilog and arranging FPGA pin assignments, design is compiled. After compilation, the code is ready for downloading to the DE2-70 board. In order to install the generating code, quartus programmer is used via USB blaster.

The project is consist of three main parts as in figure 8.1. First part is decoding analog video signal (NTSC CVBS signal). Second part is implementing all the algorithms to FPGA, the last part is digital to analog converter part for driving a VGA source.

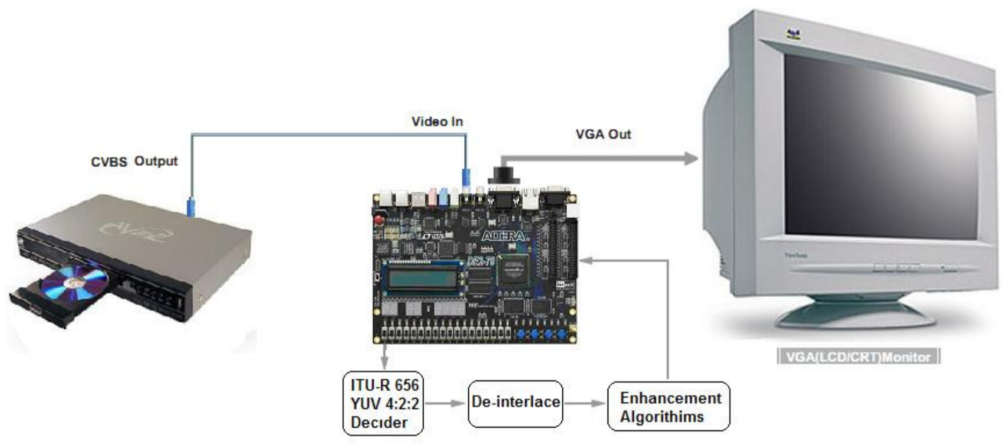


Figure 8.1 Project parts : Decoding Video, FPGA Implementation, Driving VGA Monitor

8.1.1 Decoding Analog Video Signal Part (TV Decoder)

There are two Analog Devices ADV7180 TV decoder chips on the DE2-70 board. The ADV7180 is an integrated video decoder that automatically detects and converts a standard analog baseband television signal (NTSC, PAL, and SECAM), which is called CVBS, into 4:2:2 component video data compatible with the 8-bit ITU-R BT.656 interface standard.

ADV7180 Decoder chips are programmed by a serial I2C bus, which is connected to the FPGA. The related registers and values are stored in I2C Configuration Module.

Because our code is compatible with NTSC CVBS signal, a DVD or VCD player neaded for a CVBS source. It has to be configured as fallow:

- NTSC output
- 60 Hz refresh rate
- 4:3 aspect ratio
- Non-progressive video

8.1.2 FPGA Part

Figure 8.2 shows the block diagram of the design. There are several blocks in the circuit. The TV_to_VGA block consists of the ITU-R 656 Decoder, SDRAM Frame Buffer, YUV422 to YUV444, YcrCb to RGB, and VGA Controller and image processing blocks. Blue colored and green colored blocks are designed by Altera, and it can be copied from Altera DE2-70 demoboard CD. Blue coloreed blocks are original blocks while green colored blocks are modified (some features are added like demo mode enable signal) . Image processing algorithms (noise reduction, sharpness, contrast and color enhancements) are implemented after YCbCr and RGB modules. Yellow colored blocks are written in verilog and implemented on Altera design.

The register values of the TV Decoder chip are used to configure the TV decoder via the I2C_AV_Config block, which uses the I2C protocol to communicate with the TV Decoder chip. Following the power-on sequence, the TV Decoder chip will be unstable for a time period; the Lock Detector is responsible for detecting this instability.

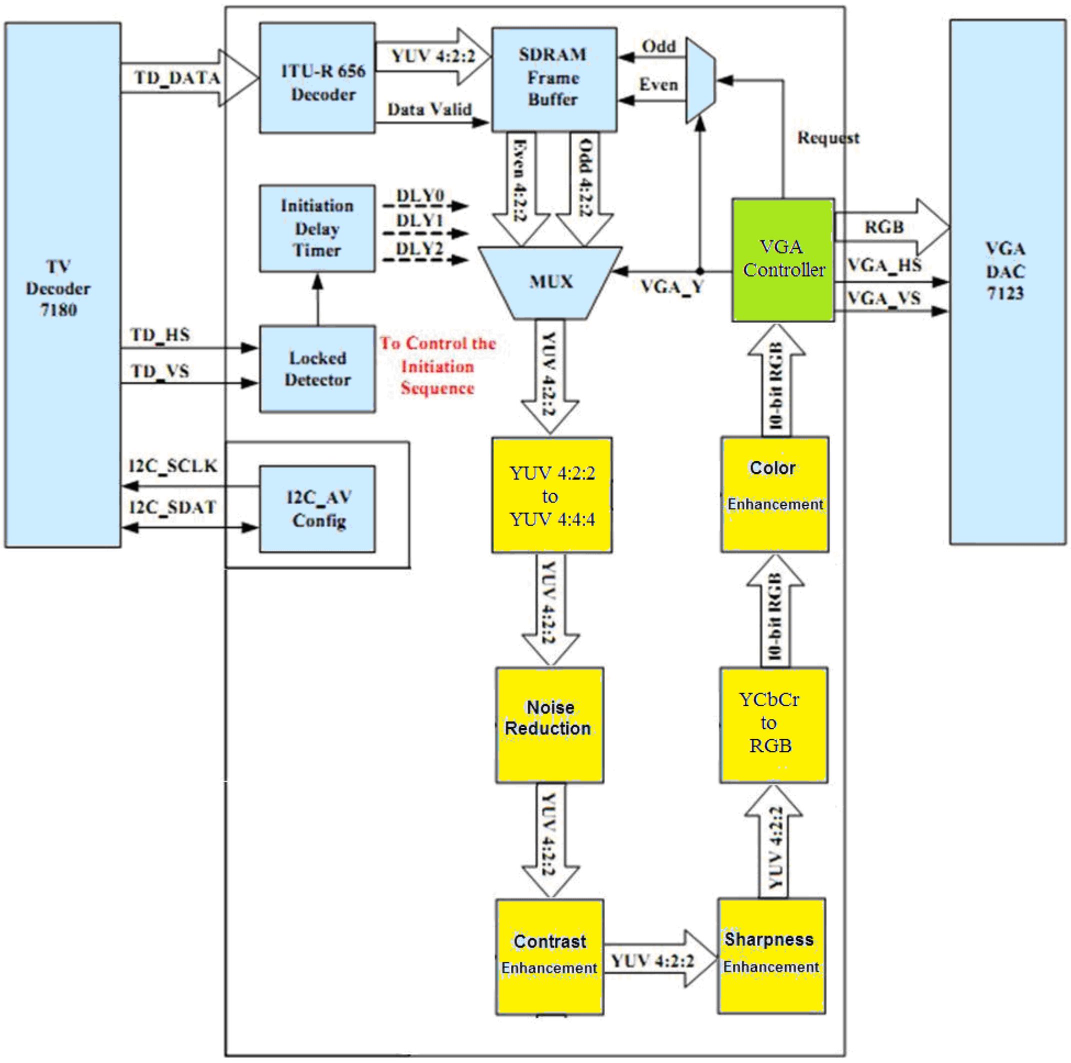


Figure 8.2 Block Diagram of the project

The ITU-R 656 Decoder block extracts YCrCb 4:2:2 (YUV 4:2:2) video signals from the ITU-R 656 data stream sent from the TV Decoder. It also generates a data valid control signal indicating the valid period of data output. Because the video signal from the TV decoder is interlaced, we need to perform de-interlacing on the data source. We used the SDRAM frame buffer and a field selection multiplexer (MUX) which is controlled by the VGA controller to perform the de-interlacing operation. Internally, the VGA Controller generates data request and odd/even selected signals to the SDRAM frame buffer and filed selection multiplexer (MUX). The YUV422 to YUV444 block converts the selected YcrCb 4:2:2 (YUV 4:2:2) video data to the YcrCb 4:4:4 (YUV 4:4:4) video data format.

Finally, the YCrCb_to_RGB block converts the YCrCb data into RGB output. The VGA Controller block generates standard horizontal sync and vertical sync signals to drive a VGA monitor.

8.1.3 Implementing A TV Encoder Part

DE2-70 board has a TV decoder but does not have a TV encoder chip. But it has a ADV7123 (10-bit high-speed triple ADCs) chip. It can be used as a TV encoder with the digital processing part implemented in the Cyclone II FPGA. Figure 8.3 shows the block diagram of TV encoder part.

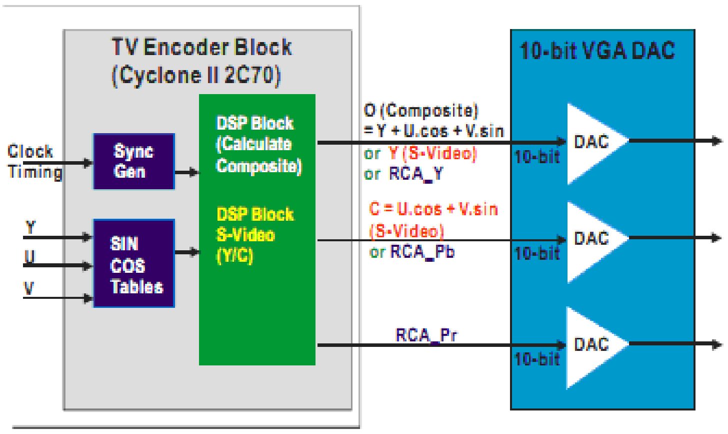


Figure 8.3 A TV Encoder that uses the Cyclone II FPGA and the ADV7123.

8.1.3.1 Using VGA

The DE2-70 board includes a 16-pin D-SUB connector for VGA output. The VGA synchronization signals are provided directly from the Cyclone II FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC is used to produce the analog data signals (red, green, and blue). It can support resolutions of up to 1600 x 1200 pixels, at 100 MHz.

Figure 8.4 illustrates the basic timing requirements for each row (horizontal) that is displayed on a VGA monitor. An active-low pulse of specific duration (time a in the figure) is applied to the horizontal synchronization (hsync) input of the monitor, which signifies the end of one row of data and the start of the next. The data (RGB) inputs on the monitor must be off (driven to 0 V) for a time period called the back porch (b) after the hsync pulse occurs, which is followed by the display interval (c). During the data display interval the RGB data drives each pixel in turn across the row being displayed. Finally, there is a time period called the front porch (d) where the RGB signals must again be off before the next hsync pulse can occur. The timing of the vertical synchronization (vsync) is the same as shown in Figure 8.4, except that a vsync pulse signifies the end of one frame and the start of the next, and the data refers to the set of rows in the frame (horizontal timing). Table 8.1 and 8.2 show, for different resolutions, the durations of time periods a, b, c, and d for both horizontal and vertical timing.

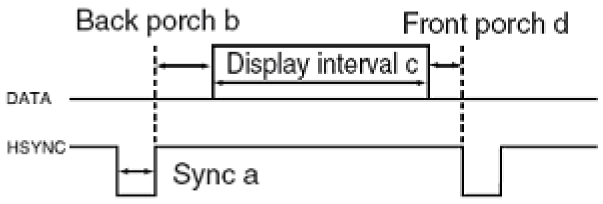


Figure 8.4 VGA Horizontal timing specifications

Table 8.1 VGA Horizontal timing specifications

VGA mode		Horizontal Timing Spec				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(Mhz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25 (640/c)
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36 (640/c)
SVGA(60Hz)	800x600	3.2	2.2	20	1	40 (800/c)
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49 (800/c)
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56 (800/c)
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65 (1024/c)
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75 (1024/c)
XGA(85Hz)	1024x768	1.0	2.2	10.8	0.5	95 (1024/c)
1280x1024(60Hz)	1280x1024	1.0	2.3	11.9	0.4	108 (1280/c)

Table 8.2 VGA Vertical timing specifications

VGA mode		Vertical Timing Spec			
Configuration	Resolution (HxV)	a(lines)	b(lines)	c(lines)	d(lines)
VGA(60Hz)	640x480	2	33	480	10
VGA(85Hz)	640x480	3	25	480	1
SVGA(60Hz)	800x600	4	23	600	1
SVGA(75Hz)	800x600	3	21	600	1
SVGA(85Hz)	800x600	3	27	600	1
XGA(60Hz)	1024x768	6	29	768	3
XGA(70Hz)	1024x768	6	29	768	3
XGA(85Hz)	1024x768	3	36	768	1
1280x1024(60Hz)	1280x1024	3	38	1024	1

8.2 Project Modules

This subsection describes the written verilog modules and their functionality.

8.2.1 Decoding Module

This module provides ITU.BT 656 YCrCb and synchronization signals decoding. The ITU-R 656 Decoder block extracts YcrCb 4:2:2 (YUV 4:2:2) video signals from the ITU-R 656 data stream sent from the TV Decoder. It also generates a data valid control signal indicating the valid period of data output.

Input of this module is 8-bit serial video data from the TV Decoder. Output of this module is 16-bit serial YCbCr video data. This module is "ITU-R 656 Decoder" in figure 8.2.

8.2.2 I2C Configuration Module

This module's name is I2C_AV_Config. As soon as the bit stream is downloaded into the FPGA, the register values of the TV Decoder chip are used to configure the TV decoder via the I2C_AV_Config block, which uses the I2C protocol to communicate with the TV Decoder chip. All the related registers and their values are stored in this module.

This module provides I2C programming, considering ADV7183 video decoder as slave and FPGA as master. This module is "I2C_AV Config" in figure 8.2.

8.2.3 Lock Detector Module

Following the power-on sequence, the TV Decoder chip will be unstable for a time period; the lock detector is responsible for detecting this instability. It counts 9 vertical sync pulses, after 9 vertical sync pulses detection, it generates TD_Stable signal to the reset module which resets all the blocks. This module is "Locked Detector" in figure 8.2.

8.2.4 Reset Module

This module is responsible for resetting the whole sytem. There are 3 reset signals, Reset0, Reset1 and Reset2. Reset module counts $(0)_{HEX}$ to $(0FFFFF)_{HEX}$. Then it generates Reset0 signals. This signal resets YUV422_to_444 Module, Contrast Enhancement Module, Sharpness Module, Noise Reduction Module and RAM Module. Then Reset Module continues counting $(0FFFFF)_{HEX}$ to $(1FFFFF)_{HEX}$, and it generates Reset1 signal. Reset1 signal resets Decoding Module (ITU-R 656 Module). After counting from $(1FFFFF)_{HEX}$ to $(2FFFFF)_{HEX}$, Reset Module generates Reset3 signal. This signal resets YCbCr2RGB Module, Color Enhancement Module and VGA Control Module. This module is "Initiation Delay Timer" in figure 8.2.

8.2.5 De-Interlacing Block

Because the video signal from the TV Decoder is interlaced, we need to perform de-interlacing on the data source to drive a VGA Monitor. We used the SDRAM Frame Buffer and a field selection multiplexer (MUX) which is controlled by the

VGA controller to perform the de-interlacing operation. Internally, the VGA Controller generates data request and odd/even selected signals to the SDRAM Frame Buffer and filed selection multiplexer (MUX).

The output of ITU-R 656 Module 640x240 odd video data and 640x240 even video data are stored in the SDRAM seperately. Then MUX side selects one line odd video data and one line even data. After completing 240 odd line and 240 even line, we have a 640x480 video signal which VGA modules need. This operation is called de-interlacing. This modules are "SDRAM Frame Buffer, MUX and VGA_Controller" in figure 8.2.

8.2.6 YUV422 To YUV444 Block

The YUV422 to YUV444 block converts the selected YCrCb 4:2:2 (YUV 4:2:2) video data to the YCrCb 4:4:4 (YUV 4:4:4) video data format by repeating chrominance information for two pixels. The input is 16-bit YCrCb serial data, the output is 24-bit paralel Y, Cb and Cr data. This module is "YUV 4:2:2 to YUV 4:4:4" in figure 8.2.

8.2.7 YCrCb to RGB Block

YCrCb_to_RGB block converts the YCrCb data into RGB output. Equation 8.1 to 8.3 are used for converting operations. In FPGA implementation, all the coefficients are multiplied by 512. After converting operations, the result is divided to 128 instead of 512 because if we multiply a 8-bit signal with 512 and devide 128, we get 10-bit output data instead of 8-bit.

$$R' = 1.164(Y - 16) + 1.596(Cr - 128) \tag{8.1}$$

$$G' = 1.164(Y - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \tag{8.2}$$

$$B' = 1.164(Y - 16) + 2.018(Cb - 128) \tag{8.3}$$

Input of this module is 24-bit YCbCr signal and the output signal is 30-bit RGB signal which is suitable for driving a VGA monitor. This module is “YCbCr to RGB” in figure 8.2.

8.2.8 VGA Controller

This block generates standard VGA sync signals VGA_HS and VGA_VS to enable the display on a VGA monitor. This module is “VGA Controller” in figure 8.2.

8.3 Image Processing Modules

8.3.1 Contrast Enhancement Module

Many contrast curves have been tried for the best result in contrast improving. The most critical question here is “ has incoming video data has more black areas or more white areas?”. According to the answer of this question, improvement can be done to desired contrast. To increase the contrast, dark areas are set darker and bright areas are set brighter, thus mid areas can be more noticeble as in figure 8.6. But some video contents have a lot of dark areas, so an other curve is designed for dark contents. Dark areas are detailed with this curve as in figure 8.8. The curve in Figure 8.6 is for high contrast, in figure 8.7 is for normal contrast, and in figure 8.8 is for dark contents. The user can select one of them according to the video content.

In figure 8.6, blue line is original curve. This means that output pixel value of this module is equal to input pixel of this module. In other words, this module is bypassed. Horizontal line is input of this module, vertical line is output of this module. Red line depicts our high contrast curve. Dark areas are set darker and bright areas are set brighter. Mid areas are contrast stretched.

Look up tables are used for FPGA implementation of contrast enhancement as in figure 8.5. 8 bit Y_in, Cb_in and Cr_in signals are input signals of the contrast enhancement module. 8 bit Y_out, Cb_out and Cr_out signals are output signals of the contrast enhancement module. Y_out signal is selected from look up table according to Y_in signal. Cb_in and Cr_in signals are connected to Cb_out and Cr_out directly. Only “Y” component of the input YCbCr is modified in this module. Some normal curve of contrast enhancement look up table values are in table 8.3.

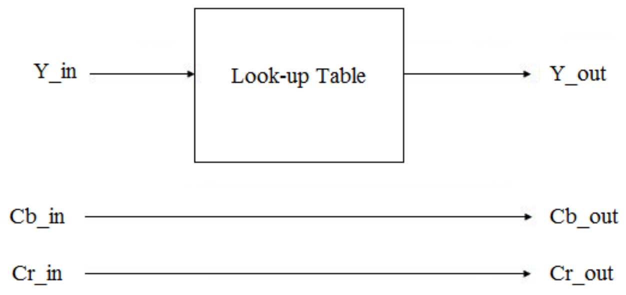


Figure 8.5 Block Diagram of Contrast Enhancement Module

Table 8.3 Contrast Enhancement (Normal Curve) Look Up Table Values

Y_in	Y_out
0	0
10	5
20	12
30	19
40	27
50	35
60	44
70	54
80	65
90	77
100	89
110	103
120	117
130	132
140	150
150	171
160	193
170	213
180	229
190	239
200	243
210	247
220	250
230	252
240	254
250	255
255	255

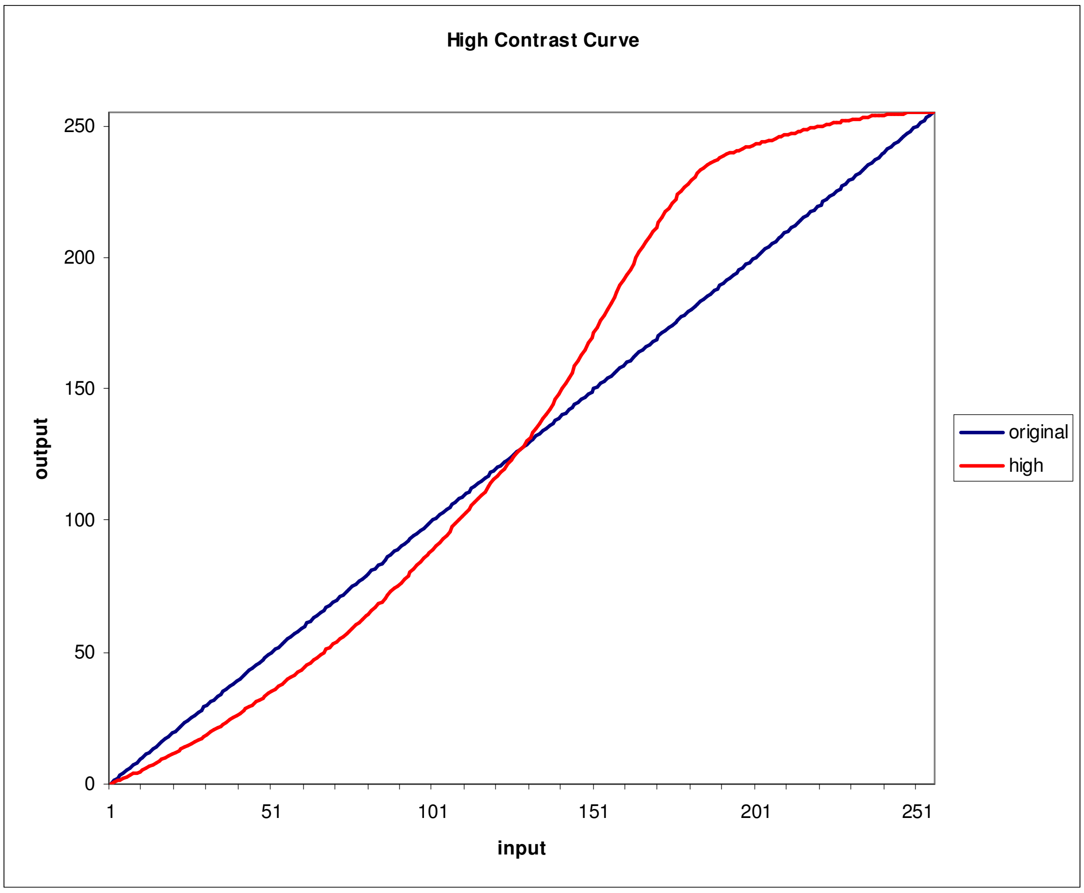


Figure 8.6 High Contrast Curve

In figure 8.7, blue line is original curve. This means that output pixel value of this module is equal to input pixel of this module. In other words, this module is bypassed. Horizontal line is input of this module, vertical line is output of this module. Red line depicts our normal contrast curve. Dark areas are set darker and bright areas are set brighter. Mid areas are remained same.

In figure 8.8, blue line is original curve. This means that output pixel value of this module is equal to input pixel of this module. In other words, this module is bypassed. Horizontal line is input of this module, vertical line is output of this module. Red line depicts our contrast curve for dark contents. Dark areas are set a little darker , mid and bright areas are stretched, so dark areas becomes more noticable.

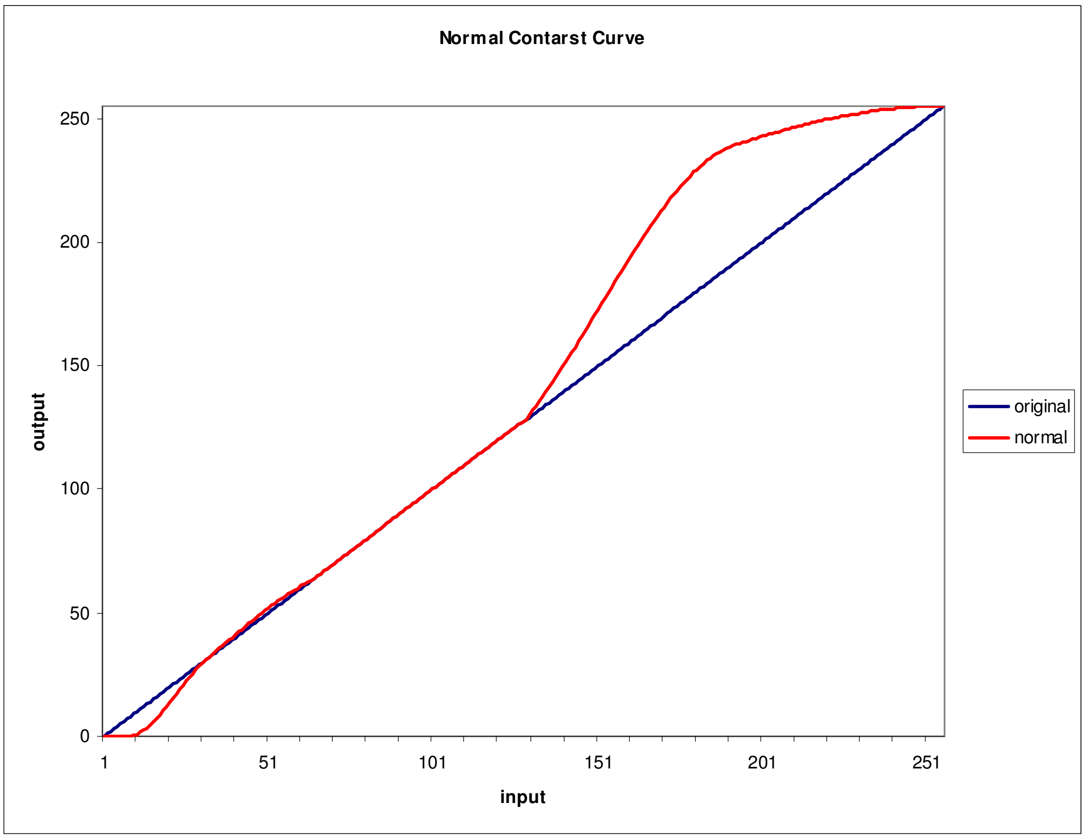


Figure 8.7 Normal Contrast Curve

Contrast enhancement module is applied just after the noise reduction module as in figure 8.2. The input of this module is 24-bit YCbCr video signal. The output signal is improved and 24-bit YCbCr video signal. Contrast enhancement algorithm is applied only “Y” signal. “Cb” and “Cr” signals are syncronized with the improved “Y” signal.

By-pass contrast enhancement option is added in this module for enhancement ON/OFF function.

There are 3 curves in this module which is selectible from the toggle switch on DE2-70 board. The switch configurations are below:

SW2 = 0,	SW3=0	Contrast Enhancement – High
SW2 = 1,	SW3=0	Contrast Enhancement – Normal
SW2= 0 or 1 ,	SW3=1	Contrast Enhancement – Bright

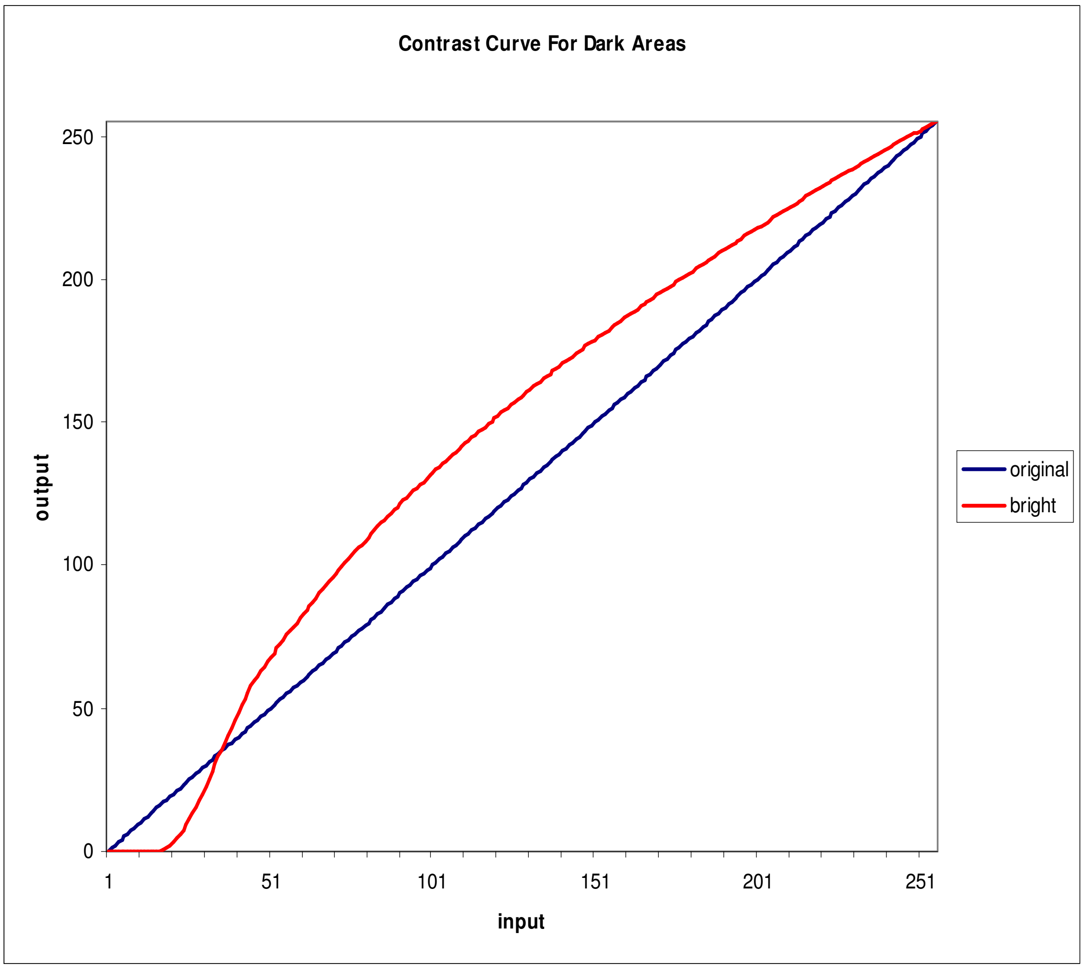
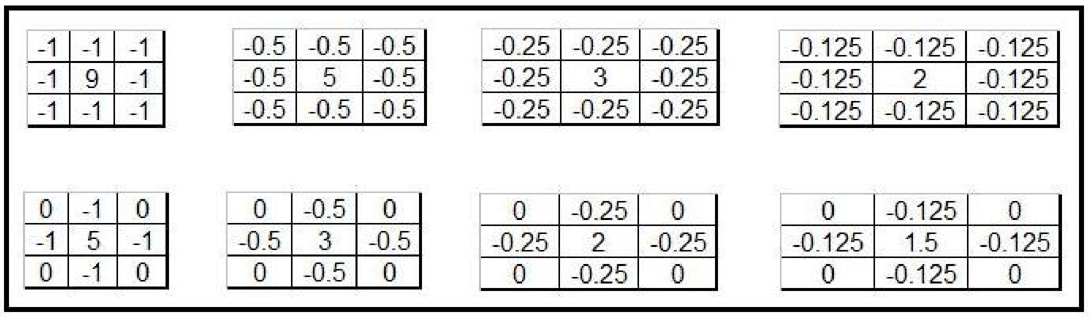


Figure 8.8 Contrast Curve for Dark Areas

8.3.2 Sharpness Module

If the picture enhanced more sharpen, some blocking noise is appears in the screen. If the video content is noisy, then sharpnes algorithms amplifies the noise. In other words, increasing sharpness increases the noise. So sharpness gives the best result in picture which doesn't have a lot of noise.

Kernels in figure 8.9 are tried for the best sharepnes solution. Some Kernels is not applicable for a CVBS signal. Because it amplifies the noise, some kernels gives very soft picture. Finally, kernel in figure 8.10 gives the best results. Kernel in left hand side of figure 8.10 is used for sharpness in medium (Normal) level, while right hand side is used for sharpness in soft level.



-1	-1	-1
-1	9	-1
-1	-1	-1

-0.5	-0.5	-0.5
-0.5	5	-0.5
-0.5	-0.5	-0.5

-0.25	-0.25	-0.25
-0.25	3	-0.25
-0.25	-0.25	-0.25

-0.125	-0.125	-0.125
-0.125	2	-0.125
-0.125	-0.125	-0.125

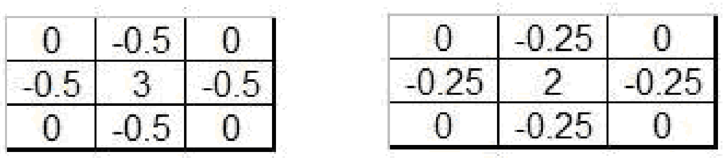
0	-1	0
-1	5	-1
0	-1	0

0	-0.5	0
-0.5	3	-0.5
0	-0.5	0

0	-0.25	0
-0.25	2	-0.25
0	-0.25	0

0	-0.125	0
-0.125	1.5	-0.125
0	-0.125	0

Figure 8.9 Sharpness Kernels



0	-0.5	0
-0.5	3	-0.5
0	-0.5	0

0	-0.25	0
-0.25	2	-0.25
0	-0.25	0

Figure 8.10 Sharpness Kernels for medium level and soft level

Sharpness module is implemented after the conntast enhancement module as in figure 8.2. The input of this module is 24-bit YCbCr video signal. The output signal is improved and 24-bit YCbCr video signal. Sharpness enhancement algorithm is applied only “Y” signal. “Cb” and “Cr” signals are syncronized with the improved “Y” signal as in figure 8.11.

There are 3 line buffers, 9 multipliers and 4 parellel adders in figure 8.11. P9, P8 and P7 are the last 3 values of line buffer 0. P6, P5 and P4 are the last 3 values of line buffer 1. P3, P2 and P1 are the last 3 values of line buffer 2. X1 to X9 values are 3x3 kernel values of sharpness enhancement. 3x3 input Y_in signal is multiplied with 3x3 kernel values. All the results are summed with parallel adders. The output signal of parallel adders (Y_out) is the output signal of this module.

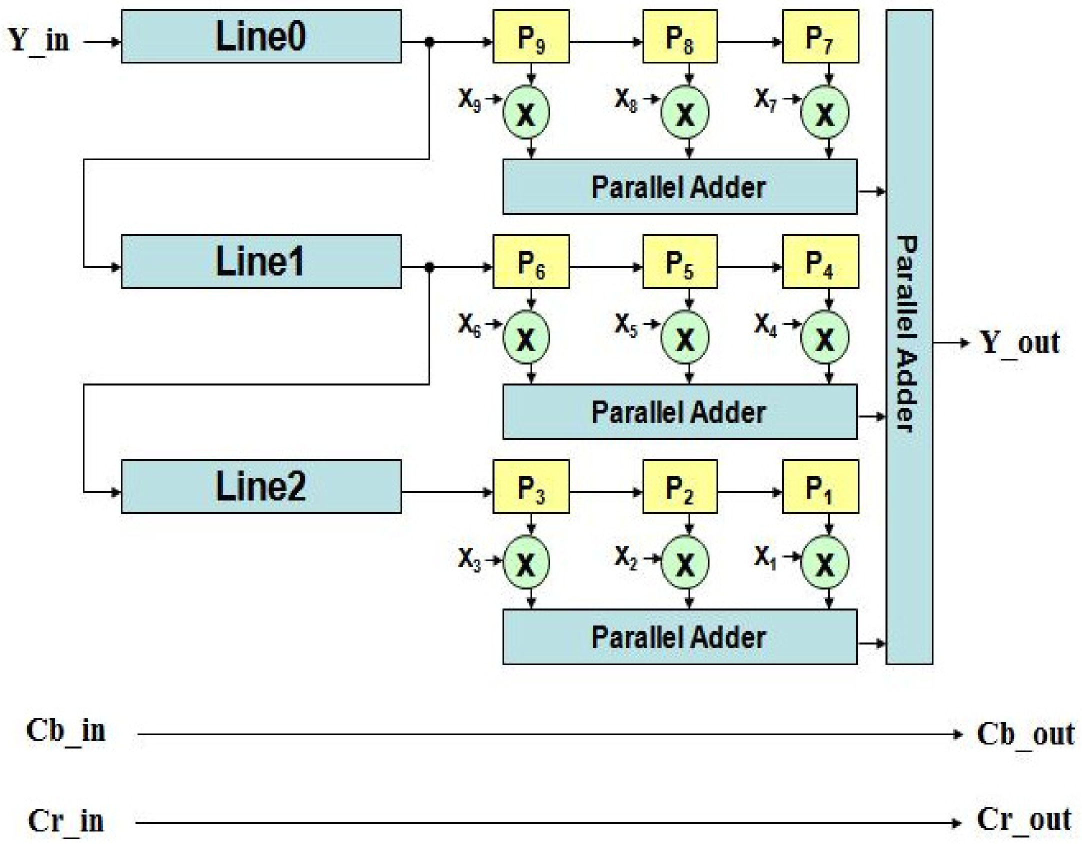


Figure 8.11 Block Diagram of Sharpness Enhancement Module

By-pass sharpness enhancement option is added in this module for enhancement ON/OFF function.

There are 2 kernels in this module which is selectible from the toggle switch on DE2-70 board. The switch configurations are below:

SW0 = 0,	Sharpness Enhancement – Normal
SW0 = 1,	Sharpness Enhancement – Soft

8.3.3 Color Enhancement Module

In this module, color correction process is managed. The goal here is to make the colors more vivid. You should be very careful while improving the color signals. Because some undesirable effects may occur accidently.

Improvement is made on 10-bit RGB signals. First, 10-bit signal is divided 10 regions and color enhancement is applied this 10 regions separately. With this method, some color blockings were seen on the screen. To overcome this problem color enhancement is made pixel by pixel. While improving green signal, extreme care should be taken, because undesired effect can appears according to incoming video data. So color improvement is applied to red and blue signals. Main goal is more vibrant red and blue colors.

There are two curves for color enhancement. They are illustrated in figure 8.12 and 8.14. One is more saturated than other.

In figure 8.12, green line is original curve. This means that output pixel value of this module is equal to input pixel of this module. In other words, this module is bypassed. Horizontal line is input of this module, vertical line is output of this module. Blue line depicts color curve in medium level for blue signal, red line depicts color curve in medium level for red signal.

In figure 8.14, blue line is original curve. This means that output pixel value of this module is equal to input pixel of this module. In other words, this module is bypassed. Horizontal line is input of this module, vertical line is output of this module. Red line depicts color curve in medium level for both blue and red signal. Red signal is more saturated in this curve from the curve before.

Color enhancement module is applied after the YCbCr_to_RGB module as in figure 8.2. The input of this module is 30-bit RGB video signal. The output signal is improved and 30-bit RGB video signal. Enhanced Red and Blue signals are synchronized with the green signal and all the signals are connected to the VGA control module.

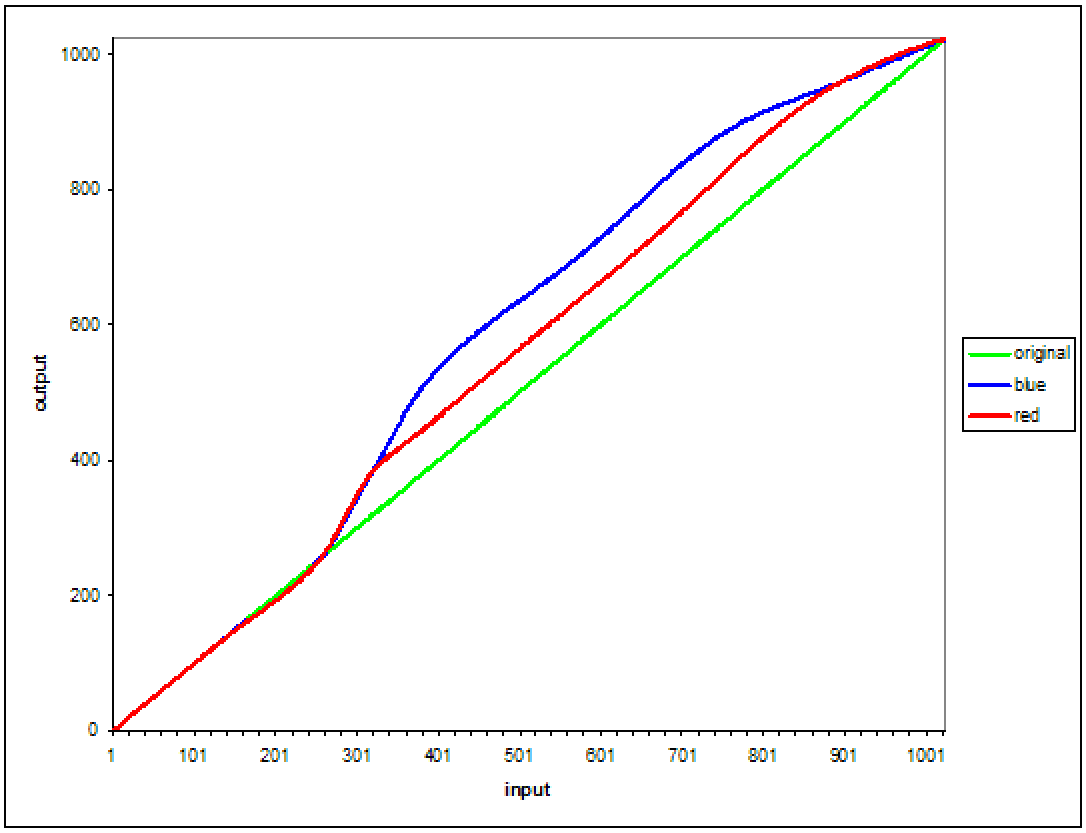


Figure 8.12 Medium Level Color Curve

Look up tables are used for FPGA implementation of contrast enhancement as in figure 8.13. 10 bit R_in, G_in and B_in signals are input signals of the contrast enhancement module. 8 bit R_out, G_out and B_out signals are output signals of the contrast enhancement module. R_out and B_out signals are selected from look up table according to R_in and B_in signals. G_in signal is connected to G_out signal directly. R and B components are modified in this module. Some normal curve of contrast enhancement look up table values are in table 8.4.

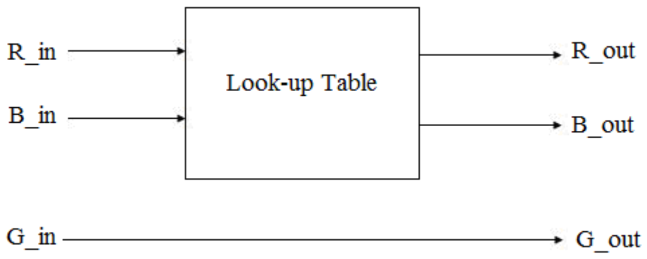


Figure 8.13 Block Diagram of Color Enhancement Module

Table 8.4 Color Enhancement (Normal Curve) Look Up Table Values

R_in	R_out	B_in	B_out
0	0	0	0
50	50	50	50
100	100	100	100
150	149	150	149
200	193	200	193
250	248	250	248
300	349	300	343
350	417	350	450
400	464	400	534
450	514	450	590
500	564	500	635
550	614	550	679
600	664	600	729
650	714	650	784
700	768	700	838
750	824	750	883
800	877	800	914
850	925	850	938
900	962	900	962
950	991	950	987
1000	1015	1000	1012
1023	1023	1023	1023

By-pass sharpness enhancement option is added in this module for enhancement ON/OFF function.

There are 2 curves in this module which is selectible from the toggle switch on DE2-70 board. The switch configurations are below:

SW1 = 0,	Color Enhancement – normal
SW1 = 1,	Color Enhancement – high

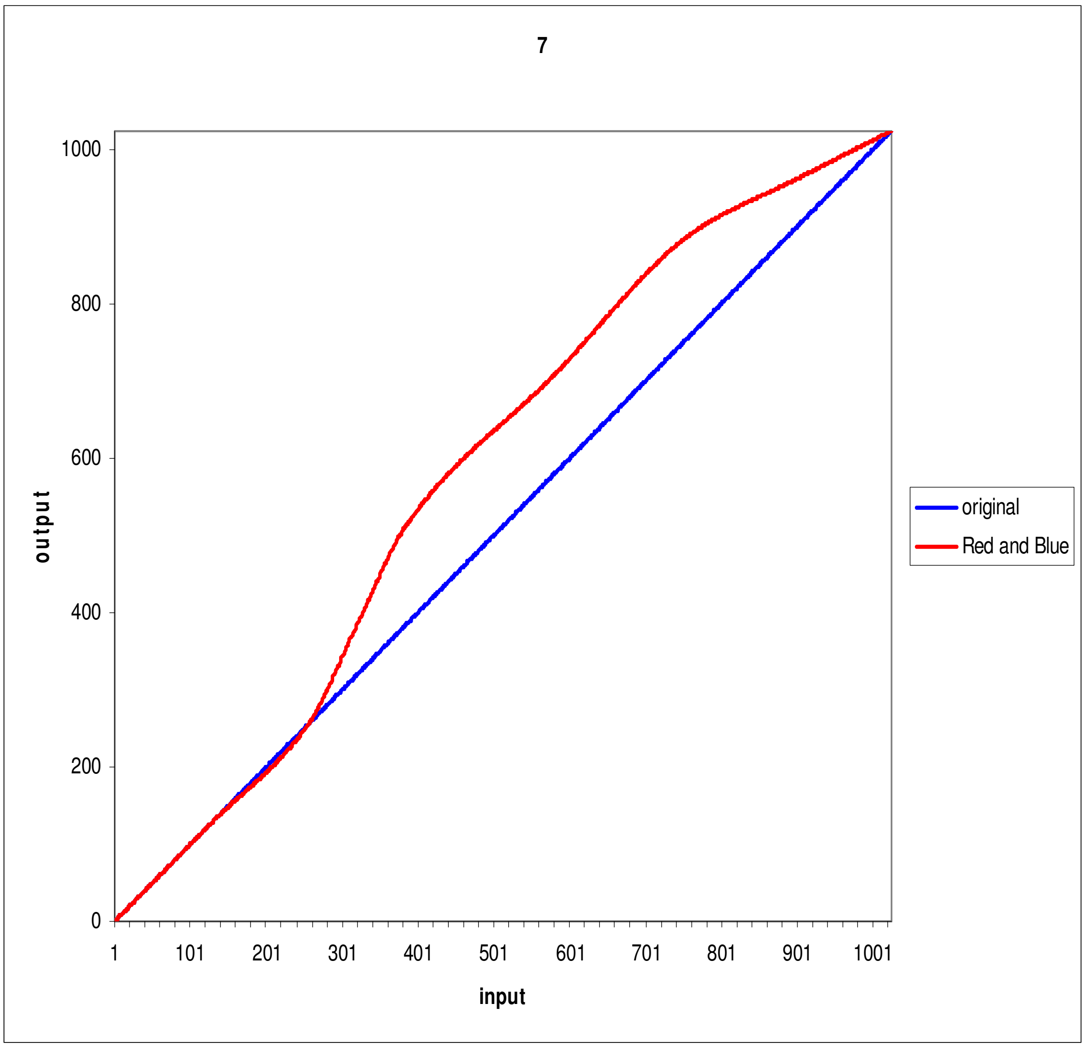


Figure 8.14 High Level Color Curve

8.3.4 Noise Reduction Module

There are many methods to reduce noise, and they are discussed in section 5.2. A linear filter (LPF) can be implemented to reduce the noise. Or more complicated but more effective non-linear filter can be implemented. Median filter is the the most famous non-linear nosie reduction filter. In this module median filter is designed for noise reduction.

This module is applied after the YUV422_to_YUV444 module as in figure 8.2. The input of this module is 24-bit YCbCr video signal. The output signal is improved and 24-bit YCbCr video signal. Noise reduction algorithm is applied only “Y” signal. “Cb” and “Cr” signals are syncronized with the improved “Y” signal.

Block diagram of this module is in figure 8.15. Block diagram is consist of 3 parts. First part finds minimum, medium and maximum values in horizontal direction, second part finds minimum, medium and maximum values in vertical direction and third part finds minimum, medium and maximum values in diagonal direction. There are 3 line buffers, 7 min/max finder module in figure 8.15. P9, P8 and P7 are the last 3 values of line buffer 0. P6, P5 and P4 are the last 3 values of line buffer 1. P3, P2 and P1 are the last 3 values of line buffer 2. Min/max finder module has 3 inputs and 3 outputs. Min/max finder module finds the minimum, medium and maximum values of its inputs.

First part of block diagram sorts values horizontally. L1_min, L1_mid and L1_max values are minimum, medium and maximum values of P9, P8 and P7. L2_min, L2_mid and L2_max values are minimum, medium and maximum values of P6, P5 and P4. L3_min, L3_mid and L3_max values are minimum, medium and maximum values of P3, P2 and P1.

Second part of block diagram sorts values vertically. Min_min, Mid_min and Max_min values are minimum, medium and maximum values of L1_min, L2_min and L3_min values. Min_mid, Mid_mid and Max_mid values are minimum, medium and maximum values of L1_mid, L2_mid and L3_mid values. Min_max, Mid_max and Max_max values are minimum, medium and maximum values of L1_max, L2_max and L3_max values.

Third part of block diagram sorts values diagonally. Y_min, Yout and Y_max values are minimum, medium and maximum values of min_max, mid_mid and max_min values. Y_out is the output of noise reduction module. It is the middle value of 3x3 input signal.

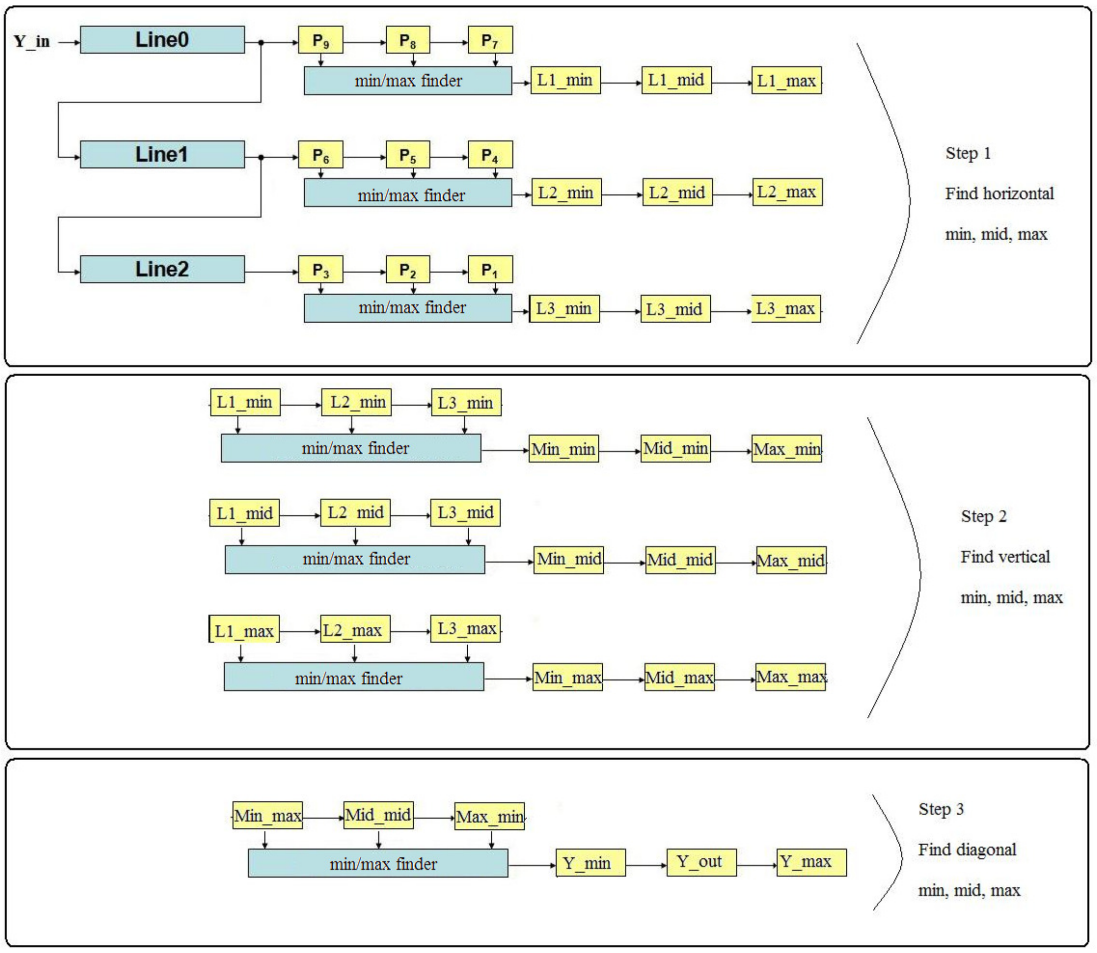


Figure 8.15 Block Diagram of Noise Reduction Module

This block should be used in noisy images. Activating this module in clear pictures reduce the affect of sharpness, so closing this block can be useful for non-noisy images. For this reason a switch is implemented for closing this module. Then user can decide to close noise reduction module.

There are 2 options in this module which is selectible from the toggle switch on DE2-70 board. The switch configurations are below:

SW15 = 0,	Noise Reduction is OFF
SW15 = 1,	Noise Reduction is ON

8.4 Results

The following figures show the results. Ther results are captured from the VGA monitor.

There are some options in this project which is selectible from the toggle switch on DE2-70 board. Toggle switch SW17 enables the enhancements (Color, Sharpness and Contrast Enhancements). If this switch is off ,other switches doesn't work except Noise Reduction. If Enhancement Mode is ON (SW17 is ON), user can change the enhancement levels via SW15, SW3, SW2, SW1 and SW0. SW16 enables the Demo Mode. If Demo Mode is active, monitor screen is divided vertically in two parts. Left hand side of screen displays the original picture, right hand side of screen displays the enhanced picture. For active Demo Mode, first Enhancement Mode should be active (SW17 is ON).

All of the option switchs which are selectible from DE270 board are below:

SW17 = 0	Enhancement OFF
SW17 = 1	Enhancement ON
SW16 = 0	Demo Mode OFF
SW16 = 1	Demo Mode ON
SW15 = 0	Noise Reduction OFF
SW15 = 1	Noise Reduction ON
SW3 = 0 & SW2 = 0	Contrast Enhancement - High
SW3 = 0 & SW2 = 1	Contrast Enhancement - Normal
SW3 = 1	Contrast Enhancement - Bright
SW1 = 0	Color Enhancement – Normal
SW1 = 1	Color Enhancement – High
SW0 = 0	Sharpness Enhancement – Normal
SW0 = 1	Sharpness Enhancement – Soft

Figure 8.16 and 8.18 are the original pictures. Figure 8.17 and 8.19 are enhanced pictures.(Color, Contrast and Sharpness Enhancements are active)

In Figure 8.20 to 8.25, Demo mode is active. When demo mode is active, left hand side of the screen is the original picture, all the algorithms are by-passed right hand side of the screen is enhanced picture. (Color, Contrast and Sharpness Enhancements are active)

Figure 8.26 to 8.29 are the results of Median Filter. Figure 8.26 and figure 8.28 are original pictures, figure 8.27 and figure 8.29 are filtered images.

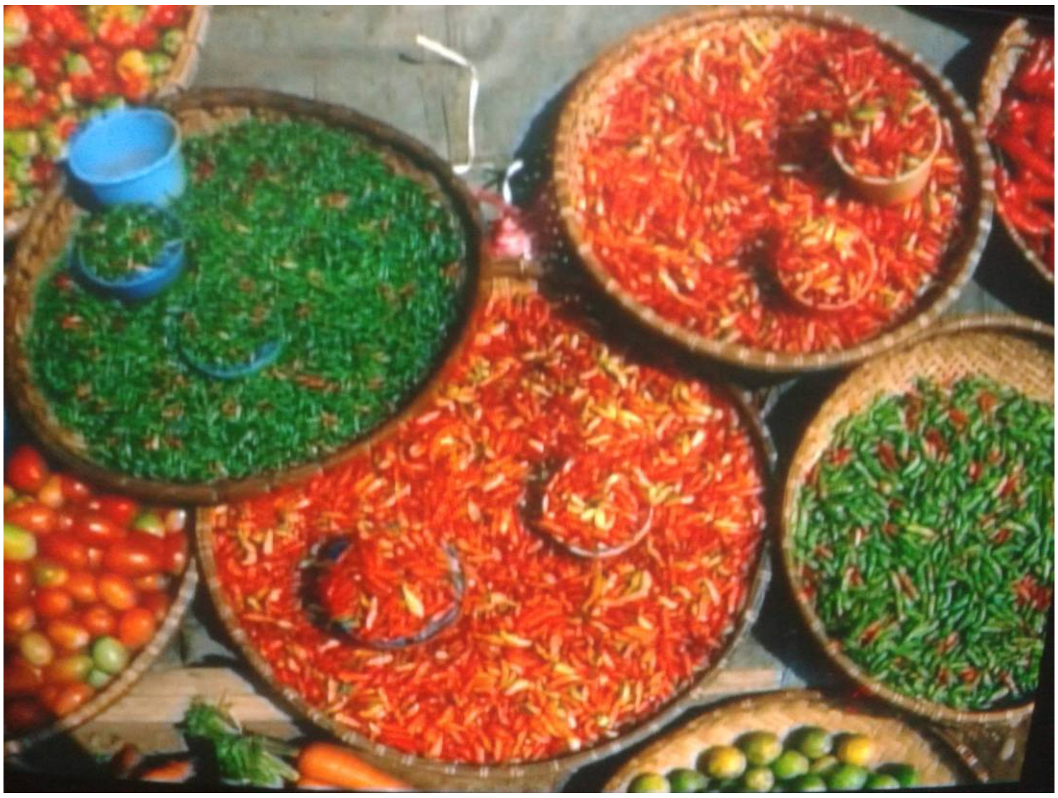


Figure 8.16 Original Picture 1

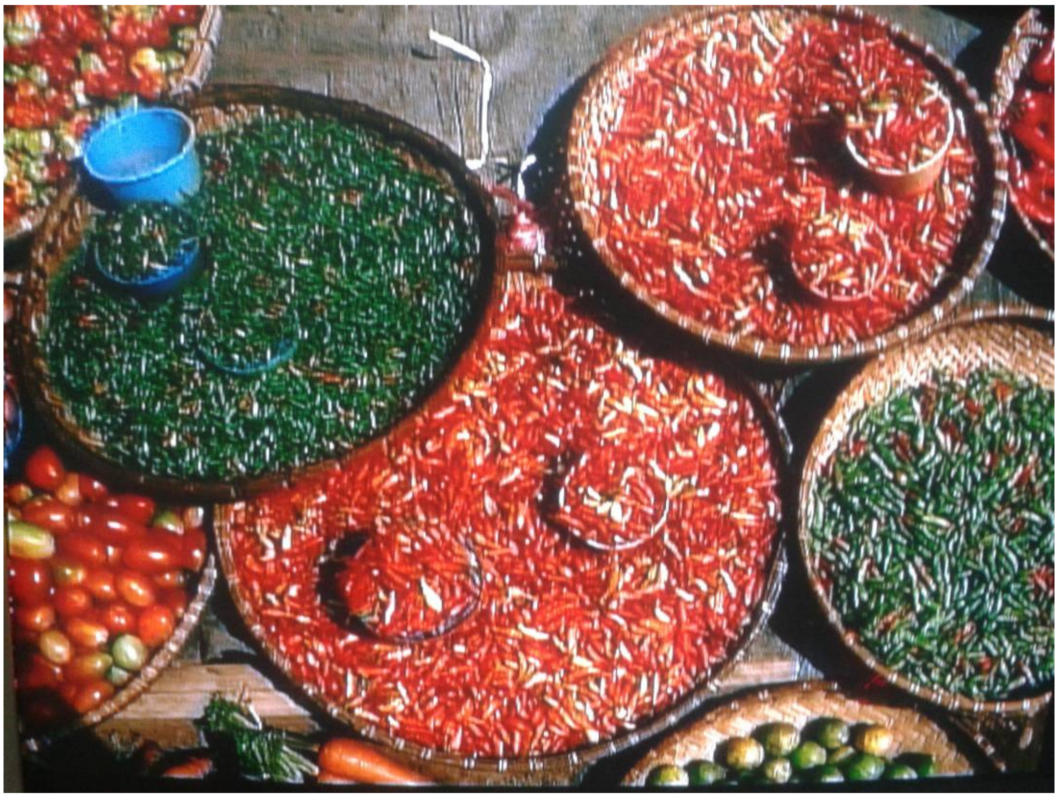


Figure 8.17 Enhanced Picture 1

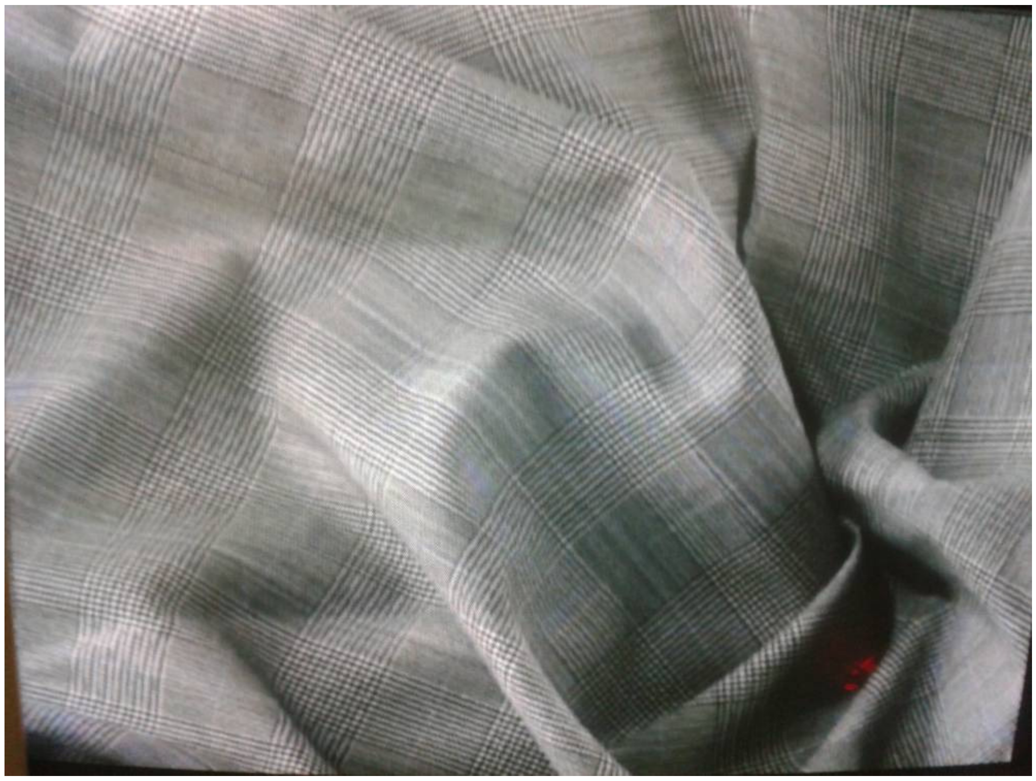


Figure 8.18 Original Picture 2

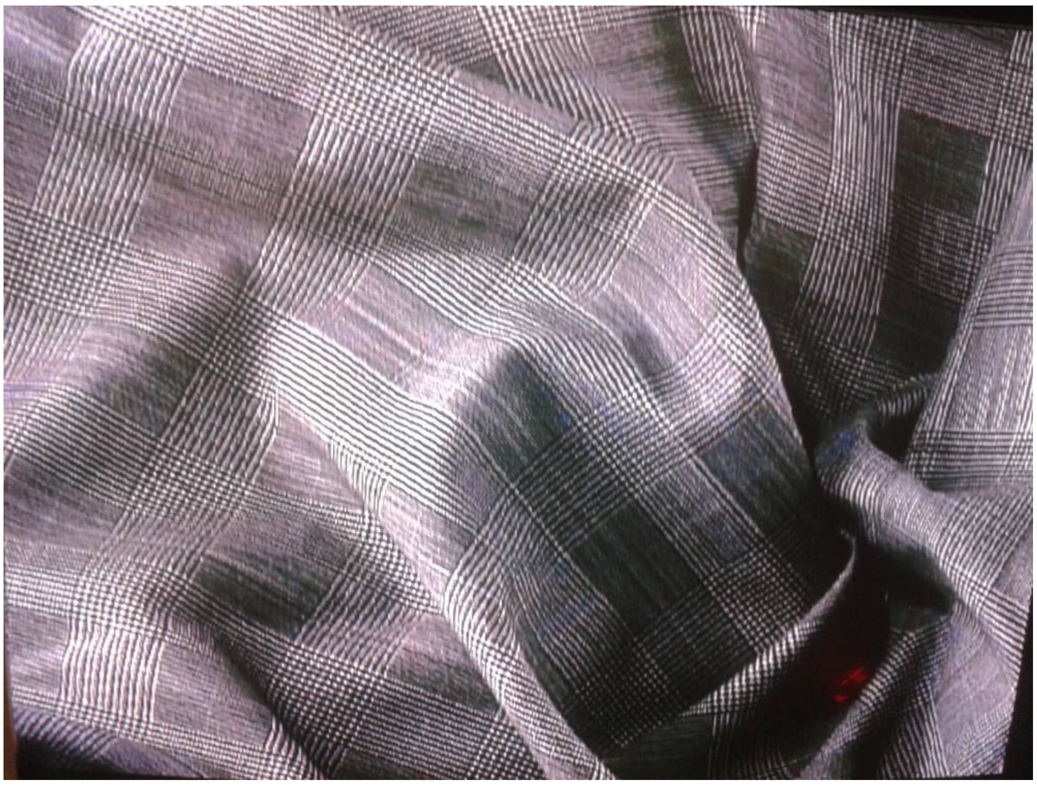


Figure 8.19 Enhanced Picture 2

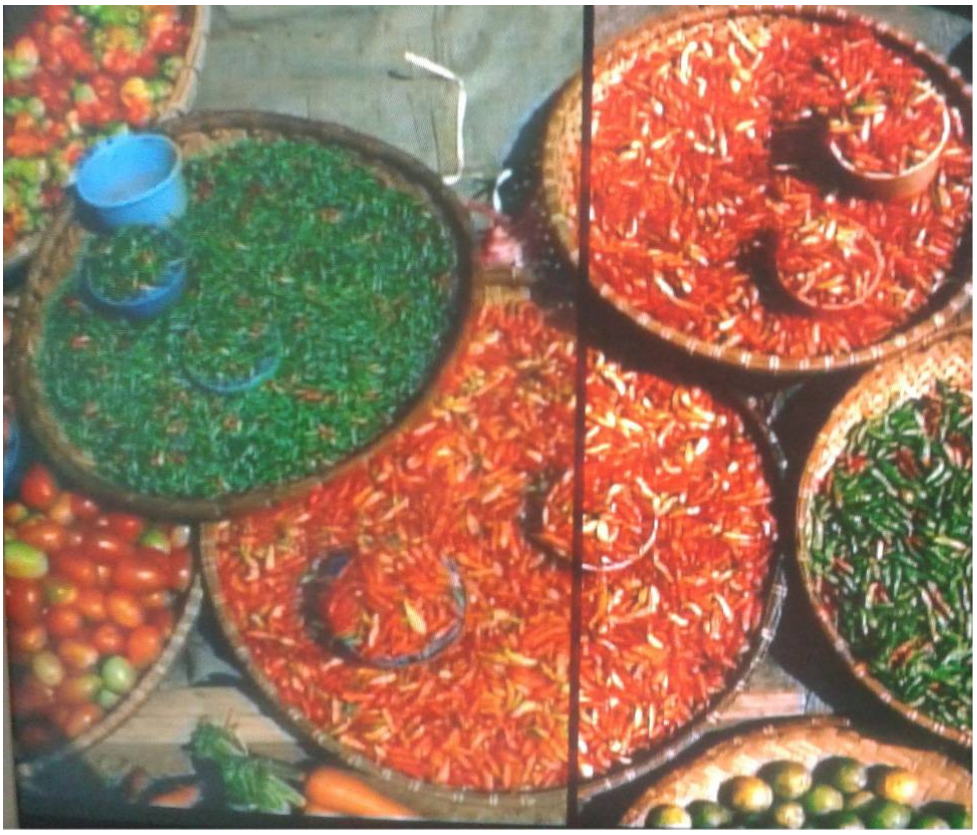


Figure 8.20 Demo Mode of Picture 2 (Left hand side is original, right hand side is enhanced.)

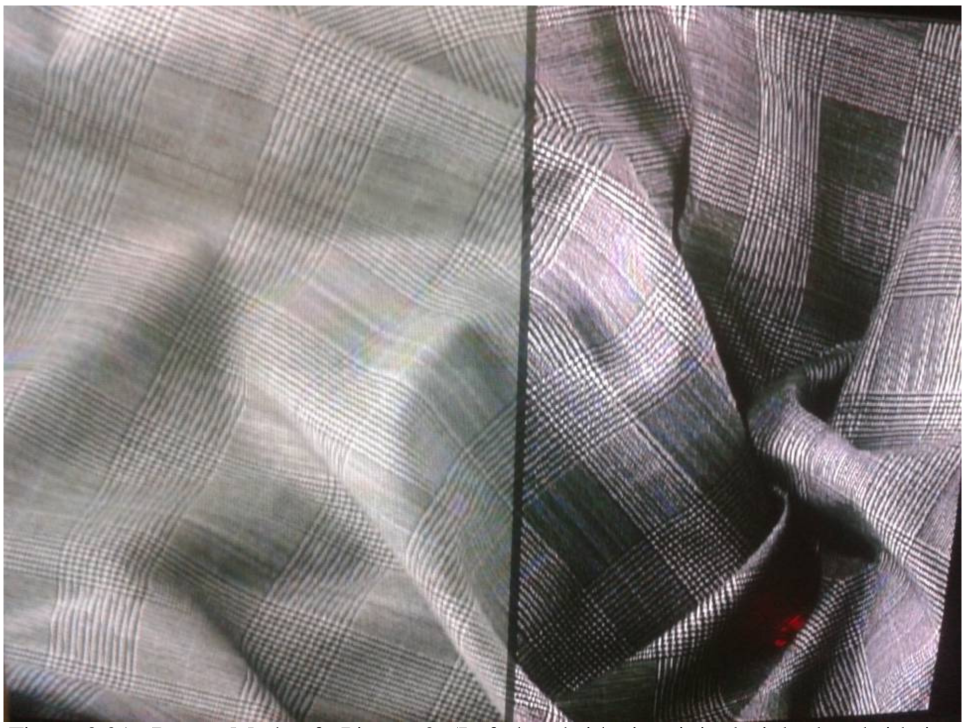


Figure 8.21 Demo Mode of Picture 2 (Left hand side is original, right hand side is enhanced.)

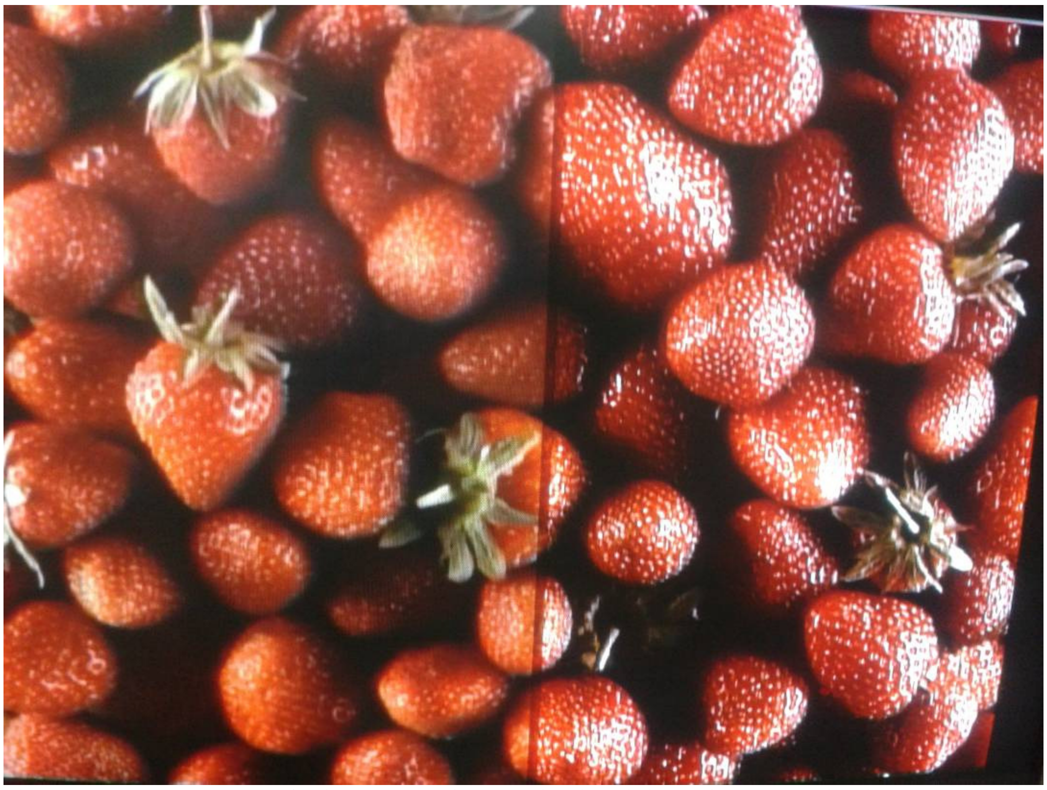


Figure 8.22 Demo Mode of Picture 3 (Left hand side is original, right hand side is enhanced.)

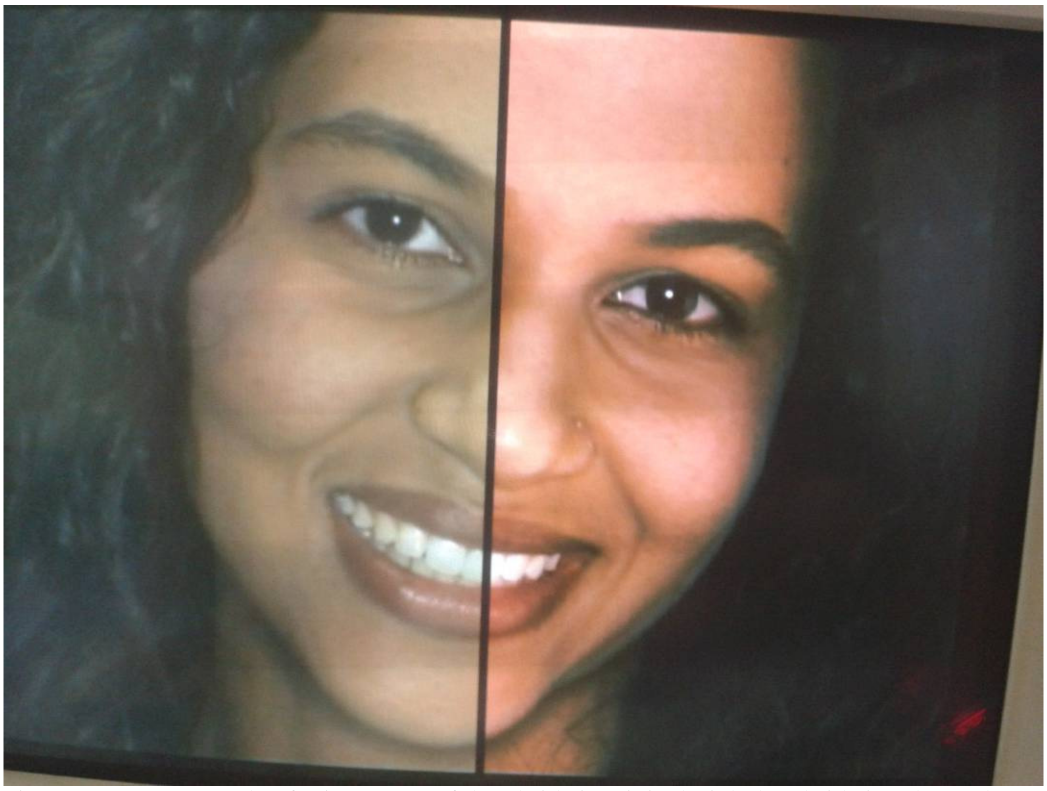


Figure 8.23 Demo Mode of Picture 4 (Left hand side is original, right hand side is enhanced.)

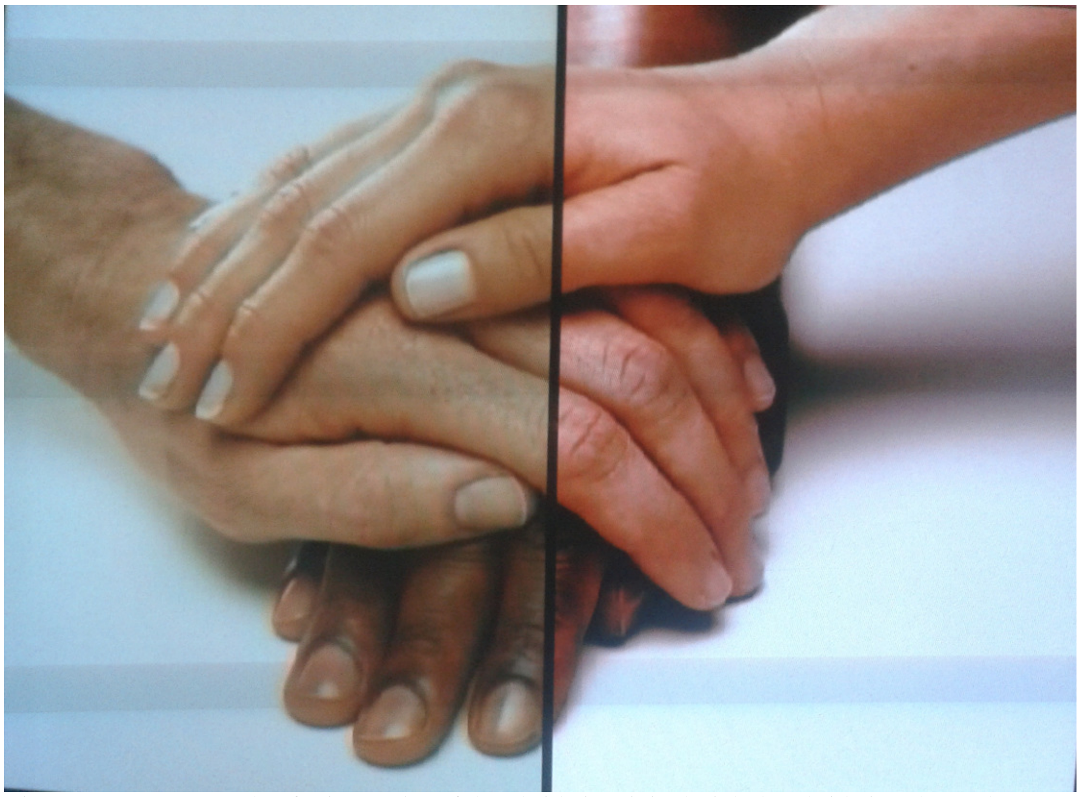


Figure 8.24 Demo Mode of Picture 5 (Left hand side is original, right hand side is enhanced.)

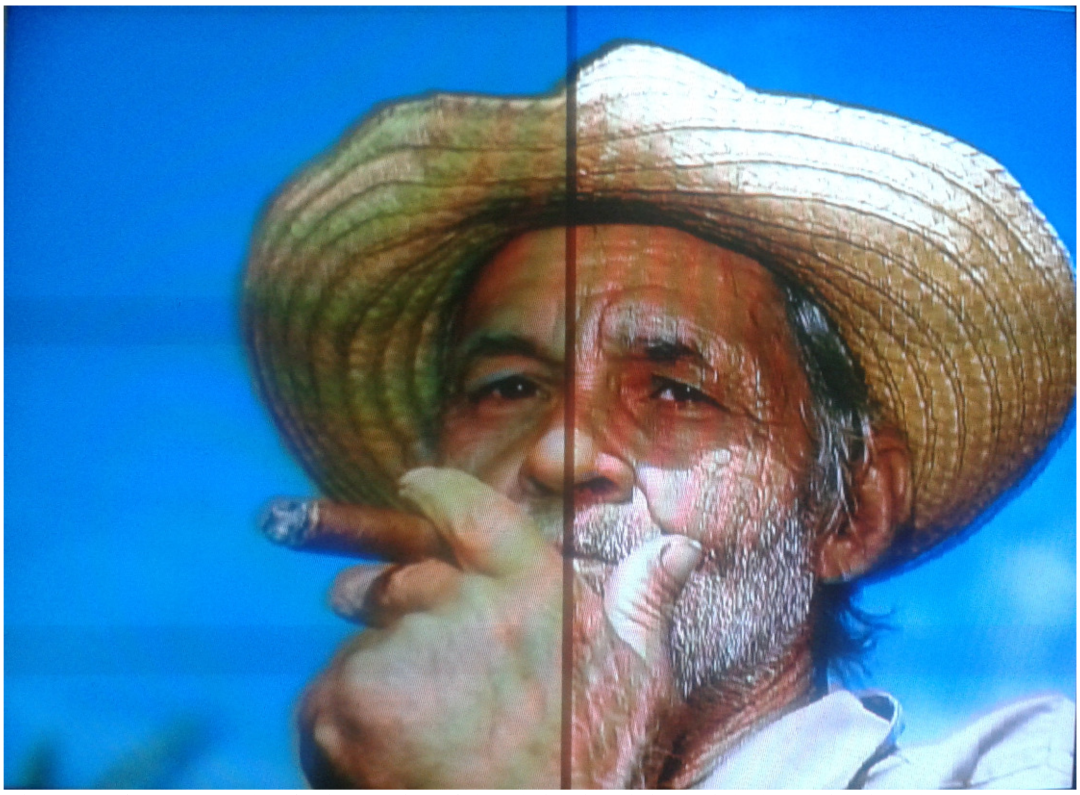


Figure 8.25 Demo Mode of Picture 6 (Left hand side is original, right hand side is enhanced.)

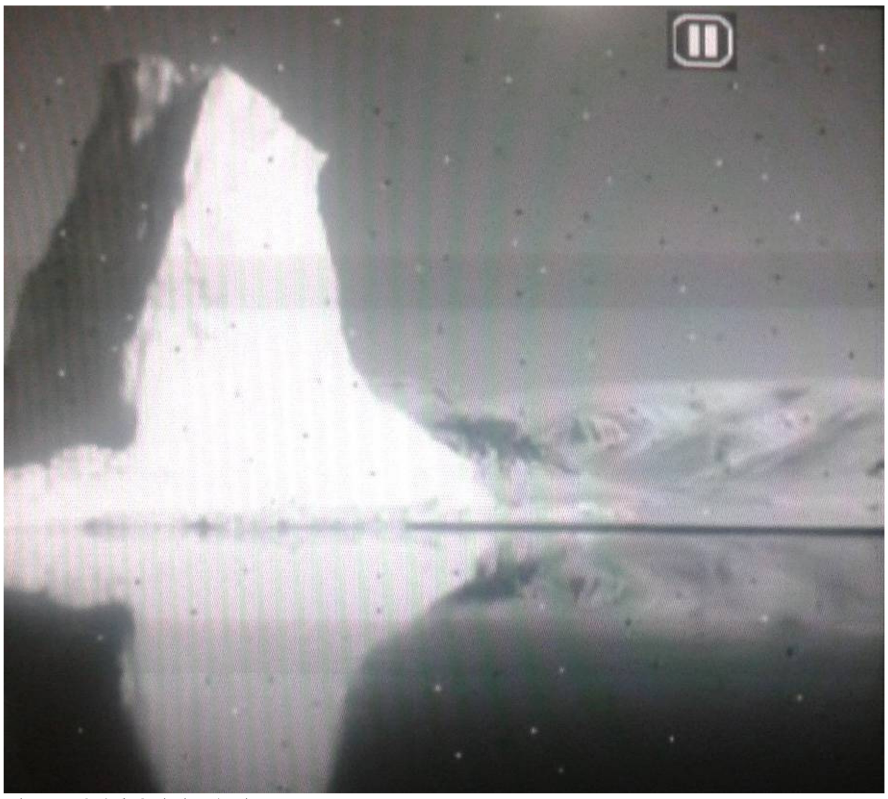


Figure 8.26 Original Picture 7

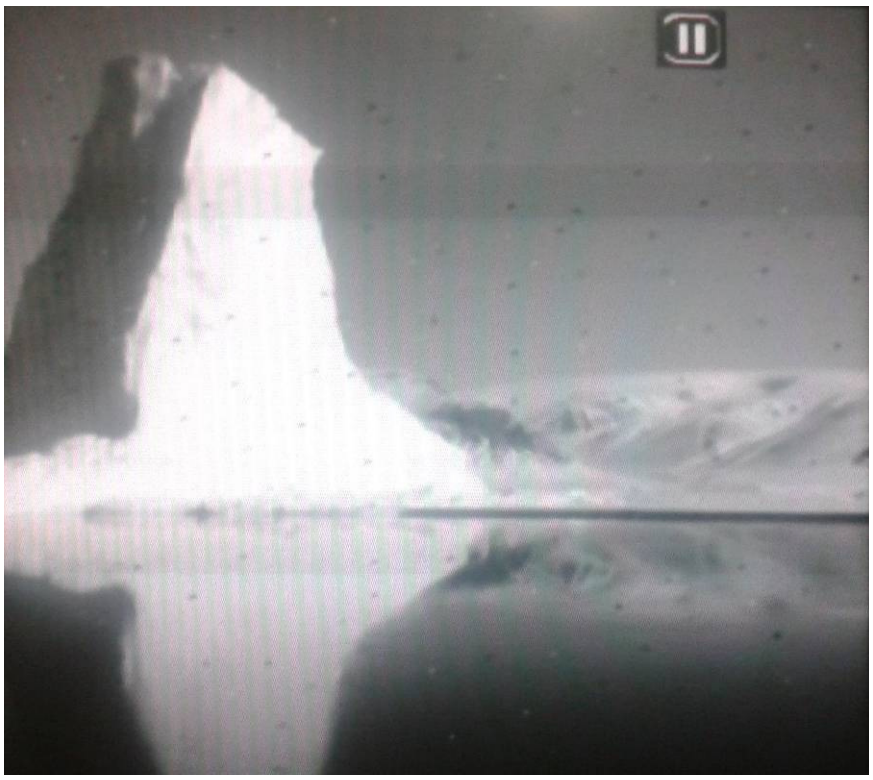


Figure 8.27 Filtered Picture 7

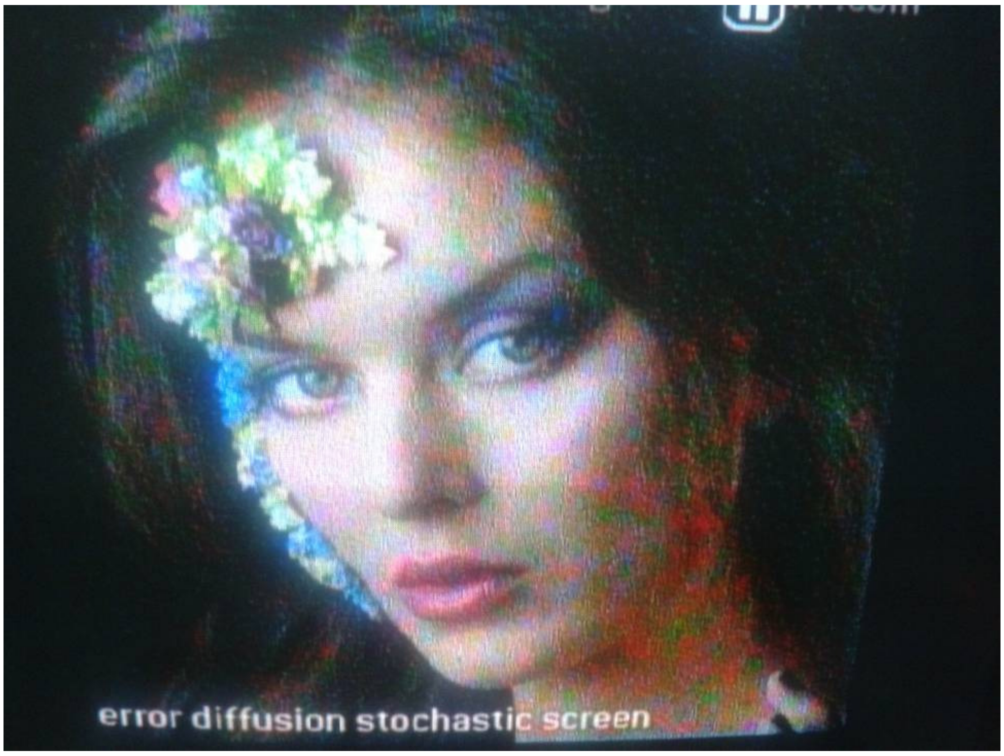


Figure 8.28 Original Picture 8,

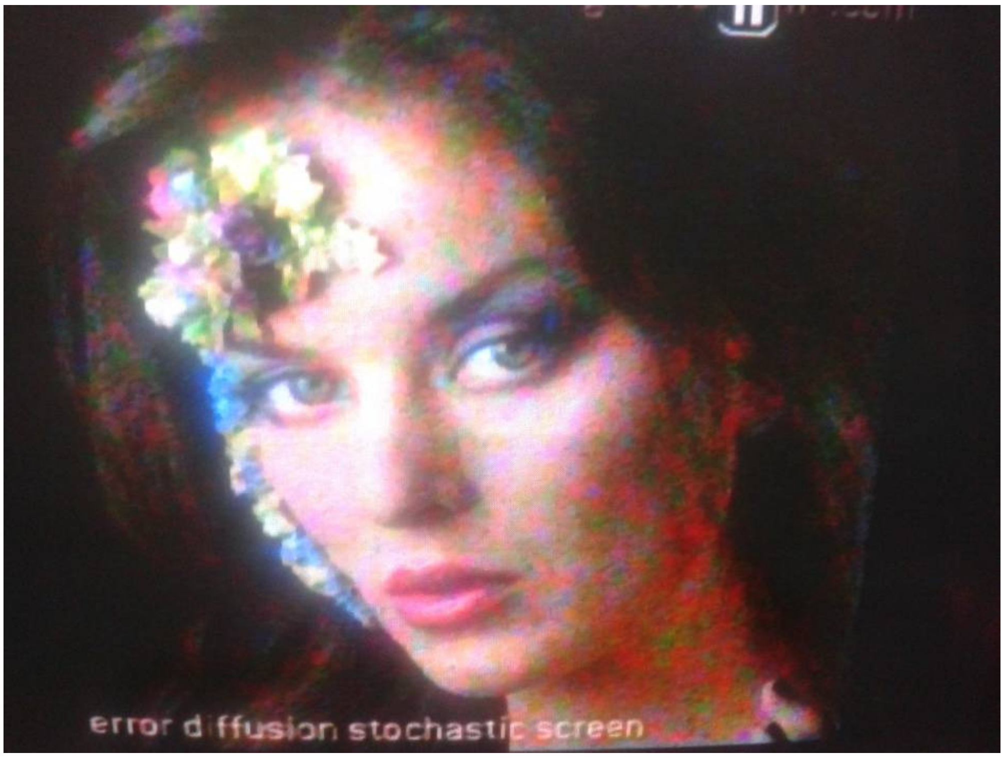


Figure 8.29 Filtered Picture 8

CHAPTER NINE
CONCLUSIONS AND FUTURE WORK

In this thesis, video processing techniques on FPGA was studied. It is aimed to design a low cost, portable and reconfigurable system. All the blocks which converts CVBS video data to VGA are designed in verilog design language, then image enhancement algorithms are implemented to the FPGA. Color, noise reduction and sharpness algorithms are tried in MATLAB and then implemented to the FPGA. Many curves and filters have been tried for the best result in improving pictures. For some image processing algorithms, digital filters are used. The reason for using finite response filter instead of infinite is the fast execution time.

The design tries to take advantage of the parallelism possible with FPGA devices. Also a great deal of knowledge was gained from the completion of this project. FPGAs are excellent for some uses, such as large number of image processing applications. The architecture of the FPGA has allowed the technology to program the design, and then reconfigure with limitless designs as the designer's needs to change. So all the algorithms can be fine tuned or designed from the beginning. Also new algorithms can be added to the system. This is the answer of why FPGA is used.

Different options is added to each image enhancement algorithms. For example, in the contrast enhancement module, there are 3 possible contrast curves which can be selected from the user. Also Sharpness Enhancement and Color enhancement blocks have different options to change the algorithms levels. Algorithms can run in aggressive or soft mode.

System development in FPGA is very economical. DE2-70 package was bought for 500 YTL from Çizgi Elektronik, İstanbul. The package included DE2-70 board and other tools like USB cable for FPGA programming and control, CD-ROMs containing Altera's Quartus® II Web Edition and the Nios® II Embedded Design Suit Evaluation Edition software. Image enhancement algorithms can be easily implemented to the DE2-70 FPGA board.

This project easily can be implemented to any TV system. Any TV manufacturer can use or improve this project to increase the image quality of the TVs.

In this thesis, only CVBS input is used. the project can be implemented to other video sources like HDMI, DVI or S-VIDEO. HDMI source can give the best result. Also this project can be modified to give HDMI or DVI output instead of VGA for a future work.

640x480 VGA out resolution is used so other resolutions like 800x600, 1366x768 etc. can be implemented to this project.

There are 4 algorithms in this thesis work. Applied algorithms can be improved. For example dynamic contrast algorithms, and dynamic noise reduction algorithms can be added to the project. Also new algorithms can be implemented, like motion detection algorithm.

REFERENCES

Acharya, T. & Ray, A.K. (2005). *Imagep processing principles and applications*. New Jersey: John Willey & Sons, Inc.

Akbal, H. Ş. (October 2005). *M.Sc. Thesis: Developing tools for the enhancement of remote sensing images and parallel programming applications.*

Al-amri, S. S., Kalyankar, N. V., & Khamitkar, S. D. (2010). Linear and non-linear contrast enhancement image. *International Journal of Computer Science and Network Security*, Vol.10 No.2.

Altera (2007). *Video and image processing design using FPGAs*. Altera Corporations

Baldock, R., & Graham, J. (Eds.). (2000). *Digital image processing* (4th ed.). Oxford: Oxford University Press.

Color balance (n.d.). Retrieved August 2011, from http://en.wikipedia.org/wiki/Color_balance.

Çallı, M. (August 2010). *M.Sc. Thesis: Development of a cost effective lvds interface test system with ethernet communication.*

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2004). *HIPR2 (Hypermedia image processing reference 2)* Retrieved August, 2011, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/freqfilt.htm

Gonzales, R.C., & Woods, R.E. (2002). *Digital image processing* (2nd ed.). New Jersey: Prentice Hall.

Günay, H. (January 2010). *M.Sc. Thesis: In partial fulfillment of the requirements for the degree of master of science in electrical and electronics engineering.*

Intersil (2002). *Application note 9728.2, BT.656 Video interface for ICs.*. Retrieved August, 2011, from http://www.intersil.com

Jack, K. (2007). *Video demystified, a handbook for the digital engineer* (5th ed.) . UK: Elsevier Inc.

Koschan, A., & Abidi, M. (2008). *Digital color image processing*. New Jersey: John Willey & Sons, Inc.

Li, J. (August 2003). *M.Sc. Thesis: A wavelet approach to edge detection.*

Maxim (2001). *Application note 734, video basics*. Retrieved August, 2011, from http://www.maxim-ic.com/an734

Phillips, D. (2000). *Image processing in C* (2nd ed.). Kansas: R & D Publications .

Pratt, W.K. (2007). *Digital image processing*. New Jersey: John Willey & Sons, Inc.

Qazi, S. (2008). *M.Sc. Thesis: Median based principal component analysis for edge detection on color images using partial derivatives of boolen functions and a new correlated color similarity measure.*

Rao, D. V., Patil S., Babu N. A., Muthukumar V. (2006). Implementation and Evaluation of Image Processing Algorithms on Reconfigurable Architecture using C based Hardware Descriptive Languages. *International Journal of Theoretical and Applied Computer Sciences, 1 (1), 9-34.*

Skogmar, K. (2003). *M.Sc. Thesis: Real-time video effects using programmable graphics cards.*

Terasic (2009). *DE2-70 User manual version 1.08*. Terasic Technologies.

Thyagarajan, K.S. (2006). *Digital image processing with application to digital cinema*. Oxford: Elsevier Inc.

Vernon, D. (1991). Machine Vision: *Automated visual inspection and robot vision*. UK: Printice Hall Internacional Ltd.