DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS ON FPGA DEMONSTRATION BOARD

by
Recep KIZILKAYA

March, 2012
İZMİR

IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS ON FPGA DEMONSTRATION BOARD

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
Recep KIZILKAYA

March, 2012
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS ON FPGA DEMONSTRATION BOARD"** completed by **RECEP KIZILKAYA** under supervision of **PROF. DR. UĞUR ÇAM** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Uğur ÇAM

Supervisor

[illegible]

Yrd.Doç.Dr. Haldun Sarnel

(Jury Member)

[illegible]

Y.Doç.Dr. ADİL ALPKOÇAK

(Jury Member)

[illegible]

Prof.Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I wish to express my deepest thanksgiving to my supervisor Prof. Dr. Uğur ÇAM who gave me the opportunity to do this research. His guidance and support encouraged me to complete this thesis study.

I would also like to thank my friends Muhammed Mustafa KAYA and Erdem ADAK because of their valuable contribution to my study.

Finally, I would like to express my special appreciation to my parents and wife because of their patience and continuous support.

Recep KIZILKAYA

IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS ON FPGA DEMONSTRATION BOARD

ABSTRACT

Image enhancement, motion and target tracking, face recognition and license plate detection are some of the image processing applications which are widely used in many fields such as military, medical sciences, astronomy etc.

Image processing is mostly implemented in software. In this thesis, implementation of image processing algorithms on FPGA was analyzed. Object and motion detection on thermal imagery, edge detection, noise reduction, histogram monitoring, nonlinear contrast adjustment and motion detection applications were successfully operated.

Altera DE2-70 development and education board was chosen to work on. Board has a number of input and output interfaces sufficient for many image-processing applications. The most important factors for choosing Altera DE2-70 board was its two composite video inputs, VGA output feature and its powerful FPGA, which facilitates to design. Verilog hardware description language (HDL) was used for modeling hardware. Quartus 9.1 program was used to synthesize the hardware models. General application includes conversion of analog video input into digital format, processing the image and transmitting the image to a VGA monitor at a resolution of 640x480.

The main objective of this study is taking advantage of the use of FPGAs in the field of image processing and demonstrating the feasibility of the efficient applications of image processing.

Keywords: Image processing, Altera DE2-70, FPGA, histogram, contrast, edge detection, object detection, motion detection, Verilog.

GÖRÜNTÜ İŞLEME ALGORITMALARININ SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ ÇALIŞMA KARTINDA GERÇEKLENMESİ

ÖZ

Görüntü iyileştirme, hareket ve hedef takibi, yüz tanıma, plaka tespiti gibi görüntü işleme uygulamaları, savunma sanayi, tıp bilimleri ve astronomi gibi birçok alanda yaygın olarak kullanılmaktadır.

Bu tezde, genelde yazılım ile uygulanan görüntü işleme algoritmalarının FPGA (sahada programlanabilir kapı dizileri) üzerinde gerçeklenebilmesi incelenmiştir. Termal görüntüde nesne ve hareket tespiti, kenar çıkarımı, gürültü azaltma, histogram görüntüleme ve doğrusal olmayan karşıtlık ayarı, ve hareketli nesne tespiti gibi uygulamalar başarıyla çalıştırılmıştır.

Çalışma kartı olarak Altera DE2-70 kartı seçilmiştir. Kart birçok uygulama için yeterli arayüze sahip bulunmaktadır. Kompozit video girişleri ve VGA çıkışının mevcut bulunması ve yeterince güçlü bir FPGA'a sahip olması tasarım yapmayı kolaylaştırması açısından seçilmesindeki etkenlerin başında gelmektedir. Donanım modellemesi için Verilog donanım tanımlama dili kullanılmıştır. Donanım modellerini sentezleme programı olarak Quartus 9.1 kullanılmıştır. Genel uygulama kompozit analog video girişini dijital formata çevirme, görüntünün işlenmesi ve bilgisayar monitörüne 640 x 480 çözünürlükte gönderilmesi işlemlerinden oluşur.

Bu çalışmanın temel amacı görüntü işleme alanında FPGA kullanımının avantajlarından faydalanarak, görüntü işlemenin verimli bir şekilde uygulanabilirliğini göstermektir

Anahtar Sözcükler: Görüntü işleme, Altera DE2-70, FPGA, histogram, karşıtlık, kenar çıkarımı, nesne tespiti, hareket tespiti, Verilog.

CONTENTS

CHAPTER ONE
INTRODUCTION

Image processing improves some features of an image, eliminates undesired portions and extracts information in accordance with the objective. Therefore processing is applied to the data if one wishes to obtain enhanced images or to collect useful information from an image.

Usually most of image processing algorithms are implemented on general-purpose processors. DSPs and ASICs can also be used. In this thesis, FPGA is proposed to implement image processing applications.

Because of the fast evolution of digital image technologies, the architecture should have characteristics like high processing performance, flexibility and low development cost. FPGAs can provide all of these so they can meet the requirements of many video and image processing applications.

An FPGA is an integrated circuit, which is used to implement any digital circuit. It contains thousands of logic blocks, which are connected internally. They are re-configurable and can be programmed by the user using hardware description languages such as VERILOG and VHDL. Due to the fact that it provides flexibility to the user during the design, use of FPGAs become increasingly common in recent days.

In this study, in order to realize image processing on images and videos, ALTERA DE2-70 development and education board was chosen. CANON SX200 camera was used as analog video source. Any VGA monitor compatible to 640x480 resolution is appropriate to analyze the effects of applied image processing algorithms.

1.1 Outline

This thesis is presented in 7 chapters.

In chapter two, brief literature review was given about studies relating to FPGA implementation of image processing applications.

Chapter three introduces fundamentals of digital image processing. Digital image and algorithms covered in applications realized in this study are described.

In chapter four, analog and digital video standards are explained. Input video, processed video and output video formats are introduced.

Chapter five gives information about FPGAs and discusses the advantages of FPGAs over other architectures on image processing. FPGA development board and general description of the system hardware are explained.

In chapter six, developed algorithms are explained in detail. Applications were tested with still images and motion images. Results of implemented image processing algorithms are evaluated.

Finally, chapter seven includes conclusion and future work.

CHAPTER TWO
LITERATURE SURVEY

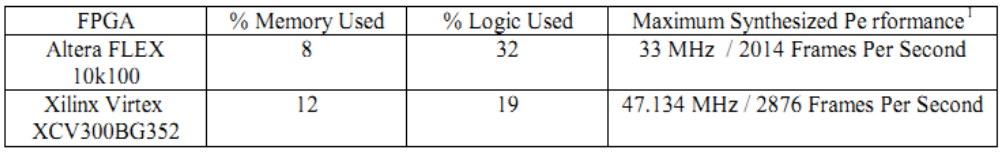
2.1 Implementation of Image Processing Algorithms on FPGA Hardware

The goal of this thesis is to develop FPGA realizations of three popular image processing algorithms on two FPGA architectures: the Altera FLEX 10K100 and the Xilinx Virtex 300. In the study median filter, morphological image processing and convolution algorithms are implemented on FPGA hardwares and on PC software program MATLAB. MATLAB was used to develop an initial version of the three image processing algorithms so that its operation could be verified and its results could be compared to the hardware version (Nelson, 2000).

Since the Xilinx Virtex is a newer generation FPGA, it was expected that it would provide superior performance over the Altera FLEX 10K FPGA. This surmise was true, and was a constant throughout the design (Nelson, 2000).

Table 2.1 (Nelson, 2000) shows the rank order filter synthesis results for the two architectures.

Table 2.1 Performance and resources for rank order filter.



FPGA	% Memory Used	% Logic Used	Maximum Synthesized Performance[1]
Altera FLEX 10k100	8	32	33 MHz / 2014 Frames Per Second
Xilinx Virtex XCV300BG352	12	19	47.134 MHz / 2876 Frames Per Second

2.2 Implementation and Evaluation of Image Processing Algorithms on Reconfigurable Architecture Using C-based Hardware Descriptive Languages

Most of the system level hardware programming languages introduced and commonly used in the industry are highly hardware specific and requires intermediate to advance hardware knowledge to design and implement the system. In order to overcome this bottleneck various C-based hardware descriptive languages have been proposed over the past decade. These languages have greatly simplified the task of designing and verifying hardware implementation of the system (Rao, Patil, Babu, & Muthukumar, 2006).

Handel-C is a new C-based language proposed that provides direct implementation of hardware from the C-based language description of the system. Handel-C language and the IDE tool introduced by Celoxica Ltd. provide both simulation and synthesis capabilities. This work evaluates the performance and efficiency of Handel-C language on image processing algorithms and is compared at simulation level with another popular C-based system level language called SystemC and at synthesis level with the industry standard Hardware Descriptive language, Verilog (Rao, et al., 2006).

The image processing algorithms considered include, image filtering, image smoothing and edge detection. Comparison parameters at simulation level include, man-hours for implementation, compile time and lines of code. Comparison parameters at synthesis level include logic resources required, maximum frequency of operation and execution time. Our evaluations show that the Handel-C implementation perform better at system and synthesis level compared to other languages considered in this work (Rao, et al., 2006). Table 2.2 (Rao, et al., 2006) and Table 2.3 (Rao, et al., 2006) summarize some results of the study.

Table 2.2 Experimental results with time and speed comparisons for Canny edge detection.

	Handel-C	PC @ 1300MHz	SA-C	PC @ 800MHz	PC with MMX @ 800MHz
FPGA CLK	16 MHz	-	32.2 MHz	-	-
Execution Time	4.2 ms	47 ms	6 ms	850 ms	135.8 ms

Table 2.3 Design efforts for Canny Edge Detection using Verilog, SystemC and Handel-C.

	VERILOG	SystemC	HANDEL-C
Design Time	56 hrs	112 hrs	56 hrs
Compilation Time	54 Sec	5 mins	50 Sec
Program Size (lines)	1318	3239	1233

2.3 FPGA Based Image Edge Detection and Segmentation

The proposed work presents FPGA based architecture for Edge Detection using Sobel operator and uses Histogram method for Segmentation. Currently the image processing algorithms has been limited to software implementation which is slower due to the limited processor speed. So a dedicated processor for edge detection and segmentation is required which was not possible until advancement in VLSI technology. Now more complex system can be integrated on a single chip providing a platform to process real time algorithms on hardware. Sobel operator is chosen due to its property of less deterioration in high levels of noise. The conventional histogram method is modified to adopt for automatically determining the threshold for different regions in the image (Chikkali & Prabhushetty, 2011).

2.4 Efficient FPGA Implementation of Image Enhancement Using Video Streams

This thesis is composed of three main parts; displaying an analog composite video input by via converting to digital VGA format, license plate localization on a video image and image enhancement on FPGA (Gunay, 2010).

When taking timing efficiency into account, image enhancement is applied only to beneficial part of the image. In this thesis work, beneficial part of the image is considered as numbered plates. Before image enhancement process, the location of the plate on the image must be found. In order to find the location of plate, a successful method, edge finding is used. It is based on the idea that the plate is found on the rows, where the brightness variation is largest (Gunay, 2010).

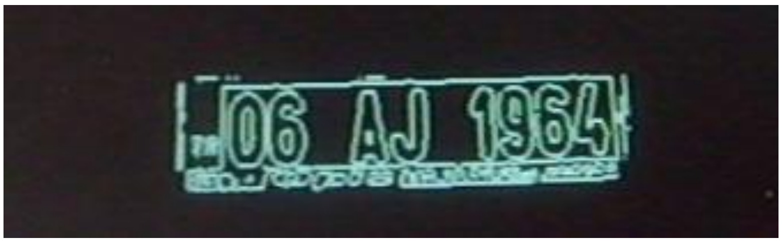


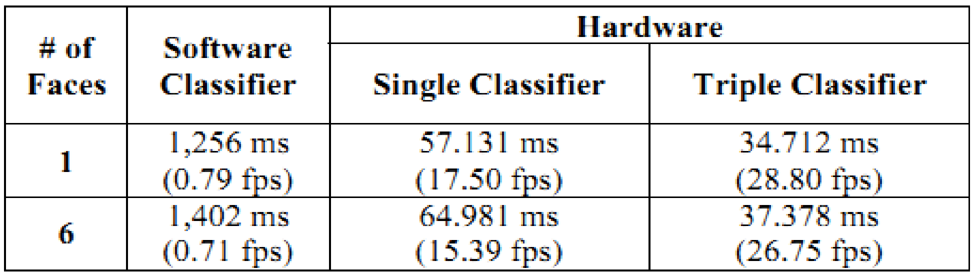
Figure 2.1 Plate localization algorithm result.

Image enhancement with rank order filter method is chosen to remove the noise on the image. Median filter, a rank order filter, is designed and simulated. To improve image quality while reducing the process time, the filter is applied only to the part of the image where the plate is. Figure 2.1 shows the plate localization algorithm result designed in FPGA (Gunay, 2010).

2.5 FPGA-Based Face Detection System Using Haar Classifiers

This paper presents hardware architecture for face detection based system on AdaBoost algorithm using Haar features. The proposed architecture for face detection has been designed using Verilog HDL and implemented in Xilinx Virtex-5 FPGA. Its performance has been measured and compared with an equivalent software implementation. Table 2.4 shows the performance of the implemented face detection system when it is applied to a camera, which produces images consisting of 320×240 pixels at 60 frames per second (Cho, Mirzaei, Oberg, & Kastner, 2009).

Table 2.4 Performance of proposed face detection system with 320×240 resolution images.



# of Faces	Software Classifier	Hardware	
		Single Classifier	Triple Classifier
1	1,256 ms (0.79 fps)	57.131 ms (17.50 fps)	34.712 ms (28.80 fps)
6	1,402 ms (0.71 fps)	64.981 ms (15.39 fps)	37.378 ms (26.75 fps)

In Figure 2.2 the experimental result of the proposed face detection system can be seen. The white squares present the detected face on the images. It shows that the face can be detected successfully (Cho, et al., 2009).

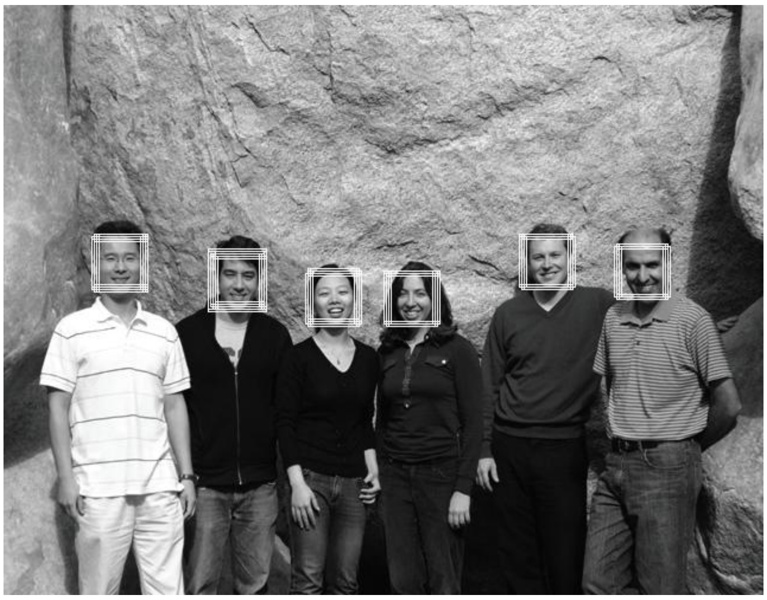


Figure 2.2 Experimental result of face detection system.

CHAPTER THREE
DIGITAL IMAGE PROCESSING BASICS

An image may be defined as a two-dimensional function f(x,y), where x and y are spatial (plane) coordinates and the amplitude of f at any pair of coordinates (x,y) is called the intensity or gray level of the image at that point. When x,y, and the amplitude values of f are all finite, discrete quantities we call the image a digital image (Gonzalez, & Woods, 2002).

Digital image processing is analysis and manipulation of images. Some objectives of image processing are to create an enhanced image, to distinguish objects, patterns and to measure some useful information in the image. Digital image processing is being used in variety of applications and there is almost no area that is not affected from image processing. Those areas are summarized in Figure 3.1

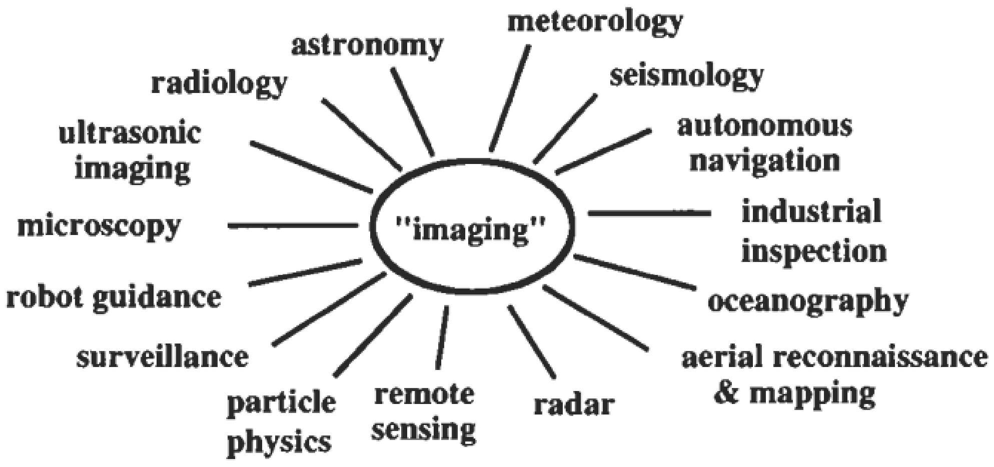


Figure 3.1 The universe of image processing applications (Bovik, 2000).

In order to process a digital image various operations are applied. There are three operations that are fundamental to digital image processing which are described in following sections. These are histogram, convolution and morphology based operations.

Operations can also be classified into three categories in terms of their implementation. These are point, local and global operations.

In point operations output value of a specific pixel just depends on the input value of its own original value. Thresholding, brightness and contrast applications are some examples for point operations. On the other hand, local operations depend on both input value of its own original value and its various shape local neighborhoods. Convolution and median filter are based on local operations. If the output value of a specific pixel depends on all the values in the input image, it is called global operation. Image compression is one of the global operation applications.

3.1 Histogram Based Operations

An 8-bit grayscale image is represented with values ranging from 0 to 255. Histogram shows the distribution of pixels to gray levels in an image. If histogram range is narrow and shows just dark, bright or a portion of gray levels, image has low contrast. Due to the low contrast all the details outside the range are lost. On the contrary, if histogram range is wide and displays most of the levels, then it is high contrast image and details are obvious.

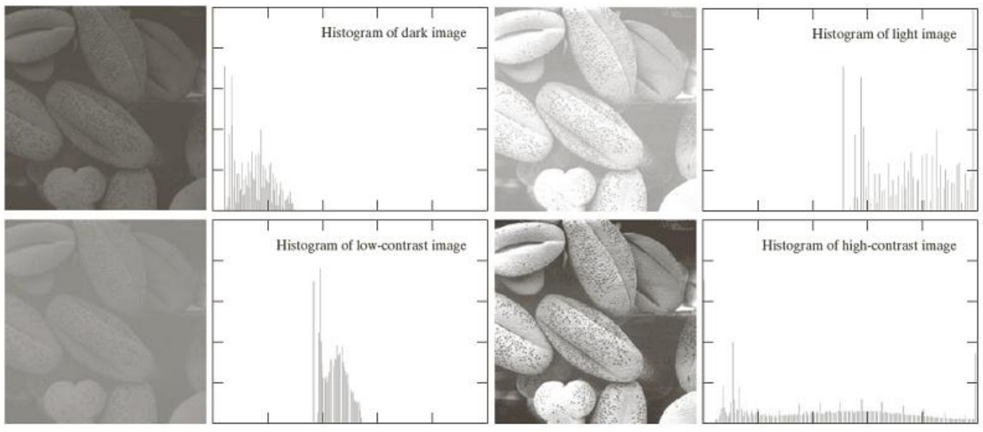


Figure 3.2 High and low contrast histograms (Gonzalez, & Woods, 2002).

High contrast and low contrast histogram examples can be seen in Figure 3.2. As understood from the example, contrast enhancement algorithms have to aim to obtain a high contrast image histogram in order to show image in detail.

3.1.1 Nonlinear Curve Adjustments

Most of images don’t use all range of gray levels. If brightness level intensity of image is not evenly distributed, this can be observed by examining the histogram. If some modification on the brightness of the image is required, nonlinear curves can be used.

The simplest form of a nonlinear curve is accomplished by moving a mid-tone location toward the upper left or lower right corner, forming a basic arc with a single bend. Also note that one end of your tones will take on more contrast while the other end will lose contrast due to the change in slope of the curve. This can be used to brighten or darken the overall image if you want to maintain your highlights and shadows at their current values (Auer, 2010).

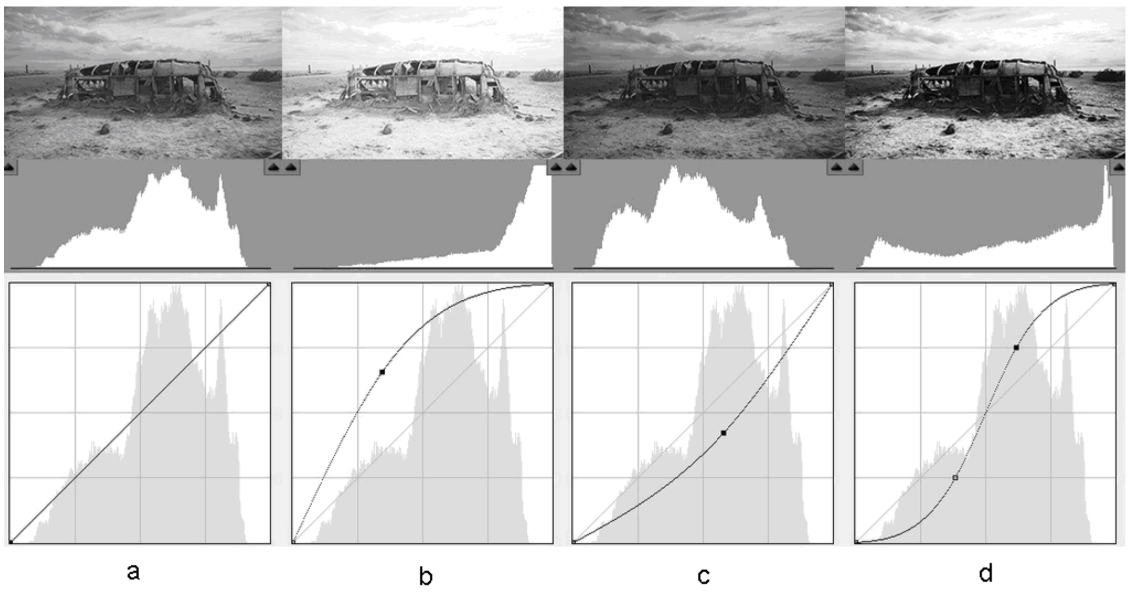


Figure 3.3 Effects of single and double bend curves on a picture (Auer, 2010).

Also known as the “S-Curve”, this curve manipulation pushes one section of tones brighter and another section of tones darker. This can be used to raise or lower the contrast of the overall image with a focus on the mid-tone areas. The bright/dark tone changes of the highlights/shadows are amplified by the mid-tone slope change so it doesn’t take much to really change the contrast. (Auer, 2010)

3.2 Convolution Based Operations

Most of image processing filters use convolution matrix. Convolution is the treatment of a matrix by another one which is called kernel (Wernicke, et al., 2010). The discrete convolution can be defined as a `shift and multiply' operation, where we shift the kernel over the image and multiply its value with the corresponding pixel values of the image. For a square kernel with size $M \times M$, we can calculate the output image with the equation (Fisher, Perkins, Walker and Wolfart, 1996) below,

$$g(i,j) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n-=\frac{M}{2}}^{\frac{M}{2}} h(m,n)\, f(i-m, j-n) \tag{3.1}$$

Kernels vary depending on the applications. Matrix size and matrix values can change. In Figure 3.4 there is an example of a convolution. The value of red pixel is altered by convolution operation.

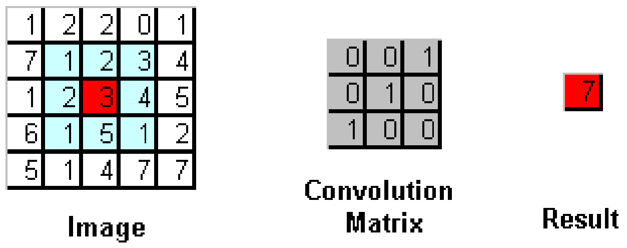


Figure 3.4 Convolution operation sample.

3.2.1 Median Filter

Median filter is a sliding window over an image and the center pixel value replaces with the median of the pixels in the window. Window can be 3x3 square or any shape. In median filtering center and neighboring pixels in the window are sorted and middle value replaces with the center pixel. The example is shown in Figure 3.5.

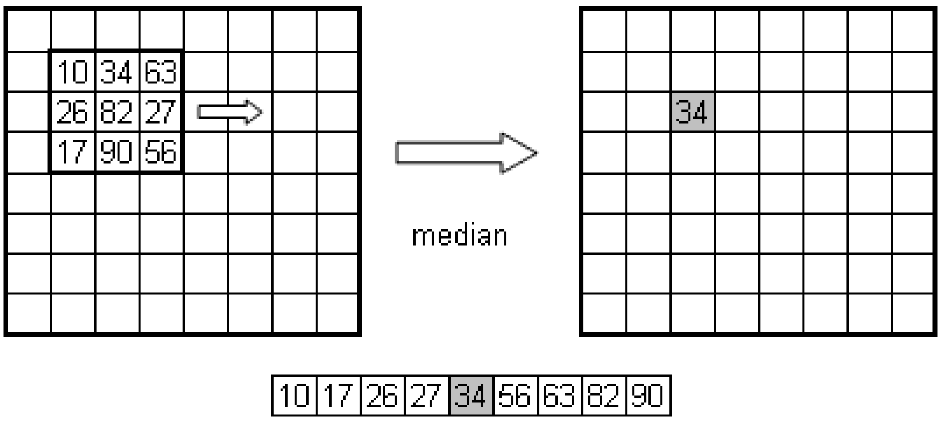


Figure 3.5 Median filter example.

Application of median filter to images is a successful noise removal technique. During noise reduction with other digital filters sharp edges cannot be protected. The advantage of median filter over other filters is that median filter preserves sharp edges during noise removal.

3.2.2 Sobel Edge Detector

Sobel edge detector uses two convolution masks to emphasize high frequency areas and calculates the gradient of image intensity. Convolution masks are shown in Figure 3.6. Masks can be applied separately to detect edges in X and Y coordinates. These values can be combined to find the magnitude and angle of orientation of gradient. In the equation below |g| is the gradient magnitude and Q is the gradient direction. If g_x and g_y values are high, also edge strength will be high. These high values show the presence of edges.

$$g_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad g_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Figure 3.6 Sobel convolution masks.

$$|g| = \sqrt{{g_x}^2 + {g_y}^2}$$
$$Q = \arctan(g_y / g_x) \tag{3.2}$$

Although the Sobel operator is slower to compute, its larger convolution kernel smoothes the input image to a greater extent and so makes the operator less sensitive to noise. The larger the width of the mask, the lower its sensitivity to noise and the operator also produces considerably higher output values for similar edges (Vincent, & Folorunso, 2009). Also because of its smoothing effect, it produces thicker edges than original edges.

3.3 Morphology Based Operations

An image can be represented by a set of pixels. Morphological operators work with two images. The image being processed is referred to as the active image, and the other image, being a kernel, is referred to as the structuring element. Each structuring element has a designed shape, which can be thought of as a probe or a filter of the active image. The active image can be modified by probing it with the structuring elements of different sizes and shapes. The elementary operations in mathematical morphology are dilation and erosion, which can be combined in sequence to produce other operations, such as opening and closing (Shih, 2009).

3.3.1 Erosion and Dilation

Erosion is typically applied to binary images but there are versions that work on grayscale images. The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels. (i.e. white pixels, typically). Thus areas of foreground pixels shrink in size, and holes within those areas become larger. If for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value as shown in Figure 3.7 (Fisher, et al., 1996).

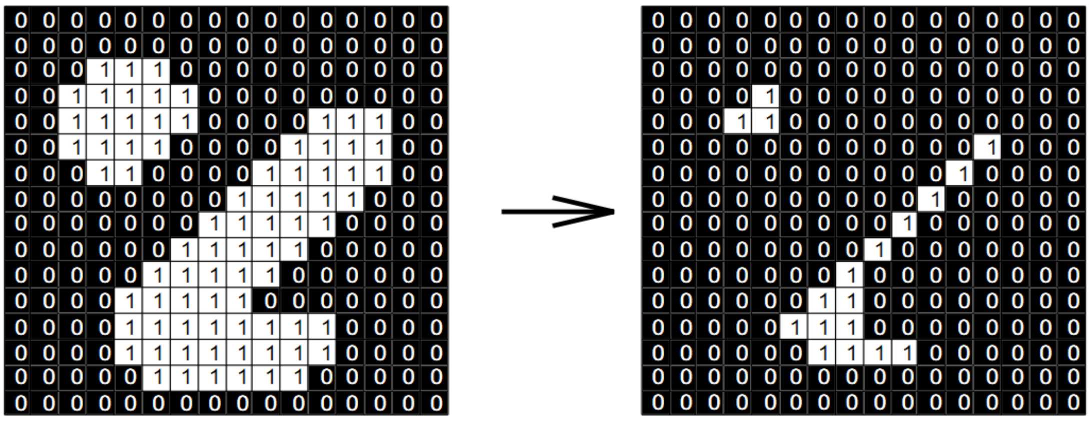


Figure 3.7 Erosion operation.

The basic effect of the dilation operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (i.e. white pixels, typically). Thus, areas of foreground pixels grow in size while holes within those regions become smaller. If at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value. If all the corresponding pixels in the image are background, however, the input pixel is left at the background value as shown in Figure 3.8 (Fisher, et al., 1996).

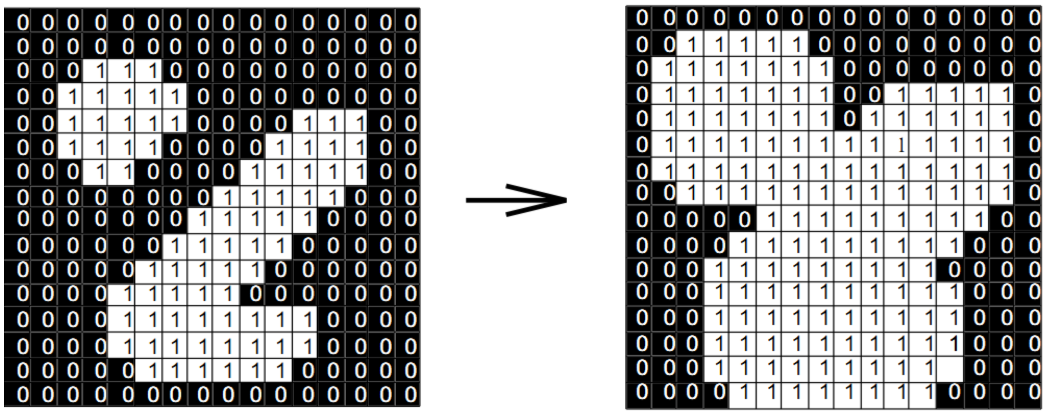


Figure 3.8 Dilation operation.

3.3.2 Opening and Closing

Opening is done by applying erosion and dilation to the image, respectively. It removes details and breaks narrow regions of the image like erosion however its effect to image is less than erosion. Firstly performed erosion removes noise on image but also creates some holes by removing narrow regions. To eliminate this effect dilation is performed. An example is shown in Figure 3.9.

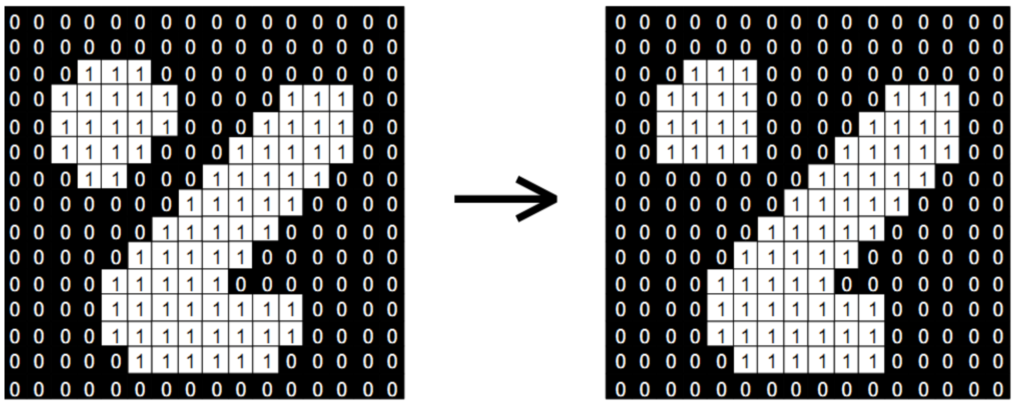


Figure 3.9 Opening operation (Fisher, et al., 1996).

Closing is done by applying dilation and erosion to the image respectively, Closing can fill holes make image solid. Effect to the original size of the image is less than dilation. It preserves shape of the original image. An example is shown in Figure 3.10 .

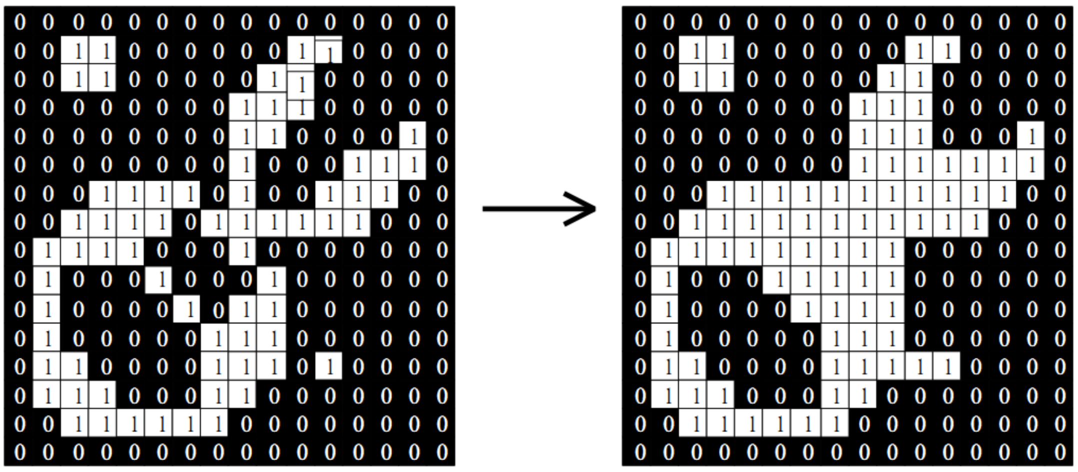


Figure 3.10 Closing operation (Fisher, et al., 1996).

Although example results of opening and closing operation are given above, exact outputs of operations can change by selected structuring element.

CHAPTER FOUR
VIDEO FUNDEMENTALS

4.1 Interlaced and Noninterlaced Video

Interlacing is displaying a frame in two sections called fields. Fields consist of equal number of horizontal lines and one of the fields is transmitted at a time. This technique was developed for transmitting TV signals because of bandwidth restrictions. Video standards NTSC, PAL, & SECAM are interlaced. NTSC frames can be displayed roughly 1/30th of a second. As shown in Figure 4.1 field 1 is followed by field 2 for each frame.

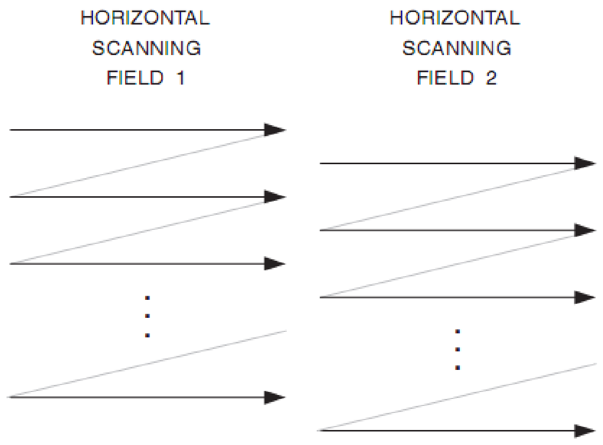


Figure 4.1 Interlaced video (Jack, 2005).

In contrast to older televisions, most of computer monitors and LCD TVs has increased frame rate. Those can display one frame in 1/60th of a second each. So video has to be converted to noninterlaced and this process is called deinterlacing.

There are two common deinterlacing techniques. First of all, scan line duplication simply duplicates the previous active scan line. Although the number of active scan lines doubled, there is no increase in vertical resolution. Scan line interpolation generates interpolated scan lines between the original active scan lines. Although the number of active scan lines doubled, the vertical resolution is not. The simplest implementation uses linear interpolation to generate a new scan line between two input scan lines (Jack, 2005). The new scan line is found by taking the average of original scan lines. Implementations are shown in Figure 4.2.

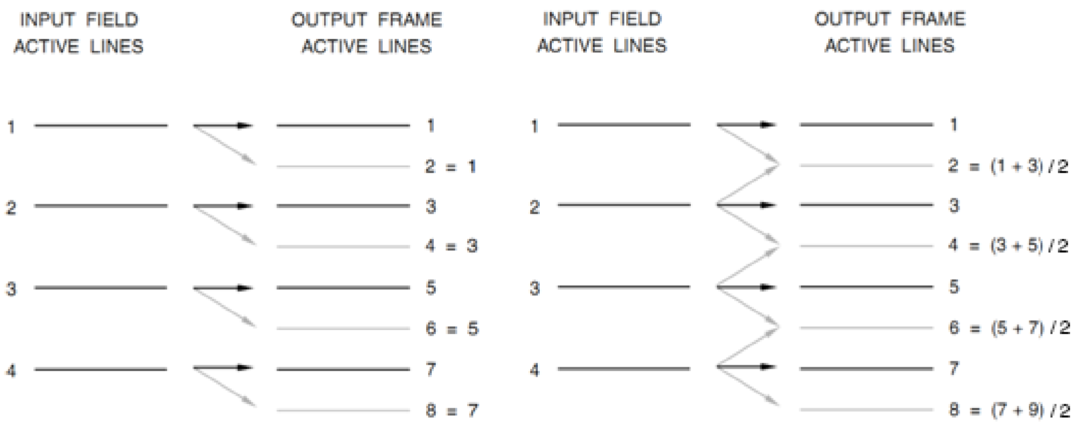


Figure 4.2 Scan line duplication and scan line interpolation techniques (Jack, 2005).

4.2 CVBS

CVBS stands for composite video baseband signal. It combines the brightness, timing and color information within one signal. Waveform of a one horizontal line white screen NTSC composite video signal was shown in Figure 4.3. Each line consists of active video and the horizontal blanking portion. In the active video portion brightness and color information are transmitted together. Amplitude of the signal gives the brightness levels and it can be maximum 1V peak to peak.

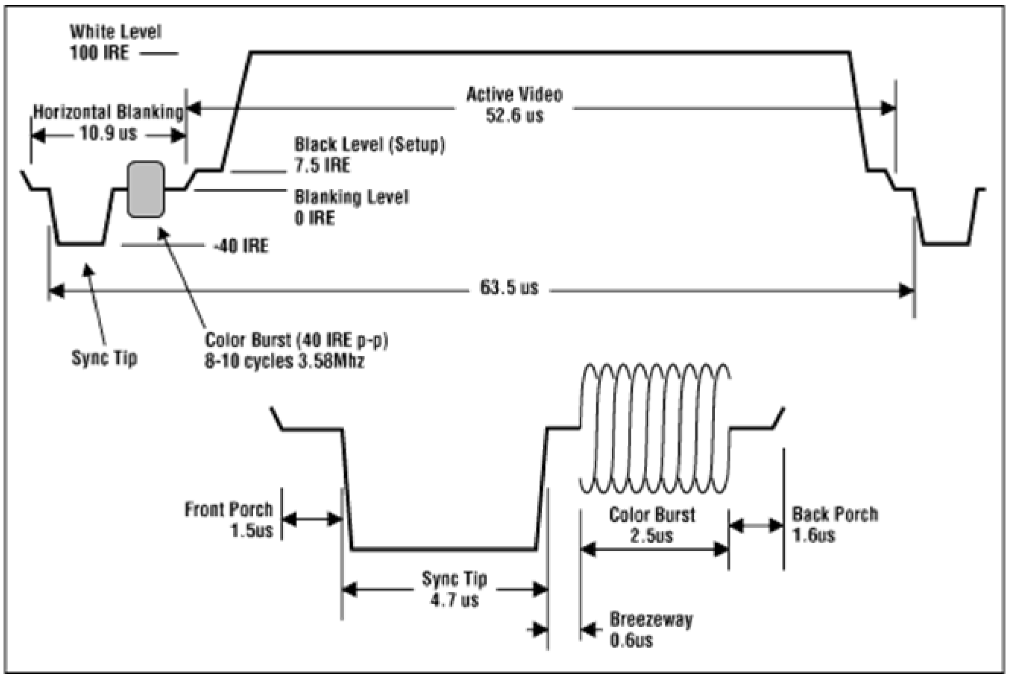


Figure 4.3 NTSC video waveform (Maxim, 2002).

Color information is a sine wave, which is added on active video portion, and its phase difference from color burst which is added on horizontal blanking determines the color. As shown In Figure 4.4 amplitude specifies the amount of brightness, and phase difference determines the color. The representation of this signal waveform on monitor is colors from white to black which is called color bar.

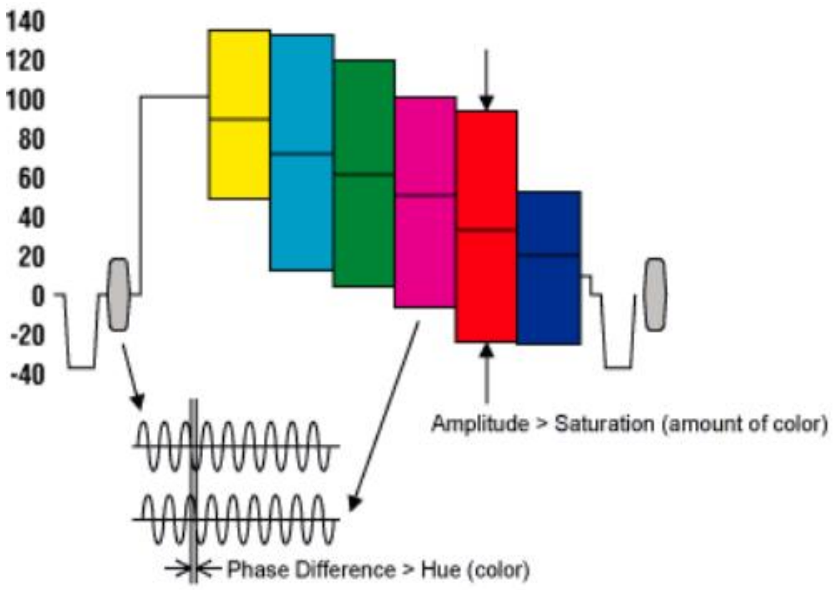

140
120
100
80
60
40
20
0
-20
-40
Amplitude > Saturation (amount of color)
Phase Difference > Hue (color)

Figure 4.4 Color video waveform (Maxim, 2002).

4.3 YCbCr and ITU-R BT.656

The YCbCr color space was developed as part of ITU-R BT.601 during the development of a world-wide digital component video standard (Jack, 2005). In YCbCr, Y represents brightness, Cb represents “blue – Y” and Cr represents “red – Y”

ITU656 is a digital video format to transmit PAL (625 lines) and NTSC (525 lines) analog standard definition TV signals. The standard uses 4:2:2 YCbCr digital video. Typical usage is between an analog video decoder and video processor in TV sets. The ITU656 parallel interface uses 8 bits of multiplexed YCbCr data and a 27 MHz - clock. The 4:2:2 YCbCr data is multiplexed into an 8-bit stream: Cb0Y0Cr0Y1Cb2Y2Cr2, etc. as shown in Figure 4.5.

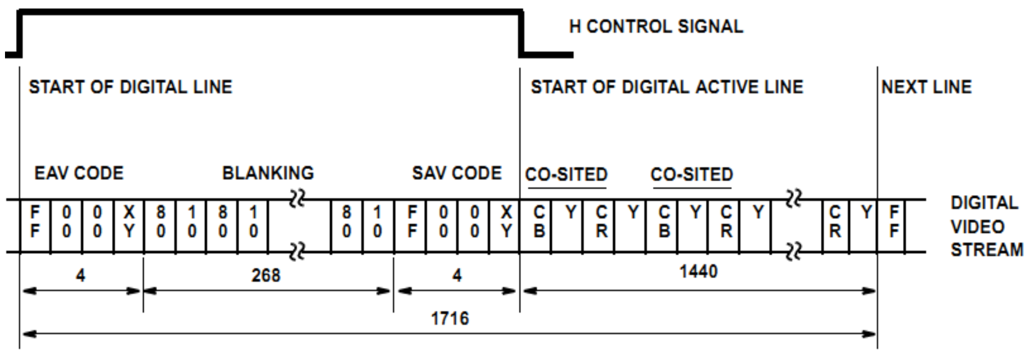


Figure 4.5 Standard ITU-R BT.656 data stream format for a single horizontal line (Intersil, 2002).

After the EAV codes, active video data of the line ends and the next line starts and SAV code is the start of the active video data within the current line. These codes in ITU-R BT.656 data stream eliminate the need for horizontal and vertical sync signals. First three byte of EAV and SAV codes are constant and last byte gives field information (field 1 or 2), and vertical and horizontal blanking states.

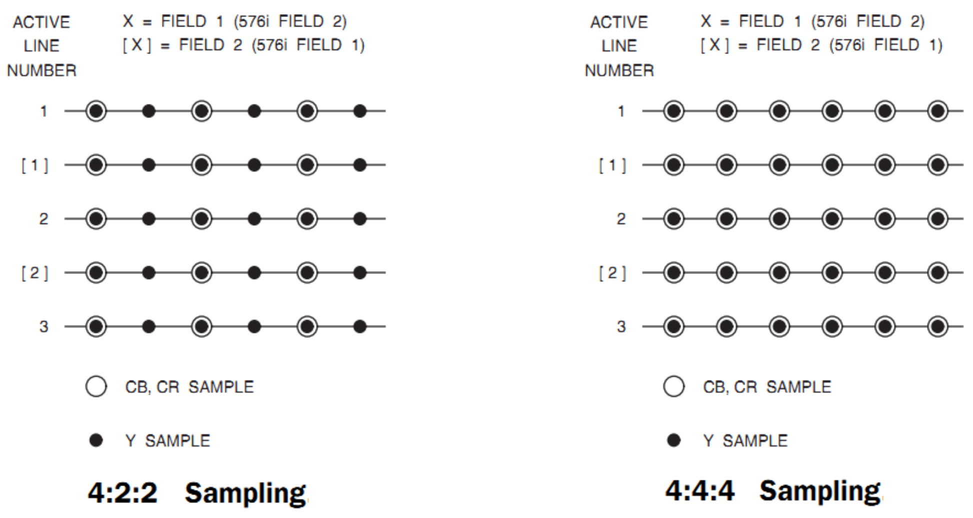


Figure 4.6 The sampling positions on the active scan lines of an interlaced video (Jack, 2005).

Each successive group of CbYCr corresponds to one pixel and out of group Y components corresponds to the next pixel. Therefore this standard is called YCbCr 4:2:2. Sampling position of YCbCr 4:2:2 can be seen on the left in Figure 4.6. If RGB process is necessary, before conversion to RGB 4:2:2 data must be converted to 4:4:4 so as to each Y sample have a corresponding Cb and Cr sample as shown on the right of Figure 4.6. The conversion is made by copying Cb and Cr values for every next single Y sample.

4.4 YCbCr and RGB Conversion

YCbCr and RGB are both common digital video signals and the difference between those color spaces is on the representation of color. YCbCr is the combination of brightness and two color difference signals; on the other hand RGB is the combination of three colors, red, green, and blue.

In order to transmit YCbCr signals to a VGA compatible display, eventually it has to be converted to RGB color space. The equation between color RGB and YCbCr is below.

$$\begin{aligned} R &= 1.164(Y_{601} - 16) + 1.596(Cr - 128) \\ G &= 1.164(Y_{601} - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \\ B &= 1.164(Y_{601} - 16) + 2.018(Cb - 128) \end{aligned} \tag{4.1}$$

CHAPTER FIVE
SYSTEM ARCHITECTURE

5.1 What is FPGA?

Field programmable gate arrays (FPGAs) are digital integrated circuits (ICs) that contain configurable (programmable) blocks of logic along with configurable interconnects between these blocks. Design engineers can configure (program) such devices to perform a tremendous variety of tasks. The “field programmable” portion of the FPGA’s name refers to the fact that its programming takes place “in the field”. This may mean that FPGAs are configured in the laboratory, or it may refer to modifying the function of a device resident in an electronic system that has already been deployed in the outside world (Maxfield, 2004).

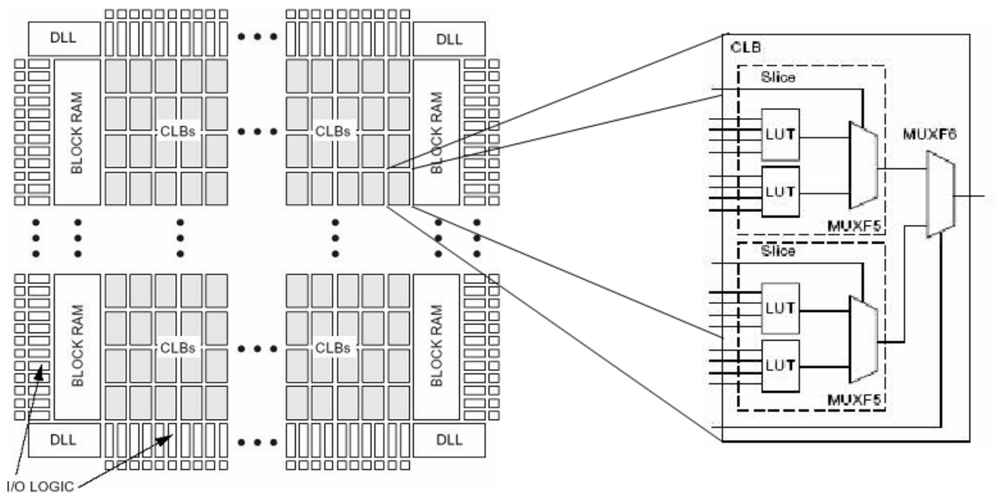


Figure 5.1 FPGA structure (Altium Designer, 2008).

In Figure 5.1 logic blocks and programmable interconnections can be seen. The function of each logic cell and interconnections are specified by hardware description languages to implement desired circuit design.

The resolutions of many video applications are increasing. In particular, resolutions used in military and medical technologies are necessarily expanding. As a result of large size of data, processing speed and compression problems are emerging.

In order to overcome problems and accomplish tasks high performance and reliable platforms like FPGA are needed.

5.2 Advantages of FPGA

There is a number of architecture that can be used for image processing. One of them is ASSP (Application Specific Standard Product) which is dedicated a specific function and designed for many customer so customer can't make changes on the products. As new technologies develop, demand for cheap and more skilled ICs rises in the competitive IC market. Hence ASSPs have a short production life approximately 3-4 years. Obsolescence may not be problem for the producers who change their product in short periods but it is a major problem for products expected to be used for long years.

Another alternative is ASICs (Application Specific Integrated Circuit). They combine specific functions like ASSP but produced for a particular customer. ASIC production is an expensive process. Because of long duration of the design and high engineering costs it is not preferred by many manufacturers. This high development costs can be acceptable for just companies those have high capacity production.

Lastly the most significant disadvantage of DSP and other processors is their processing speed. For many circuit applications they can be sufficient however even though day by day computer processors are getting faster, for complex circuit applications like HD TV, their performance can be low.

High technology image processing applications need more processing performance. In this case a faster technology than processors is needed and this necessity can be met by FPGA, what I propose in this thesis. Next sections discuss the advantages of FPGA.

5.2.1 Performance

Taking advantage of hardware parallelism, FPGAs exceed the computing power of digital signal processors (DSPs) by breaking the paradigm of sequential execution and accomplishing more per clock cycle. Controlling inputs and outputs (I/O) at the hardware level provides faster response times and specialized functionality to closely match application requirements (National Instruments, 2010).

5.2.2 Time to Market

FPGA technology offers flexibility and rapid prototyping capabilities in the face of increased time-to-market concerns. You can test an idea or concept and verify it in hardware without going through the long fabrication process of custom ASIC design. You can then implement incremental changes and iterate on an FPGA design within hours instead of weeks. Commercial off-the-shelf (COTS) hardware is also available with different types of I/O already connected to a user-programmable FPGA chip (National Instruments, 2010).

5.2.3 Cost

The nonrecurring engineering (NRE) expense of custom ASIC design far exceeds that of FPGA-based hardware solutions. The large initial investment in ASICs is easy to justify for OEMs shipping thousands of chips per year, but many end users need custom hardware functionality for the tens to hundreds of systems in development. The very nature of programmable silicon means that there is no cost for fabrication or long lead times for assembly. As system requirements often change over time, the cost of making incremental changes to FPGA designs are quite negligible when compared to the large expense of respinning an ASIC (National Instruments, 2010).

5.2.4 Reliability

While software tools provide the programming environment, FPGA circuitry is truly a “hard” implementation of program execution. Processor-based systems often involve several layers of abstraction to help schedule tasks and share resources among multiple processes. The driver layer controls hardware resources and the operating system manages memory and processor bandwidth. For any given processor core, only one instruction can execute at a time, and processor-based systems are continually at risk of time-critical tasks pre-empting one another. FPGAs, which do not use operating systems, minimize reliability concerns with true parallel execution and deterministic hardware dedicated to every task (National Instruments, 2010).

5.2.5 Long-term Maintenance

FPGA chips are field-upgradable and do not require the time and expense involved with ASIC redesign. Digital communication protocols, for example, have specifications that can change over time, and ASIC-based interfaces may cause maintenance and forward compatibility challenges. Being reconfigurable, FPGA chips are able to keep up with future modifications that might be necessary. As a product or system matures, you can make functional enhancements without spending time redesigning hardware or modifying the board layout (National Instruments, 2010).

5.3 FPGA Demonstration Board

5.3.1 Hardware Architecture

DE2-70 Development and Education board is very beneficial tool to learn about FPGAs. The circuits and Altera Cyclone® II 2C70 FPGA equipped with almost 70,000 logic elements on the board allow users to implement many simple and complicated applications. A photograph of the DE2-70 board is shown in Figure 5.2.

It depicts the layout of the board and indicates the location of the connectors and key components (Terasic, 2009).

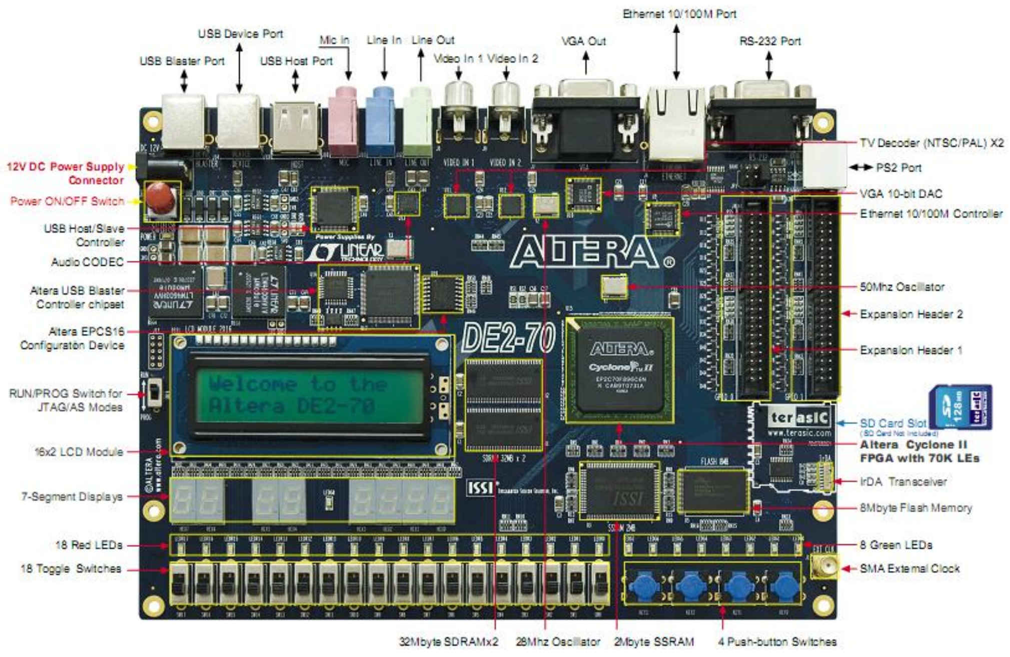


Figure 5.2 The DE2-70 board.

To provide maximum flexibility for the user, all connections are made through the Cyclone II FPGA device. Thus, it is very easy for user to configure the FPGA to implement any system design. Block diagram of the FPGA board is shown in Figure 5.3. As seen, it is very simple to connect any input to outputs of the system by HDL coding (Terasic, 2009).

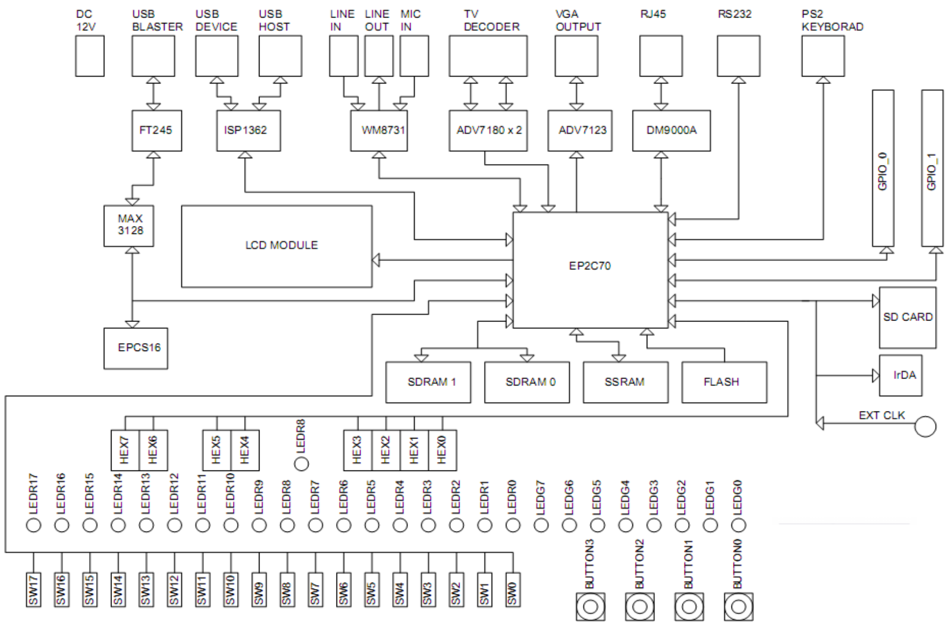


Figure 5.3 Block diagram of DE2-70 board (Terasic, 2009).

Following is more detailed information about the most important blocks used to implement projects.

Cyclone II 2C70 FPGA

- 68,416 Les (Logic elements)
- 250 M4K RAM blocks
- 1,152,000 total RAM bits
- 150 embedded multipliers
- 4 PLLs (Phase locked loops)
- 622 user I/O pins

SDRAM

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

VGA output

- Uses the ADV7123 140-MHz triple 10-bit high-speed video DAC
- Supports up to 1600 x 1200 at 100-Hz refresh rate

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 digital output formats : 8-bit ITU-R BT.656 YCbCr 4:2:2 output

5.3.2 TV Box Demonstration

DE2-70 Board kit contains illustrative demonstrations for programmable logic device design software called Quartus. In the project DE2-70 TV demonstration is used to realize image processing algorithms. The hardware layout of the DE2-70 TV demonstration can be seen in Figure 5.4.

The DE2-70 board has software support for standard I/O interfaces and a control panel facility for accessing various components. Also, software is provided for a number of reference designs and demonstrations that illustrate the advanced capabilities of the DE2-70 board (Terasic, 2009).

One of the demonstrations is TV Box Demonstration, which is useful to prepare this thesis. This demonstration plays video input from a CVBS source (DVD, camera etc.) using the VGA output and one TV decoder on the DE2-70 board. Figure 5.4 shows the block diagram of the design. There are two major blocks in the circuit, called I2C_AV_Config and TV_to_VGA. The TV_to_VGA block consists of the ITU-R 656 Decoder, SDRAM Frame Buffer, YUV422 to YUV444, YCbCr to RGB, and VGA Controller. The figure also shows the TV Decoder (ADV7180) and the VGA DAC (ADV7123) chips used (Terasic, 2009).

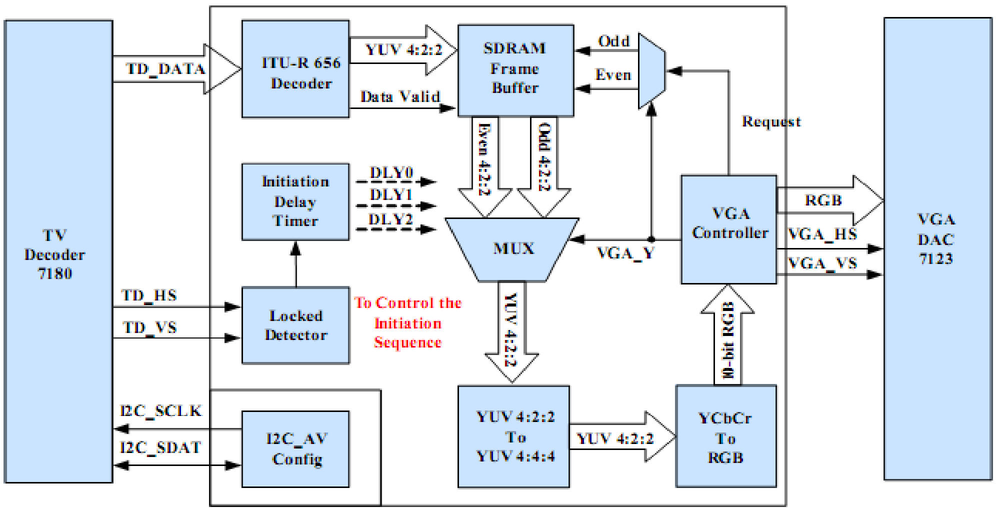


Figure 5.4 Hardware architecture of project.

TV demonstration plays input video on VGA monitors. After power-on I2C block communicates with TV decoder and registers are initialized to make it recognize the CVBS video. Then CVBS video is converted to ITU-R 656 format. Locked detector makes sure that TV decoder is ready. ITU-R 656 decoder extract YCbCr 4:2:2 data and interlaced video stored to SDRAM. While input video is streaming, the odd and even lines de-interlaced and frame is sent to next block. The conversion process from 4:2:2 to 4:4:4 take place in YUV 4:2:2 to YUV 4:4:4 block to make Y sample has a corresponding Cb and Cr sample. Next block converts data to RGB. Finally VGA controller block produce timing signals and send to digital to analog converter together with data. When timing signals are active, another signal called Request becomes active so that data in the SDRAM starts to be transmitted toward VGA Controller (Terasic, 2009).

5.4 Hardware Description

5.4.1 Design Flow

In the design process, hardware description languages such as Verilog and VHDL, describes circuit behaviors that will be implemented in the FPGA. Synthesis process configures logic blocks and creates interconnections between these blocks from HDL code. FPGA design software generates a file to load into FPGA. After it is synthesized by FPGA design software (Quartus, etc.) desired circuit can be programmed in the FPGA usually via usb cables. Designer can observe signals during simulation stage by writing a test environment using HDL. In this stage expected functions of code are verified and in case of unexpected results designer changes the HDL code. An FPGA Design flow can be seen in Figure 5.5.

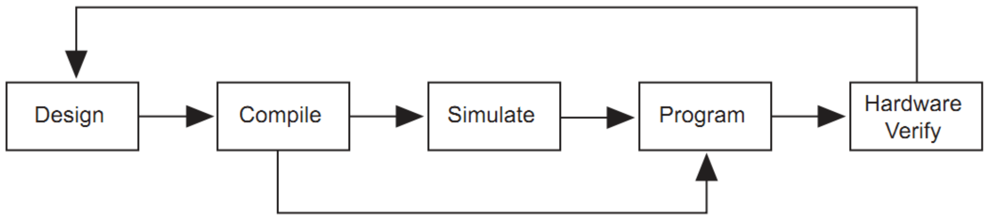


Figure 5.5 Design Flow (Altera, 2008).

5.4.2 Block Description

A design may contain several blocks. These blocks are connected with each other to create complete design. This approach makes design more understandable. Blocks may also contain other blocks inside and items declared in a block are not allowed to be used outside the block. In Figure 5.6 two blocks from the edge detection application and their connections can be seen. As you see they have some common connections. For example outputs of YCbCr2RGB block such as mGreen,mRed, mBlue are connected as inputs to edge_detect block.

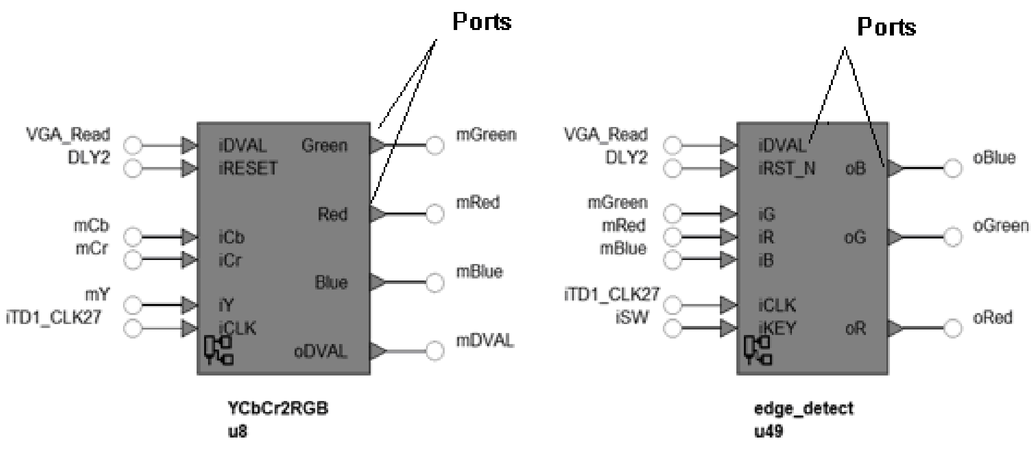


Figure 5.6 Connections between blocks

In the top file, block connections, wires and if necessary combinational circuits are defined. Input and output ports in a block allow communication between other blocks. Verilog description of blocks shown in Figure 5.6 can be seen below. Signals that connects blocks are specified as wires. In parenthesis wire connections between ports of blocks are specified.

```
wire    [9:0]   mRed;
wire    [9:0]   mGreen;
wire    [9:0]   mBlue;
…
…
YCbCr2RGB               u8  (         //   Output Side
                        .Red(mRed),
                        .Green(mGreen),
                        .Blue(mBlue),
                        .oDVAL(mDVAL),
                                      //   Input Side
                        .iY(mY),
                        .iCb(mCb),
                        .iCr(mCr),
                        .iDVAL(VGA_Read),
                                      //   Control Signal
                        .iRESET(!DLY2),
                        .iCLK(iTD1_CLK27));

edge_detect           u49     (       //   Input Side
                        .iR(mRed),
                        .iG(mGreen),
                        .iB(mBlue),
                        .iDVAL(VGA_Read),
                                      //   Output Side
                        .oR(oRed),
                        .oG(oGreen),
                        .oB(oBlue),
                                      //   Control Signals
                        .iRST_N(DLY2),
                        .iCLK(iTD1_CLK27),
                        .iKEY(iSW[17:0]));
```

5.4.3 Verilog Code Sample

A shortened version of edge detection block verilog codes is given below. First part describes input and output ports of the design. Second part declares internal wires or registers used in the block and describes the operational function of the block. According to switch position on DE2-70 TV board image shown on the monitor changes.

In the edge detection block, two blocks, which derived from Altera Verilog library called line buffer and square root blocks are used.

```
//  FIRST PART ---------------------------------------------
module edge_detect (
                    iR,
                    iG,
                    iB,
                    oR,
                    oG,
                    oB,
                    iDVAL,
                    iRST_N,
                    iCLK,
                    iKEY);
//  Inputs
input [9:0]     iR;
input [9:0]     iG;
input [9:0]     iB;
input            iRST_N;
input            iCLK;
input           iDVAL;
input [17:0]     iKEY;
//  Outputs
output [9:0]    oR;
output [9:0]    oG;
output [9:0]    oB;

//  SECOND PART-------------------------------------------
//  Internal registers/wires
reg [9:0] Red;
reg [9:0] Green;
reg [9:0] Blue;
reg [9:0] cRed;
reg [9:0] cGreen;
reg [9:0] cBlue;
wire [9:0] a0;
wire [9:0] b0;
wire [9:0] c0;
reg [9:0] a1;
reg [9:0] b1;
reg [9:0] c1;
reg [9:0] a2;
reg [9:0] b2;
reg [9:0] c2;
reg [9:0] a3;
reg [9:0] b3;
reg [9:0] c3;
wire [26:0] input_sqrt;
wire [13:0] sqrt;
wire [9:0] BW;
reg [12:0] oBW;
reg [12:0] oBZ;
reg [12:0] oBZZ;
reg [12:0] oX;
reg [12:0] oY;
reg [26:0] oZ;
reg [12:0] X_edge;
reg [12:0] Y_edge;
reg [12:0] X1;
reg [12:0] X2;
reg [12:0] Y1;
reg [12:0] Y2;
//   Approximate grayscale conversion and assignments
assign BW     =   (iR>> 2) + (iB >> 2) + (iG >> 1);
assign oR      =  oBW[12:3]    ;
assign oG      =  oBW[12:3]    ;
assign oB      =  oBW[12:3]    ;
assign  input_sqrt     =   oZ;

always@(posedge iCLK)
begin
    if(!iRST_N)
//   Initialize registers if RESET occurs
    begin
        X1          <= 0;
        X2          <= 0;
        Y1          <= 0;
        Y2          <= 0;
        X_edge      <= 0;
        Y_edge      <= 0;
        oZ          <= 0;
        oBW         <= 0;
        oBZ         <= 0;
        oX          <= 0;
        oY          <= 0;
    end
    else
    begin
        if(iDVAL==1'b1)
        begin
            c1<=c0;
            c2<=c1;
            c3<=c2;
            b1<=b0;
            b2<=b1;
            b3<=b2;
            a1<=a0;
            a2<=a1;
            a3<=a2;

// Sobel calculations
            X1 = (a1+(a2<<1)+a3);
            X2 = (c1+(c2<<1)+c3);
            Y1 = (a3+(b3<<1)+c3);
            Y2 = (a1+(b1<<1)+c1);

            if (X1 > X2)
                X_edge= X1 - X2;
            else
                X_edge = X2 - X1;
            if (Y1 > Y2)
                Y_edge=Y1 - Y2;
            else
```

```
                Y_edge=Y2 - Y1;

            oZ = (X_edge*X_edge) + (Y_edge*Y_edge);
            oBZ = sqrt[13:1];

//  Horizontal edges
            if(iKEY[2:0]==3'b001)
            begin
                ...
                oBW=X_edge;
                ...
            end

//  Vertical edges
            else if(iKEY[2:0]==3'b010)
            begin
                ...
                oBW=Y_edge;
                ...
            end
//  Gradient magnitude
            else if(iKEY[2:0]==3'b100)
            begin
                ...
                oBW=oBZ;
                ...
            end
        end
    end

//  Other blocks and their connections inside edge detection block
//Three line buffers for windowing operation
line_buf u0   (   .clken(iDVAL),
                  .clock(iCLK),
                  .shiftin(BW),
                  .taps0x(a0),
                  .taps1x(b0),
                  .taps2x(c0));

//Square root calculation
sqrt    u1  (      .clk(iCLK),
                  .radical(input_sqrt),
                  .q(sqrt),
                  .remainder());

endmodule
```

CHAPTER SIX
IMAGE PROCESSING APPLICATIONS

6.1 Object and Motion Detection in Thermal Imagery

A thermal camera captures heat radiated in the infrared range of electromagnetic spectrum by objects, invisible to the human eye. Infrared radiation differences are represented by colors in a thermal camera so that we interpret temperature differences. Actually thermal camera was invented for human detection and vision enhancement in military, its usage became widespread. For example, it is used for troubleshooting and maintenance in industrial applications.

Color is one of the most important elements of image processing. Color itself can be corrected or enhanced during image processing. At the same time different applications can be based on color. Several applications such as face detection and motion detection can use color as a tool. Project described below use color to detect objects in a thermal image. It was implemented for detection of red, which shows the hottest object. Object and motion detection block accepts RGB signals as input, finds red pixels and transmits to VGA controller. In the application code, the color values can be altered to detect the desired colors.

In Figure 6.1 a thermal image which contains two objects that radiated infrared light can be seen. As you see, different colors represent different temperature ranges. Blue shows the coldest areas, red shows the hottest areas and yellow shows the mid-temperature areas. Activating SW 0 (Switch 0) on DE2-70 TV board starts detection. In Figure 6.2, successfully detected objects can be seen. In various applications, this algorithm can be implemented to detect humans.

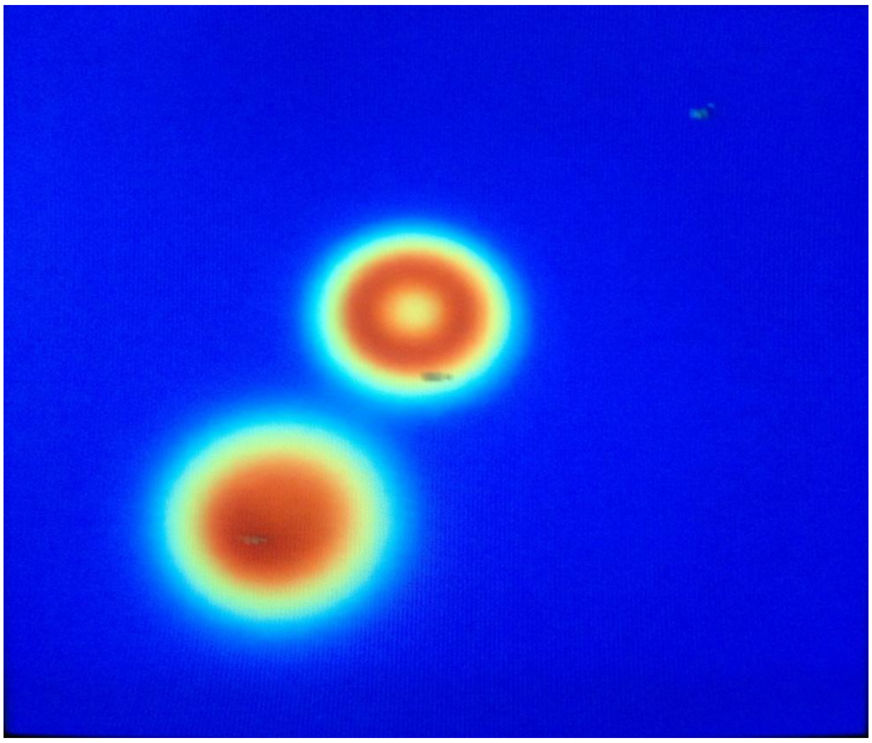


Figure 6.1 Thermal image.

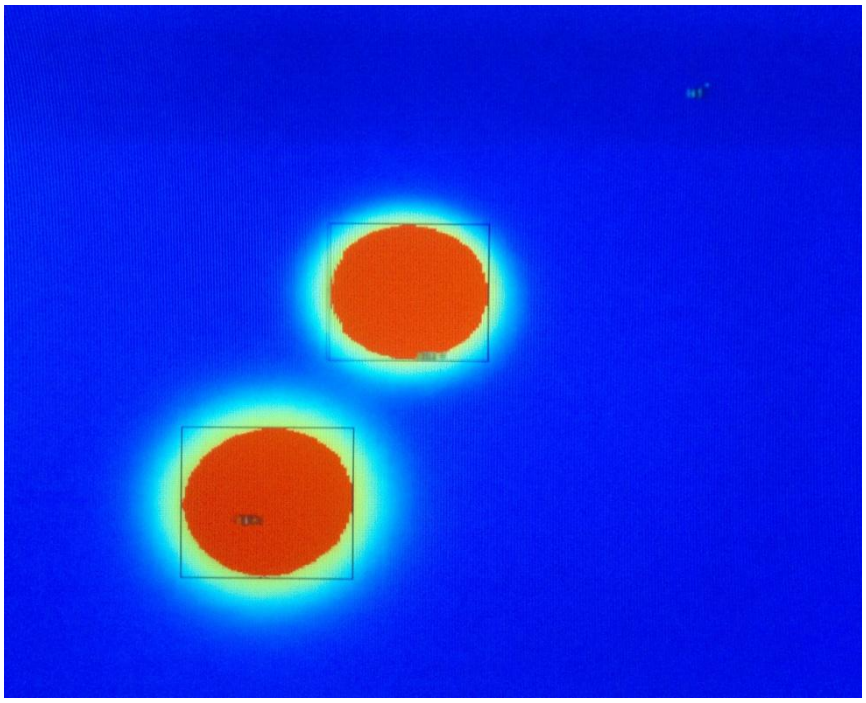


Figure 6.2 Detection of objects.

At the time of initial detection of desired color, that point’s coordinates are stored and top boundary of the object has been identified. Next detected pixels on the image provide to find left, right and bottom boundaries. The edge points are marked and a black square is drawn around the object. In every frame, the square is updated. The detected pixels in the range were converted to pure red to segment the object out of background. In this application detection up to two objects is implemented however any number of objects can be detected through the modification on the HDL code.

By activating SW 1, motion and its direction can be detected. After an object has been identified, positions of enclosing square in consecutive frames are compared. Changes in the center of the square indicate the direction of object on the white frame as shown in Figure 6.3. Motion detection can be applied for just one object or group of objects.

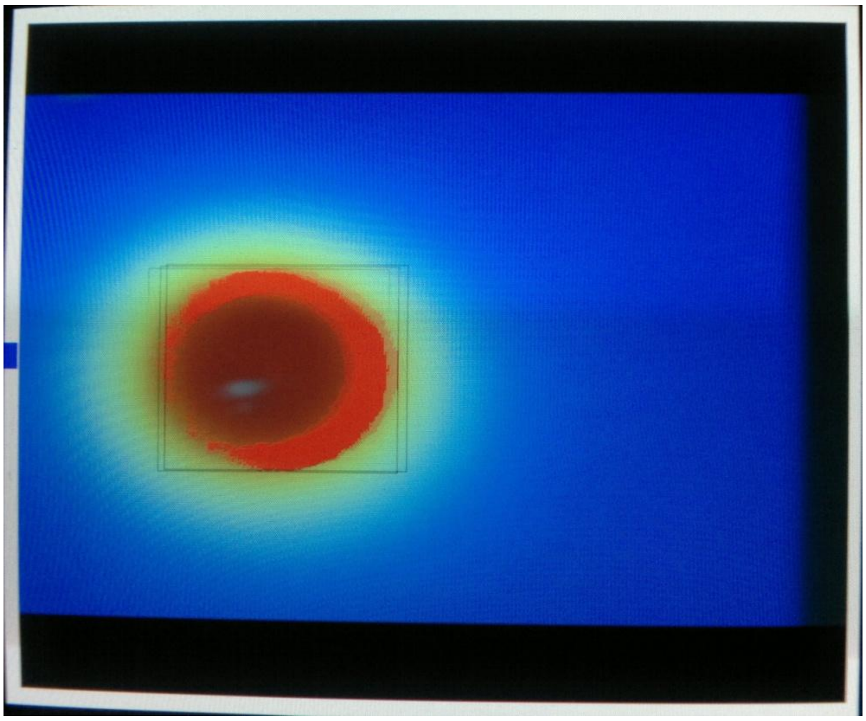


Figure 6.3 Moving hot object.

In order to develop better algorithms for motion and object detection based on color, range of color has to be selected carefully.

6.2 Contrast Enhancement

6.2.1 Real Time Histogram Display

Real time histogram display algorithm was applied to RGB images. RGB images are converted to grayscale. In fact, conversion is unnecessary because 'Y' signal is already present. After conversion, 10-bit grayscale signal was divided into 256 portions. Every portion represents four gray levels. The pixel values correspond to this portions were counted and counts were reduced at the same rate and displayed as a bar graph on motion picture. In Figure 6.4 the result of the algorithm can be seen in color image. The histogram is calculated just for luminance, not for color channels, and located on the left bottom of the picture. Every second, histogram is updated at least 30 times together with the image. SW 0 on the DE2-70 TV board makes histogram visible.

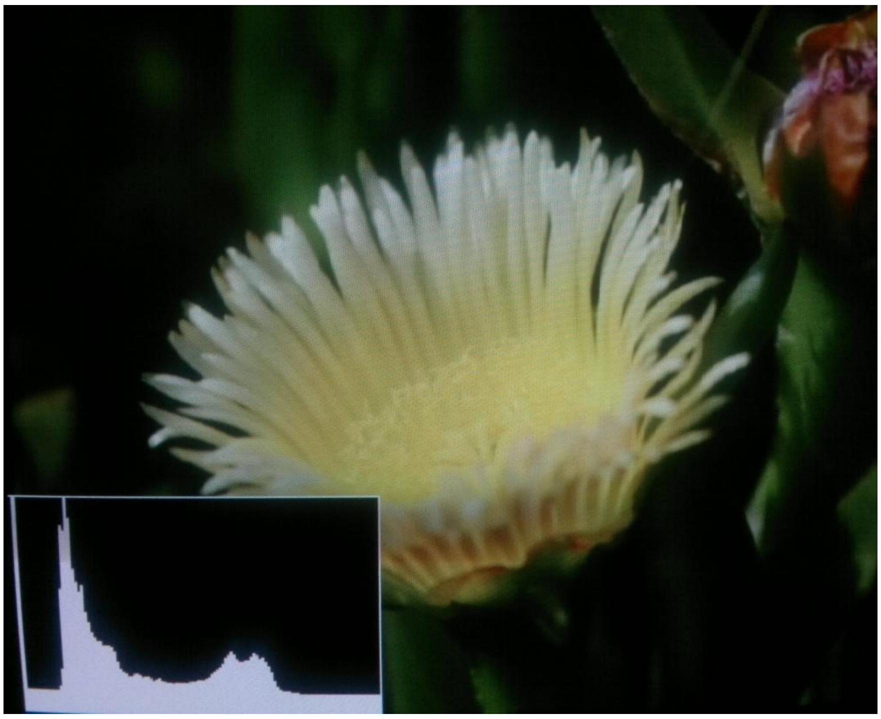


Figure 6.4 Histogram monitoring on FPGA.

Picture settings menu of SX200 camera can be seen in Figure 6.5. Histogram of the camera and FPGA histogram implementation can be compared. It can be clearly said that camera shows a similar histogram pattern with FPGA demonstration.

However, dynamic range of the camera output is compressed from two sides of the grayscale levels, which lead lower contrast than original.

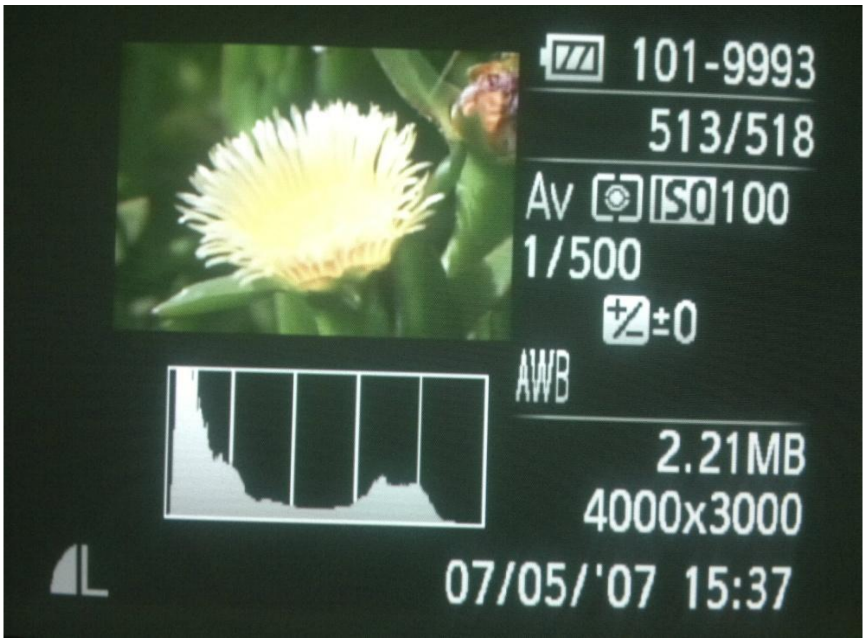


Figure 6.5 Picture settings menu of camera.

6.2.2 Contrast Enhancement

If there is a low contrast image contrast enhancement can be applied to retain details which were lost. There are various curve shapes mentioned in section 3.1.1 and I applied all of the curves on the image. The low contrast image can be seen in Figure 6.6. The number of existent gray levels is almost half of the number of total gray levels.

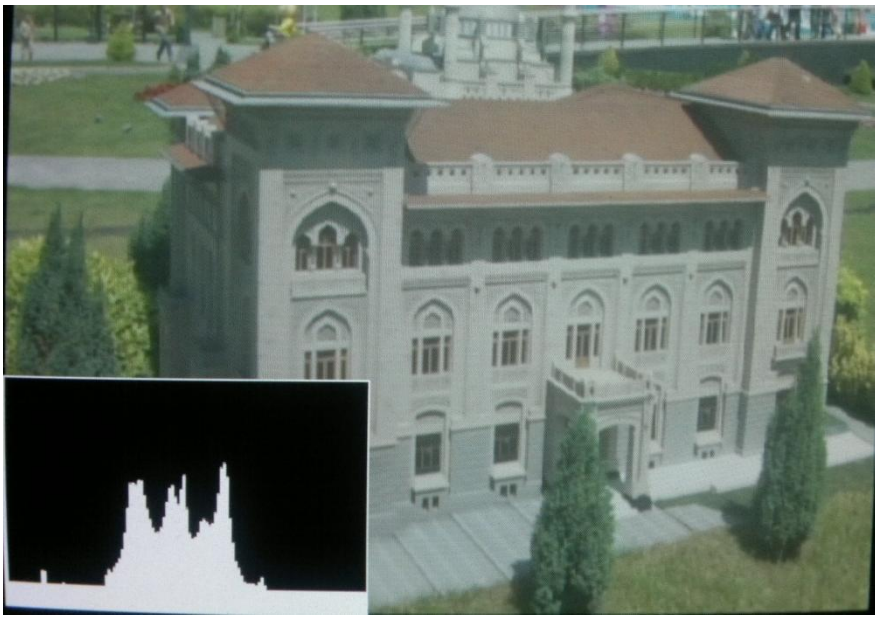


Figure 6.6 Low contrast image.

6.2.2.1 Grayscale Contrast Enhancement

The curve in Figure 6.7 was applied to input video RGB signals to obtain a high contrast image. RGB signals are converted to 10-bit grayscale. X Axis corresponds to 10 bit input gray levels which is equal to 1024 different values and Y axis is its contrast enhanced gray levels. Due to the curve shape, dark areas will be darker bright areas will be brighter and middle levels will be stretched in two directions after contrast enhancement. Input values are between 100 and 800. Because my camera's composite video output is analyzed by histogram monitoring and it is noted that it cannot exceed these values. So gray level 100 will be 0(darkest) and 800 will be 1023(brightest).

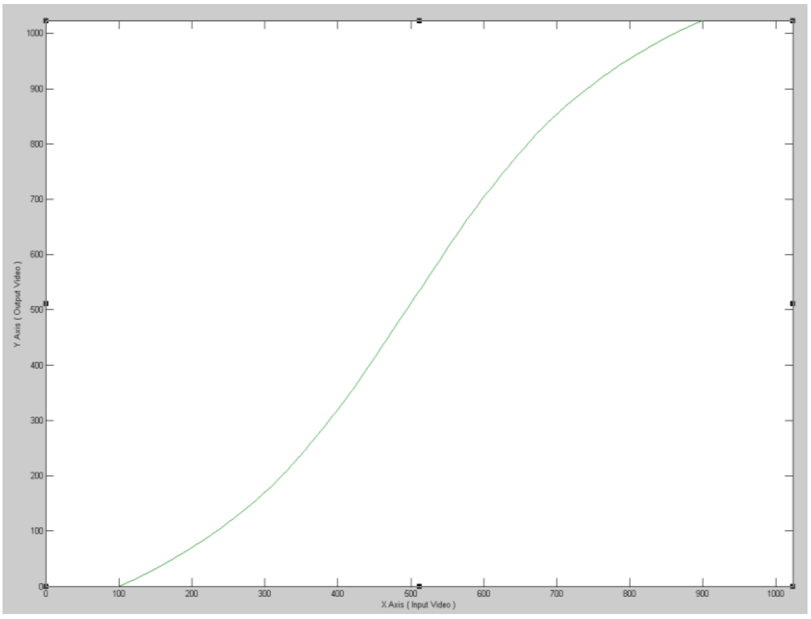


Figure 6.7 High contrast curve.

In Figure 6.8, on the left, there is a grayscale version of original picture and on the right, there is an enhanced version. So differences between pictures and the effect of curve can be evaluated easily. Additionally, histogram of the enhanced picture is given. When it is compared with histogram in Figure 6.8, stretching effect of adjustment is obvious and image details are relatively more visible.

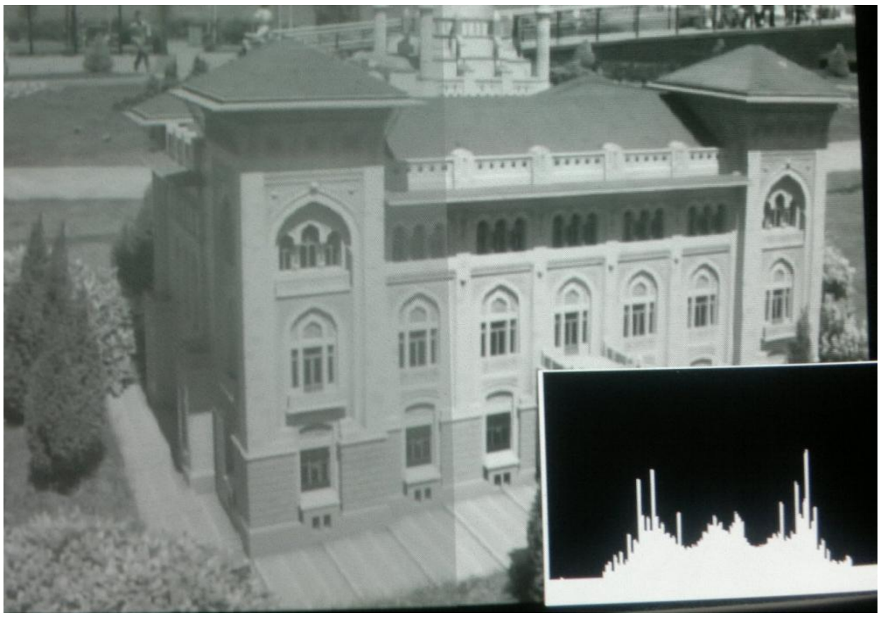


Figure 6.8 Grayscale contrast enhancement.

6.2.2.2 Color Image Contrast Enhancement

Y signal is a copy of grayscale image The DE2-70 TV board has 8 bits of Y signal so the difference from the grayscale enhancement is only the number of bits. Y signal can also be used as input to VGA controller instead of RGB. In this case, 2 least significant bits must be added to Y signal and once again grayscale contrast enhancement can be performed.

The similar algorithm used in grayscale contrast enhancement is applied on YCbCr signals. First, histogram was stretched to eliminate narrow grayscale output range of camera, and then high contrast curve was applied to increase contrast of the image. Curves respectively can be seen in Figure 6.9.

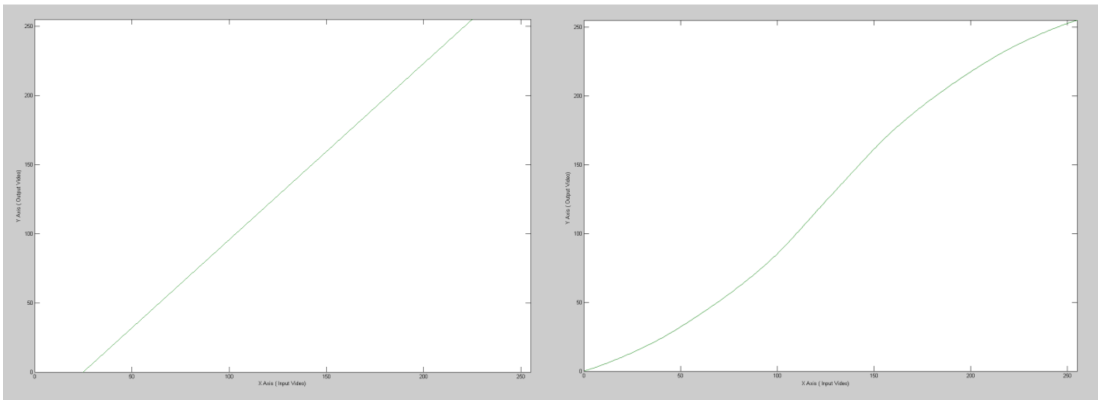


Figure 6.9 Stretching and high contrast curves.

SW 0 enables the stretching effect and then SW 1 applies high contrast curve. SW 5 divides screen so modified image can be seen on the right and original picture on the left. The effect of curves on color image can be seen in Figure 6.10. It is clear that modified image is quite better than the original image. After curve adjustment at luminance channel, colors also seem enhanced.

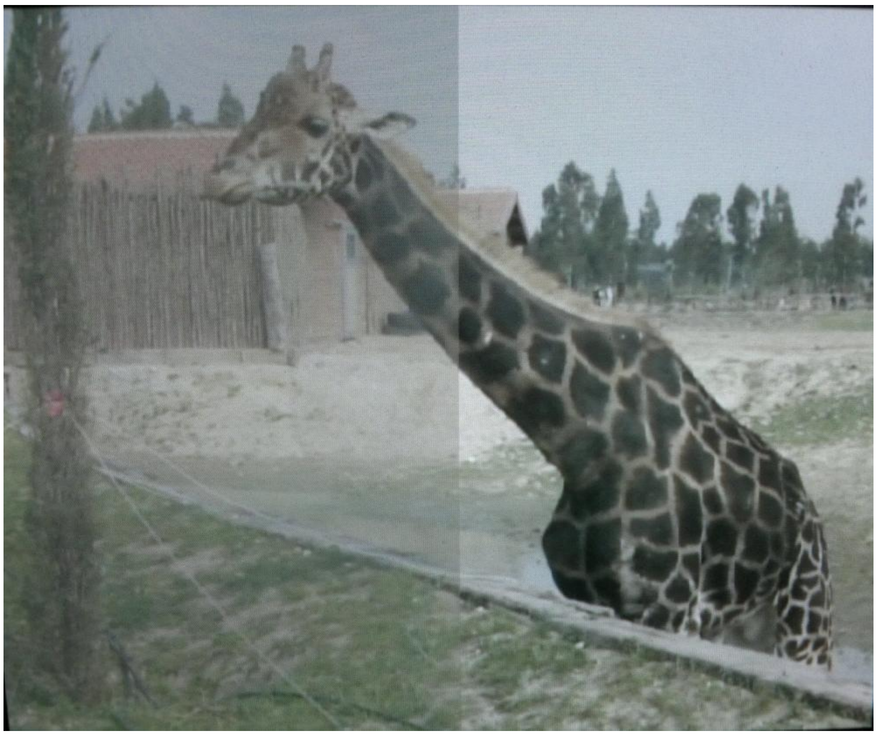


Figure 6.10 Color image contrast enhancement.

Sometimes it is not necessary to convert an image to a high contrast image, instead brightness change can be needed. In Figure 6.11, curve on the left makes images darker and curve on the right makes images brighter. SW 2 and SW 3 activate these features and results can be seen in Figure 6.12.

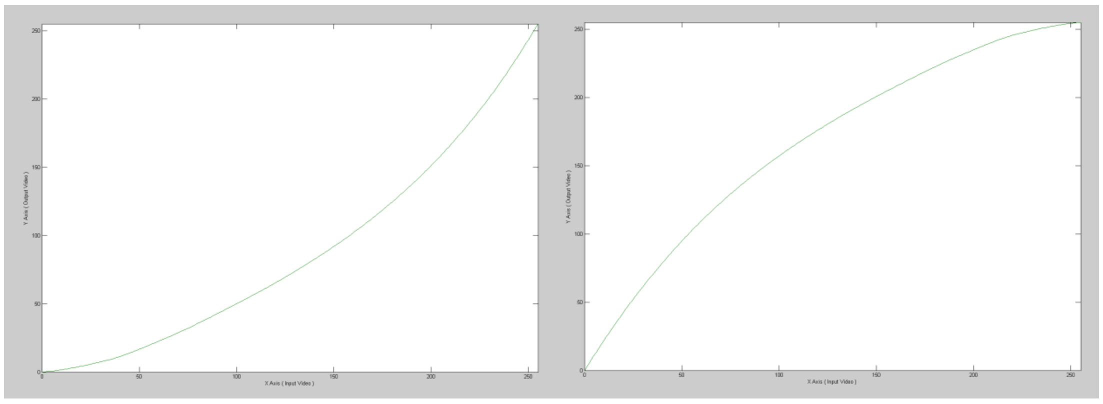


Figure 6.11 Brightness curves.

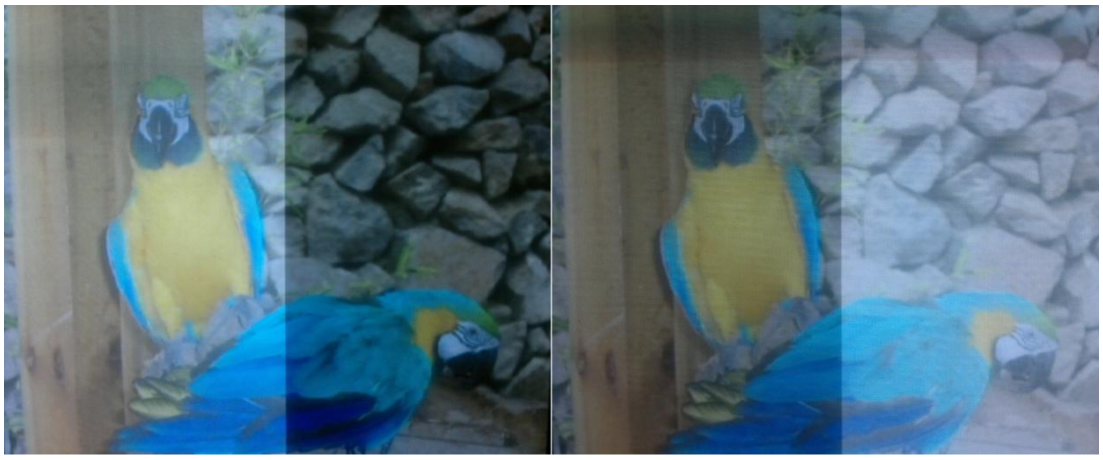


Figure 6.12 Result of applied brightness curves.

6.3 Noise Removal

6.3.1 Salt-and-Pepper Noise

Noisy pixels have abnormal values compared to neighboring pixels. Salt and pepper noise can be described as the bright pixels in dark areas and dark pixels in bright areas. Analog to digital convertor errors and dead pixels on a camera sensor can be a reason for formation of this type of noise. In order to remove salt and pepper noise mostly median filter is used. I added salt and pepper noise to the picture in a specific order in Figure 6.13 so that removal of noisy pixels can be observed clearly.

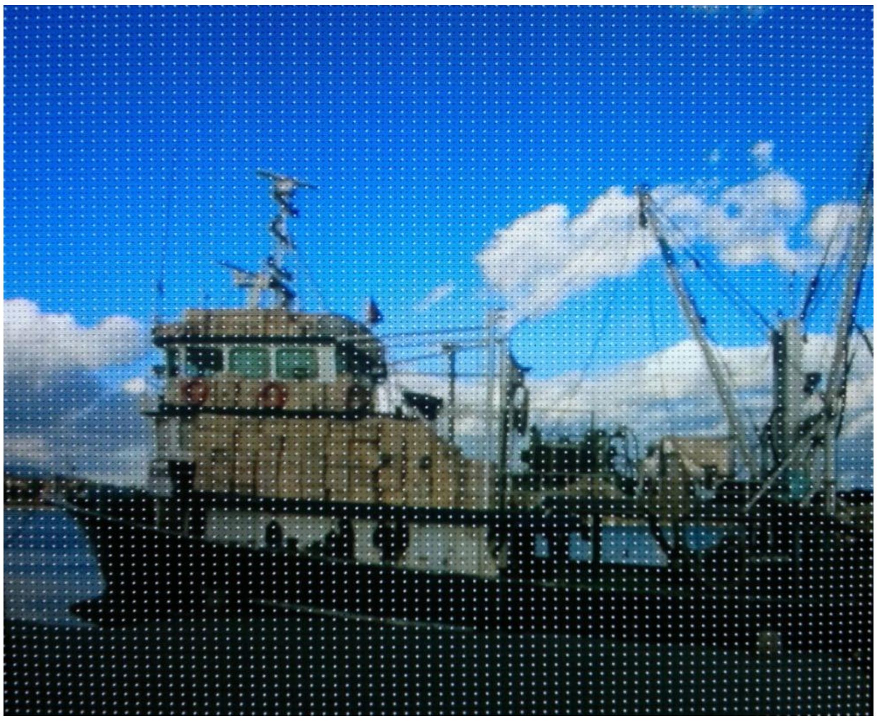


Figure 6.13 Noisy image.

6.3.2 Median Filtering

The median filtering algorithm need to carry out median value computations in real time. To determine median value the pixel values have to be sorted in ascending order and pixel value in the middle has to be selected. This complex calculation needs too much computational power. As a result the performance of software running on general purpose processors is not good. Even though hardware implementation of median filtering use many comparison blocks in FPGA, its speed and real time result is very satisfactory.

In this application two blocks were created. The first one is salt & pepper noise block, which adds noise to image in an already determined pattern. The second block is the noise removal block, which includes median filters. Block diagram of the application can be seen in Figure 6.14.

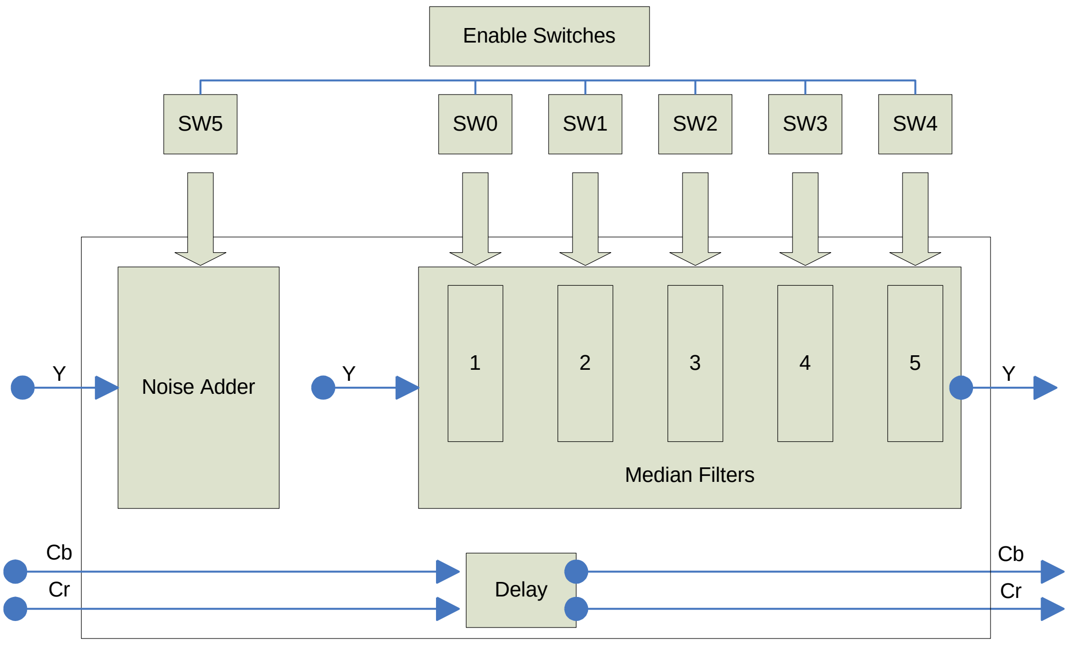


Figure 6.14 Block diagram of noise removal application.

To apply median filtering on the image, a pixel at an arbitrary coordinate and its neighbors must be accessible. The structure shown in Figure 6.15 allows us to access necessary pixels. 640 pixel length line buffers store 3 lines at a time and lets us access to all neighbors of pixels being processed. Instead of transmitting b2 pixel to monitor, median is calculated among a1, a2, a3, b1, b2, b3, c1, c2, c3 pixels and median is replaced by b2. As you see in this application no color process exists, just a time delay is applied to color signals as required.

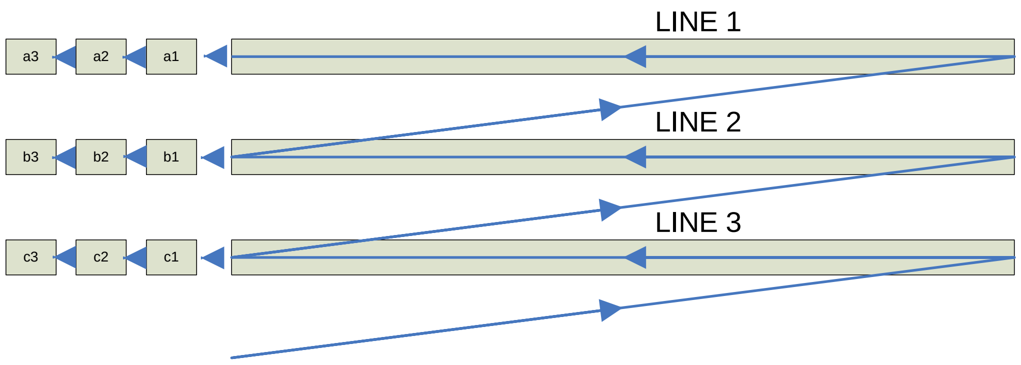


Figure 6.15 Line buffers.

Median filter smoothes image with the activation of SW 0. If SW 5 is activated, additional noise is applied to the input image. In this application, noise added image shown in Figure 6.13 is filtered. As you see in Figure 6.16 the resulted image contains almost no noise except noises on the edges.

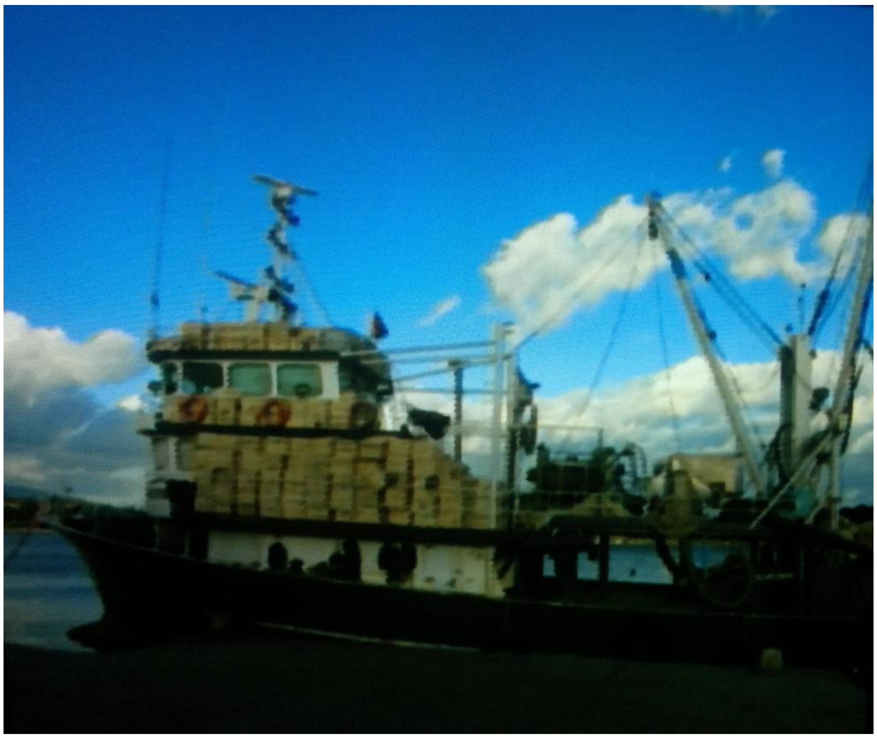


Figure 6.16 Median filtered image.

Noise removal block lets us filter image up to five times via activation of switches SW 0 to SW 4.The number of toggled switches determines the number of median filtering. Due to the fact that median filter cannot distinguish details from noise, details are lost with noise removal process. In Figure 6.17 median filtering is applied five times, hence almost all of the detail is lost. Five times filtering produced very severe smoothing.

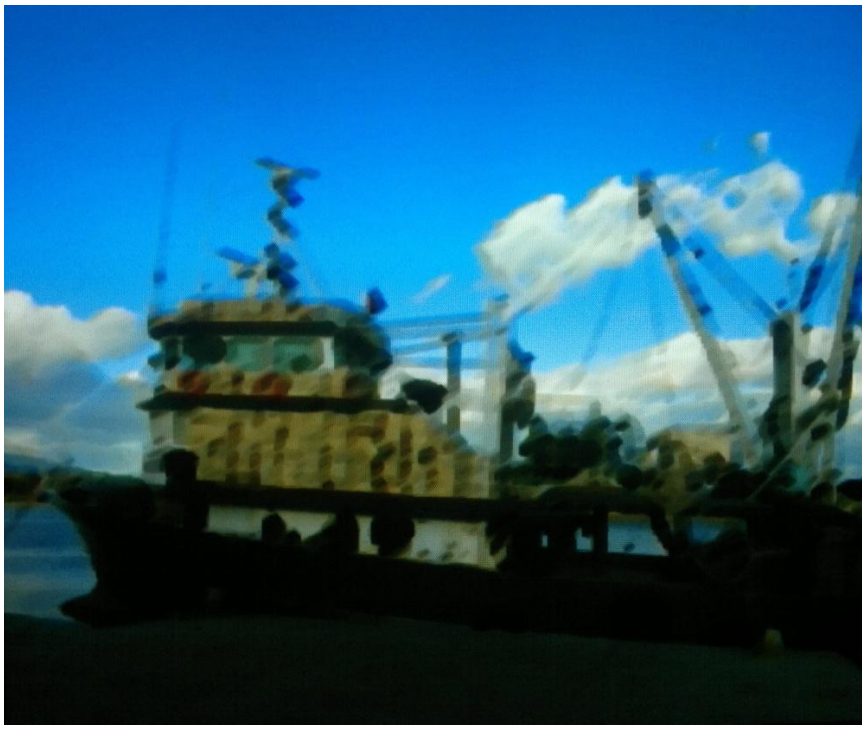


Figure 6.17 Median filtered image (five times).

6.4 Edge Detection

Edge detection can be used to extract important features of the objects in an image. Extracted features can be lines, letters and shapes, which are used in high level image processing applications, such as pattern recognition, motion estimation and texture analysis.

An edge is defined as the significant changes which cause discontinuities in intensity on the pixels value of an image. Edges usually exist on the boundaries of two objects in an image. Also the geometric shapes and other factors can cause the formation of edges. Edge detection process reveals these kinds of structures of the objects and removes other data. Edge information can be found by analyzing the neighborhood of all the pixels in the image. If the pixel has neighborhoods brighter or darker than pre-determined threshold more likely there is an edge at that pixel.

There are four types of edge,

Step edge: The image intensity abruptly changes from one value to one side of the discontinuity to a different value on the opposite side (Jain, Kasturi, & Schunck, 1995).

Ramp edge: A step edge where the intensity change is not instantaneous but occur over a finite distance (Jain et al., 1995).

Ridge edge: The image intensity abruptly changes value but then returns to the starting value within some short distance (Jain et al., 1995).

Roof edge: A ridge edge where the intensity change is not instantaneous but occur over a finite distance (Jain et al., 1995).

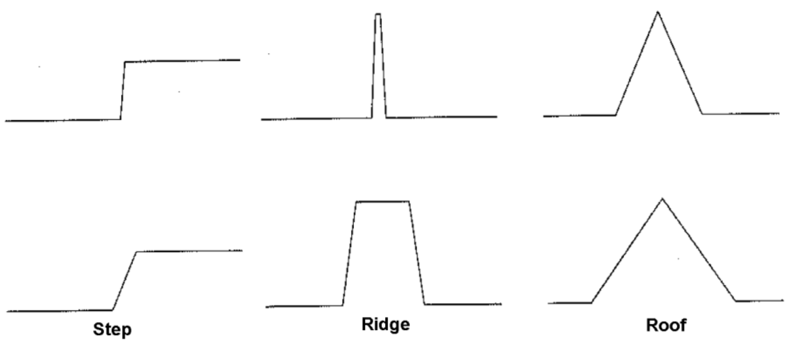


Figure 6.18 Step, Ridge and Roof Edge (Trucco, & Verri, 1998).

6.4.1 Application and Results

Digital images can be corrupted by noise. First of all, this noise should be removed. Even though noise reduction filters smoothes the edge, real edges do not disappear. In order to make detection of edges easy significant intensity changes should be strengthen by sharpening of the image. Lastly, image pixels don’t have just values of 0 or 255 for an 8 bit image. Every pixel has a nonzero value. To find edges the values below a threshold value must be converted to 0 and the others to 255.

Block diagram of edge detection application is shown in Figure 6.19. Edge detection block accepts 10 bits RGB signals and after RGB to gray scale conversion, grayscale signal is sent to three line buffer which is sufficient for windowing operation. Sobel operator calculation for horizontal and vertical edges is performed and they are separately shown on VGA monitor by changing switch configurations.

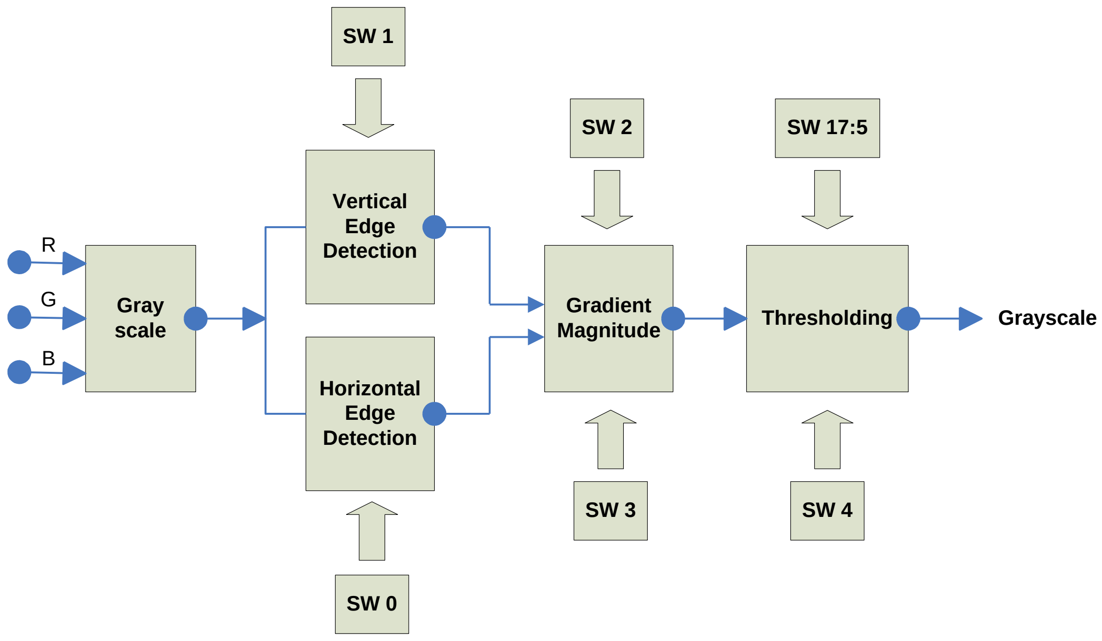


Figure 6.19 Block diagram of edge detection application.

SW 0 activates horizontal edge demonstration, SW 1 activates vertical edges demonstration, and SW 2 activates total Sobel edge detection. SW 3 makes edges brighter and increases the visibility of edges for horizontal and vertical edges and if it is desired SW 4 can be used to remove detected edges which are below a specified gray level. The gray level threshold can be set using 13 switches between SW 5 to SW 17. In this application image shown in Figure 6.20 was processed to find edges.

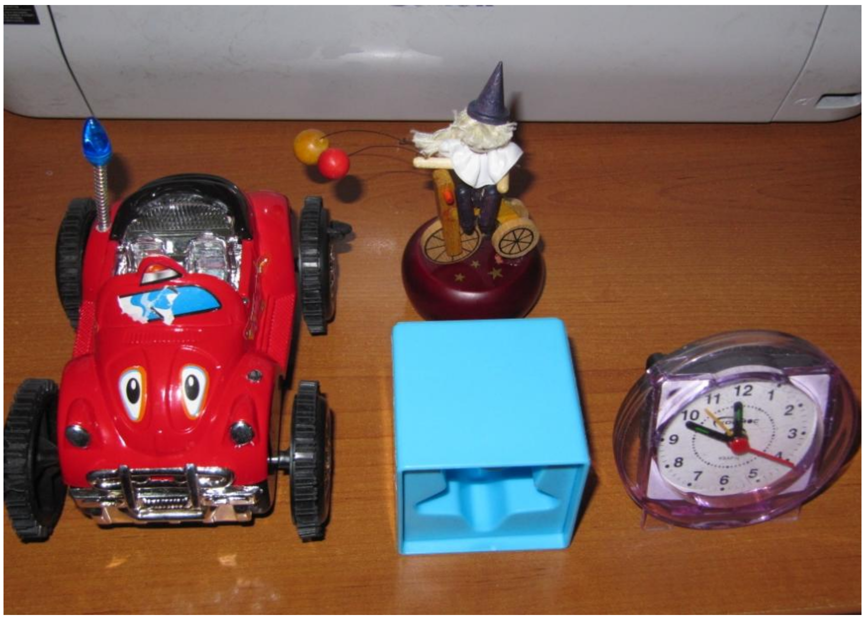


Figure 6.20 Original image.

In Figure 6.21 result of vertical edge detection can be seen. As you see real straight horizontal edges cannot be detected. In Figure 6.22 horizontal edge detection is present and almost no vertical edge is visible.

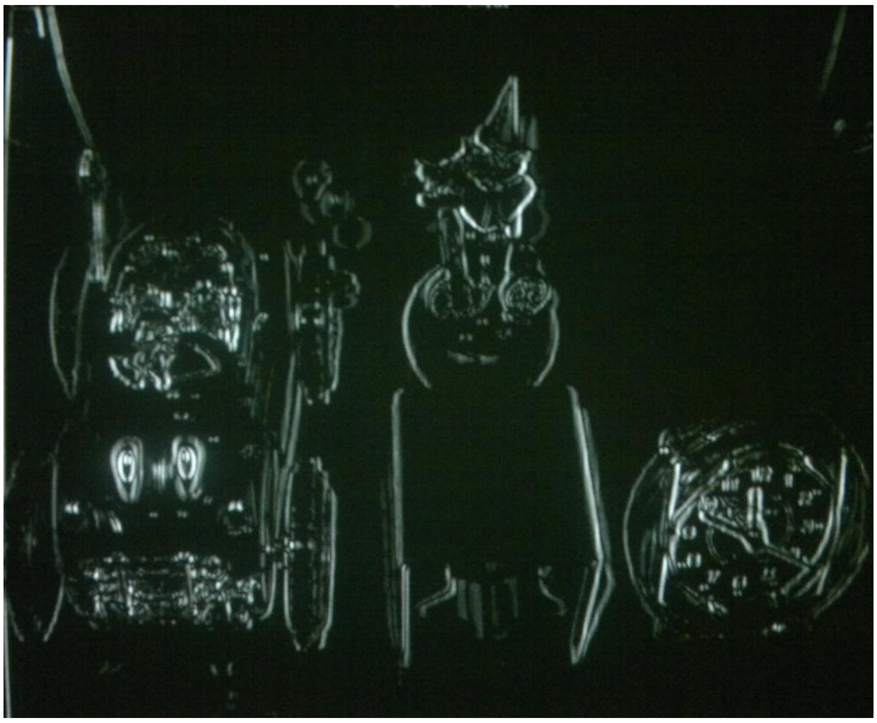


Figure 6.21 Result of vertical Sobel edge detector.

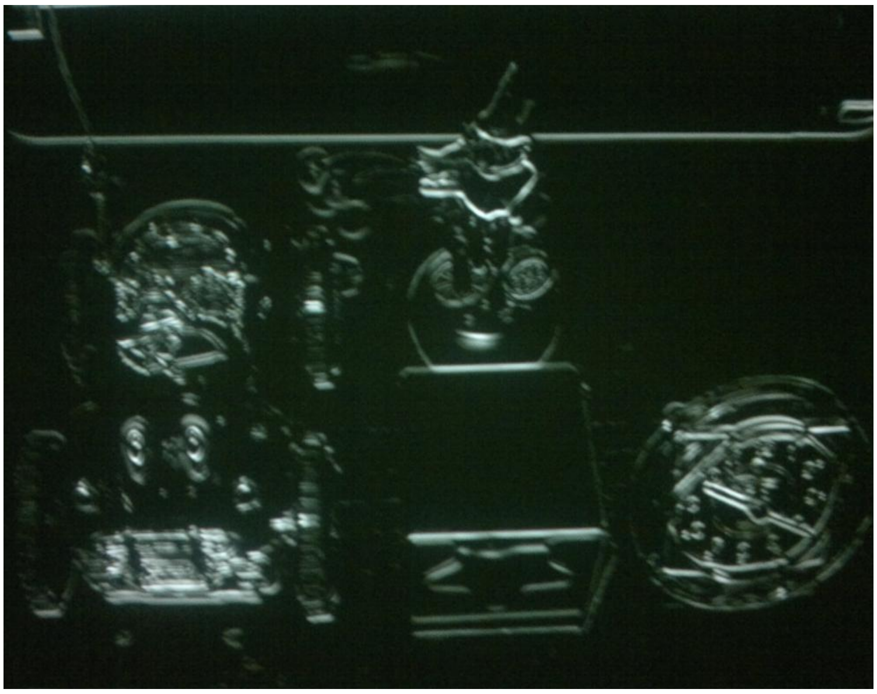


Figure 6.22 Result of horizontal Sobel edge detector.

In the produced image, brightness levels of edges show how much strong the edges are. After calculation of gradient magnitude of horizontal and vertical edges, the final image is obtained. In Figure 6.23 image was processed and the amplitude of edges was increased to make edges more visible.

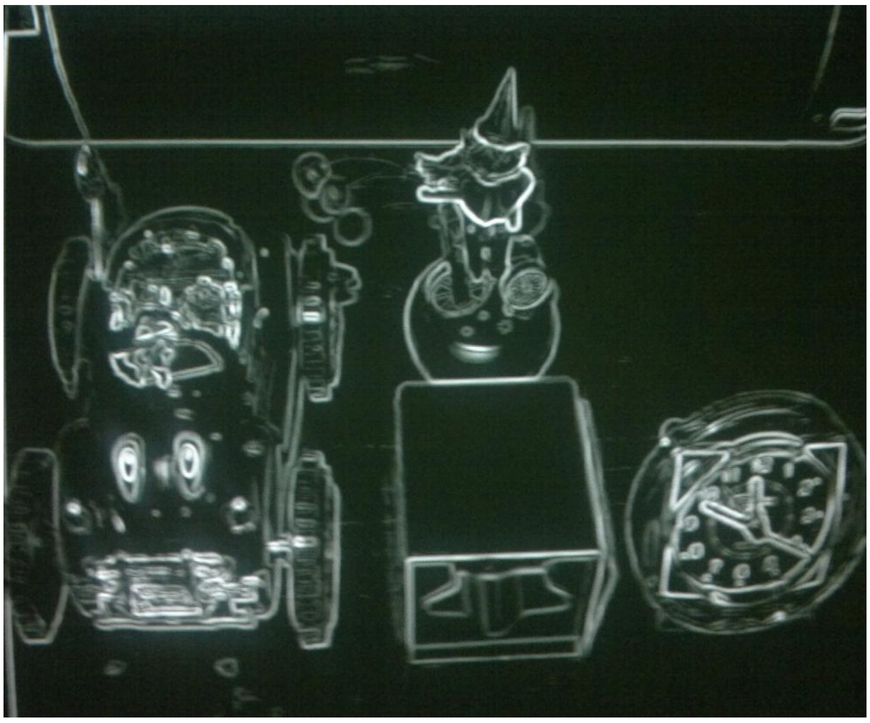


Figure 6.23 Result of Sobel edge detector.

In this thesis, just result of application on a still image was shown and compared with the original image. As seen from the results, horizontal and vertical edges were correctly detected. At the same time, real-time edge extraction on motion image was also tested and results seem satisfactory.

6.5 Motion Detection Based on Frame Differencing

Detection of movements is an important task on image processing. Many application such as video identification and video surveillance, need to obtain movements. In this study, frame-differencing method was used to detect motion.

Frame differencing is a common motion detection technique and usually a static camera is used to find moving objects. In this technique, consecutive frames are checked whether there is a difference between them. The difference for a still camera mostly represents the motion. Noise, rain or illumination changes can be considered as motion. In order to eliminate them and to detect real movements filtering and thresholding can be necessary.

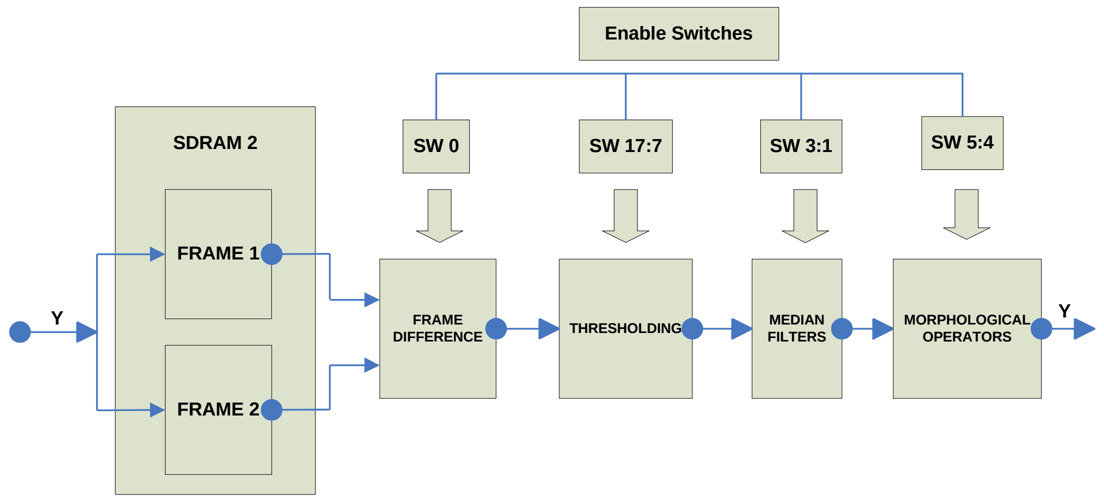


Figure 6.24 Block diagram of motion detection algorithm.

In Figure 6.24 block diagram of motion detection algorithm can be seen. To implement the algorithm second SDRAM in DE2-70 board is used. Consecutive grayscale frames are stored, and if SW 0 is active, absolute difference between frames are calculated, and if SW 0 is not active, original grayscale, raw frames are transmitted to VGA monitor. Without thresholding almost on every pixel motion detection occurs. After adjusting threshold value through SW 17 to SW 7 noise is removed and useful portion of motion is left. If threshold value is selected too high, moving object can be lost, hence an acceptable value should be selected. Then, if noise still exists, median filters can be used three times. Switches from SW 1 to SW 3 activate three median filters. Lastly, morphological dilation and erosion operators can be applied by activating SW 4 and SW 5. This final step makes moving objects more solid.

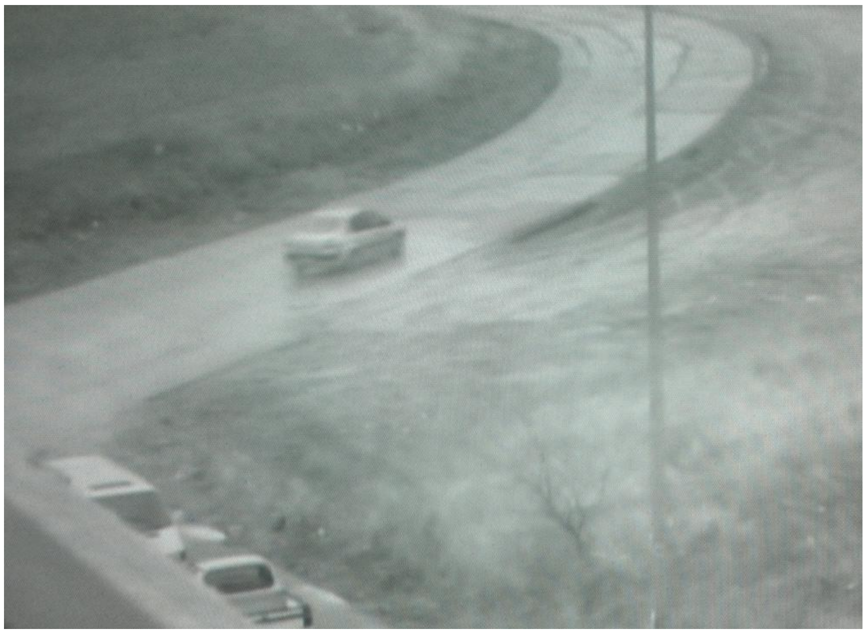


Figure 6.25 Moving object.

In Figure 6.25 a moving car is seen. After frame differencing, thresholding was applied and most of the noise was removed. Then, image was smoothed by median filters in order to remove the rest of the noise, for example moving small objects would disappear. Then, in order to make moving object to be seen clearly closing operator was applied. It filled the holes and makes moving object more visible.

The result of the motion detection algorithm can be seen in Figure 6.26. In favorable weather conditions, just moving object was detected but obviously, it is not a complete car shape.

After evaluating results of the algorithm, two drawbacks of the algorithm can be clearly seen. First of all, if object is not fast enough only boundaries of the moving object are detected in this process. Secondly, algorithm has faced difficulties in detecting small objects like human because in order to extract the most useful moving object, small objects have to be filtered out.

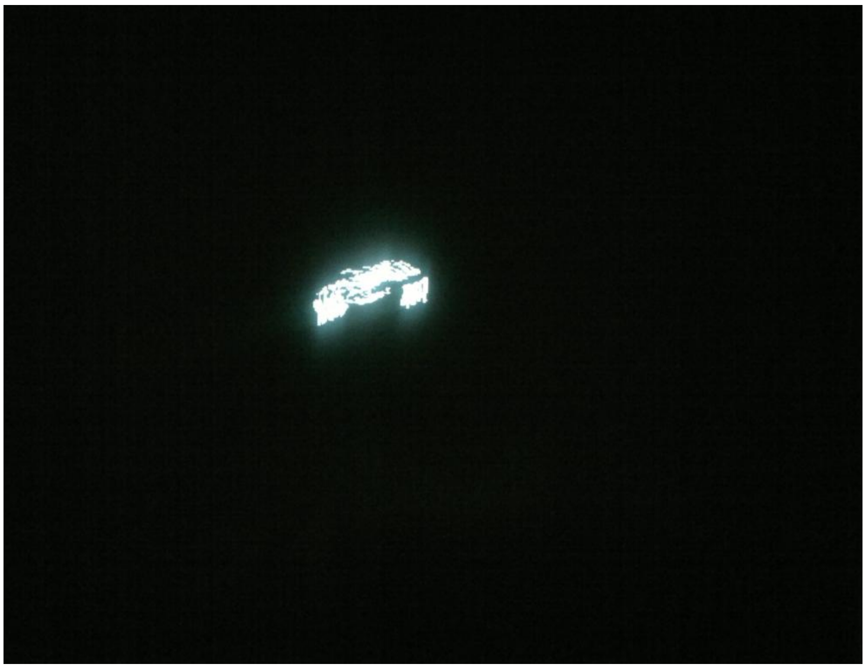


Figure 6.26 Detected moving object.

Another alternative algorithm for detection of movements is background subtraction method, which may achieve better results. In this method frame differencing is applied between original frame and pre-defined stored or averaged background, so the difference will be exactly moving objects.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

Image processing is a common practice in a great number of fields. Mostly it is performed on general-purpose processors, but if images and pixel bit sizes expand, software may face performance problems and become slow to execute image-processing applications. In this case advantages of FPGA come into prominence. Its processing speed is one of its most important characteristics. Because of its parallel nature, designers can create simultaneous parallel circuits and these different processing operations don’t use same resources. As a result they run all independently from each other. In this project, in order to utilize its advantages, an FPGA platform was used to perform image-processing algorithms.

Successfully implemented applications are, respectively, color detection and color based motion detection, edge detection, histogram monitoring, non linear contrast adjustments, noise removal and motion detection based on frame differencing. Applications did not go deeper into details hence they may not give exactly clear and desired results. The applications implemented on FPGA need to be developed because they are not tested in the field. After tests, many other requirements may arise.

Algorithms were emerged while knowledge about the subject of this was gained during the thesis study in time. This knowledge includes principles of FPGAs, VERILOG hardware description language, Quartus 9.1 and finally simple concepts and theory about image processing. Designed algorithms worked properly on the hardware for still and real time moving images. Obtaining successful results showed me that more complex new algorithms are feasible and must encourage us to improve the algorithms covered in the study. Many of the currently used image-processing application methods can be replaced by FPGA. In this way, the advantages of an FPGA can be benefited.

In consumer electronics, for instance in TV market, techniques related to image correction have been studied for years. Many companies have their own techniques and brands, but others purchase image processors from Integrated Circuit manufacturers. If FPGA based reprogrammable systems are used, TV manufacturers can create their own styles on color, sharpness, contrast, etc. by experiencing different applications. Result of experiments may even lead to develop their own video processors. After successful evaluation and testing, high volume producers can switch to ASICs for mass production.

TV demonstration of DE2-70 FPGA board supports only 640x480 resolution and NTSC standard input video. If I2C commands are set, PAL video also can be used as input instead of NTSC. Also in order to obtain higher resolutions in FPGA board, digital inputs like DVI can be used.

To improve performance, design may contain DSPs and FPGAs together. Complex mathematical calculations, which are considered as a burden for FPGAs, can be performed by DSPs and image processing algorithms, where speed is a necessity, can be implemented in FPGAs.

In the spatial domain some image processing operations such as image filtering may need high computational power. These operations may be easier in frequency domain. Fourier transformations can be investigated to obtain frequency representation of an image.

In conclusion, after sufficient working about FPGA design, many companies in different industries may develop and produce national products so that they reduce dependency on foreign countries.

REFERENCES

Altera (2008). *My first FPGA design tutorial.* Retrieved April 18, 2011, from http://www.altera.com/literature/tt/tt_my_first_fpga.pdf

Altium (2008). *FPGA design basics.* Retrieved April 12, 2011, from http://www.altium.com/community/trainingcenter/en/training-manuals.cfm

Auer, B. (February 12, 2010). *Nonlinear curve adjustments and histograms.* Retrieved August 26, 2011, from http://blog.epicedits.com/2010/02/12.

Bovik, A. (Ed.). (2000). *Handbook of image and video processing.* Texas: Academic Press.

Chikkali, P. S., & Prabhushetty, K. (2011). FPGA based Image Edge Detection and Segmentation. *International journal of advanced engineering sciences and technologies, 9* (2), 187-192.

Cho, J., Mirzaei, S., Oberg, J., & Kastner, R. (2009). FPGA based face detection system using haar classifiers. *FPGA '09 Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays,*103-112.

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (1996). *Hypermedia image processing reference.* Chichester: John Wiley & Sons, Inc

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

Gunay, H. (2010). *M.Sc. Thesis: Efficient FPGA Implementation of Image Enhancement using Video Streams.* Middle East Technical University.

Hong-Li, Z., Xian-Rong, W., & Hong-Yan, Z. (2011). Research on defect detection in rubber rings. *American Journal of Engineering and Technology Research, 11* (9), 746-750.

Intersil (2002). *Application note 9728.2, BT.656 Video interface for ICs.* Retrieved May 18, 2011, from http://www.intersil.com/data/an/an9728.pdf.

Jack, K. (2005). *Video demystified, a handbook for the digital engineer.* Massachusetts : Elsevier

Jain, R., Kasturi, R., & Schunck, B. G. (1995). *Machine vision*. USA: McGraw-Hill

Maxfield, C. (2004). *The design warrior's guide to FPGAs*. Oxford: Elseiver

Maxim (2002). *Application note 734, video basics*. Retrieved August 17, 2011, from http://www.maxim-ic.com/an734.

National Instruments (December 20, 2010). *Introduction to FPGA technology: top five benefits.* Retrieved May 17, 2011, from http://zone.ni.com/devzone/cda/tut/p/id/6984.

Nelson, A. E. (2000). M.Sc. Thesis: Implementation of image processing algorithms on FPGA hardware. Vanderbilt University.

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.

Shih, F.Y. (2009). *Image processing and mathematical morphology.* Florida: CRC Press.

Terasic (2009). *DE2-70 User manual*. Retrieved August 29, 2010 http://www.terasic.com

Trucco, E., & Verri, A. (1998). *Introductory techniques for 3-D computer vision.* New Jersey: Prentice Hall.

Vincent, O.R., & Folorunso, A. (2009). A Descriptive Algorithm for Sobel Image Edge Detection. *Proceedings of Informing Science & IT Education Conference.* Macon, GA.

Wernicke, A., Joost, R., Ehlert, U., Ciampa, M., et al. (2010). *GIMP User Manual*. Retrieved September 12, 2011, from http://docs.gimp.org/en/index.html.