DIGITAL SIGNAL PROCESSOR (DSP) ACCELERATED IMAGE ENHANCEMENT FOR INFRARED IMAGING SYSTEMS

A MASTER'S THESIS

in

Electrical & Electronics Engineering

Atılım University

by

NURPINAR AKDENİZ

JULY 2011

DIGITAL SIGNAL PROCESSOR (DSP) ACCELERATED IMAGE ENHANCEMENT FOR INFRARED IMAGING SYSTEMS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

NURPINAR AKDENİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

JULY 2011

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ibrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Elif Aydın

Head of Department

This is to certify that we have read the thesis "DIGITAL SIGNAL PROCESSOR (DSP) ACCELERATED IMAGE ENHANCEMENT FOR INFRARED IMAGING SYSTEMS" submitted by Nurpınar Akdeniz and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Hakan Tora

Supervisor

Examining Committee Members

Asst. Prof. Dr. Hakan Tora _____

Asst. Prof. Dr. Mehmet Efe Özbek _____

Asst. Prof. Dr. Atila Bostan _____

Date: July 22, 2011

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Nurpınar Akdeniz

Signature:

ABSTRACT

DIGITAL SIGNAL PROCESSOR (DSP) ACCELERATED

IMAGE ENHANCEMENT FOR

INFRARED IMAGING SYSTEMS

Akdeniz, Nurpinar

M.S., Electrical & Electronics Engineering Department

Supervisor: Asst. Prof. Dr. Hakan Tora

July 2011, 49 pages

This study evaluates the implementation of Balanced Contrast Limited Adaptive Histogram Equalization and Contrast Enhancement (BCLAHE+CE) for infrared images on an embedded platform. The developed software is integrated onto a BeagleBoard system and optimization techniques are investigated to meet the real-time streaming performance required by operator display. The distributed software is run on a general purpose processor and a digital signal processor (DSP), passing the computational intensive operations to the DSP. Performance analysis is done by different parameter sets and the accuracy of the new implementation is compared to theoretical results from MATLAB.

Keywords: Contrast Enhancement, histogram equalization, infrared imaging, DSP, ARM, SoC, OMAP.

ÖZ

KIZILÖTESİ GÖRÜNTÜLEME SİSTEMLERİ İÇİN SAYISAL İŞARET İŞLEMCİ (Sİİ) İLE HIZLANDIRILMIŞ GÖRÜNTÜ İYİLEŞTİRME

Akdeniz, Nurpınar

Yüksek Lisans, Elektrik - Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Prof. Dr. HakanTora

Temmuz 2011, 49 sayfa

Bu çalışma, kızılötesi görüntüler için Dengeli Karşıtlık Uyarlamalı Histogram Denkleme ve Yeğinlik Pekiştirme tekniğinin gömülü platform üzerine uyarlanmasını ele alır. Geliştirilen yazılım BeagleBoard üzerine entegre edilerek, operatör ekranı için gerekli olan performansı sağlayacak iyileştirme teknikleri araştırılmıştır. Genel amaçlı işlemci ve sayısal işaret işlemcisi (Sİİ) arasında paylaştırılan dağıtık yazılım, hesaplamanın yoğun olduğu işlemlerde Sİİ kaynaklarını kullanır. Performans analizleri, farklı parametreler için yapılmış ve tekniğin doğruluğu MATLAB çıktıları ile karşılaştırılmıştır.

Anahtar Kelimeler: Karşıtlık pekiştirme, yeğinlik pekiştirme, kızılötesi görüntüleme, DSP, ARM, SoC, OMAP.

To My Parents and David

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Asst. Prof. Dr. Hakan Tora for his guidance and insight throughout the research. I would also like to express my sincere gratitude to Asst. Prof. Dr. Mehmet Efe Özbek for the discussions and his guidance on embedded platforms.

Special thanks to my parents Süleyman and Melek for their continuous support during this period.

Very special thanks to David for his comments and tips on software development. Also for helping me in developing much of the material presented here.

Big thanks to Şafak and Fatma for reviewing my thesis and their comments; Ali Onur for discussions on problems in image enhancement; Kerem for supplying me the BeagleBoard and its components that are necessary for this thesis.

Special thanks to Radio Hacettepe for motivating me by their music throughout the nights I worked hard.

And finally a special dedication for my dog Köpük for giving me her unconditional and endless love especially whenever I needed the positive energy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHE	-	Adaptive Histogram Equalization
API	-	Application Program Interface
ARM	-	Advanced RISC Machine
BCCE	-	BCLAHE + Contrast Enhancement
CLAHE	-	Contrast Limited Adaptive Histogram Equalization
BCLAHE	-	Balanced CLAHE
DSP	-	Digital Signal Processor
FPGA	-	Field Programmable Gate Array
FPS	-	Frames per second
GPP	-	General Purpose Processor
HE	-	Histogram equalization
IR	-	Infra-red
MMC	-	Multimedia Card
MPU	-	Microprocessors Unit
OMAP	-	Open Multimedia Application Processor
OS	-	Operating System
RISC	-	Reduced Instruction Set Computer
RPC	-	Remote Procedure Call
RTOS	-	Real-time OS
SBC	-	Single Board Computer
SD	-	Secure Digital

SDRAM	-	Synchronous Dual Access Memory
SoC	-	System on Chips
TI	-	Texas Instruments
UMBM	-	Unsharp-Masking-Based Method
VLIW	-	Very Long Instruction Word

CHAPTER 1

INTRODUCTION

Thermal infrared cameras are an important component of military systems. One of the most prominent usage areas in the military is thermal scope and night vision goggles. For such systems, it is important to develop efficient image processing algorithms for the low contrast and high dynamic range characteristics of infrared images. The sophisticated image enhancement techniques are developed on various platforms. Among these platforms, FPGAs are the foremost technology for performance critical applications at a cost of development complexity and their higher prices. A hybrid FPGA+DSP system gives further development flexibility. DSP systems are specialized for real time signal processing and their performance is limited by CPU speed. To increase the execution performance, a convenient solution is using a DSP and a General Purpose Processor (GPP) depending on the specific application. This study investigates the real time performance of a visualization technique named BCLAHE+CE (BCCE) [1]. This technique is chosen for its robustness against the existence of horizon effect, saturation of display by the existence of very hot objects which are often encountered in IR images. An unsharp masking technique with BCLAHE is also implemented to compare the performance with BCCE. The developed software is integrated onto a low cost, high performance OMAP3530 platform that consists of ARM and DSP processors. Performance of the system is improved by optimization techniques.

1.1. Scope of the Thesis

This study evaluates real-time implementation of IR image enhancement algorithm on DSP+GPP system. The performance evaluation is based on the Frames Per Second (FPS) that is sufficient for an operator display. The methodology of distributed software is investigated for DSP+GPP embedded platform. Software was

tested on *BeagleBoard* which is a low cost, high performance system-on-chip (SoC) based on low power OMAP 3530 platform.

1.2. Organization of the Thesis

Chapter 2 presents background information on infrared technology, image enhancement algorithms, an overview of hardware and software technology used on the development and experiment systems as well as a look at present image enhancement implementations for embedded systems. Chapter 3 explains how the development and experiment environment is set up and gives detailed information on the proposed software design. The optimization techniques are also mentioned in this chapter. The results and performance of the overall system are given in Chapter 4. Finally, Chapter 5 includes the conclusion and remarks are stated on the performance of the system.

CHAPTER 2

BACKGROUND

2.1. Infra-red Technology

Thermal imaging is the use of an infrared camera to "see" thermal energy emitted from an object. Infrared energy cannot be detected by the human eye because the energy of the photons is too low to stimulate the photoreceptors in the eye. Also, thermal IR light does not get to the retina, because it is highly absorbed by the eye's lens and fluids. Infrared allows us to see what our eyes cannot - invisible heat radiation emitted by all objects regardless of lighting conditions. Thermal imaging cameras produce images of invisible infrared or "heat" radiation. Based on temperature differences between objects, thermal imaging produces a clear image [2]. Recent infrared imaging systems have sensor sensitivity as high as 16 bit. In other words, every measured intensity can hold a range of values between 0 and 65535. The term "intensity" refers to the gray level that ranges from black to white and is the only attribute of an IR image. The high dynamic range of IR images makes it difficult to get the important details in the original image when simple linear mapping is used for a typical IR display when it supports 255 gray levels. Knowing that the visual perception of human eye is between about 64 gray levels at a time [3], it is necessary to apply visualization techniques while preserving the important details. The software that accomplishes this has high computational load depending on the resolution of image, and the choice of techniques.

2.2. Image Enhancement

Image enhancement is the process of manipulating an image so that the result is more suitable than the original for a specific application [4]. The specific application of this study is visual interpretation of infrared images. Image enhancement is done by

intensity transformations, spatial filtering or filtering in the frequency domain. This study uses histogram equalization to increase the contrast between the target and background, and spatial filtering for image sharpening.

2.2.1. Histogram Equalization

A histogram tells about the statistics of the image. For an image of size N by M and with the intensities between [0, L-1], normalized histogram is,

$$p(r_k) = n_k/MN \quad k = 0, 1, 2, .., L\text{-}1. \tag{2-1}$$

where r_k is the k^{th} intensity value and n_k is the number of pixels in the image with intensity r_k. In other words, $p(r_k)$ is an estimate of the probability of occurrence of intensity level r_k in an image. Histograms may be viewed graphically simply as plots of $p(r_k)$ versus r_k.

Histogram statistics can be used to enhance the contrast of the image. Histogram equalization (HE) is a method that expands the gray levels of more pixels while compressing the gray levels of fewer pixels. This is done by mapping the histogram of the original image to the desired histogram by using a histogram transformation function. Fundamental transformation functions for desired histogram shapes are listed in [Table 1]. This study is based on the uniform distribution model.

Table 1: Histogram modification transfer functions [5]

Output Probability Density Model		Transfer Function[a]
Uniform	$p_g(g) = \frac{1}{g_{max} - g_{min}} \quad g_{min} \le g \le g_{max}$	$g = (g_{max} - g_{min})P_f(f) + g_{min}$
Exponential	$p_g(g) = \alpha \exp\{-\alpha(g - g_{min})\} \quad g \le g_{min}$	$g = g_{min} - \frac{1}{\alpha}\ln\{1 - P_f(f)\}$
Rayleigh	$p_g(g) = \frac{g - g_{min}}{\alpha^2}\exp\left\{-\frac{(g - g_{min})^2}{2\alpha^2}\right\} \quad g \ge g_{min}$	$g = g_{min} + \left[2\alpha^2 \ln\left\{\frac{1}{1 - P_f(f)}\right\}\right]^{1/2}$
Hyperbolic (Cube root)	$p_g(g) = \frac{1}{3}\frac{g^{-2/3}}{g_{max}^{1/3} - g_{min}^{1/3}}$	$g = \left[g_{max}^{1/3} - g_{min}^{1/3}[P_f(f)] + g_{max}^{1/3}\right]^3$
Hyperbolic (Logarithmic)	$p_g(g) = \frac{1}{g[\ln\{g_{max}\} - \ln\{g_{min}\}]}$	$g = g_{min}\left(\frac{g_{max}}{g_{min}}\right)^{P_f(f)}$

Output intensity values should never be mapped to more than one value and this is guaranteed by the monotonically increasing characteristic of transformation functions.

When the transformation function is applied to the whole image, the technique is called *Global Histogram Equalization*. Global HE is the simplest method and it has the least computational complexity. With intensity levels, r, ranging between [0, L-1] where 0 is black and L-1 is white, the output intensity level, s, is of the form

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \quad \text{k=0, 1, 2, …, L-1} \qquad (2\text{-}2)$$

Knowing that T(r) is a monotonically increasing function, it is possible to define the inverse mapping function $s=T^{-1}(r)$. All of the pixels in the input image whose gray levels lie within the range [r, r+Δr] will have their pixel intensities reassigned such that they lie within the range of [s, s+Δs]. As shown in Figure 1, the transformation function is such that the surface area under the input histogram curve, p_f, from r to r+Δr is the same as the surface area under the output histogram curve, p_g, from s to s+Δs. The plot of the output histogram p_g is rotated by 90 degrees [6].

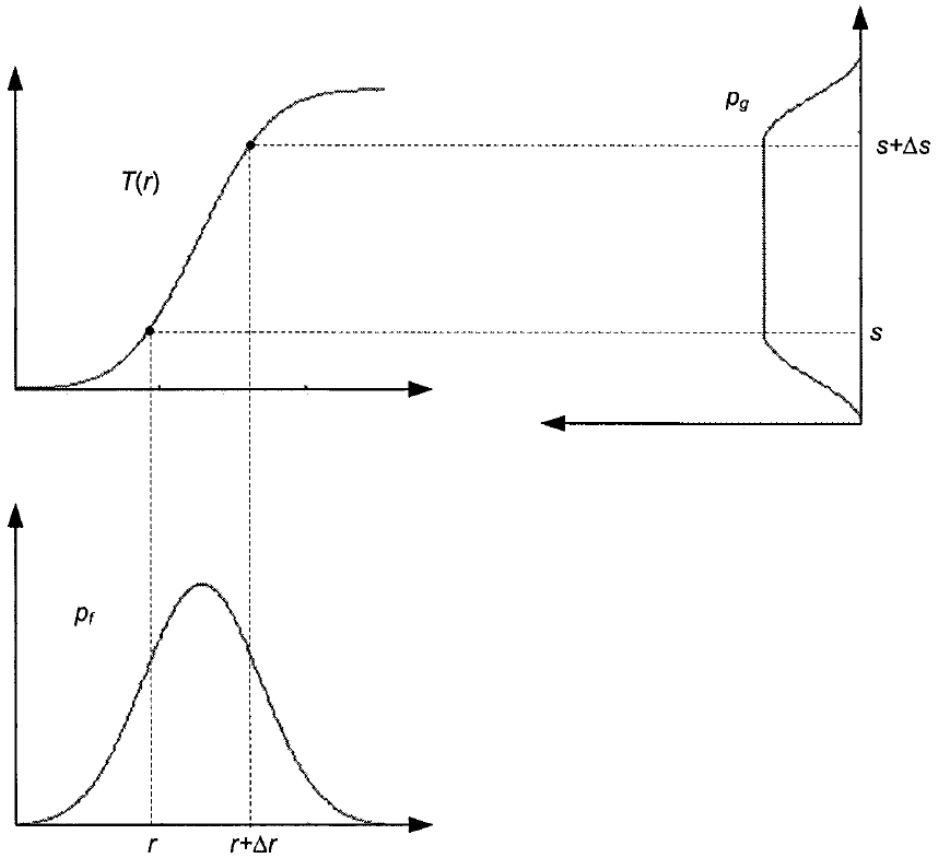
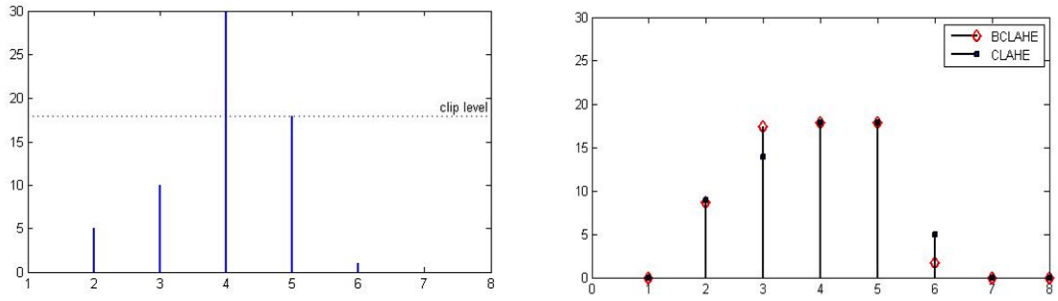


Figure 1: Histogram equalization in action.

Depending on the type of image, the shape of the histogram of the input image may consist of a large peak that might be caused by noise, background data or low contrast of the image. For such situations, the application of global histogram equalization would increase the unwanted noise in the output image or hide the important details in the input image because small areas have negligible contribution in the transformation curve. To overcome these drawbacks, adaptive histogram equalization (AHE) methods were invented independently by Ketcham et al. [7], Hummel [8] and Pizer [9]. Generally, output intensity values are adaptively calculated using the transformation curves that are obtained by sliding window over the image. In another AHE method the input image is subdivided into partially overlapping or non-overlapping tiles and equalization curves are obtained for each tile. The AHE method not only enhances image contrast but also increases noise. Limiting the contrast enhancement will limit the noise in the image signal. Limitation of contrast enhancement is known as Contrast Limited Adaptive Histogram Equalization (CLAHE) and can be used to restrict the slope of the mapping function. In other words, limiting the slope of the histogram mapping function is equivalent to clipping the height of the histogram [10]. Thus the maximum slope is equal to the clip limit. A slope of 1 involves no enhancement, and higher slopes give increasingly higher intensity. In both adaptive and global histogram equalization methods, the output histogram value is proportional to the input histogram value. This is more visible when we look at the mapping curve. When large flat areas occur in the image, the slope of the mapping function for these regions will be high. As a consequence, most of the output levels will be used to map large flat areas and the small relevant areas are less affected. This is a drawback for IR applications where low temperature variations are often encountered. Balanced CLAHE (BCLAHE) overcomes this problem by suppressing mapping of large flat areas while improving the less frequent regions.

BCLAHE is the histogram equalization method used in this study. This method is the same as traditional CLAHE except that the excess over the clip level is distributed proportionally over the bins under clip level. The application of CLAHE and BCLAHE is shown in Figure 2 for 3 bit image to simplify it. The histogram of the

input image is clipped at level 18 and the excess amount is distributed equally for CLAHE and proportionally for BCLAHE. BCLAHE is successful for histogram shapes with very high peaks where the clipping is done at 1% of the highest peak in the histogram.



(a) (b)

Figure 2: Application of CLAHE and BCLAHE (b) on histogram (a).

To apply BCLAHE locally, the original image is subdivided into non-overlapping rectangles and BCLAHE is applied to each tile. Mapping curves are calculated for each tile. It is reasonable that the pixels near the pixel whose mapping is being calculated should affect the mapping more than those farther away. Therefore, for every pixel in the original image, the new value is calculated by multiplying the mapping values for that pixel's gray value with the inverse of the distance of the pixel to each tile's center allowing the pixels to be affected more by pixels at a closer distance. The contribution of the distance is known as weighted AHE in the literature. Distance weight for pixel i and tile n is calculated by,

$$w(n,i) = \frac{1/d_{n,i}}{\sum_{n}^{NTiles} 1/d_{n,i}}, \qquad (2\text{-}3)$$

and weights are applied to output image using the equation,

$$p^{NEW}(i) = \sum_{n} w(n,i) * T_n(p_i) . \qquad (2\text{-}4)$$

Since we want all the details in the image, linear scaling is used on the equalized histogram to map the input intensity range [0-65535] to the output intensity range [0-255] as depicted in Figure 3

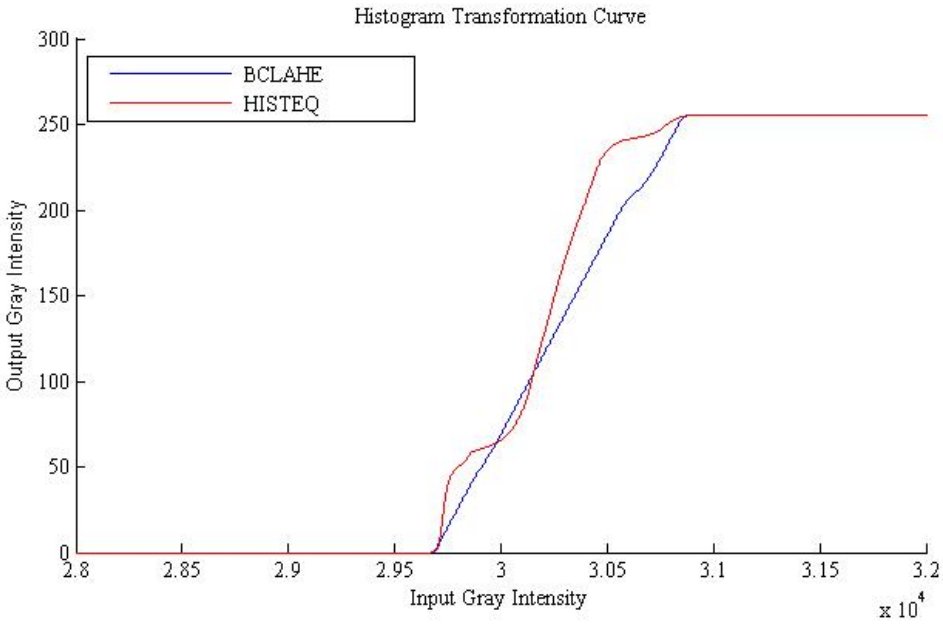


Figure 3: Histogram transformation curve for BCLAHE and histeq.

To comply with the proposed method, BCLAHE is applied locally in this study.

Application of histogram equalization methods to IR images are compared in Figure 4. Linear scaling in (a) is the simplest method to obtain 16 bit to 8 bit scaled image. The contrast is improved by scaling pixel values that are between minimum and maximum in the original image to 0 and 255. The image in (b) is the application of MATLAB's *histeq* method to the image in (a). *Histeq* uses global histogram equalization based on the uniform distribution model. When applied to original 16 bit image, the output image was far from being enhanced. The output of CLAHE given in (c) is produced by MATLAB's *adapthisteq* method using the input image in (a). In this method it is seen that contrast is increased and more details are visible in this method. But it can also be seen that this method also increased the noise. In picture

(d), the output of BCLAHE is given. The original image is subdivided into 3x3 rectangular regions and BCLAHE is applied. Compared to (c), details are preserved while noise is suppressed.

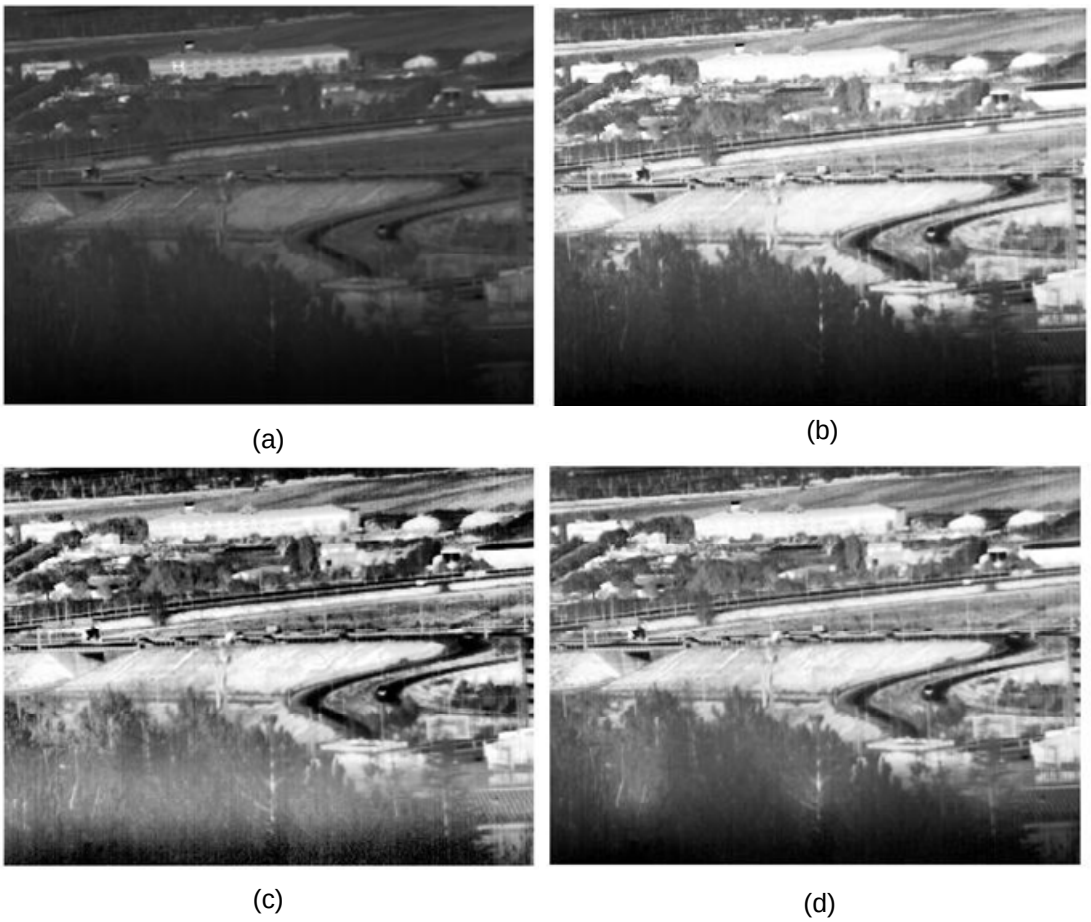


Figure 4: Different histogram equalization methods applied to IR image.

2.2.2. Contrast Enhancement

Contrast enhancement is a technique to sharpen images, used widely in image processing. Image sharpening is the subtraction of an unsharp (smoothed) version of the image from the original image and adding the result to the original image. The general formulation is

$$f^{out}(x,y) = f^{in}(x,y) + \alpha[f^{in}(x,y) - f_M^{in}(x,y)] \qquad (2\text{-}5)$$

The unsharp image $f_M^{in}(x,y)$ is obtained by lowpass filtering the input image. The value of α controls the contribution of sharpening. For α=1, the process is called

unsharp masking, when α>1 it is referred to as highboost filtering. Choosing α<1 de-emphasizes the contribution of the unsharp mask.

Lowpass filters diminish high spatial frequency components, thereby reducing the visual effect of edges as the surrounding pixel intensities blur together. Highpass filters have the opposite effect and attenuate low-frequency components, so regions in an image of constant brightness are mapped to zero and the fine details of an image are emphasized. 2D filtering an image with a lowpass kernel is the 2-dimensional convolution operation.

$$f_M^{in}(x,y) = f(x,y) * h(x,y) \tag{2-6}$$

A common lowpass kernel $h_{LP}(x,y)$ is,

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2-7}$$

which replaces the center pixel value with the average of it and the adjacent pixels.

A highpass filter is obtained from a given lowpass filter using the equation

$$h_{HP}(x,y) = 1 - h_{LP}(x,y) \tag{2-8}$$

Using (2-7) and (2-8), convolution kernel, $h_{HP}(x,y)$, becomes,

$$h_{HP} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \frac{1}{9}\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

BCCE defines the input-output relationship of the image as

$$f^{out}(x,y) = p(f^{in}(x,y)).\left[\frac{f^{in}(x,y)}{f_M^{in}(x,y)}\right]^{\alpha} \tag{2-9}$$

In the first term, p is the transfer function obtained by BCLAHE while second term is the local contrast of the image and α is the contrast enhancement factor.

This study evaluates the performance of two methods corresponding to the equations given in (2-5) and (2-9).

The input image given in Figure 4.d is enhanced by the two methods separately and the output images are given in Figure 5. (a) shows the result of BCCE is applied with α=20 and (b) shows the result of Unsharp Masking Based Method (UMBM) with α=128. Edges in image (b) are more sharpened comparing to (a) while the noise is attenuated more in (a) compared to (b).

Figure 5: Application of BCCE (a), and UMBM (b)

2.3. BeagleBoard

BeagleBoard, is a single board computer (SBC) equipped with Texas Instrument (TI)'s OMAP3530 processor. With the on board peripherals, shown in Figure 6 it gives us an easy to develop and a low cost embedded platform solution. The cost of BeagleBoard Rev B7, which is used in the thesis, is 125$ at the time of writing.

The heart of the BeagleBoard, the OMAP3530 processor, comes in a Package on Package (POP) where the memory, NAND and SDRAM, are mounted on top of the

OMAP3530. POP memory provides NAND (256MB) and SDRAM (128MB) and are the only memory components of BeagleBoard.

However additional memory can be installed on BeagleBoard by means of an SD/MMC slot, a high speed USB OTG port or a powered USB hub to drive a USB memory stick or hard drive. The SD/MMC connector is provided as a means for expansion and can support devices such as: WiFi cards, cameras, Bluetooth cards, GPS modules, SD memory cards, MMC memory cards, SDIO cards, MMCMobile cards and miniSD cards. MMC or SD cards can also be used as boot media.

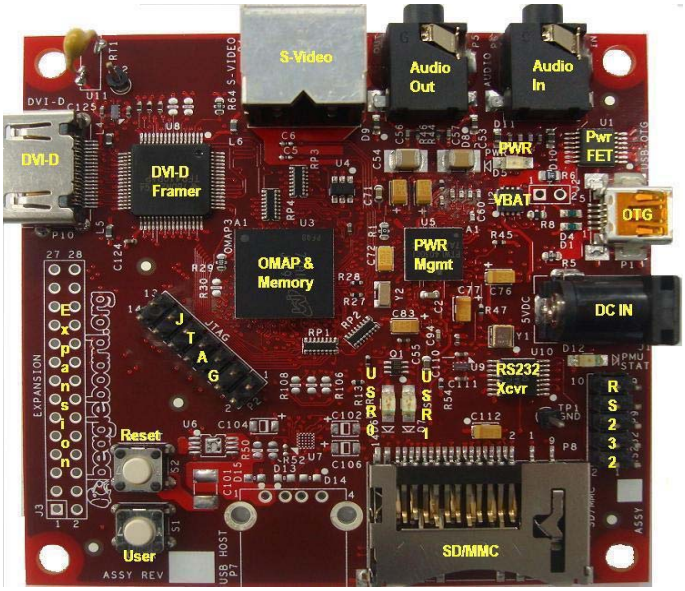


Figure 6: Top view of BeagleBoard and its basic components [11].

The power required by BeagleBoard is supplied by a 5 Volt DC supply or, by connecting the USB OTG port to the USB port of a PC, or by connecting to a powered USB hub. Support for RS232 via UART3 is provided by a 10 pin header on the BeagleBoard for access to an onboard RS232 transceiver. The size of the BeagleBoard is 7.6 cm x 7.9 cm.

2.4. OMAP 3530 Microprocessor

The OMAP3530 is a high-performance multimedia application device based on the enhanced OMAP™ 3 architecture. A high level block diagram of the OMAP3530 is

shown in Figure 7. The OMAP3530 supports high-level operating systems (OSs), such as Windows CE and Linux. This OMAP3530 device includes state-of-the-art power-management techniques required for high-performance low power products designed for portable and embedded devices.

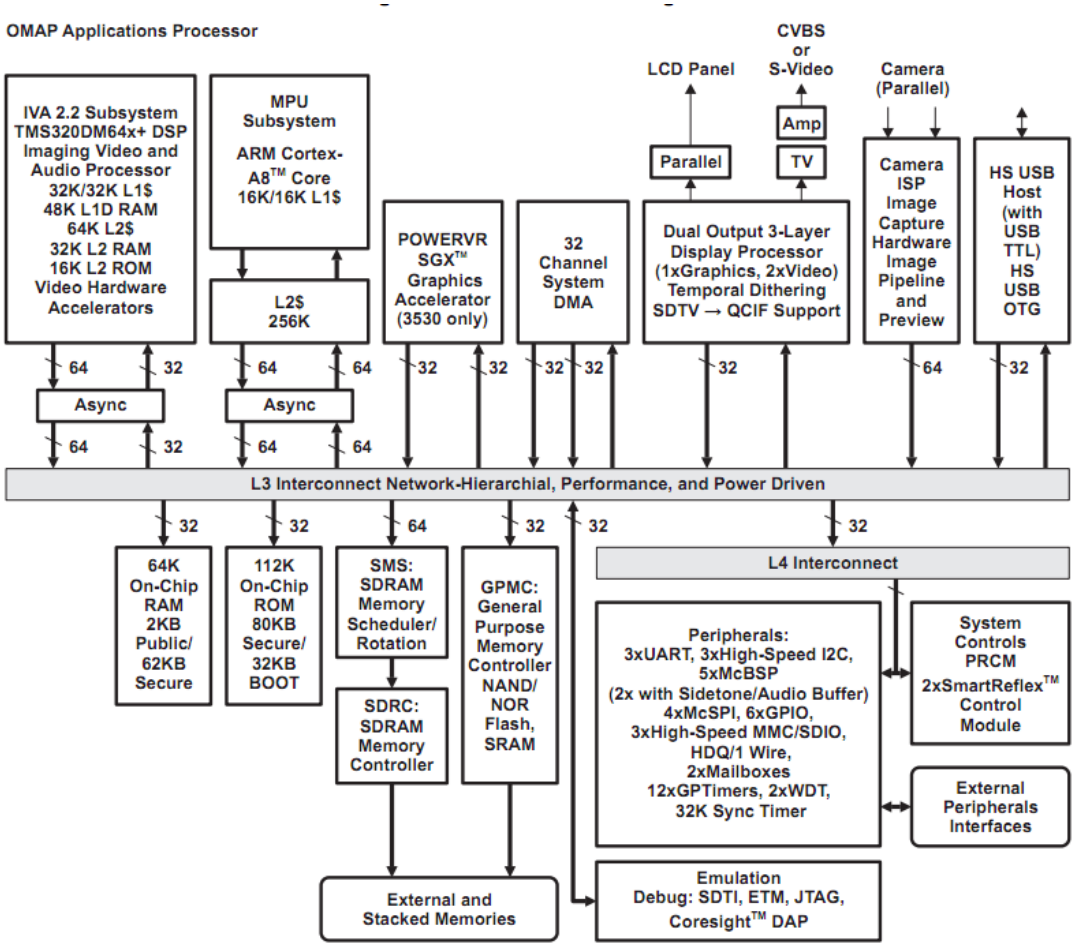


Figure 7: OMAP3530 functional diagram [12].

The OMAP3530 is a dual-core microprocessor system. The subsystems that compose the device are:

- ARM Cortex-A8™ Microprocessor Unit (MPU) (Up to 720 MHz),
- TI C64x+ DSP (Up to 520 MHz),
- POWERVR SGX™ subsystem for 3D graphics acceleration,
- Level 3 (L3) and level 4 (L4) interconnects for high speed data transfer with memory controllers (either external or on-chip ones).

The active power consumption is reduced due to automatic control of the operating voltage of individual modules and by supporting the SmartReflex™ technology.

2.5. Cortex-A8 Processor

The Cortex-A8 processor is a microprocessor designed by ARM Holdings based on the ARMv7-A, a 32-bit Reduced Instruction Set Computer (RISC) Instruction Set Architecture (ISA).

The Cortex-A8 is a low-power, high-performance single core microprocessor designed for portable devices [13] with the following main features [14]:

- Pipeline for executing ARM integer instructions;
- NEON pipeline for executing Advanced SIMD and VFP (optional) instruction sets;
- Dynamic branch prediction with branch target address cache, global history buffer, and 8-entry return stack;
- Memory Management Unit (MMU) and separate instruction and data Translation Look-aside Buffers (TLBs) of 32 entries each;
- Static and dynamic power management;
- Level 1 instruction and data caches of 16KB or 32KB configurable size;
- Level 2 cache of 0KB, 128KB through 1MB configurable size.

The ARM® NEON™ technology [15] is a 128 bit SIMD architecture extension. SIMD is a class of parallel execution that exploits parallel operations on data. This technology provide a significant acceleration in the performances of multimedia and signal processing algorithms such as video encoding/decoding, 2D/3D graphics, gaming, audio and speech processing, and image processing. Tight coupling to the ARM processor provides a single instruction stream and a unified view of memory, presenting a single development platform target with a simpler tool flow. NEON can be integrated into a project by using royalty free OpenMAX DL library, using vectorizing compilers, using C intrinsics, or using Assembly level code. OpenMAX DL supports functions such as FIR, IIR, FFT, Dot Product, and Color space conversion.

In this project, the code dedicated to ARM is investigated for parallelism and the structure of the algorithm is mostly based on the if-then-else operation which is not suitable for parallelism in this context. For this reason, the developed software doesn't make use of NEON technology in this study.

2.6. TMS320C64x+ DSP

TMS320 family consists of fixed-point, floating-point and multiprocessor DSPs. From TMS320 DSP family, TMS320C6000 DSPs have a performance of 8000 million instructions per second (MIPS). With the advanced VLIW CPU, C6000 series DSPs include six arithmetic logic units and two multipliers. With eight functional units, it executes eight instructions per cycle. The C64x+ DSP is a fixed-point DSP from the TM320CC6000 family. Each multiplier can perform one 32 x 32 bits or two 16 x 16 bits or four 8 x 8 bits multiplications every clock cycle [16]. Fixed-point architecture implies that floating point operations are not executed in hardware, but rather are emulated by software. A set of libraries are provided by TI for programmers. The IQmath library [17] is a collection of highly optimized mathematical functions (written as C/C++ routines) aimed for porting floating-point algorithms to fixed-point code that can be executed by the C64x+ hardware. The DSP library or DSPLIB for short is a collection of high-level optimized DSP functions for the TMS320C64x device [18]. Most of these routines are used for signal processing, especially in computationally expensive real-time applications. The functions in DSPLIB are organized into seven different categories: Adaptive filtering, correlation, Fast Fourier Transform (FFT), filtering and convolution, math, matrix and miscellaneous. The rich set of software routines are included in the Image Video Processing Library (IMGLIB) [19]. IMGLIB is organized into three different functional categories as follows: Compression and decompression, image analysis, and picture filtering/format conversions. Fast Run-time Support (FastRTS) Library [20] is an optimized floating-point function library for the fixed-point TMS320C62x/64x devices and used for computationally intensive real-time applications where optimal execution speed is critical.

2.7. OMAP3530 Operating Systems

2.7.1. Ångström Distribution

Ångström is a Linux distribution supporting various embedded devices. The Ångström distribution is the result of unifying the OpenEmbedded, OpenZaurus, and OpenSimpad projects. Ångstrom can be built for an embedded platform by the OpenEmbedded framework. The OpenEmbedded project uses *bitbake* build packages by means of cross-compiling. Cross-compilation is the procedure when development and embedded platforms have different compilers and the resources of embedded platform are limited by its processor speed, storage space or memory space.
In this study, Ångström distribution is configured on host computer and installed onto the BeagleBoard.

2.7.2. DSP/BIOS™ Real-Time OS

The Texas Instruments DSP/BIOS is a real-time multi-tasking kernel that was designed to run on DSP platforms, specifically the TMS320C6000, TMS320C5000, and TMS320C28x families. Without a license fee, DSP/BIOS supports many functions in order to support complex applications within the constraints and deadlines typical for real-time applications. This study uses the default DSP/BIOS configuration and mainly concentrates on using C6Accel.

2.8. C6Accel

C6Accel is a DSP library which includes many core functions. These core functions are useful in various signal processing applications including digital signal processing routines such as FFTs and filters. They also include image processing routines such as edge detection and color space conversion as well as math routines such as Logarithm, SQRT and trigonometric functions.

C6Accel specifically targets ARM+DSP devices which run Linux on the ARM. ARM can allow an application to offload computation intense tasks to DSP. Another reason for using C6Accel in this study is that it allows us not to worry about DSP's complex memory management and DSP's real-time issues. C6Accel abstracts these

complexities away by providing abstracted wrapper API calls that hide these complexities from an ARM user.

The application code (or the middleware it uses) calls C6Accel using wrapper library APIs. These APIs then pass through the codec engine interface and invoke the C6Accel algorithm on the DSP which identifies the function call and executes the appropriate functionality on the DSP using the parameters passed from the application.

SoC View given in Figure 8 shows its integration over ARM+DSP platform.

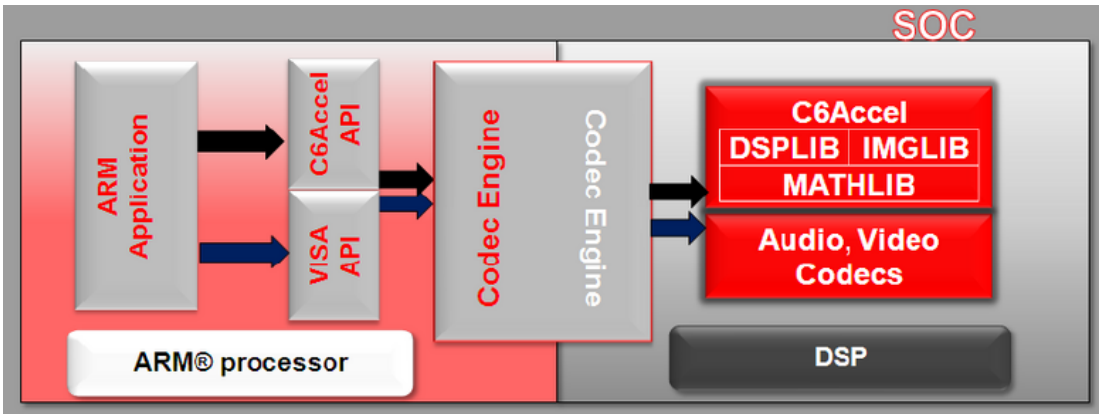


Figure 8: SoC view with C6Accel [21].

There is an overhead of calling API functions. The overhead is due to invalidating cache of large buffers between DSP and ARM as well as calling the API function. C6Accel supports synchronous and asynchronous calling modes. To reduce the overhead, the ARM side code could be designed to make calls in asynchronous mode. This enables parallel processing on DSP and ARM [Figure 9].

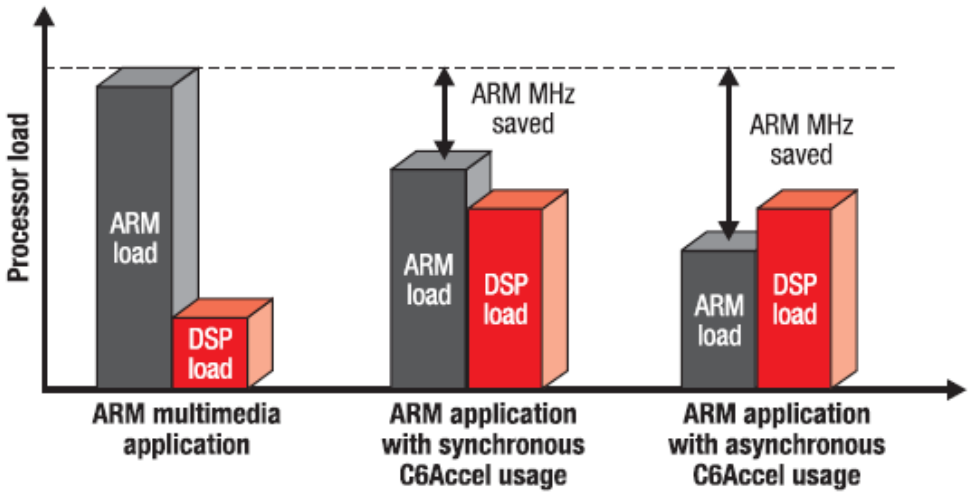


Figure 9: Processor load view to demonstrate benefits of C6Accel usage.

DSP libraries that are linked to C6Accel are,

- C64x+DSPLIB DSP Signal Processing Library version 2.10,
- IMGLIB-2 DSP Image/Video Processing Library Release version 2.0.1,
- IQMath Release version 2.1.3,
- C67xFastRTS Fast Run Time Support Library Release: Version 1.03,
- C674x DSPLIB Digital Signal Processing Library Release: Version 1.2.

A list of benchmark results for OMAP3530 is given in [22].

2.9. CMEM

The DSP and the ARM exchange information using a shared memory space in the DDR2 which is defined in the CMEM module. CMEM is an API and library for managing one or more blocks of physically contiguous memory. It also provides address translation services (e.g. virtual to physical translation) and user-mode cache management APIs. This physically contiguous memory is useful as data buffers that will be shared with another processor[23]. The ARM sees these locations as a virtual address through the MMU while the DSP sees it as the physical address. Using its pool-based configuration, CMEM enables users to avoid memory fragmentation, and ensures large physically contiguous memory blocks are available even after a system has been running for very long periods of time.

2.10. OpenCV

OpenCV is a powerful and easy to use computer vision library, aimed to be efficient for real time applications. Initially developed by Intel it is now supported by the open source community. Written in C and C++, it is available for Linux, Windows and Mac OS X. OpenCV is structured into five main components. The CV component contains the basic image processing and higher-level computer vision algorithms; ML is the machine learning library, which includes many statistical classifiers and clustering tools. HighGUI contains I/O routines and functions for storing and loading video and images, and CXCore contains the basic data structures and content [24]. Loading and displaying frames is handled with the HighGUI component in this study. Handle to images that are passed to DSP has to be stored in contiguous memory provided by the CMem module. The project *OpenCV DSP Acceleration* [25] is adopted in this study because it includes the necessary patches for integration with DSP. This simplified the use of OpenCV functions without a change in its calling.

2.11. Previous Work

This section mentions previous studies on histogram equalization based image enhancement techniques for embedded platforms. The literature survey showed that most studies are based on FPGA solutions. A DSP+GPP solution has not been found in literature other than this work. A hybrid study of FPGA+DSP is the only combined solution that was found so far [26].

In [26], a real-time Plateau histogram equalization method for infrared images is realized on an FPGA+DSP platform. The FPGA calculates the histogram statistics from input signal and DSP uses the input signal and histogram output from FPGA to create the look-up table (LUT) according to the algorithm selected. DSP then updates the LUT which is input to video output system. No performance analysis is associated with that study.

A HE circuit for infrared image enhancement based on FPGA is proposed in [27]. The circuit is composed of four main units: 16-bit binary counters, 8-to-256 special purpose decoder (SD), multiplexers, and a timing and control module. The first 8 bits of 16 bit counter are used to get statistics of the histogram; the second task is to

simultaneously accumulate the current gray level with the statistics of all pixels whose values are less than the current pixel. The result of this accumulation process is the final HE values. The SD is used to enable the counter assigned to the current pixel and all the counters assigned to pixels whose values are greater than the current one so that the computation of histogram statistics and transformation function are performed at the same time. Simulation results for a 250 MHz clock show that for a 256x256x8 bit image, the complete computation time is 0.263 ms.

In [28] a contrast enhancement algorithm based on Plateau histogram for IR images is implemented on FPGA. That system can process 25 frames of 128x128x12 bits infrared images per second.

The report in [29] investigates state-of-the-art algorithms for surveillance from the field of computer vision in respect of DSP-based embedded platforms, especially issues related to parallelism mechanisms and fixed-point arithmetic. That study specializes on object detection and recognition.

CHAPTER 3

PROPOSED METHOD

3.1. Introduction

The aim of the thesis is to enhance IR images in real-time. The real-time requirement for operator display should not be less than 15 FPS. The operating system on ARM is not a hard real-time system, thus the performance is evaluated as a soft real time system. A hard-real time system is used for time critical applications and the application is not interrupted by high-level tasks. The whole application is first written to test on ARM. Running the whole application on ARM doesn't meet the execution performance that we intend. Later, we seek a better approach that makes use of a DSP in an efficient way. Therefore, tasks are rearranged so as to leave the sequential execution on ARM, and parallel execution on DSP. Moving to fixed-point arithmetic instead of floating-point arithmetic made a big improvement in terms of execution performance. Against the loss in precision in fixed-point arithmetic, the results are checked to make sure it gives the same accuracy or with acceptable rounding errors. After several benchmarks for different software distribution approaches, the best performance is met and optimizations are done for this design. Where possible, the DSP is run in parallel (asynchronous mode) by ARM to utilize two cores at the same time.

3.2. Development and Experimental Environments

The Ubuntu 10.10 Linux distribution is used as the development environment on the host computer. The host computer is used to build the Ångström image, the ARM application, the DSP application and to move binary files to the BeagleBoard through Ethernet. This is done through a Secure Shell (SSH) connection. Since the

BeagleBoard does not provide an Ethernet interface, a USB hub is connected to its mini USB port and a USB network device was used on the USB hub. The experiment setup is shown in Figure 10. Other peripherals connected to the USB hub are a keyboard and a mouse. The RS232 serial protocol is used to execute commands. For serial terminal emulation, Minicom is set up on the host computer. Minicom is a text-based modem control and terminal emulation program for Unix-like operating systems. The software for DSP and ARM is written in C, to make development straight-forward and efficient.

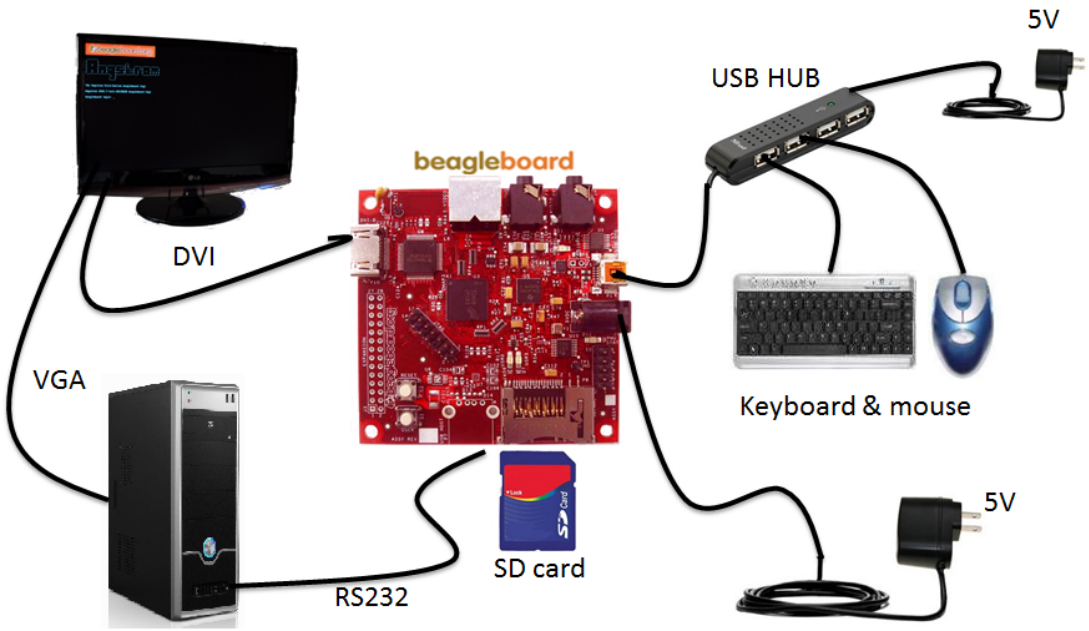


Figure 10: Experiment setup

The tools that are used to build up the embedded system are mentioned below.

- Codec-Engine 2.25.01.06, necessary to add our DSP kernels to the C6Accel source so they are integrated with the codec engine as explained in [30]. The CMem module which is used to assign contiguous memory from the user space also comes with this package.
- OpenCV-2.2, Open Computer Vision library, OpenCV, is used in the project to load and display images.
- DSPLink 1.64 is installed as a step from OpenCV-DSP-Acceleration Project.

- Code Generation Tool 7.0.1 TI's collection of tools needed to build DSP applications. It contains an optimizing C compiler and assembly language tools (assembler, linker, archiver, hex conversion utilities, etc.)
- Local Power Manager 1.24.02 used to create kernel modules for power management system services.
- eXpress DSP Components (xdctools) 3.16.00.18 collection of components to create, test, deploy, and install DSP algorithms for target device.
- C6Accel 1.00.00.04 collection of APIs for ARM application to call DSP kernels. Image processing functions from this package are used as well as integrated the DSP kernel that is developed as part of this study.
- GCC 4.3.3 toolchain with prefix arm-angstrom-linux-gnueabi is used to produce software for ARM architecture. Toolchain is compliant with the GNU Embedded Application Binary Interface (EABI). An ABI is a low-level description of the interface between the applications and the operating system hosting them. Although it is mentioned in [31] that ARM's commercial compiler RVCT produces the fastest executable, I preserved GCC because it is GNU licensed.

As mentioned before, Ångström for ARM and DSP/BIOS RTOS for DSP are chosen. Among a number of OSs supported by BeagleBoard, Ångström is chosen because it is an open source Linux distribution, is stable, updated frequently and well documented, and an easy to use package manager makes it attractive. Ångström uses OpenEmbedded as package building system which allows creating a complete Linux Distribution for embedded systems. OpenEmbedded uses *bitbake* for package management. Following the procedures from the OpenCV-DSP-Acceleration Project, the base-image Ångström-2010.7 is built from scratch.

In this study BeagleBoard uses an SD-Card as boot media. As explained in [32], the SD-Card is partitioned into FAT and Ext3 partitions. The FAT partition stores the boot manager and the kernel image while the Ext3 partition stores the Linux root file system. The kernel version used is 2.6.32. On the DSP side, the BIOS version used is the DSP/BIOS 5.41.07.24.

Table 2: Revision of hardware components in the experiment setup

Hardware	Version
BeagleBoard	B7.2
OMAP3530	ES3.0
ARM Cortex A8	r1p3

Output for /proc/cpuinfo is as following:

```
root@beagleboard:~# cat /proc/cpuinfo
Processor       : ARMv7 Processor rev 3 (v7l)
BogoMIPS        : 490.52
Features        : swp half thumb fastmult vfp edsp thumbee neon vfpv3
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x1
CPU part        : 0xc08
CPU revision    : 3
Hardware        : OMAP3 Beagle Board
Revision        : 0020
Serial          : 0000000000000000
```

The processor speed for both ARM and DSP are left untouched which are 500 MHz for ARM and 360 MHz for DSP. 500 MHz is the default used because it is a balance of performance and longevity. Although it is possible to change the clock permanently with boot parameters, higher frequencies brings high core voltages [33].

This study also includes outputs from MATLAB to test the accuracy of the proposed approach.

3.3. Application Software

The main objective of this thesis is to produce an algorithm that is fast enough to meet real time criteria while preserving the correctness of the original image

enhancement approach. The final design uses the advantage of typical IR image characteristics which will be mentioned later. Software is developed for both the ARM and the DSP side. High level tasks of ARM are, user interfacing, encoding/decoding 16 bit images in TIFF format, sending the output image to the display and initializing routines. ARM is capable of sending asynchronous calls to the DSP, allowing the ARM to continue its job while the DSP is running in parallel. Since the algorithm in this study runs mostly sequentially, meaning that the input to a method depends on the output of previous method, it is difficult to make the design so that ARM and DSP are loaded efficiently. So that, the preparation of input data for the DSP task is left as low level task to ARM. The tasks that are shared between ARM and DSP are illustrated in Figure 11. In the figure, gray and white boxes are used for DSP and ARM tasks respectively, and a dashed line is used to show that the task is run in parallel with ARM. The input and output parameters are also illustrated.

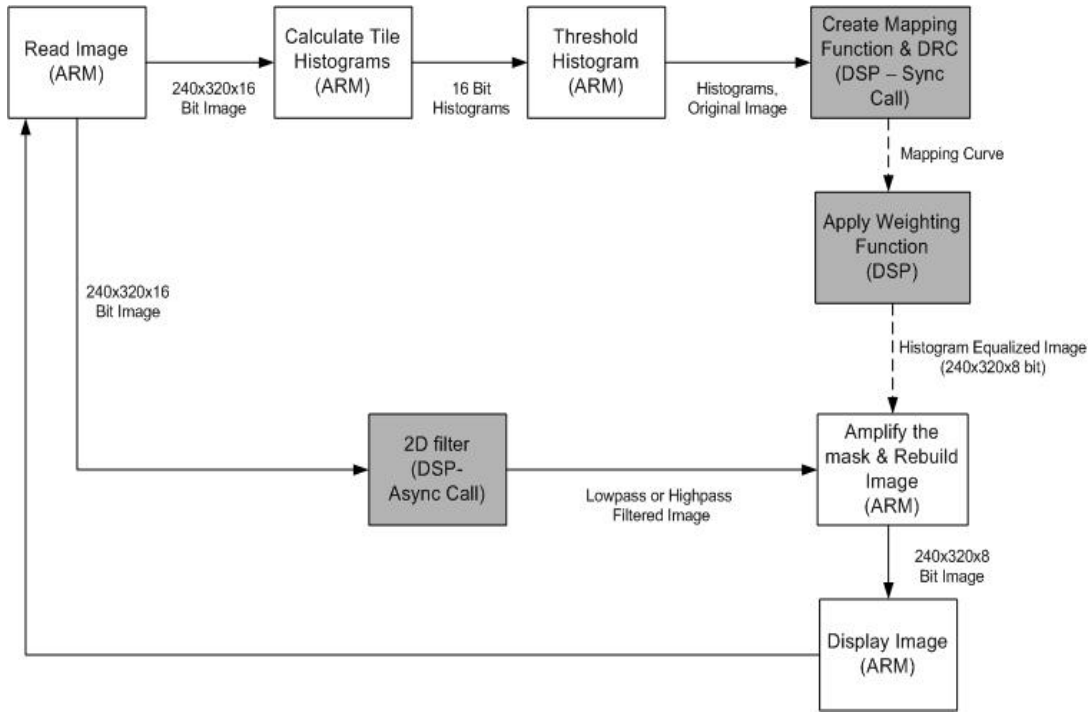


Figure 11: Main flow of operation

When the process for a frame is finished, the whole process is started following the capture of a new frame and new histogram values are calculated again. In other

words, there is no dependency between frames. The flow of the operation is explained by dividing it into parts:

- initialization,
- read image,
- calculate tile histograms,
- clip histogram,
- process histogram,
- contrast enhancement,
- display image.

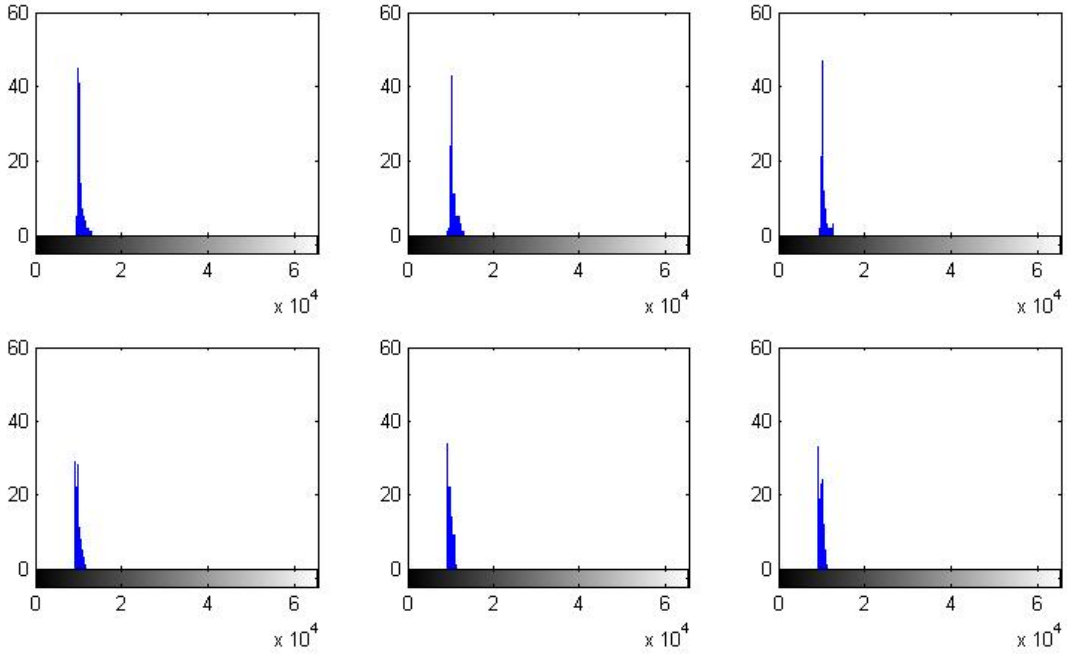
Corresponding flow charts are given for the main parts.

Initialization: The codec engine server thread required by DSP calls is started and the buffers shared between DSP and ARM are allocated in the contiguous memory. These buffers include a lookup table for distance weights of size *[tile count] x [image size]* bytes, a histogram buffer of *[tile count]* × *[number of bins in the histogram]* × *2* bytes since the values are stored in 16-bit values, a buffer to hold the filtered version of the image of size *[image size]* × *2* bytes for 16-bit values, a buffer for the captured image of size *[image size]* × *2* bytes for 16-bit values, and a buffer for the enhanced image of size *[image size]* × *1* bytes because the output image is of 8 bit resolution.
After memory allocation, the lookup table to hold distance weights is created from equation (2-3).

Read image: The image is read in tiff format using the OpenCV method cvLoadImage.

Calculate tile histograms: A two dimensional matrix is used to hold histogram values of each tile. Every column in the matrix holds the histogram values of each tile. Histogram size is 2^k, where k is the number of bits to hold intensity values.

(a)



(b)

Figure 12: (a) Subdivided image and (b) corresponding histograms.

Clip histogram: For *n* tiles and for each value in the histogram matrix, it calculates the excess amount over the clip level and the total area under the clip level *s*, using equations (3-1) and (3-2),

$$h^{CLIPPED}(n,i) = \begin{cases} h(n,i) & \forall n,i : h(n,i) < s \\ s & \forall n,i : h(n,i) > s \end{cases} \tag{3-1}$$

$$\begin{aligned} Excess(n) &= \sum_i h(n,i) - s \qquad \forall n,i : h(n,i) > s \\ Total(n) &= \sum_i h(n,i) \qquad\quad \forall n,i : h(n,i) < s \end{aligned} \tag{3-2}$$

The user defined clip level, s, is global for one frame. After the distribution of the excess amount, it is possible that new histogram values exceed the clip level. This requires a recheck over the histogram and redistribution and this needs to be repeated as long as the distribution keeps generating new values above the clip level. This can be done more efficiently by first finding the largest histogram value that will not exceed the clip level after redistribution. Then the values between the largest histogram value and the clip level will be set to the clip level and values lower than that will be increased by the distribution amount. In order to find the largest histogram value that will have distribution amount, the number of occurrences of histogram values are calculated using (3-3) during the calculation of excess amount.

$$H(n,h(n,i)) = \sum 1 \qquad \forall n,i : h(n,i) < s \tag{3-3}$$

After calculating excess, total and *H* values for all tiles, the largest histogram value that will not exceed the clip level after excess distribution is calculated. The flow chart of this process is given in Figure 13. In this function *H* is decremented by one until *H* doesn't exceed the clip level and the values of *Excess* and *Total* are updated for every iteration and finally the new distribution factor is calculated. The distribution factor is the contribution of excess over the histogram value and calculated using the equation (3-4).

$$\begin{aligned} \mathrm{h}^{\mathrm{NEW}} &= \mathrm{h} \times \mathrm{Factor}, \\ \mathrm{Factor} &= 1 + \frac{\mathrm{Excess}}{\mathrm{Total}} \end{aligned} \tag{3-4}$$

The flow chart for obtaining the equalized histogram values is shown in Figure 14.

Equalized histogram values are then passed to a DSP task for generating the new image.

Local contrast enhancement: Two approaches for local contrast enhancement are tested. First, the proposed BCCE method is tested. For this method lowpass-filtered version of the original image is obtained using two dimensional convolution. TI's image processing library comes in C6Accel and includes 2D convolution method for 16 bit images. This method is optimized for ARM to give better performance and its signature is given as;

*int C6accel_IMG_conv_3x3_i16s_c16s_Frame(C6accel_Handle hC6accel, const short *restrict imgin_ptr, short *restrict imgout_ptr, short rows, short width, short pitch, const short *restrict mask_ptr, short shift)*

The parameters passed to DSP method have the following definitions:
hC6accel : handle to the C6Accel,
imgin_ptr: The original image buffer with 16 bit values passed to DSP,
imgout_ptr: Convolved image, the output buffer passed back to ARM,
rows: Number of rows to process. Rowsize -2,
width: Number of outputs to be calculated. ColSize -2,
pitch: Number of columns in the input image,
mask_ptr : 3x3 mask
shift: The amount that output values to be shifted.

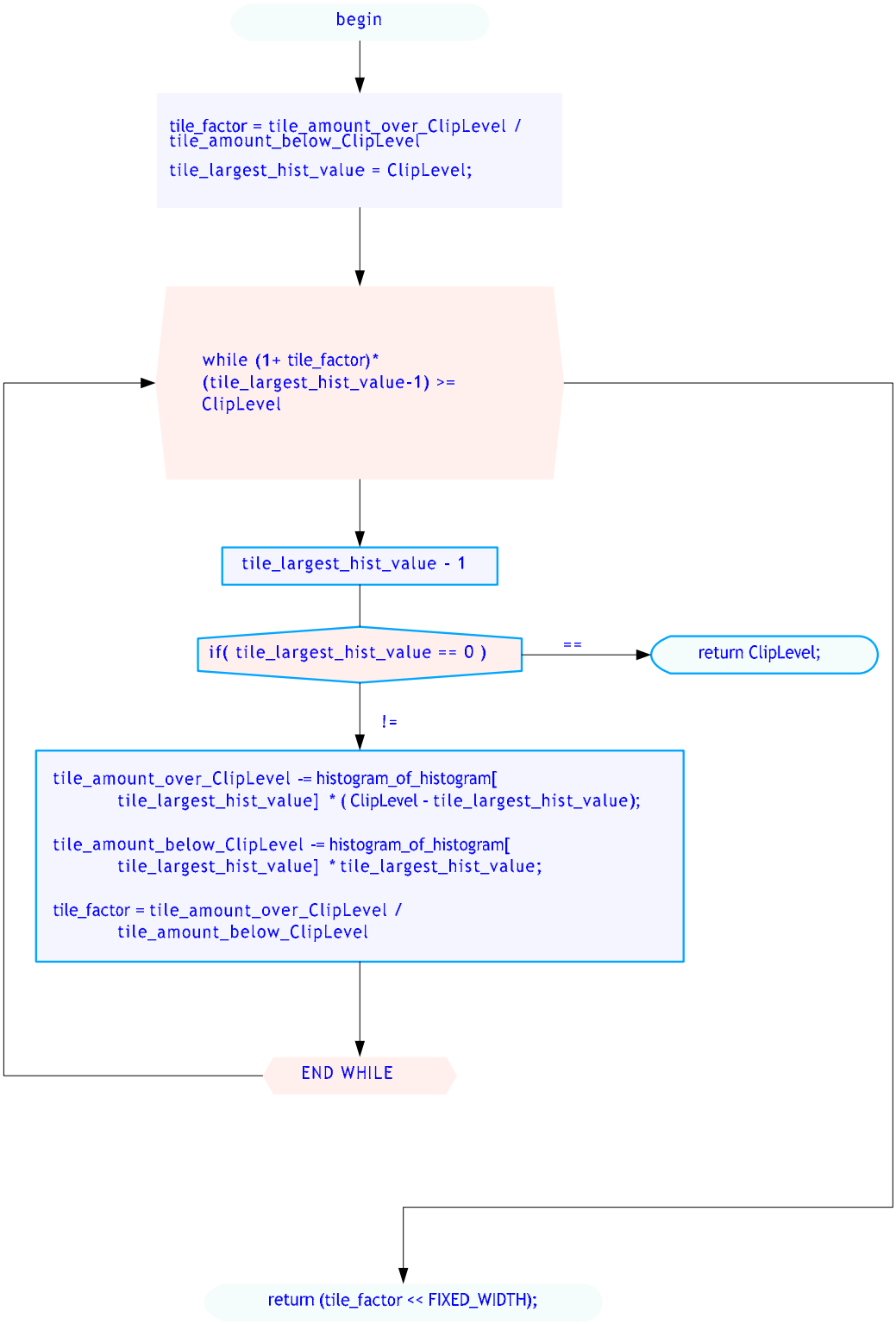


Figure 13: Flow chart for factor calculation for single tile.

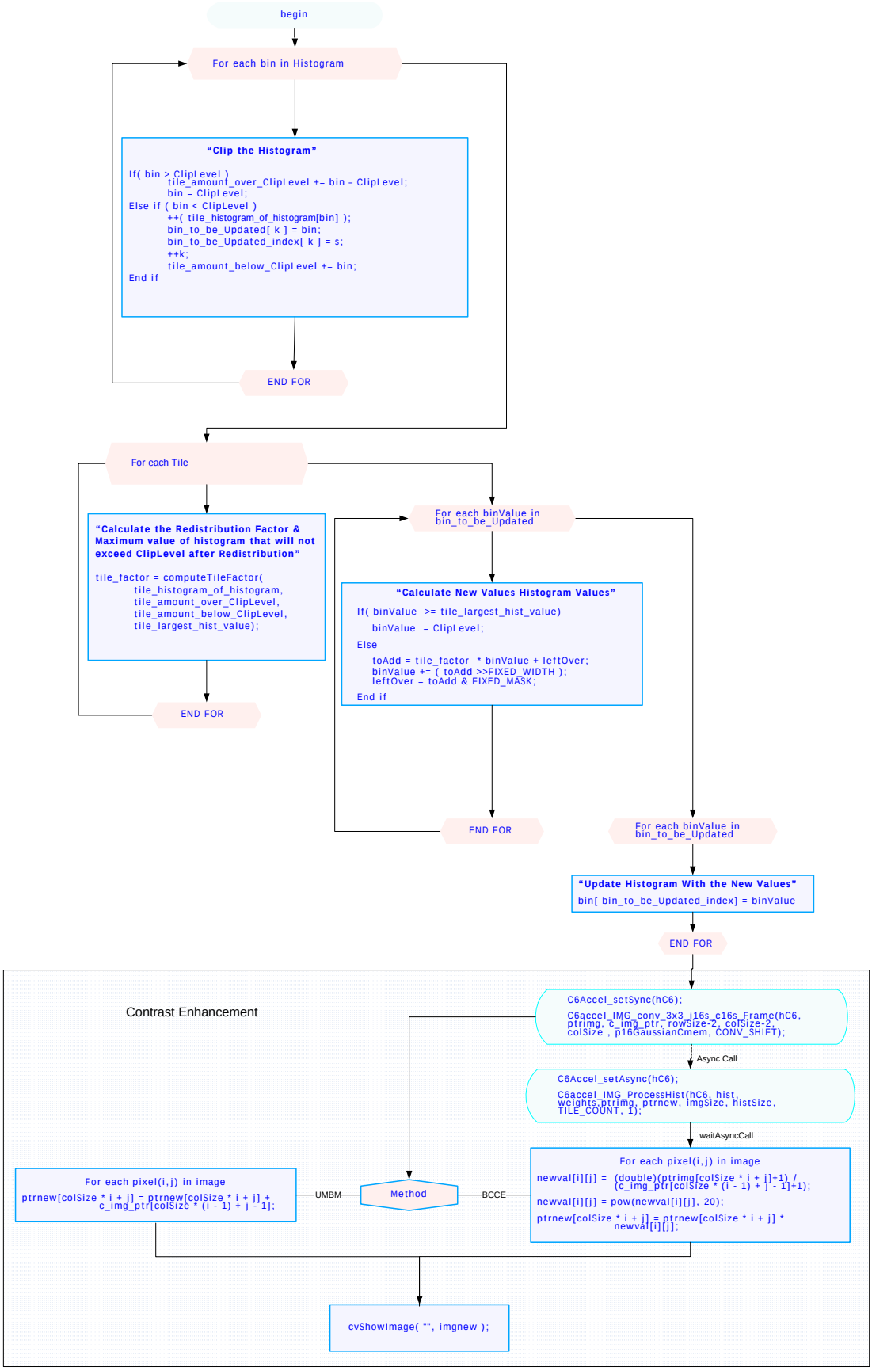


Figure 14: Flow chart for histogram clipping & contrast enhancement.

The local contrast image is obtained by dividing the input image values by the lowpass-filtered image values. The local contrast improvement factor α=20 gave the best output image during tests and it is used as default in the method. Next, the result is multiplied by the histogram equalized image and the enhanced image is sent to the display. To apply UMBM, filtered image is added to the histogram equalized image and the convolution mask passed to the DSP is,

$$\begin{pmatrix} -5 & -5 & -5 \\ -5 & 40 & -5 \\ -5 & -5 & -5 \end{pmatrix}$$

The mask passed to the DSP was converted into Q15[1] format. The conversion is done by shifting the parameters by 7 or multiplying by 2^7 and adding 0.5 for rounding error. Passing the *shift* parameter as 7, the required output data is obtained.

Flow chart in Figure 14 includes the code flow for both UMBM and BCCE.

Process histogram: This part of the software is run on DSP with input parameters from ARM, 16 bit equalized histogram values, the look up table of distance weights shifted by 8 to make fixed point operation, the original image buffer with 16 bit pixel values, and the output parameter to ARM is the histogram equalized image with 8 bit pixel values. For every tile, mapping curves are created using the histogram values.
Next, new pixel values are obtained using the mapping curves and the distance weights lookup table. Since the local contrast enhancement method takes the equalized and scaled image in the last step, *async* mode is enabled for this method to allow ARM to continue doing other calculations. The flow chart for this process is given in Figure 15.

Display Image: The image is displayed using the OpenCV method cvShowImage.

[1] Fractional Q formats are one way of codifying fixed-point representations. One of the most common representations is the so-called "Q15" format, or more specifically Q0.15, where 16-bit short integers are treated as having their radix point between the most-significant bit (MSB) and the second MSB, leaving a fractional component of 15 bits [6].

Booting System and Running Application:

The application requires the following modules loaded to run: DSPLink, CMem and LPM. CMem requires extra parameters and an example of it is given below.

insmod cmemk.ko phys_start=0x86300000 phys_end=0x87300000 pools=9x76800,18x65536,2x76800,1x76800 allowOverlap=1

Physical start and end addresses are in hex format and defines memory region that is reserved for contiguous memory. The sizes of the pools are decided according to the buffers allocated in the contiguous memory during the initialization in the code. This phase is done once before running the application.

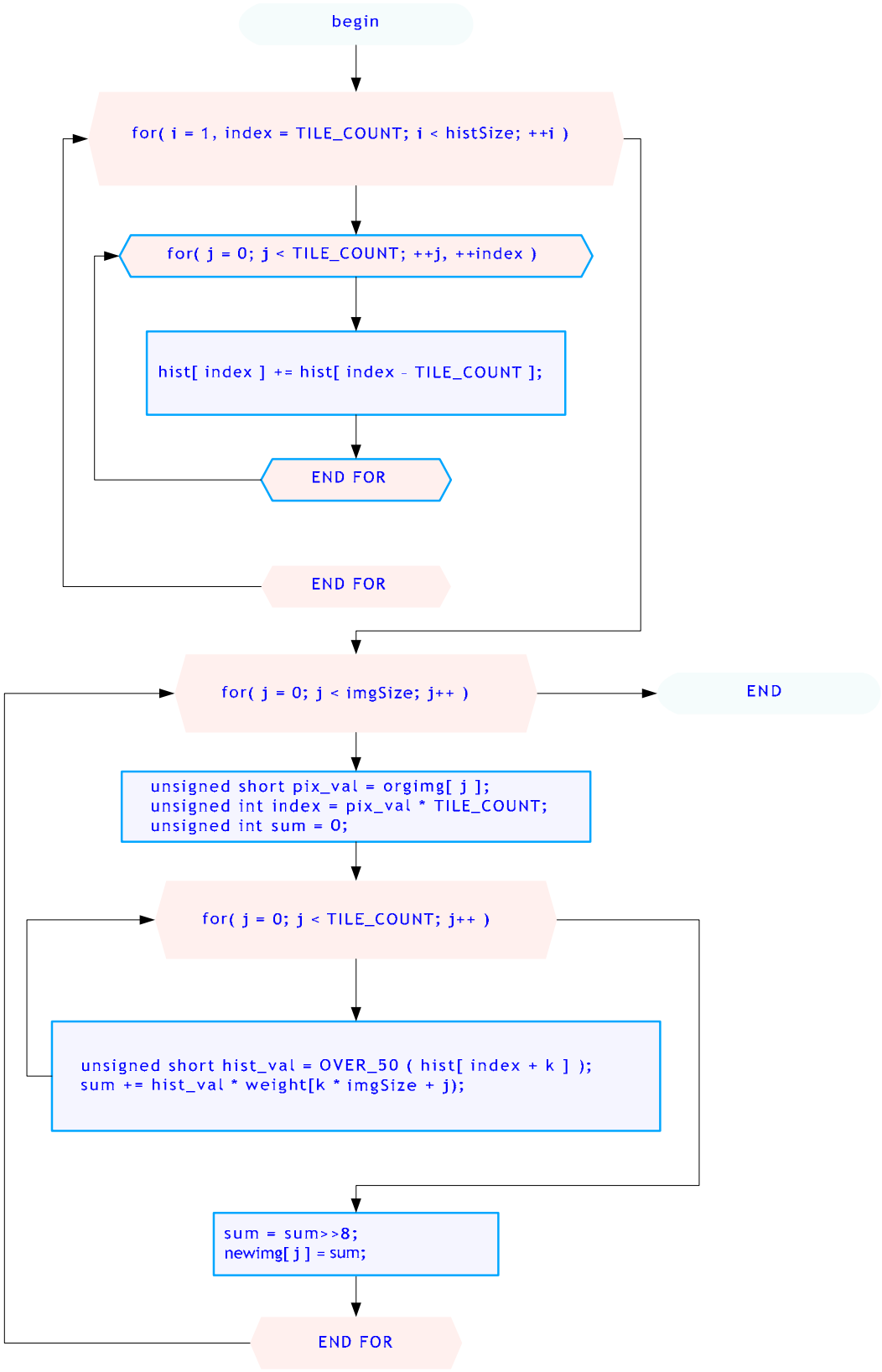


Figure 15: Flow chart for calculating mapping curves and histogram equalized image.

3.4. Optimization Techniques

The software is first developed to run purely on ARM and the execution time for the computational parts is measured. The timing results showed that ARM alone is not capable of finishing one frame in less than 200 milliseconds and the results are obtained before optimizations. Use of DSP CPU has become attractive and the software is designed to share the tasks between CPUs efficiently.

3.4.1. DSP Optimizations

One of the design decisions was passing big data chunks at once instead of making several calls because of the overhead of a DSP call. The overhead is due to invalidating from and writing back to the cache of the buffers so that the DSP and ARM sees the right data. Nonetheless, C6Accel is still a good choice for its performance on big buffer sizes [34]. Most CPU cycles are spent on large arrays like distance weights and tile histograms. One way to increase the execution performance is to make operations on data sets that are in the same memory region and will thus be loaded in the cache at the same time. This is known as memory striping and is suitable to apply to our code. To reduce the memory access time using this technique, arrays are structured in a manner that consecutive memory accesses are within the cached buffer. Hereby, the histogram array is laid out such that every column holds histogram values of a single tile or, equivalently, every row holds a single value of every tile. Another optimization technique applied is loop unrolling. This is applied to the iteration through every pixel in the image when new pixel values are calculated by each tile's mapping curve and distance weights to each tile's center. Iterating for each tile is replaced by iterating for each pixel. This can be seen in Figure 15. In the figure, the macro definition OVER_50 is used for fixed-point division, which is another improvement in terms of execution performance since floating-point operations are only emulated by software thus consuming more CPU cycle. For that parameter set in the application, OVER_50 makes the division for value, x, by adding the eight times right shifted value to the six times right shifted value. It corresponds to multiplication by 256/12720 = 0.02 or, equivalently, division by 50. Hence the name OVER_50.

These are the DSP code optimization techniques used in our application. Other optimization techniques found not applicable are parallel instructions, software pipelining and intrinsic C functions.

TI's C compiler, cl6x, is able to perform various optimizations. DSP software is compiled with flags "-mv6400+ -PM -o3". "mv6400+" tells the compiler to optimize for c64x+ platforms. "-PM –o3" tells the compiler to do program-level optimization. The use of compiler optimizations increased execution speed by around 8%.

3.4.2. ARM optimizations

A typical IR image has low contrast characteristics, thus the histogram of a low contrast, high dynamic range image has a narrow distribution. This can be seen in Figure 12. The histogram is non-zero for around 4000 values out of 65535. Because zero values are not processed during histogram clipping, word-wide access is used to skip multiple consecutive zero values simultaneously. This increased the execution speed around two times. Another improvement is achieved by the use of single precision floating point values instead of double precision floating point values.. During contrast enhancement, the arithmetic power operation was done on double values. For this calculation the *pow* method in math library was used first. But *pow* is known to be slow because it uses a floating point value for the exponent. In our case the exponent is 20 so the method exponentiation by squaring was preferred. Execution speed is increased by 15% when the results are stored in floating point variables. Fixed point operation is preferred where applicable. The code part given as a flow chart in Figure 14 has this advantage. Lastly, compiler options for optimizations are sought. Giving the GCC compiler as much information as possible it can do the optimization. The compiler options used for our system are:
"-march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fno-strict-aliasing -fexpensive-optimizations -frename-registers -fomit-frame-pointer -O2 -ggdb0 -fPIC -DPLATFORM=3530".

CHAPTER 4

PERFORMANCE ANALYSIS AND TEST RESULTS

Execution performance is measured in time using the C function, gettimeofday, defined in the GNU C library. This function measures time in micro seconds resolution. The measurement is done throughout the process of one frame excluding the OpenCV methods that are used to read and display images. Also, the Ethernet network service is disabled during the measurement to free the CPU resources.

The measurements are done on images of dimensions 320 by 240 pixels. BCLAHE, BCLAHE+CE and BCLAHE+UMBM are tested with different parameters and the performance affected by different parameters is compared. Histogram size and number of tiles are the parameters that influence the process time. Therefore, the algorithm is tested for 14 bit and 16 bit images as well as for 6 and 9 tiles.

Performance comparison of 14 and 16 bit images for 6 tiles is given in Table 3. Performance comparison of 14 and 16 bit images for 9 tiles is given in Table 4. FPS values are calculated using the measured execution time given in tables.

Table 3: Results for 6 tiles

	Execution Time (ms.) / **FPS** 16 bit	Execution Time (ms.) / **FPS** 14 bit
BCLAHE	35 / **28**	18 / **55**
BCLAHE+ CE	57 / **17**	49 / **20**
BCLAHE+ UMBM	44 / **22**	27 / **37**

Table 4: Results for 9 tiles

	Execution Time (ms.) / **FPS** 16 bit	Execution Time (ms.) / **FPS** 14 bit
BCLAHE	46 / **21**	23 / **43**
BCLAHE+ CE	64 / **15**	52 / **19**
BCLAHE+ UMBM	56 / **17**	32 / **31**

Performance analysis shows that, execution time increases by 50% for 14 bit images for BCLAHE method, 16% for BCLAHE+CE method and 40% for BCLAHE+UMBM method. The best performance is achieved for 14 bit image with 6 tiles. The system is capable of 15 FPS for the worst case when images are 16 bit and number of tiles is 9.

The accuracy of images is compared using output images from BeagleBoard and MATLAB. Due to the use of fixed-point calculations as an optimization technique rounding errors may occur. Output images from two systems are found identical except that the pixels on the borders are by default not processed by 2D convolution filter of DSP library. Therefore the comparison is done excluding the borders of image. Zero padding is not preferred due to its performance cost and assuming the edges of image don't contain important information for operator display.

Output images from the BeagleBoard for which performance is measured are given in Figure 16 to Figure 21. The use of the distance weights method for the elimination of tile borders was adopted from [1]. This method produced spot effects around the center of tiles. This can be better seen in Figure 22 when 16 tiles are used.

(a)

(b)

Figure 16: BCLAHE applied to 16 bit (a) and 14 bit (b) images; 9 tiles

(a)

(b)

Figure 17: BCLAHE applied to 16 bit (a) and 14 bit (b) images; 6 tiles

(a)

(b)

Figure 18: BCCE applied to16 bit (a) and 14 bit (b) images; 9 tiles

(a)

(b)

Figure 19: BCCE applied to16 bit (a) and 14 bit (b) images; 6 tiles

(a)

(b)

Figure 20: Unsharp mask applied to16 bit (a) and 14-bit (b) images; 9 tiles

(a)

(b)

Figure 21: Unsharp mask applied to16 bit (a) and 14-bit (b) images; 6 tiles

Figure 22: Unsharp mask applied to 16 bit; 16 tiles

CHAPTER 5

CONCLUSION

The objective of this thesis project was to investigate the performance of IR image enhancement on OMAP3530 platform. First, the performance of BCLAHE is tested on ARM. Initial results showed that ARM was not fast enough to process one frame in 67 ms. (or 15 FPS) which is the lowest execution time criteria in this study. Next, the software is redesigned to offload computationally intensive functions onto DSP. Performance criteria were met through the use of DSP and ARM together. Then contrast enhancement is incorporated in the software. The use of parallel execution of DSP and ARM reduced the overall performance. Finally, code and compiler optimization techniques are investigated and the required performance is achieved. Benchmarking is done for different parameters and the worst scenario is use of 16 bit images and 9 tiles. The objective of this thesis study has been successfully achieved for images with intensity resolution not more than 16 bit and a number of tiles not more than 9.

Future work can be to port a real time operating system on ARM so that the operating system will not interrupt the application and give the highest priority to the software.

REFERENCES

[1] F. Branchitta, M. Diani, A. Porta and G. Corsini. Dynamic-range compression and contrast enhancement in infrared imaging systems. *Opt. Eng.,* 47:(1-14), 2008

[2] Application Story, FLIR. Seeing in total darkness in a maritime environment. http://www.flir.com/uploadedfiles/Eurasia/MMC/Appl_Stories/AS_0006_EN.pdf

[3] K. R. Castleman, Digital Image Processing, Chap. 3, pp. 37-38, Prentice Hall, Englewood Cliffs, NJ (1996).

[4] Gonzales, R.C., Woods, R.E.: Digital Image Processing. Prentice-Hall, Inc., Englewood Cliffs (2008)

[5] William K. Pratt: Digital Image Processing: PIKS Inside, Third Edition, 2001

[6] Shehrzad Qureshi, Embedded Image Processing on the TMS320C6000™ DSP, Labcyte Inc., Springer, 2005.

[7] D. J. Ketcham, R. W. Lowe, and J. W. Weber, Real-time image enhancement techniques, in *Seminar on Image Processing*, Hughes Aircraft, Pacific Grove, California, 1976, pp. 1-6.

[8] R. A. Hummel, Image enhancement by histogram transformation, Comput. Graphics Image Process. **6**, 1977, 184-195.

[9] S. M. Pizer, Intensity mappings for the display of medical images, in *Functional Mapping of Organ Systems and Other Computer Topics*, Society of Nuclear Medicine, 1981.

[10] S. M. Pizer, E. P. Amburn, D. A. John, C. Robert, G. Ari, G.Trey, R. Bart Ter Haar, and B.Z. John. "Adaptive histogram equalization and its variations," Comput. Vision Graph. Image Process., vol. 39, no. 3, pp. 355-368, 1987, 29046.

[11] *BB_SRM - BeagleBoard System Reference Manual - Revision B7.2*, Nov 17, 2008. URL http://beagle.s3.amazonaws.com/BBSRM_7_2_0.pdf

[12] Texas Instruments. *OMAP35x Applications Processor: Technical Reference Manual*, April 2010. Reference: SPRUF98T. Revised July 2011.

[13] http://www.arm.com/community/software-enablement/index.php

[14] ARM. Cortex A8 Technical Reference Manual, Rev. r3p2, DDI 0344K. p.1-2. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0344k/ch01s09s04.html

[15] ARM NEON. http://www.arm.com/products/processors/technologies/neon.php

[16] Texas Instruments. *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide*. July 2010. Reference: SPRU732J

[17] Texas Instruments. TMS320C64x+ IQmath Library User's Guide, December 2008. Reference: SPRUGG9.

[18] Texas Instruments. *TMS320C64x+ DSP Little-Endian DSP Library Programmer's Reference*. Revised March 2008. Reference: SPRU565B

[19] Texas Instruments. *TMS320C64x+ DSP Image/Video Processing Library Programmer's Guide*. Revised May 2008. Reference: SPRUF30A

[20] Texas Instruments. *TMS320C62x/64x FastRTS Library Programmer's Reference*. February 2003. Reference: SPRU653

[21] Texas Instruments. C6Accel White Paper. September 2010. Reference: SPRY144

[22] Texas Instruments. Benchmarks of C6Accel on OMAP3530. URL: http://processors.wiki.ti.com/images/9/9a/Benchmarks_sync_omap3530.pdf

[23] Texas Instruments. CMEM Overview. URL: http://processors.wiki.ti.com/index.php/CMEM_Overview

[24] Learning OpenCV, Gary Bradski and Adrian Kaehler.

[25] OpenCV DSP Acceleration Project. URL: http://code.google.com/p/opencv-dsp-acceleration

[26] Dongsheng, G., Nansheng, Y., Defu, P. Min, H., Xiaoyan, S. and Reolan, Z. A DSP+FPGA based real-time histogram equalization system of infrared image, *Proc. Spie*, **4602**, 2001.

[27] A. M. Alsuwalem, "A Novel FPGA Based Real-time histogram Equalization Circuit for Infrared Image Enhancement", *J. of Active and Passive Electronic Devices,* Vol. 3, pp. 311-321.

[28] W. Bing-jian, L. Shang-qian, L.Qing, Z. Hui-xin, "A real-time contrast enhancement algorithm for infrared images based on plateau histogram", Infrared Physics & Technology 48 pp. 77-82, 2005.

[29] Clemens Arth, "Visual survelliance on DSP based embedded platforms", PhD Thesis, Graz University of Technology, Austria, March 2008.

[30] Texas Instruments. C6Accel Advanced Users Guide. URL: http://processors.wiki.ti.com/index.php/C6Accel_Advanced_Users_Guide#Adding_a_new_kernel.2Flibrary_to_C6Accel

[31] ARM compiler shoot-out. URL: http://hardwarebug.org/2009/08/05/arm-compiler-shoot-out/#more-150

[32] Setting up Angstrom Distribution on BeagleBoard. URL: http://www.angstrom-distribution.org/demo/beagleboard/

[33] Embedded Linux Wiki. BeagleBoard. URL: http://elinux.org/BeagleBoard

[34] TI, C6EZAccel FAQ, http://processors.wiki.ti.com/index.php/C6EZAccel_FAQ