PERFORMANCE-ANALYSIS-BASED ACCELERATION OF IMAGE QUALITY ASSESSMENT

By
THIEN DUC PHAN
Bachelor of Science in Information Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Master of Science
December, 2014

PERFORMANCE-ANALYSIS-BASED ACCELERATION OF IMAGE QUALITY ASSESSMENT

Dissertation Approved:

Dr. Damon M. Chandler

Advisor

Dr. Guolian Fan

Dr. Keith Teague

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Damon M. Chandler, for his limitless patience and expert guidance. He is the greatest advisor I have ever had; without him, my research would not have been this far. This work would have never come to be truth without his guidance.

I would also like to thank Dr. Guoliang Fan, a great professor that teaches a couple of very difficult but so interesting and useful courses, computer vision and pattern recognition and machine learning, for all his knowledge and support. These courses also provided some ideas for Eric Larson to complete Most Apparent Distortion (MAD) algorithm, which is an important factor of my thesis.

I would like to thank Dr. Keith Teague for all his help and support, and for all his comments and guidelines for this thesis document. In the past, now, and future, I still need you on my side. Thank you.

I also would like to thank Dr. Sohum Sohoni and Eric Larson, who have helped me to finish the paper *Performance-Analysis-Based Acceleration of Image Quality Assessment* to be in IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI) conference.

This material is based upon work supported by, or in part by, the National Science Foundation, *Performance-Analysis-Based Acceleration of Image Quality Assessment*, Award #1054612, and by the U.S. Army Research Laboratory (USARL) and the U.S. Army Research Office (USARO) under contract/grant number W911NF-10-1-0015.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: THIEN DUC PHAN

Date of Degree: DECEMBER, 2014

Title of Study: PERFORMANCE-ANALYSIS-BASED ACCELERATION OF IMAGE QUALITY ASSESSMENT

Major Field: Electrical Engineering

Algorithms for image/video quality assessment (QA) aim to predict the qualities of images in a manner that agrees with subjective quality ratings. Over the last several decades, the major impetus in QA research has focused on improving predictive performance; very few studies have focused on analyzing and improving the runtime performance of QA algorithms. Modern algorithms of image/video quality assessment commonly employed two stages: (1) a local frequency-based decomposition, and (2) block-based statistical comparisons between the frequency coefficients of the reference and distorted images. These two stages constitute the bulk of the computation and runtime required for QA. This research thesis presents a performance analysis of and techniques for accelerating these stages. We also specifically analyze and accelerate one representative QA algorithm, Most Apparent Distortion (MAD), which was developed by Eric Larson and Damon Chandler in 2010 [1]. We identify the bottlenecks from the above-mentioned stages, and we present methods of acceleration using generalized integral image, inline expansion, a GPGPU implementation, and other code modifications. We show how a combination of these approaches can yield a speedup of 47x.

The content of the report is divided into five different chapters. In Chapter 1, we present a general overview of QA algorithms, current work on improving the computational performance and execution time of QA algorithms, and an introduction to our work. In Chapter 2, we describe MAD algorithm, the first performance analysis, and the systems used to test the performance. In Chapter 3, we present generalized integral image and inline expansion techniques. In this chapter, we also provide the results of each technique in terms of speeding up running time. Chapter 4 provides GPGPU and some other code optimization techniques with the timing results. Finally, the conclusion are proposed in the Chapter 5 to summarize the report.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Current research on image/video quality assessment (QA) spans different areas, including improving predictive performance, applying image quality assessment (IQA) to video quality assessment (VQA), applying IQA and VQA into image/video acquisition, transmission, compression, restoration and enhancement; very few studies have focused on analyzing and improving the runtime performance of QA algorithms. However, as QA moves from the research community into more mainstream applications, the bottlenecks of current algorithms are preventing widespread adoption. This report focuses on these two goals: analyzing and improving the runtime performance of one IQA algorithm, Most Apparent Distortion (MAD) [1], which is currently the best predictive performance IQA algorithm. While our implementation is specific to MAD, our methodology is straightforward to apply to a variety of QA algorithms; the analysis and results presented here can also provide insight into how other related algorithms might be accelerated.

In this chapter, we first introduce some background information on IQA and VQA algorithms in terms of runtime performance. There are many different approaches, yet, these QA algorithms share the same stages in algorithmic theory. We then present the need for acceleration of QA and current work on this area. Finally, our contribution to the research community will be proposed in detail in the following chapters.

1.1 Background on image/video quality assessment algorithms

Most IQA algorithms are so-called *full-reference* (FR) algorithms, which take as input a reference image and a processed (usually distorted) image, and yield as output either a scalar value denoting the overall visual quality or a spatial map denoting the local quality of each image region ([2, 3, 4, 5, 6, 1]). More recently, researchers have begun to develop *no-reference* (NR) ([7, 8, 9]) and *reduced-reference* (RR) algorithms ([10, 11, 12]), which attempt to yield the same quality estimates either by using only the processed/distorted image (no-reference IQA), or by using the processed/distorted image and only partial information about the reference image (reduced-reference IQA).

VQA algorithms can be classified into the same categories (FR, NR, or RR VQA). A natural technique is is to apply existing IQA algorithms to each frame of the video and to pool the per-frame results across time. A better approach is to extract and compare visual/quality features from localized space-time regions or groups of video frames or using spatiotemporal slices or motion information ([13, 14, 15]). See [16, 17, 18, 19] for more recent reviews on IQA/VQA.

Despite the fact that all IQA and VQA algorithms employ different approaches, they share two common algorithmic operations: (1) a local frequency-based decomposition (filtering/filterbanks or transforming), and (2) block-based statistical comparisons between the frequency coefficients of the reference and distorted images/videos (statistical computation). The block diagrams of some popular algorithms are provided in Figure 1.1-1.3 for us to have a general overview of these stages.

The Figure 1.1 shows the block diagram of the multi-scale structural similarity algorithm (MS-SSIM) [5]. MS-SSIM extends the original SSIM algorithm [20], which estimates the image quality based on the structure difference, by applying and combining SSIM for multiple scales. Specifically, the algorithm is implemented with five

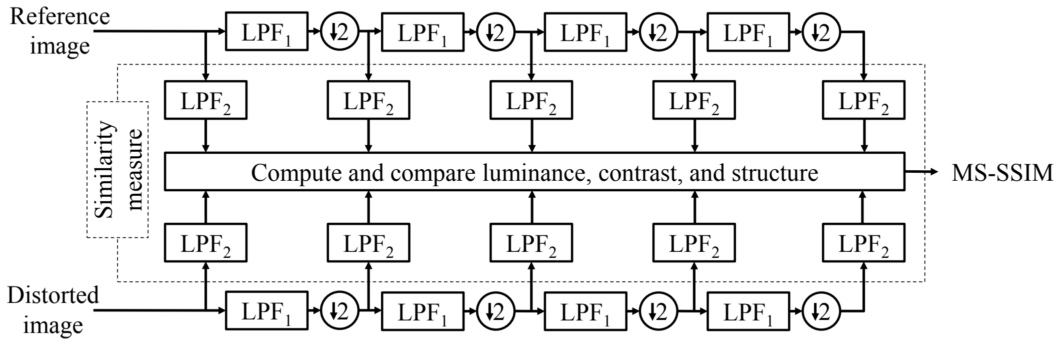


Figure 1.1: Diagram of the MS-SSIM algorithm. LPF_1 is a low-pass filter of size 2×2. $\downarrow 2$ is a downsampling by a factor of two. LPF_2 is a low-pass filter of size 11×11. The reference and distorted images serve as the first scale. The other four scales are obtained by applying LPF_1 and $\downarrow 2$ repeatedly. For each scale, the similarity between two images is measured by applying LPF_2 to prevent artifacts. Finally, the MS-SSIM index is formed via a combination of the luminance, contrast, and structure comparisons from different scales.

scales, in which the reference and distorted images serve as the first scale. To obtain the other four scales, a low-pass filter, LPF_1, and a downsampling by a factor of two are applied repeatedly. For each scale, a low-pass filter, LPF_2, of size 11×11 is applied to prevent artifacts from the discontinuous truncation of the image. This is the filtering/transforming stage. In the statistical comparisons stage, the luminance, contrast, and structure are computed and compared to yield a different map for each scale. Five maps of five scales are then combined and and collapsed to obtain the final MS-SSIM quality index.

The Figure 1.2 shows the block diagram of visual information fidelity algorithm (VIF) [6]. Using natural scene statistic models, VIF quantifies the loss of image information due to the distortion process by considering the relationship between image information, the amount of information shared between a reference and a distorted image, and visual quality. As seen from the figure 1.2, in the filtering/transforming stage, VIF uses the Steerable Pyramid [21] to model the image information in wavelet domain. This transforming step is employed with four scales and six orientations of decomposition and yields 24 subbands. In the statistical computation stage, each

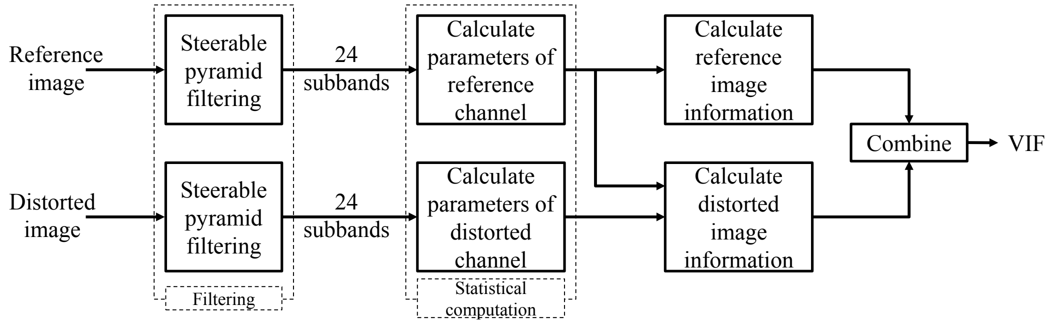

Reference image
Steerable pyramid filtering
24 subbands
Calculate parameters of reference channel
Calculate reference image information
Combine
VIF
Distorted image
Steerable pyramid filtering
24 subbands
Calculate parameters of distorted channel
Calculate distorted image information
Filtering
Statistical computation

Figure 1.2: The block diagram of VIF algorithm. First, two input images are filtered via a six-orientation and four-level Steerable Pyramid. The parameters of reference and distorted channels are calculated from the filtered images. Finally, the information of reference and distorted images are calculated and collapsed into a VIF index.

reference subband is modeled using a Gaussian scale mixtures model, as one random field (RF) which is a product of two independent RFs: a positive scalar and a Gaussian vector with zero-mean and a covariance matrix. The distorted subbands are modeled using the same strategy, with an additional noise: a stationary additive zero mean Gaussian noise RF in the same wavelet domain. From the parameters calculated here, the reference and distorted image information are given. Finally, the image information is summed over subbands, and the VIF index is given by the ratio between the distorted image information and reference image information.

The Figure 1.3 shows the block diagram of Most Apparent Distortion (MAD) algorithm [1]. MAD uses two strategies to estimate image quality. First, a detection-based strategy is used for nearthreshold distortions. In this case, the HVS attempts to look past the image and look for the distortions. Second, an appearance-based strategy is used for clearly visible distortions, and thus the HVS attempts to look past the distortion and look for the images subject matter. As seen from the Figure 1.3, MAD also consists of two stages, filtering/transforming and statistical computation. The detection-based strategy employs a contrast sensitivity function (CSF) (filtering/transforming) before computing visible distortion map (statistical computation). The appearance-based strategy employs a Log-Gabor filtering (filter-

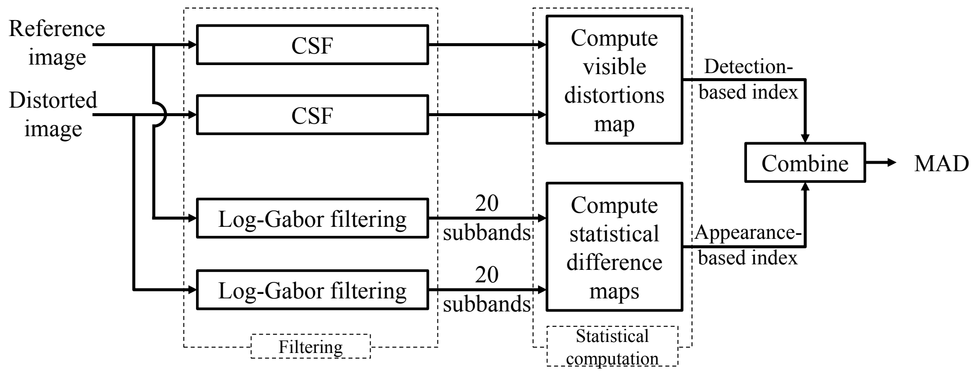


Figure 1.3: The diagram of MAD algorithm. For detection-based stage, reference and distorted images are first filtered using a contrast sensitivity function. The distortion map is then computed from filtered images and collapsed via a MSE measure to obtain a detection-based index. For the appearance-based stage, both images first are filtered using Log-Gabor with five scales and four orientations. The statistical difference map is computed from the 20 filtered subbands and then collapsed into a appearance-based index. Finally, the MAD index is given by taking a weighted geometric mean of the appearance-based index and detection-based index.

ing/transforming), and then applies a statistical difference map computing for each pair of reference and distorted subbands (statistical computation). MAD then collapses all the maps and combines two indices from two strategies into a single MAD quality index via a weighted geometric mean.

Current research on QA focuses on improving predictive performance; there are very few studies have focused on analyzing and improving the runtime performance of QA algorithms. Our report focuses on the goal of analyzing and improving the runtime performance of IQA algorithm. We chose Most Apparent Distortion (MAD) [1], which is currently the best predictive performance IQA algorithm. While our implementation is specific to MAD, our methodology is straightforward to apply to a variety of QA algorithms; the analysis and results presented here can also provide insight into how other related algorithms might be accelerated.

1.2 The need for acceleration of QA

IQA algorithms have been employed for a wide variety of applications ranging from image compression (e.g., [22, 23, 24]), to denoising (e.g., [25]), to gauging intelligibility in sign language video [26]. Modern IQA algorithms such as VIF [27], MS-SSIM [28], and MAD [29] are quite effective at QA, and they have been adapted for those applications. However, they require a relatively large run-time—on the order of seconds for even modest-sized images (e.g., less than 1MPixels). Nowadays, even a smartphone can capture/process a very large image (e.g., 4-20 MPixels), and thus, the run-time is very important because users do not want to wait too long to process an image.

These IQA algorithms are also adapted to process frames of video (e.g., MOVIE [30], ST-MAD [15]). For a 25 frame per second video of size 768×432, which is 10-second long, MOVIE needs approximately five hours to compute the quality index. For the same video, ST-MAD also needs approximately eight hours. It must be frustrating for the developers to test these VQA algorithms, just because of waiting time. Therefore, the run-time issues become of greater importance.

1.3 Current work

IQA algorithms have been used extensively in image compression and computer vision, and thus, a considerable amount of research has focused on accelerating two-dimensional image transforms which provide local frequency-based decompositions. For example, the Discrete Cosine Transform (DCT) has been accelerated at the algorithm level by using variations of the same techniques used in the Fast Fourier Transform (FFT) (e.g., [31]) and by exploiting various algebraic and structural properties of the transform, e.g., via recursion [32], lifting [33], matrix factorization [34], cyclic convolution [35], and many other techniques (see [36] for a review). Numerous techniques for hardware-based acceleration of the DCT have also been proposed using

GPGPU-based and FPGA-based implementations (e.g., [37, 38, 39, 40]). Algorithm- and hardware-based acceleration has also been researched for the Discrete Wavelet Transform (DWT) (e.g., [41, 42, 43]) and Gabor transform (e.g., [44, 45, 46, 47]).

Techniques for accelerating the computation of local statistics in images has also been researched, though to a much lesser extent than the transforms. One technique, called *integral image*, which was originally developed in the context of computer graphics [48], has emerged as a popular approach for computing block-based sums of any two-dimensional matrix of values (e.g., a matrix of pixels or coefficients). The integral image, also known as the *summed area table*, requires first computing a table which has the same dimensions as the input matrix, and in which each value in the table represents the sum of all matrix values above and to the left of the current position. Thereafter, the sum of values within any block of the matrix can be rapidly computed via addition/subtraction of three values in the table. This technique has been employed widely, for example, fast feature-based block matching [49], rapid object detection [50], and local adaptive thresholding [51] .

In [52], Chen and Bovik presented the Fast SSIM and Fast MS-SSIM algorithms, which are accelerated versions of SSIM and MS-SSIM, respectively. Three modifications were used for Fast SSIM: (1) The luminance component of each block was computed by using an integral image. (2) The contrast and structure components of each block were computed based on 2×2 Roberts gradient operators. (3) The Gaussian-weighting window used in the contrast and structure components was replaced with an integer approximation. For Fast MS-SSIM, a further algorithm-level modification of skipping the contrast and structure computations at the finest scale was proposed. By using these modifications, Fast SSIM and Fast MS-SSIM were shown to be, respectively, 2.7x and 10x faster than their original counterparts on 768x432 frames from videos of the LIVE Video Quality database [53]. Although algorithm-level modifications were used, the authors demonstrated that these modifications had negligible impact on predictive performance; testing on the LIVE Image Quality and Video Quality databases revealed effectively no impact on Spearman rank-order correlation coefficient, Pearson correlation coefficient, and root-mean-square error. By further implementing the calculations of the contrast and structure components via Intel SSE2 (SIMD) instructions, speedups of approximately 5x for Fast SSIM and 14x for Fast MS-SSIM were reported. In addition, speedups of approximately 17x for Fast SSIM and 50x for Fast MS-SSIM were reported by further employing parallelization via a multithreaded implementation.

In [54], Okarma and Mazurek presented GPGPU techniques for accelerating SSIM, MS-SSIM, and CVQM (a video quality assessment algorithm developed previously by Okarma, which uses SSIM, MS-SSIM, and VIF to estimate quality). To accelerate the computation of both SSIM and MS-SSIM, the authors described a CUDA-based implementation in which separate GPU threads were used for computing SSIM or MS-SSIM on strategically sized fragments of the image. To overcome CUDA's memory-bandwidth limitations, the computed quality estimates for the fragments were stored in GPU registers and transferred only once to the system memory. Okarma and Mazurek reported that their GPGPU-based implementations resulted in 150x and 35x speedups of SSIM and MS-SSIM, respectively.

1.4 What we do

Several papers have focused on accelerating the filtering/transforming stage (e.g., [31, 32, 33, 34]), some have focused on local statistics computation (e.g., [48, 49]), some other have focused in the hardware and multithreaded implementation (e.g., [54, 37, 38, 39, 40]), assuming that the original algorithms are fully optimized, only a few have done speeding up a whole IQA algorithm (such as in [52]). From a developer viewpoint, software optimization needs to be considered before any hardware or multithreading coding. Moreover, we also need to avoid changing the algorithm's predictive performance, (not to use approximation such as an integer approximation of the Gaussian window in [52]).

In this report, we present a performance analysis of and methods for accelerating a representative QA algorithm recently developed by Eric Larson and Damon Chandler, MAD [1]. MAD employs a log-Gabor decomposition and a comparison of local statistical differences between blocks of log-Gabor coefficients of the reference and distorted images; it is thus an appropriate representative algorithm for the stages performed in QA.

We present a technique to identify the bottlenecks in MAD, and then we present four methods for accelerating the algorithm by attacking each of the found bottlenecks: (1) Using generalized integral image for the statistical computations; (2) using procedure expansion and strength reduction; (3) using a general-purpose-GPU (GPGPU) implementation of the log-Gabor decomposition; and (4) precomputation and caching of the log-Gabor filters. As we will show, a combination of these approaches can result in a nearly 47x speed increase. While our implementation is specific to MAD, our methodology is straightforward to apply to a variety of QA algorithms; the analysis and results presented here can also provide insight into how other related algorithms might be accelerated.

The rest of the report is organized as follows: In Chapter 2, we describe MAD algorithm, the first performance analysis, and the bottleneck results. We also describe the systems which are used to test the run-time performance in this chapter. In Chapter 3, we first present the integral image technique, and then a new-developed integral image technique, generalized integral images, for higher moments. The inline expansion technique and the results of each technique (in terms of speeding up running time) are also provided in this chapter. Chapter 4 provides GPGPU and some other code optimization techniques with the timing results. Finally, the conclusion is proposed in the Chapter 5 to summarize the report.

CHAPTER 2

ALGORITHM AND ANALYSIS

2.1 Most Apparent Distortion (MAD) algorithm

The Most Apparent Distortion (MAD) algorithm [1], agues that human employs multiple strategies to assess image quality. Specifically, for slightly distorted (*high-quality*) images, human looks for distortions, and for heavily distorted (*low-quality*) images, human leans to the task of image contents recognition. Consequently, the MAD algorithm consists of two assessment stages: (1) a detection-based stage, which estimates quality based on the extent to which the distortions are visible; and (2) an appearance-based stage, which estimates quality based on the extent to which the *image* is recognizable.

The block diagram of MAD is shown in Fig. 2.1 with details of the two stages. The detection-based stage (compute detection-based difference map) uses a masking-weighted block-based MSE computed in the lightness domain (contrast sensitivity function, and then a least mean square). The appearance-based stage (Compute appearance-based map) computes the average difference between the block-based log-Gabor statistics of the original image to those of the distorted image.

The appearance-based stage, which employs a computational neural model using a log-Gabor filterbank with both even-symmetric and odd-symmetric filters applied using the FFT [55], is believed to the slowest stage of MAD. The even and odd filter outputs are combined to yield magnitude-only subband values. The standard deviation, skewness, and kurtosis computed for each 16×16 block (with 75% overlap between blocks) of each subband of the original image are compared to corresponding

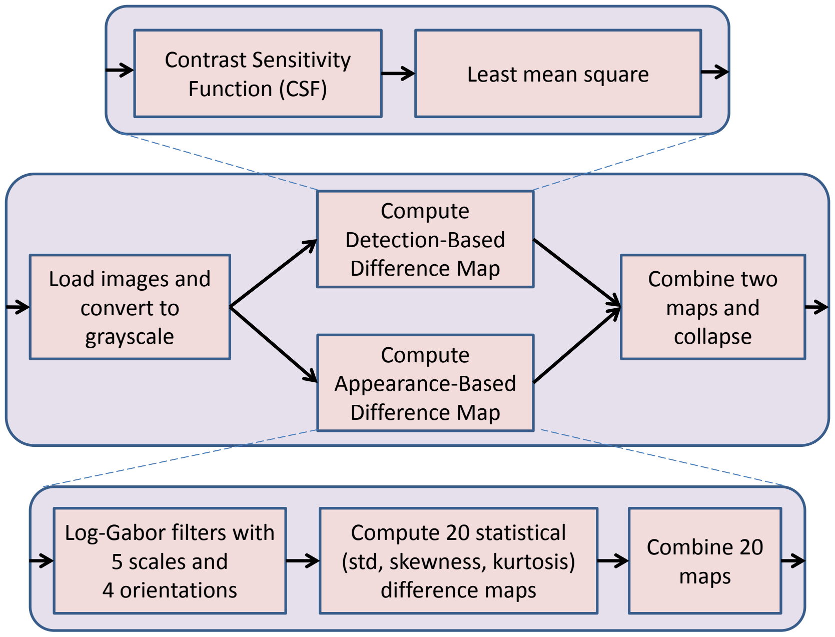

Contrast Sensitivity Function (CSF)
Least mean square
Load images and convert to grayscale
Compute Detection-Based Difference Map
Compute Appearance-Based Difference Map
Combine two maps and collapse
Log-Gabor filters with 5 scales and 4 orientations
Compute 20 statistical (std, skewness, kurtosis) difference maps
Combine 20 maps

Figure 2.1: Block diagram of the MAD algorithm.

values computed for the distorted image. The differences between these statistics are then collapsed via a 2-norm to yield a scalar output corresponding to appearance-based differences.

2.2 Testing systems

The performance analysis was executed on a Dell Inspiron 580 desktop, and for verification on two additional systems, which are a laptop and a server. We also need a good graphic card for testing GPU performance, only the desktop, which includes Quadro Q9400 graphic card can do this. Details are provided in Figures 2.2-2.4. Three systems were chosen with different configurations, such as memory (4 GB, 5 GB, or 8 GB), operating systems (Windows 7 or Windows server 2003), architectures (32-bit vs. 64-bit), and simulator versions (Matlab 2009a vs. Matlab 2011a). A few basic specifications are provided in Table 2.1 for these systems.

Table 2.1: Three systems with different configurations, such as memory (4 GB, 5 GB, or 8 GB), operating systems (Windows 7 or Windows server 2003), architectures (32-bit vs. 64-bit), and simulator versions (Matlab 2009a vs. Matlab 2011a). System 1 was used for the main performance analysis; Systems 2 and 3 were used for the additional timing results.

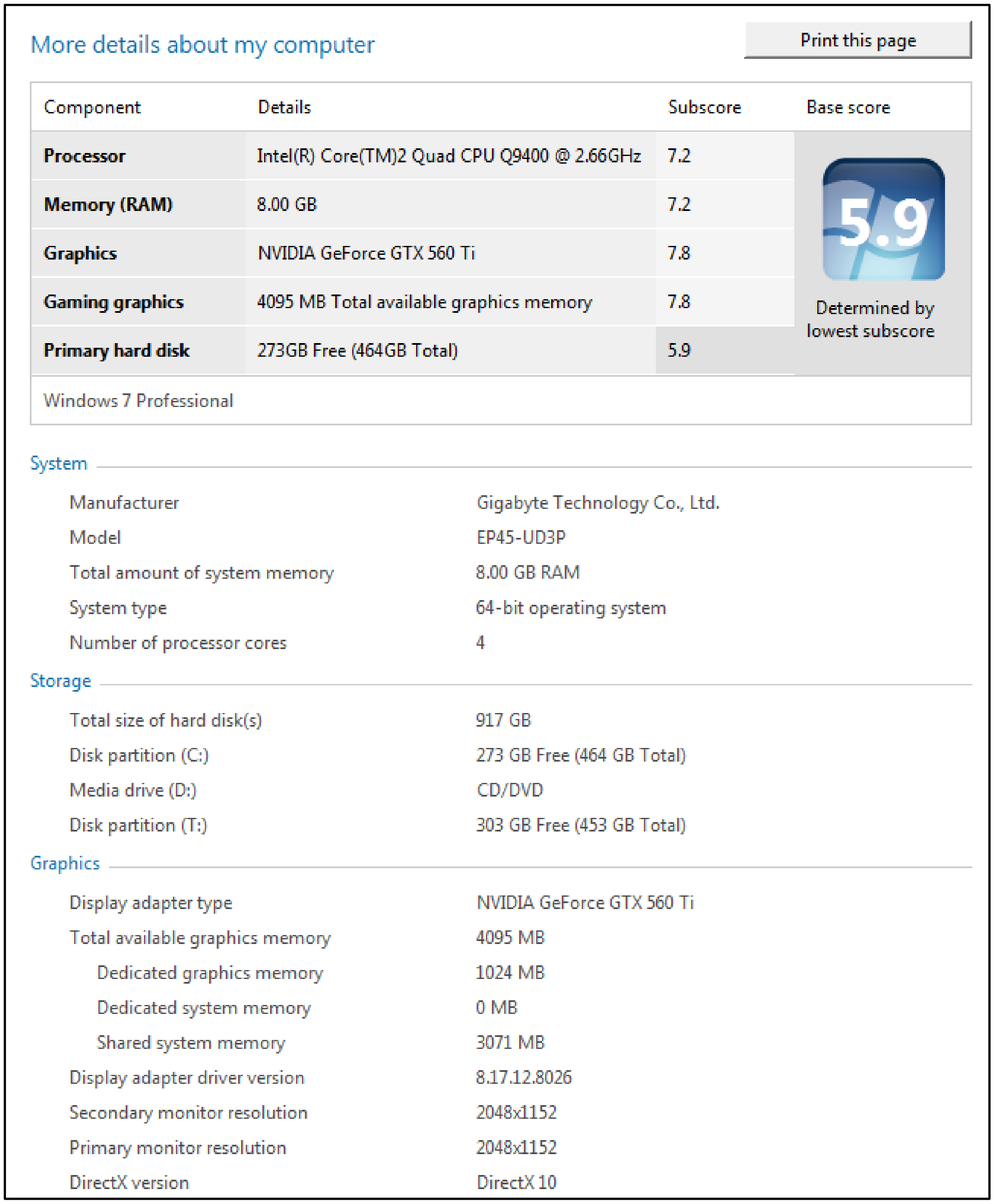
	CPU and RAM	OS and Matlab
System 1 (Desktop)	Intel Core 2 Quad Q9400 2.66 GHz 8 GB 1333-MHz DDR3	Win. 7, 64-bit Matlab 2011a
System 2 (Laptop)	Intel Core 2 Duo T6400 2 GHz 4 GB 1333-MHz DDR3	Win. 7, 32-bit Matlab 2009a
System 3 (Server)	Two Intel Xeon 5050 3 GHz 5 GB 667-MHz DDR2	Win. Server 2003 Matlab 2009a

2.3 Original MAD performance

The original version of MAD was implemented in Matlab, with C++ MEX files used for computation of the block-based statistics. To identify bottlenecks, we performed a timing analysis using the profiler in Matlab. The MEX files were compiled using Microsoft Visual C++ 2008 using Matlab's default optimization flags `/O2 /Oy-`.

We ran MAD on 180, 512×512-pixel images from the CSIQ database [56]. Figure 2.5 shows the average results from this analysis for System 1 (similar distributions of time were observed on all three systems); the average total execution time per image is 55.85 seconds.

Figure 2.5 shows hotspots (bottlenecks) of several blocks of MAD. The more time-consuming blocks, the hotter color they are. As seen in Figure 2.5(a), over 98% of this time is spent in the appearance-based stage. This stage consists of three sub-stages: a log-Gabor decomposition, computation of block-based statistics of the log-Gabor subbands to generate statistical difference maps, and combining/collapsing the maps into a final scalar output. As shown in Figure 2.5(b), computation of the statistical difference maps alone consumes over 93% of MAD's total execution time. Thus, performance improvement should focus on this part of the algorithm, statistical



More details about my computer

Component	Details	Subscore	Base score
Processor	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz	7.2	5.9 Determined by lowest subscore
Memory (RAM)	8.00 GB	7.2	
Graphics	NVIDIA GeForce GTX 560 Ti	7.8	
Gaming graphics	4095 MB Total available graphics memory	7.8	
Primary hard disk	273GB Free (464GB Total)	5.9	

Windows 7 Professional

System

Manufacturer	Gigabyte Technology Co., Ltd.
Model	EP45-UD3P
Total amount of system memory	8.00 GB RAM
System type	64-bit operating system
Number of processor cores	4

Storage

Total size of hard disk(s)	917 GB
Disk partition (C:)	273 GB Free (464 GB Total)
Media drive (D:)	CD/DVD
Disk partition (T:)	303 GB Free (453 GB Total)

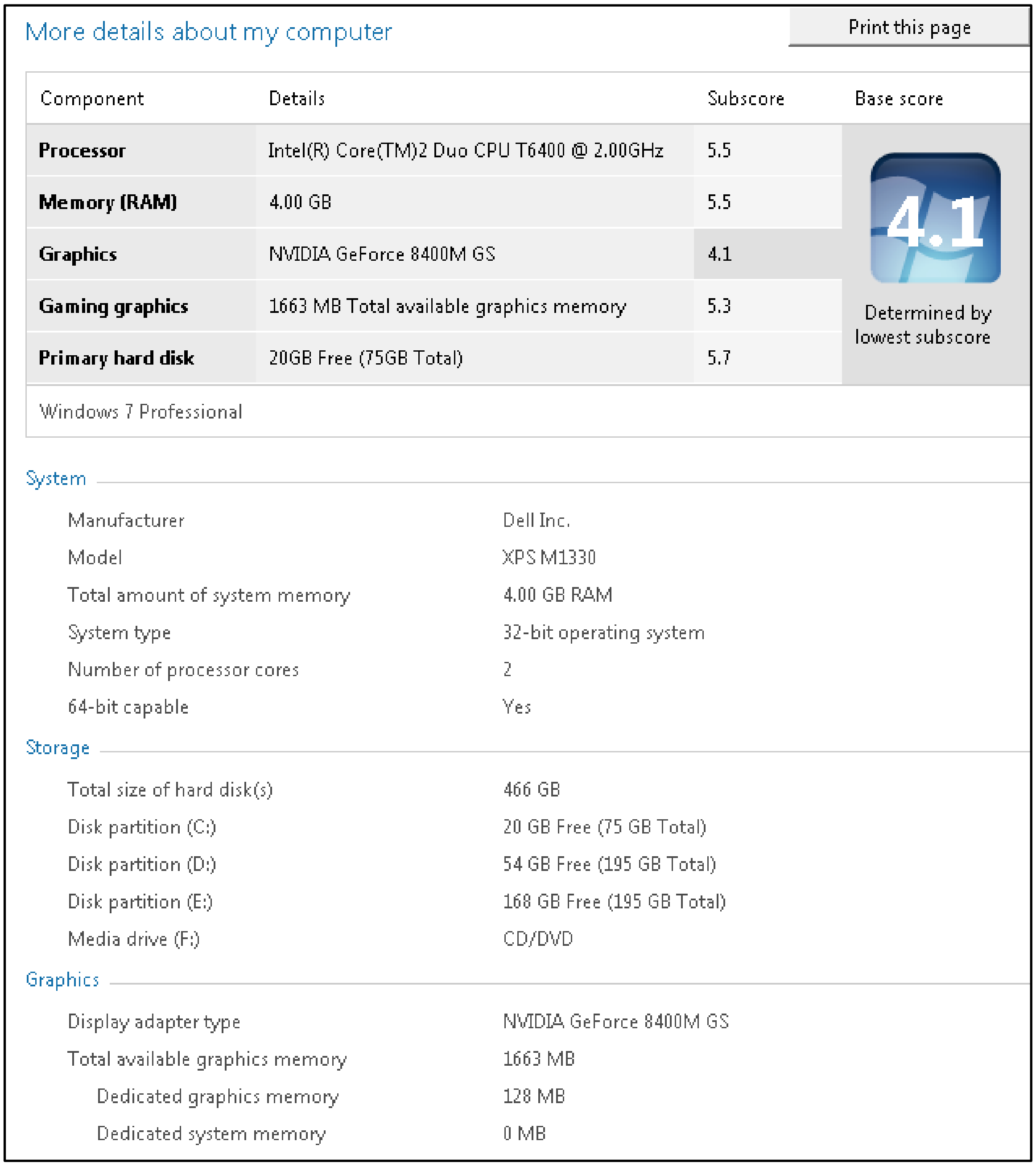
Graphics

Display adapter type	NVIDIA GeForce GTX 560 Ti
Total available graphics memory	4095 MB
Dedicated graphics memory	1024 MB
Dedicated system memory	0 MB
Shared system memory	3071 MB
Display adapter driver version	8.17.12.8026
Secondary monitor resolution	2048x1152
Primary monitor resolution	2048x1152
DirectX version	DirectX 10

Figure 2.2: System 1, the desktop employed in the experiment.

difference maps computation.

The following chapters will provide in details some techniques to speed up the MAD algorithm, such as generalized integral images, inline expansion, GPU and code



Component	Details	Subscore	Base score
Processor	Intel(R) Core(TM)2 Duo CPU T6400 @ 2.00GHz	5.5	4.1 Determined by lowest subscore
Memory (RAM)	4.00 GB	5.5	
Graphics	NVIDIA GeForce 8400M GS	4.1	
Gaming graphics	1663 MB Total available graphics memory	5.3	
Primary hard disk	20GB Free (75GB Total)	5.7	

Windows 7 Professional

System

Manufacturer	Dell Inc.
Model	XPS M1330
Total amount of system memory	4.00 GB RAM
System type	32-bit operating system
Number of processor cores	2
64-bit capable	Yes

Storage

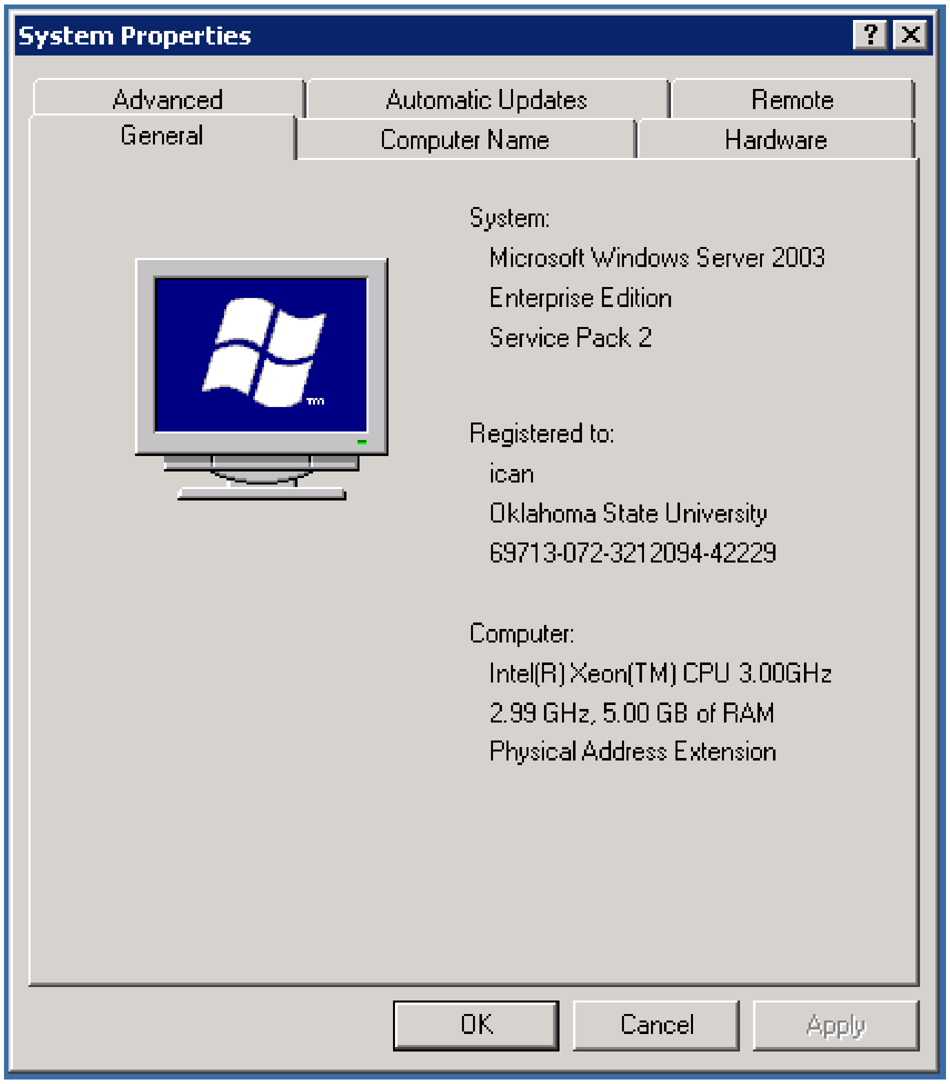
Total size of hard disk(s)	466 GB
Disk partition (C:)	20 GB Free (75 GB Total)
Disk partition (D:)	54 GB Free (195 GB Total)
Disk partition (E:)	168 GB Free (195 GB Total)
Media drive (F:)	CD/DVD

Graphics

Display adapter type	NVIDIA GeForce 8400M GS
Total available graphics memory	1663 MB
Dedicated graphics memory	128 MB
Dedicated system memory	0 MB

Figure 2.3: System 2, the laptop employed in the experiment.

optimizations. The contribution of each of the technique to the speed-up process is also provided on these chapters.



System:
Microsoft Windows Server 2003
Enterprise Edition
Service Pack 2

Registered to:
ican
Oklahoma State University
69713-072-3212094-42229

Computer:
Intel(R) Xeon(TM) CPU 3.00GHz
2.99 GHz, 5.00 GB of RAM
Physical Address Extension

Figure 2.4: System 3, the server employed in the experiment.

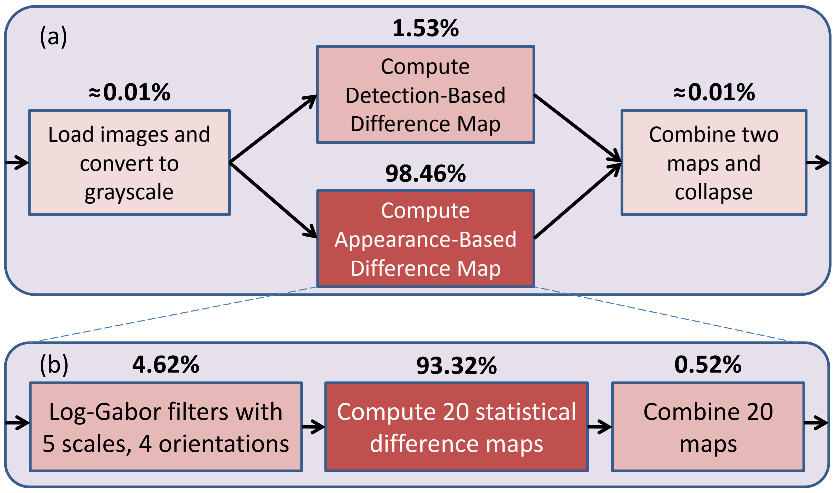

(a)
≈0.01%
Load images and convert to grayscale
1.53%
Compute Detection-Based Difference Map
98.46%
Compute Appearance-Based Difference Map
≈0.01%
Combine two maps and collapse
(b)
4.62%
Log-Gabor filters with 5 scales, 4 orientations
93.32%
Compute 20 statistical difference maps
0.52%
Combine 20 maps

Figure 2.5: (a) Profiling analysis showing average execution time in each stage for System 1; the average total execution time per image is 55.85 seconds. (b) Breakdown of the bottleneck, the appearance-based stage.

CHAPTER 3

GENERALIZED INTEGRAL IMAGES AND INLINE EXPANSION

This chapter provides two methods to speed up statistical difference maps computation stage: *Generalized Integral images* and *inline expansion*. The generalized integral images technique attempts to reduce the computational complexity (e.g. reduce the number of addition, subtraction, multiplication, division and other operations), and the inline expansion is employed to save the overhead related to the procedure call, and for strength reduction.

3.1 Integral image

To accelerate the computation of the statistical difference maps, we employ and build upon a technique called *integral image* originally developed in the context of computer graphics [48]. The integral image, which is also known as the *summed area table*, is an algorithm for quickly computing the sum of values within any block of an image.

The integral image was designed to compute a block's sum, whereas MAD requires standard deviation, skewness, and kurtosis. In this section, we first review the integral image technique, and then we present our new integral images, which developed for quickly computing local standard deviation, skewness and kurtosis.

3.1.1 Original integral image

Let I denote an image (or subband) for which one needs to compute block-based sums. The integral image M has the same dimensions as I, but with each pixel value at position (x, y) given by:

$$M(x, y) = \sum_{y'=1}^{y} \sum_{x'=1}^{x} I(x', y'). \tag{3.1}$$

Another word, the value at any point (x, y) in the integral image M (or summed area table) is just the sum of all the pixels in I in the area from (1,1) to (x, y).

The integral image M can be computed efficiently in a single pass over the image I, using the fact that the value in M at (x, y) can be computed from its previous values, at (x-1,y-1), (x,y-1), and (x-1,y), and a value in I at (x,y) as follow:

$$M(x, y) = I(x, y) + M(x-1, y) + M(x, y-1) - M(x-1, y-1) \tag{3.2}$$

Given M as defined above, the sum s of all pixel values located in the rectangle (x_1,y_1), (x_2,y_2) is given by

$$\begin{aligned} s &= \sum_{y'=y_1}^{y_2} \sum_{x'=x_1}^{x_2} I(x', y') \\ &= \sum_{y'=1}^{y_2} \sum_{x'=1}^{x_2} I(x', y') + \sum_{y'=1}^{y_1} \sum_{x'=1}^{x_1} I(x', y') - \sum_{y'=1}^{y_1} \sum_{x'=1}^{x_2} I(x', y') - \sum_{y'=1}^{y_2} \sum_{x'=1}^{x_1} I(x', y') \\ &= M(x_2, y_2) + M(x_1, y_1) - M(x_1, y_2) - M(x_2, y_1) \end{aligned}$$

The formula of the sum s gives us a faster way to calculate the sum of any block by calculating four operations (additions and subtractions), by pre-calculating integral image M one time.

A further reduction in computation can be achieved by capitalizing on the fact that MAD does not need to compute the statistics of *every* block. MAD calculates the statistical maps using blocks of size 16×16 pixels, with 12 pixels of overlap between neighboring blocks. Accordingly, we do not need the entire integral image, just subsets of it required to support the computation of the statistics within these particular blocks. In this case, the integral image M has dimensions 1/4 of the size of I, and is given as follows:

$$M(x,y) = \sum_{y' \leq 4y} \sum_{x' \leq 4x} I(x', y'). \tag{3.3}$$

3.1.2 Generalized integral images

The MAD algorithm requires standard deviation, skewness and kurtosis of of all blocks of size 16×16 pixels, with 12 pixels of overlap between neighboring blocks. Therefore, we need an upgrade of integral image technique, generalized integral images, for those statistical computations. Specifically, let $\mathbf{b}$ denote a $N_1 \times N_2$ block of I. The standard deviation, skewness, and kurtosis of $\mathbf{b}$ are given by

$$\sigma_{\mathbf{b}} = \sqrt{\frac{1}{N_1 N_2} \sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^2}, \tag{3.4}$$

$$\varsigma_{\mathbf{b}} = \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^3, \tag{3.5}$$

$$\kappa_{\mathbf{b}} = \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^4} \sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^4, \tag{3.6}$$

where b_i and $\bar{\mathbf{b}}$ denote the i^{th} pixel and mean of $\mathbf{b}$.

We developed generalized integral images, the extension of the integral image, to accelerate the computation of Equations (3.4)-(3.6). This modification requires computing integral images for the image and powers of the image up to power 4. Specifically, let M_1, M_2, M_3, and M_4 denote the integral image computed for I, I^2, I^3, and I^4, respectively. Let s_1, s_2, s_3, and s_4 denote sums of the values (over the same coordinates as block $\mathbf{b}$) in I, I^2, I^3, and I^4, respectively. The integral images M_1, M_2, M_3, and M_4 are given by:

$$M_i(x,y) = \sum_{y' \leq 4y} \sum_{x' \leq 4x} I^i(x', y'). \tag{3.7}$$

To compute the standard deviation, we manipulate Equation 3.4 as follows:

$$\begin{aligned}
\sigma_{\mathbf{b}} &= \sqrt{\frac{1}{N_1N_2}\sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^2}, \\
&= \sqrt{\frac{1}{N_1N_2}\sum_i \left(\mathbf{b}_i^2 - 2\mathbf{b}_i\bar{\mathbf{b}} + \bar{\mathbf{b}}^2\right)}, \\
&= \sqrt{\frac{1}{N_1N_2}\left(\sum_i \mathbf{b}_i^2 - 2\sum_i \mathbf{b}_i\bar{\mathbf{b}} + N_1N_2\bar{\mathbf{b}}^2\right)}, \\
&= \sqrt{\frac{1}{N_1N_2}\left(s_2 - 2s_1\frac{s_1}{N_1N_2} + N_1N_2\left(\frac{s_1}{N_1N_2}\right)^2\right)}, \\
&= \sqrt{\frac{1}{N_1N_2}\left(s_2 - \frac{s_1^2}{N_1N_2}\right)}. \qquad (3.8)
\end{aligned}$$

Originally, to compute $\sigma_{\mathbf{b}}$ as in Equation 3.4, it requires approximately $N_1 \times N_2$ additions for the mean $\bar{b}$, $N_1 \times N_2$ subtractions of b_i and $\bar{b}$, $N_1 \times N_2$ times for square operation, $N_1 \times N_2$ times to compute the sum squared, two divisions and one square root. However, with the integral images M_1 and M_2 (to compute the sums s_1 and s_2 efficiently with four additions and subtractions), the $\sigma_{\mathbf{b}}$ in Equation 3.8 can be computed with one square operation, one subtractions, four divisions, and one square root. We reduced the number of operations required, and thus reduce computational complexity.

Similar manipulation of Equations 3.5 for local skewness $\varsigma_{\mathbf{b}}$ is given by:

$$
\begin{aligned}
\varsigma_{\mathbf{b}} &= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^3, \\
&= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \sum_i \left(\mathbf{b}_i^3 - 3\mathbf{b}_i^2\bar{\mathbf{b}} + 3\mathbf{b}_i\bar{\mathbf{b}}^2 - \bar{\mathbf{b}}^3\right), \\
&= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \left(\sum_i \mathbf{b}_i^3 - 3\sum_i \mathbf{b}_i^2\bar{\mathbf{b}} + 3\sum_i \mathbf{b}_i\bar{\mathbf{b}}^2 - \sum_i \bar{\mathbf{b}}^3\right), \\
&= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \left(\sum_i \mathbf{b}_i^3 - 3\bar{\mathbf{b}}\sum_i \mathbf{b}_i^2 + 3\bar{\mathbf{b}}^2\sum_i \mathbf{b}_i - N_1 N_2\bar{\mathbf{b}}^3\right), \\
&= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \left(s_3 - 3\bar{\mathbf{b}}s_2 + 3\bar{\mathbf{b}}^2 s_1 - N_1 N_2\bar{\mathbf{b}}^3\right), \\
&= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \left(s_3 - 3(\frac{s_1}{N_1 N_2})s_2 + 3(\frac{s_1}{N_1 N_2})^2 s_1 - N_1 N_2(\frac{s_1}{N_1 N_2})^3\right), \\
&= \frac{1}{N_1 N_2 \sigma_{\mathbf{b}}^3} \left(s_3 - 3\frac{s_1 s_2}{N_1 N_2} + 2\frac{s_1^3}{(N_1 N_2)^2}\right). \qquad (3.9)
\end{aligned}
$$

Again, if we compare Equations 3.5 and 3.9 in terms of number of operations. We can see that Equation 3.5 requires multiple times $N_1 \times N_2$ operations. However, Equation 3.9 requires only a few operations to obtain $\varsigma_{\mathbf{b}}$ values.

Similar manipulation of Equations 3.6 for local kurtosis $\kappa_{\mathbf{b}}$ is given by:

$$
\begin{aligned}
\kappa_{\mathbf{b}} &= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^4,, \\
&= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\sum_i \left(\mathbf{b}_i^4 - 4\mathbf{b}_i^3\bar{\mathbf{b}} + 6\mathbf{b}_i^2\bar{\mathbf{b}}^2 - 4\mathbf{b}_i\bar{\mathbf{b}}^3 + \bar{\mathbf{b}}^4\right), \\
&= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\left(\sum_i \mathbf{b}_i^4 - 4\sum_i \mathbf{b}_i^3\bar{\mathbf{b}} + 6\sum_i \mathbf{b}_i^2\bar{\mathbf{b}}^2 - 4\sum_i \mathbf{b}_i\bar{\mathbf{b}}^3 + \sum_i \bar{\mathbf{b}}^4\right), \\
&= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\left(\sum_i \mathbf{b}_i^4 - 4\bar{\mathbf{b}}\sum_i \mathbf{b}_i^3 + 6\bar{\mathbf{b}}^2\sum_i \mathbf{b}_i^2 - 4\bar{\mathbf{b}}^3\sum_i \mathbf{b}_i + N_1N_2\bar{\mathbf{b}}^4\right), \\
&= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\left(s_4 - 4\bar{\mathbf{b}}s_3 + 6\bar{\mathbf{b}}^2 s_2 - 4\bar{\mathbf{b}}^3 s_1 + N_1N_2\bar{\mathbf{b}}^4\right), \\
&= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\left(s_4 - 4(\frac{s_1}{N_1N_2})s_3 + 6(\frac{s_1}{N_1N_2})^2 s_2 - 4(\frac{s_1}{N_1N_2})^3 s_1 + N_1N_2(\frac{s_1}{N_1N_2})^4\right), \\
&= \frac{1}{N_1N_2\sigma_{\mathbf{b}}^4}\left(s_4 - 4\frac{s_1s_3}{N_1N_2} + 6\frac{s_2s_1^2}{(N_1N_2)^2} - 3\frac{s_1^4}{(N_1N_2)^3}\right). \qquad (3.10)
\end{aligned}
$$

One more time, if we compare Equations 3.6 and 3.10 in terms of number of operations. We can see that Equation 3.6 requires multiple times $N_1 \times N_2$ operations. However, Equation 3.10 requires only a small number of operations to obtain $\kappa_{\mathbf{b}}$ values.

The generalized integral images technique can be extended to higher moments. The key point of extending is expanding $\sum_i \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^m$ to use with m integral images, $M_1, M_2, ...M_m$ for any m. What we have presented is for $m = 1, 2, 3, 4$, but it can be extended to any $m > 4$.

The results of applying the new integral images technique will be presented in Section 3.3, in order to compare to *Procedure Inlining and Strength Reduction* technique.

3.2 Procedure inlining and strength reduction

In the original implementation of MAD, Equations 3.4, 3.5, and 3.6 are implemented using the `pow` function from the Standard C++ Library. Another avenue for accelerating the computation of the statistics is to replace the function call with a direct multiplication.

We therefore replaced the calls to `pow` with an inline expansion involving a direct multiplication (e.g., `pow(x,3)` was replaced with `x*x*x`). This modification not only saves the overhead related to the procedure call, but also offers strength reduction.

The `pow` functions in C++ and Matlab are expensive function. To compare the use of `pow` and direct multiplication, let look at the Matlab code:

```
tic;
a=rand(1024);
b=a.∧2;
toc;
Elapsed time is 0.035528 seconds.
```

and

```
tic;
a=rand(1024);
b=a.*a;
toc;
>> Elapsed time is 0.035528 seconds.
```

This case, there is not much different between `pow` of two and direct multiplication. However, if it is power of three:

```
tic;
a=rand(1024);
b=a.∧3;
toc;
Elapsed time is 0.105884 seconds.
```

and

```
tic;
a=rand(1024);
b=a.*a.*a;
toc;
>> Elapsed time is 0.042218 seconds.
```

The power function in Matlab shows a poor performance. The same pattern can be seen for power of four, via:

```
tic;
a=rand(1024);
b=a.∧4;
toc;
Elapsed time is 0.111544 seconds.
```

and

```
tic;
a=rand(1024);
b=a.*a.*a.*a;
toc;
>> Elapsed time is 0.042512 seconds.
```

3.3 Results

The timing results of these modifications are shown in Table 3.1 for all three systems, and the average speedup. The first column show the modification types (integral image and inline expansion individually, and a combination of the two). The first and second columns for each system show run-time in seconds and speedup, respectively.

Via Table 3.1, we can see that both approaches clearly provide a significant acceleration. However, it is important to note that the integral image (II) and inline expansion (IE) offer different forms of acceleration. The II aims to minimize the number of power (and other) computations, whereas the IE aims to make each power computation faster. Thus, even though the results shown in Table 3.1 indicate that the IE offers more acceleration, the II can offer more acceleration when the block size or number of blocks increases.[1] It is therefore prudent to use both approaches. Indeed, as shown in Table 3.1, when both modifications are employed, MAD is accelerated by nearly 11x on average (17x for System 1).

Figure 3.1 shows an updated version of Figure 2.5, indicating the distribution of time required by each portion of MAD when using II+IE. These latter results indicate that the bottleneck is no longer in the computation of the statistics, but is rather in the computation of the log-Gabor decomposition. In the following section, we describe a GPGPU-based acceleration of the log-Gabor decomposition. Several code optimizations and results of all the techniques are also provided in Section 4.2.

[1]For example, when using 16×16 blocks with 14 pixels (rather than 12 pixels) of overlap between neighboring blocks, the time required for the II approach increases by approximately 10%, while the time required for the IE approach increases by approximately 190%.

Table 3.1: Timing results of various modifications. The first and second columns for each system show run-time in seconds and speedup, respectively. II = Integral Image; IE = Inline Expansion of `pow`.

Modification	System 1		System 2		System 3		Average	
None	55.8	1.0x	56.42	1.0x	62.23	1.0x	58.15	1.0x
II	6.77	8.2x	8.84	6.4x	10.98	5.7x	8.86	6.6x
IE	3.77	14.8x	7.90	7.1x	9.45	6.6x	7.04	8.3x
II + IE	3.21	17.4x	5.46	10.3x	7.38	8.4x	5.35	10.9x

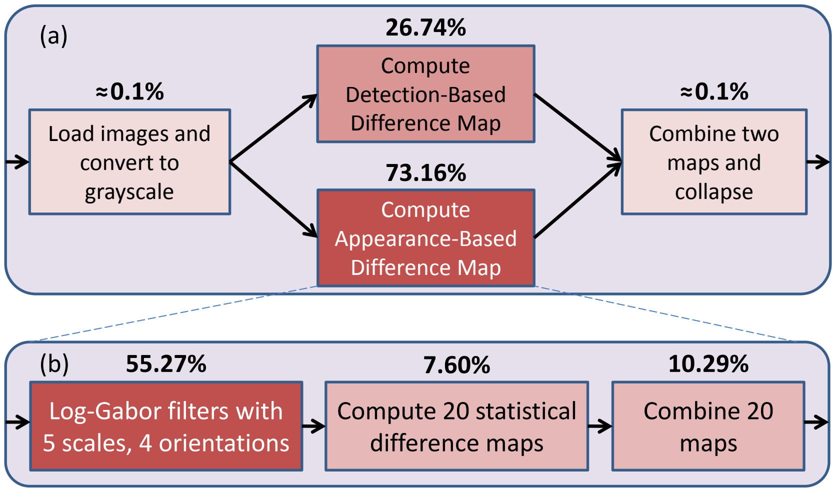

(a)
≈0.1%
Load images and convert to grayscale
26.74%
Compute Detection-Based Difference Map
73.16%
Compute Appearance-Based Difference Map
≈0.1%
Combine two maps and collapse
(b)
55.27%
Log-Gabor filters with 5 scales, 4 orientations
7.60%
Compute 20 statistical difference maps
10.29%
Combine 20 maps

Figure 3.1: Updated version of Figure 2.5; the average total run-time per image is 3.21 secs (System 1). The log-Gabor filtering is now the bottleneck, and target for following chapter.

CHAPTER 4

GPGPU AND OTHER CODE OPTIMIZATIONS

4.1 GPGPU and code optimizations

MAD's log-Gabor decomposition is performed for both the original image and the distorted image using twenty log-Gabor filters which span five scales and four orientations; the filtering results in 40 total subbands (20 subbands each for the original and distorted image). This process is well suited to a parallel implementation because each subband can be computed separately from all other subbands.

In MAD, the log-Gabor decomposition is implemented entirely in Matlab. To parallelize this decomposition, we employed Matlab's GPGPU (CUDA) facilities [57]. The GPGPU implementation requires specific hardware, which is a supported graphic card for CUDA-enabled NVIDIA GPUs (with compute capability at least 1.3). Such graphic card requires power supply and hardware compatibility. Therefore, only System 1, which is a desktop was tested.

The graphic card used in the system 1 was an NVIDIA GeForce GTX 560 Ti with 1 GB of RAM with Version 8.17.12.8026 of NVIDIA's driver. This card has compute capability of 2.1, which is greater than the minimum requirement of 1.3. Table 4.1 provides basic information of this graphic card. More detail information can be found in [58].

Matlab GPGPU is similar to we have already known about Matlab on CPU. We will need to bring variables from RAM (of CPU) to GPU first, and then apply Matlab *GPU functions* on GPU variables. We can bring values from GPU memory back to CPU memory when we finish GPU calculating. The following Matlab code demon-

Table 4.1: NVIDIA GeForce GTX 560 Ti specifications

Graphics card version	GTX 560 Ti
CUDA Cores	384
Graphics Clock (MHz)	822
Processor Clock (MHz)	1645
Texture Fill Rate (billion/sec)	52.5
Memory Clock (MHz)	4008 Gbps
Standard Memory Config	1024
Memory Interface	GDDR5
Memory Interface Width	256-bit
Memory Bandwidth (GB/sec)	128
Compute Capability	2.1

strates how to bring an array back and forth between CPU and GPU in Matlab:

```
>> CPUarr = rand(1024);
>> GPUarr = gpuArray(CPUarr)
GPUarr =
parallel.gpu.GPUArray:
---------------------
Size:  [1024 1024]
ClassUnderlying:  ’double’
Complexity:  ’real’
CPUarr1 = gather(GPUarr); % bring back to CPU
```

In Matlab, if a GPU-supported function receives GPU variables (GPU arrays or values) as the inputs, Matlab will switch to GPU function, and it will performs operation on GPU, and return values on GPU memory. For example: `xx=fft(CPUarr);` will perform on CPU, and return values on RAM and `yy=fft(GPUarr);` will perform on GPU and return values on GPU memory. The list of built-in functions, which are GPU-supported was limited at the time this study was performed. The newest list of GPU-supported functions can be found online at [59].

Table 4.2: Timing results of GPGPU and other code modifications (System 1 only). CO = Code Optimization; IE-DT = Inline Expansion of `pow` in MAD's detection-based stage. When all of the acceleration techniques are employed, MAD's run-time is accelerated by 47x.

Modification	Run-time in secs.	Speedup vs. Orig.	Speedup vs. II+IE
II + IE (repeated from Table 3.1)	3.21	17.4x	1.0x
II + IE + GPGPU	2.04	27.3x	1.6x
II + IE + GPGPU + CO	1.75	31.9x	1.8x
II + IE + GPGPU + CO + IE-DT	1.19	47.0x	2.7x

The log-Gabor function employed in MAD is not an GPU function. However it includes mainly GPU-supported built-in functions, for example, `sin`, `cos`, `atan2`, `fft2`, `ifft2`, `fftshift`, `log` and other mathematical operations. Therefore, we just need to change the inputs to be GPU arrays, where needed.

Further optimization in log-Gabor function can be archived by precomputing the filters and caching them in memory. This technique aims to reduce computational complexity, while GPU aims to employ a faster computation unit. Therefore, we will have an addition performance speedup to see what is the real power of GPGPU applying on log-Gabor stage.

4.2 Results

Table 4.2 shows the results of the GPGPU implementation (second row) along with two additional code modifications. The third row shows the results when the log-Gabor code is further optimized by precomputing the filters and caching them in RAM (CO = code optimization). The last row shows the results of applying inline expansion of the `pow` function to all parts of MAD (including the preprocessing and detection-based stages) (IE-DT = Inline Expansion in MAD's detection-based stage). When all of these acceleration techniques are employed, MAD's run-time is accelerated by 47x.

Showing in the first row of Table 4.2 is the result of previous chapter, using integral images and inline expansion techniques. This results a speedup vs. original version of 17x faster. The algorithm at this point requires only 3.21 seconds for one image, which is efficient enough to be applied on video (30 frames per second) in a reasonable time.

In the second row of Table 4.2, the GPGPU implementation (applied on top of II+IE techniques) provides a lower-than-expected result, which is 1.6x speedup over II+IE alone. This version of MAD reaches 2.04 seconds per image. The GPGPU implementation required copying the images to the GPU, and then copying the resulting log-Gabor subbands from GPU back to the CPU for the statistical computations. We suspect that the overhead involved with these memory transfers reduced the overall performance gain. This finding is consistent with our previous study on CUDA [60], which revealed that the memory bandwidth between the system and the GPU can create a bottleneck that reduces the potential gains.

The third row of Table 4.2 provides the results of II, IE, GPGPU, and CO techniques. This version consumes 1.75 seconds for each image, and it yields 31.9x times speedup vs. original version, and 1.8x times speedup comparing to the II+IE version.

The last row of Table 4.2 shows us that when all of these acceleration techniques are employed, MAD's run-time is accelerated by 47x (compare to original MAD) or 2.7x (compare to II+IE version). The running time requires for one image on average is only 1.19 seconds.

Figure 4.1 shows an updated version of Figure 3.1, indicating the new distribution of times. Notice that the log-Gabor decomposition has been reduced from its previous value of approximately 55% to the new value of 22%. The appearance-based and detection-based stages of MAD now consume a more balanced share of the execution time.

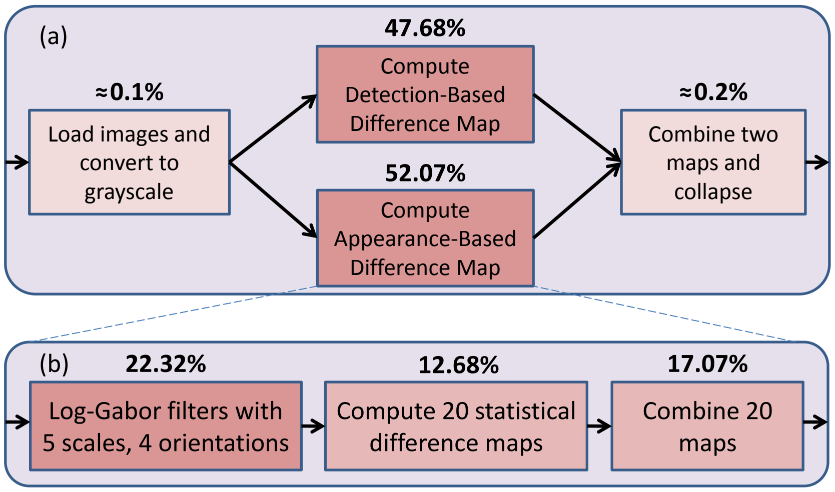

(a)
47.68%
Compute Detection-Based Difference Map
≈0.1%
Load images and convert to grayscale
52.07%
Compute Appearance-Based Difference Map
≈0.2%
Combine two maps and collapse
(b)
22.32%
Log-Gabor filters with 5 scales, 4 orientations
12.68%
Compute 20 statistical difference maps
17.07%
Combine 20 maps

Figure 4.1: Updated version of Figure 3.1 using all acceleration techniques; the average total execution time per image is 1.19 seconds, which is 47x times faster than the original version. The log-Gabor decomposition has reduced to 22% of the total time (previously 55%).

CHAPTER 5

Conclusion

This work presented techniques for accelerating the two most computationally expensive stages employed in many QA algorithms: local frequency-based decomposition and local statistical comparisons between the frequency coefficients of the reference and distorted images. We specifically analyzed and accelerated one representative QA algorithm, MAD [29]. The results of our performance analysis showed that the bottlenecks stem from these two stages and we presented four methods of acceleration, generalized integral images, inline expansion, GPGPU and code optimizations. Specifically, our new integral image technique can be efficiently applied back to computer graphics, where the original integral image had been employed. Although this study focused on one specific algorithm, our methodology and acceleration techniques are applicable to a variety of QA algorithms.

Several papers of other researchers have benefited from the speed-up MAD version from this work. With a 47x times faster runtime, the new MAD version could be applied and studied on a larger image database or on videos with a reasonable running time. For example, Singh *et. al.* [61] presented F-MAD, a feature-based extension of the MAD algorithm for still images, or Vu *et. al.* [62, 63] developed new video quality assessment algorithm based on MAD.

This work can be extended several ways by using performance counters and other profiling techniques. Such an analysis will yield further insights into the hardware resource usage of the different stages. Another avenue for acceleration that we are currently investigating is to change the algorithm in an effort to reduce its computational complexity (e.g., reducing the number of the filters used in the decomposition, or simplifying the statistical computations used to compare the coefficients). The latter approaches are likely to change the output of the algorithm; however, it is still possible to achieve a reasonable tradeoff between predictive performance and overall run-time.

REFERENCES

[1] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *Journal of Electronic Imaging*, vol. 19, no. 1, p. 011006, 2010.

[2] S. Daly, "Visible differences predictor: an algorithm for the assessment of image fidelity," in *Digital Images and Human Vision* (A. B. Watson, ed.), pp. 179–206, 1993.

[3] P. C. Teo and D. J. Heeger, "Perceptual image distortion," *Proc. SPIE*, vol. 2179, pp. 127–141, 1994.

[4] C. J. van den Branden Lambrecht, "A working spatio-temporal model of the human visual system for image representation and quality assessment applications," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 2291–2294, May 1996.

[5] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1398 – 1402 Vol.2, Nov 2003.

[6] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006.

[7] A. K. Moorthy and A. C. Bovik, "Blind image quality assessment: from natural scene statistics to perceptual quality.," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3350–64, 2011.

[8] A. Mittal, G. S. Muralidhar, J. Ghosh, and A. C. Bovik, “Blind image quality assessment without human training using latent quality factors,” *IEEE Signal Processing Letters*, vol. 19, no. 2, pp. 75–78, 2012.

[9] M. Saad, A. Bovik, and C. Charrier, “Blind image quality assessment: A natural scene statistics approach in the DCT domain,” *Image Processing, IEEE Transactions on*, vol. 21, no. 8, pp. 3339–3352, 2012.

[10] Z. Wang and E. P. Simoncelli, “Reduced-reference image quality assessment using a wavelet-domain natural image statistic model,” vol. 5666, January 2005.

[11] Z. W. Qiang Li, “Reduced-reference image quality assessment using divisive normalization-based image representation,” *IEEE Journal of Selected Topics in Signal Processing*, April 2009.

[12] F. Z. Lin Ma, Songnan Li and K. N. Ngan, “Reduced-reference image quality assessment using reorganized dct-based image representation,” *IEEE Trans. on Multimedia*, vol. 13, August 2011.

[13] H. R. Sheikh and A. C. Bovik, “A visual information fidelity approach to video quality assessment,” in *The First International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, pp. 23–25, 2005.

[14] Z. Wang and Q. Li, “Video quality assessment using a statistical model of human visual speed perception,” *JOSA A*, vol. 24, no. 12, pp. B61–B69, 2007.

[15] C. Vu and D. M. Chandler, “Main subject detection via adaptive feature refinement,” *Journal of Electronic Imaging*, vol. 20, March 2011.

[16] K. Seshadrinathan and A. C. Bovik, “Automatic prediction of perceptual quality of multimedia signals–a survey,” *Multimedia Tools Appl.*, vol. 51, pp. 163–186, Jan. 2011.

[17] W. Lin and C. C. Jay Kuo, “Perceptual visual quality metrics: A survey,” *J. Vis. Comun. Image Represent.*, vol. 22, pp. 297–312, May 2011.

[18] A. C. Bovik, “Automatic prediction of perceptual image and video quality,” *Proceedings of the IEEE*, 2012.

[19] D. M. Chandler, “Seven challenges in image quality assessment: past, present, and future research,” *ISRN Signal Processing*, vol. 2013, 2013.

[20] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, pp. 600–612, 2004.

[21] Internet, “Available online.” http://www.cns.nyu.edu/ eero/steerpyr/.

[22] C. J. van den Branden Lambrecht, “A working spatio-temporal model of the human visual system for image representation and quality assessment applications,” in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 2291–2294, May 1996.

[23] Z. Wang, A. C. Bovik, and L. Lu, “Wavelet-based foveated image quality measurement for region of interest image coding,” *Proc. IEEE Int. Conf. on Image Processing*, vol. 2001.

[24] K. Yang and H. Jiang, “Optimized-SSIM based quantization in optical remote sensing image compression,” in *Proceedings of the 2011 Sixth International Conference on Image and Graphics*, ICIG ’11, (Washington, DC, USA), pp. 117–122, IEEE Computer Society, 2011.

[25] A. Rehman, M. Rostami, Z. Wang, D. Brunet, and E. R. Vrscay, “Ssim-inspired image restoration using sparse representation,” *EURASIP J. Adv. Sig. Proc.*, vol. 2012, p. 16, 2012.

[26] F. Ciaramello, A. Cavender, S. Hemami, E. Riskin, and R. Ladner, "Predicting intelligibility of compressed american sign language video with objective quality metrics," in *2006 International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2006.

[27] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006.

[28] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc 37th Asilomar Conf on Signals, Systems and Computers*, vol. 2, (Pacific Grove, CA), pp. 1398–1402, IEEE Computer Society, Nov 9-12 2003.

[29] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *Journal of Electronic Imaging*, vol. 19, no. 1, 2010.

[30] K. Seshadrinathan and A. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *Image Processing, IEEE Transactions on*, vol. 19, pp. 335 –350, Feb 2010.

[31] W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *Communications, IEEE Transactions on*, vol. 25, pp. 1004 – 1009, sep 1977.

[32] H. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, pp. 1455 – 1461, oct 1987.

[33] J. Liang and T. D. Tran, "Fast multiplierless approximation of the DCT with the lifting scheme," *IEEE Trans. on Signal Processing*, vol. 49, pp. 3032–3044, 2000.

[34] W. Yuan, P. Hao, and C. Xu, "Matrix factorization for fast DCT algorithms," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 3, p. III, may 2006.

[35] C. Cheng and K. Parhi, "Hardware efficient fast DCT based on novel cyclic convolution structures," *Signal Processing, IEEE Transactions on*, vol. 54, pp. 4419 –4434, nov. 2006.

[36] V. Britanak, P. Yip, and K. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations.* Academic, 2007.

[37] D. Trainor, J. Heron, and R. Woods, "Implementation of the 2D DCT using a Xilinx XC6264 FPGA," in *Signal Processing Systems, 1997. SIPS 97 - Design and Implementation., 1997 IEEE Workshop on*, pp. 541 –550, nov 1997.

[38] G. Kiryukhin and M. Celenk, "Implementation of 2D-DCT on XC4000 series FPGA using DFT-based DSFG and DA architectures," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 3, pp. 302 –305 vol.3, 2001.

[39] B. Fang, G. Shen, S. Li, and H. Chen, "Techniques for efficient DCT/iDCT implementation on generic GPU," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pp. 1126 – 1129 Vol. 2, may 2005.

[40] S. Tokdemir and S. Belkasim, "Parallel processing of DCT on GPU," in *Data Compression Conference (DCC), 2011*, p. 479, march 2011.

[41] T.-T. Wong, C.-S. Leung, P.-A. Heng, and J. Wang, "Discrete wavelet transform on consumer-level graphics hardware," *Multimedia, IEEE Transactions on*, vol. 9, pp. 668 –673, april 2007.

[42] C. Tenllado, J. Setoain, M. Prieto, L. Pinuel, and F. Tirado, “Parallel implementation of the 2D discrete wavelet transform on graphics processing units: Filter bank versus lifting,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 299–310, 2008.

[43] J. Franco, G. Bernabe, J. Fernandez, and M. Acacio, “A parallel implementation of the 2D wavelet transform using CUDA,” in *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pp. 111 – 118, feb. 2009.

[44] M. Unser, “Fast gabor-like windowed fourier and continuous wavelet transforms,” *Signal Processing Letters, IEEE*, vol. 1, pp. 76 –79, may 1994.

[45] L. Tao and H. K. Kwan, “Fast parallel approach for 2-D DHT-based real-valued discrete gabor transform,” *Image Processing, IEEE Transactions on*, vol. 18, pp. 2790 –2796, dec. 2009.

[46] X. Wang and B. Shi, “GPU implemention of fast gabor filters,” in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 373 –376, 30 2010-june 2 2010.

[47] L. Tao and H. K. Kwan, “Multirate-based fast parallel algorithms for 2-D DHT-based real-valued discrete gabor transform,” *Image Processing, IEEE Transactions on*, vol. 21, pp. 3306 –3311, july 2012.

[48] F. C. Crow, “Summed-area tables for texture mapping,” *SIGGRAPH Comput. Graph.*, vol. 18, pp. 207–212, Jan. 1984.

[49] J.-S. Kim and R.-H. Park, “A fast feature-based block matching algorithm using integral projections,” *Selected Areas in Communications, IEEE Journal on*, vol. 10, no. 5, pp. 968–971, 1992.

[50] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[51] F. Shafait, D. Keysers, and T. M. Breuel, "Efficient implementation of local adaptive thresholding techniques using integral images," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 6815 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Jan. 2008.

[52] M.-J. Chen and A. C. Bovik, "Fast structural similarity index algorithm," *J. Real-Time Image Process.*, vol. 6, pp. 281–287, Dec. 2011.

[53] H. R. Sheikh, Z. Wang, A. C. Bovik, and L. K. Cormack, "Image and video quality assessment research at live." Online. http://live.ece.utexas.edu/research/quality/.

[54] K. Okarma and P. Mazurek, "GPGPU based estimation of the combined video quality metric," in *Image Processing and Communications Challenges 3* (R. Choras, ed.), vol. 102 of *Advances in Intelligent and Soft Computing*, pp. 285–292, Springer Berlin / Heidelberg, 2011.

[55] P. D. Kovesi, "MATLAB and Octave functions for computer vision and image processing." Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[56] C. P. I. Q. L. O. S. University, "CSIQ image database," 2009. http://vision.okstate.edu/csiq/.

[57] Nvidia graphic card, "Matlab GPU." http://www.nvidia.com/object/tesla-matlab-accelerations.html.

[58] Nvidia graphic card, "Matlab GPU." http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-560ti/specifications.

[59] Matlab, "Matlab GPU-supported functions." http://www.mathworks.com/help/distcomp/run-built-in-functions-on-a-gpu.html.

[60] B. Gordon, S. Sohoni, and D. Chandler, "Data handling inefficiencies between cuda, 3d rendering, and system memory," in *Workload Characterization (IISWC), 2010 IEEE International Symposium on*, pp. 1 –10, dec. 2010.

[61] P. Singh and D. M. Chandler, "F-mad: a feature-based extension of the most apparent distortion algorithm for image quality assessment," in *IS&T/SPIE Electronic Imaging*, pp. 86530I–86530I, International Society for Optics and Photonics, 2013.

[62] P. V. Vu and D. M. Chandler, "Video quality assessment based on motion dissimilarity," in *Seventh Int. Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2013.

[63] P. V. Vu and D. M. Chandler, "Vis3: an algorithm for video quality assessment via analysis of spatial and spatiotemporal slices," *Journal of Electronic Imaging*, vol. 23, no. 1, pp. 013016–013016, 2014.

VITA

THIEN DUC PHAN

Candidate for the Degree of

Master of Science

Dissertation: PERFORMANCE-ANALYSIS-BASED ACCELERATION OF IMAGE QUALITY ASSESSMENT

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Ha Tinh City, Ha Tinh Province, Vietnam on April 11, 1985.

Education:
Received the B.S. degree from Hanoi University of Technology,
Hanoi, Vietnam, 2008, in Information Technology
Completed the requirements for the degree of Master of Science with a major in Electrical Engineering Oklahoma State University in December, 2014.

Experience:
Research Assistant, Oklahoma State University 2010-2014.

Papers:
TD Phan, S Sohoni, DM Chandler, and EC Larson, "Performance-analysis-based acceleration of image quality assessment", *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, 2012.

CT Vu, TD Phan, PS Banga, and DM Chandler, "On the quality assessment of enhanced images: A database, analysis, and strategies for augmenting existing methods", *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, 2012.