STRATEGIES FOR IMPROVING EFFICIENCY AND EFFICACY OF IMAGE QUALITY ASSESSMENT ALGORITHMS

By

THIEN DUC PHAN

Bachelor of Science in Information Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
2008

Master of Science in School of
Electrical and Computer Engineering
Oklahoma State University
Oklahoma, USA
2014

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Doctor of Philosophy
May, 2015

STRATEGIES FOR IMPROVING EFFICIENCY AND EFFICACY OF IMAGE QUALITY ASSESSMENT ALGORITHMS

Dissertation Approved:

Dr. Damon M. Chandler

Advisor

Dr. Guoliang Fan

Dr. Keith A. Teague

Dr. R. Russell Rhinehart

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; I would not have been who I am/where I am at today without him. This work would have never become truth without his guidance.

I would like to thank Dr. Guoliang Fan, a great professor of two 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, another Dr. Chandler's student, 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 dissertation. In the past, now, and future, I still need you on my side. Thank you.

I would like to thank Dr. R. Russell Rhinehart, who has given me knowledge in the course Optimization Applications. This course made me think differently about optimization. Its usefulness remains relevant now and in the future. Dr. Rhinehart also have read my previous reports and given so many helpful comments. Thank you so much for your insightful feedbacks.

I also would like to thank Phong Vu, Cuong Vu, and Dr. Sohum Sohoni, who have helped me to finish one conference, *On the use of image quality estimators for improved JPEG2000 coding*, and two journals, *A spectral and spatial measure of local perceived sharpness in natural images* and *Microarchitectural analysis of image quality assessment algorithms*, and of course, Dr. Chandler, who have been helping me with

these three above and a few more papers, which provided the main ideas for this thesis report. I have had an incredible time working with you. It has been the best time I have ever had in my student life that I will never forget.

Furthermore, I would like to thank Eric Larson because of his MAD algorithm, a FR IQA algorithm. What a pity that Eric had left OSU before I came. However, his work with Dr. Chandler inspires not only me, but also Phong Vu and other researchers to keep working on Image and Video Quality Assessment.

This material is based upon work supported by, or in part by, the National Science Foundation, Award #1054612 and #0917014, 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. Many thanks for the money, and for giving me an opportunity to be a great person working with great people in great projects.

Last, but not least, I would like to thank my wife, my family, and my labmates, who have been so supportive when I was in OSU. A few words cannot describe all my feeling; I'll talk to you later.

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: May, 2015

Title of Study: STRATEGIES FOR IMPROVING EFFICIENCY AND EFFICACY OF IMAGE QUALITY ASSESSMENT ALGORITHMS

Major Field: Electrical Engineering

Image quality assessment (IQA) research aims to predict the qualities of images in a manner that agrees with subjective quality ratings. Over the last several decades, the major impetus in IQA research has focused on improving prediction efficacy globally (across images) of distortion-specific types or general types; very few studies have explored local image quality (within images), or IQA algorithm for improved JPEG2000 coding. Even fewer studies have focused on analyzing and improving the runtime performance of IQA algorithms. Moreover, *reduced-reference* (RR) IQA is also a new field to be explored, when the transmitting bandwidth is limited, side information about original image was received with distorted image at the receiver. This report explored these four topics. For local image quality, we provided a local sharpness database, and we analyzed the database along with current sharpness metrics. We revealed that human highly agreed when rating sharpness of small blocks. Overall, this sharpness database is a true representation of human subjective ratings and current sharpness algorithms could reach 0.87 in terms of SROCC score. For JPEG2000 coding using IQA, we provided a new JPEG2000 image database, which includes only same total distortion images. Analysis of existing IQA algorithms on this database revealed that even though current algorithms perform reasonably well on JPEG2000-compressed images in popular image-quality databases, they often fail to predict the correct rankings on our database's images. Based on the framework of Most Apparent Distortion (MAD), a new algorithm, MAD_{DWT} is then proposed using local DWT coefficient statistics to predict the perceived distortion due to subband quantization. MAD_{DWT} outperforms all others algorithms on this database, and shows a promising use in JPEG2000 coding. For efficiency of IQA algorithms, this paper is the first to examine IQA algorithms from the perspective of their interaction with the underlying hardware and microarchitectural resources, and to perform a systematic performance analysis using state-of-the-art tools and techniques from other computing disciplines. We implemented four popular *full-reference* IQA algorithms and two *no-reference* algorithms in C++ based on the code provided by their respective authors. Hotspot analysis and microarchitectural analysis of each algorithm were performed and compared. Despite the fact that all six algorithms share common algorithmic operations (e.g., filterbanks and statistical computations), our results revealed that different IQA algorithms overwhelm different microarchitectural resources and give rise to different types of bottlenecks. For RR IQA, we also provide a new framework based on multiscale sharpness map. This framework employs multiscale sharpness maps as reduced information. As we will demonstrate, our framework with 2% reduced information can outperform other frameworks, which employ from 2% to 3% reduced information. Our framework is also competitive to current state-of-the-art FR algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Background on IQA

The quality of digital images is rarely perfects. When we see it on the Internet, on the TV, on the phone or other displaying devices, there is a high chance that the images got distorted. The distortions could be at any stage, e.g., during acquisition, compression, transmission, decompression, or processing. Therefore, it is important for the system to quantify the degradation in order to maintain, to control, and/or to enhance the quality of the image automatically.

Although, it is difficult to get all people agreed on the same opinion, the quality assessment task seems to be easy for human. Giving one person one reference (original) image, and one distorted image, he or she can assess image quality easily. Even when one portion of the reference image or no reference image is provided, human can still assess quality. This task remains challenge for computer.

The crucial ground-truth information is needed, and thus several *image databases* have been provided. Current IQA research also has been focused developing *IQA algorithms*, and applying IQA algorithms. Several large image databases have been provided publicly (smaller-size and oriented databases are not listed here).

The LIVE database [1] was developed in 2003 at the University of Texas at Austin, USA. It contains 29 reference images and 779 distorted images. The distorted images were generated from five different distortion types: JPEG compression, JPEG2000 compression, additive Gaussian white noise, Gaussian blurring, and JPEG2000 with bit errors via a simulated Rayleigh fading channel.

The CSIQ database [2] was developed in 2008 at Oklahoma State University, USA. It contains 30 reference images and 866 distorted images, which were generated from six distortion types: JPEG compression, JPEG2000 compression, additive Gaussian white noise, additive Gaussian pink noise (150 distorted images), Gaussian blurring, and global contrast decrements.

The TID2008 database [3] developed at the Tampere University of Technology, Finland. It contains 25 reference images and 1700 distorted images generated. There are 17 distortion types in the database (e.g., different types of noise, blur, denoising, JPEG and JPEG2000 compression, transmission of JPEG, JPEG2000 images with errors, local distortions, luminance, and contrast changes). There were four different levels of each distortion type.

More recently, Ponomarenko *el. at* provided the TID2013 [4] image database, which is larger than TID2008. 24 distortion types and five level of distortions are included in this database to form 3000 images total. New distortion types have been have been introduced: change of color saturation, Multiplicative Gaussian noise, Comfort noise, Lossy compression of noisy images, Image color quantization with dither, Chromatic aberrations, Sparse sampling and reconstruction.

Many techniques are employed to develop IQA algorithms. We list here a few, more state-of-the-art algorithms are provided in Section 4.2.

The SSIM algorithm was proposed by Wang *et al.* [5] and published in 2004. SSIM first measures correlation, luminance, and contrast of reference and distorted images. The image quality is then captured by three aspects of information loss: luminance distortion, contrast distortion, and structural distortion. The MS-SSIM [6] algorithm, proposed by the same authors, is an extension of SSIM for multiple scales. MS-SSIM argues that the correct scale depends on the viewing conditions (e.g., display resolution and viewing distance). Basically, MS-SSIM employs SSIM for first scale (the reference and distorted images) and four smaller scales (filtered

and downsampled versions of the inputs) and it uses different weights for different scales.

The Most Apparent Distortion algorithm (MAD) was developed by Larson and Chandler [7] in 2010. MAD uses two strategies to estimate image quality. First, a detection-based strategy is used for near-threshold distortions. In this case, the image is most apparent, and thus the HVS attempts to look past the image and look for the distortions. Second, an appearance-based strategy is used for clearly visible distortions. In this case, the distortions are most apparent, and thus the HVS attempts to look past the distortion and look for the image's subject matter.

Today, IQA research has emerged as an active subdiscpline of image processing, and many of the resulting techniques and algorithms have begun to benefit a wide variety of applications ranging from image compression ([8–10]), to denoising ([11]), to gauging intelligibility in sign language video [12], to synthesized texture [13].

Most IQA algorithms are so-called *full-reference* 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. The FR IQA algorithm assumes that the original (reference) image is available for use. For example, we have to-be-compressed image and lossy compressed image. In this case, the compressed (distorted) image will be compared to the original on purposes, such as quality monitoring, compressed rate maintaining.

Recently, researchers have begun to develop *no-reference* IQA algorithms, which attempt to yield the same quality estimates by using only the processed/distorted image. The NR IQA states that the original image may not be available. For example, when watching TV, the images we see are processed (distorted), and an algorithm that can measure the quality can help the TV to control the quality.

More recently, a new field, *reduced-reference* IQA, has been explored. RR IQA

algorithm assesses quality by using the processed/distorted image and *only partial information* about the reference image. In this case, the scenario is: The sender has a limited bandwidth to send partial information about the original image along with distorted image.

The additional background information on each research topic is presented at the beginning of each chapter, separately.

1.2 IQA Challenges

Most of the IQA algorithms have been focused on improved prediction accuracy for the best fit to those above-mentioned image databases. Some algorithms have been developed for specific distortion types, for example Gaussian blurred, or JPEG/JPEG2000. However, most consumer photographs contain particular regions which are perceived to be sharper than others. Indeed, most professional photographers attempt to maximize perceived sharpness of some specific areas. Different from the global sharpness prediction, which is ability to quantify the perceived sharpness of an image, the local sharpness prediction is also useful for a variety of image processing applications, for example, auto-enhancement and auto-focus camera, and main subject detection.

Even though almost all existing sharpness measure algorithm can be modified to run in a block-based fashion to generate a local sharpness map, none of them was actually designed to generate a map. Moreover, there was no sharpness map database to analyze these algorithms. Exploring local perceived sharpness is one of the goals of this report.

An IQA algorithm that performs well in predicting image quality on current image quality databases does not guarantee the same success in a coding context. During JPEG2000 encoding, it is possible to generate images with vastly different visual qualities depending on how the distortions are allocated to different frequency bands.

Moreover, most IQA algorithms have been tested only on standard MSE-optimal

JPEG2000 images found in popular image quality databases. Thus, these FR IQA algorithms have been tested only MSE-optimal images. A new database of JPEG2000 image will be useful to value the performance of current algorithm. This process will be interesting to know how to obtain good quality images given a total distortion.

Current IQA algorithms have been shown to perform quite well at gauging quality. Some of the best-performing full-reference algorithms such as MS-SSIM [14], VIF [15], and MAD [7] have been shown to generate estimates of quality that correlate highly with human ratings of quality, typically yielding Spearman and Pearson correlation coefficients in excess of 0.9. Research in no-reference and reduced-reference IQA is much less mature; however, recent methods such as DIIVINE [16], BLIINDS-II [17], and BRISQUE [18] can yield quality estimates which also correlate highly with human ratings of quality, sometimes yielding correlation coefficients which rival the full-reference methods. The efficiency of these algorithms has not been taken care of. None of them could reach real time processing. Most of them need from three seconds to one minute to assess quality of one 512×512 image.

There are very few studies have focused on analyzing and improving the runtime performance of IQA algorithms. Their interaction with the underlying hardware and microarchitectural resources has not been explored. To what extent are the bottlenecks in IQA algorithms attributable to the decomposition and statistical computation stages vs. more algorithm-specific auxiliary computations? To what extent are the bottlenecks attributable to computational complexity vs. limitations in memory bandwidth? Are there generic implementation techniques or microarchitectural modifications that can be used to accelerate all or at least several IQA algorithms? The answers of these questions are important. However, there is no study that can give us the answer. The research in the microarchitecture will answer this.

Progress on RR IQA has been fastened in recent years and shown promising results. Many RR algorithms, developed with different approaches, have shown to be

competitive with FR algorithm, typically yielding Spearman and Pearson correlation coefficients in excess of 0.88, comparing 0.90 for FR IQA algorithm.

The human visual system often does not require all information about the reference image to assess image quality. Therefore, if we can find good RR features, RR IQA can be as good as FR IQA can. This is the main difference between no-reference IQA and reduced-reference IQA. The NR IQA could never be as good as FR, but the RR can. However, two questions remain challenges for RR research: what kind of reduced information should be used and how much information should be used in RR task.

1.3 Dissertation Overview

The work presented in this dissertation focuses on four topics: (1) exploring local perceived sharpness, (2) IQA algorithms for JPEG2000 coding, (3) performing microarchitectural analysis of IQA algorithms, and (4) a new RR IQA algorithm.

Local sharpness work involves creating a first local sharpness database ever. This sharpness database will be helpful for local sharpness context, which is also important for image auto-enhancement and main subject detection research fields. Current sharpness metrics focus on improving global sharpness prediction (i.e., giving one single scalar for one image, and to compare images together). These sharpness metrics can be modified to yield local sharpness map by dividing the input image into small blocks and running sharpness metrics for each block. However, performing well at global sharpness prediction does not guaranty a good result for local sharpness. Therefore, we need a ground-truth to compare all algorithms. The local sharpness database also can answer some important questions, for example, what do human think about sharpness (since term *sharpness* lacks a precise technical definition), what do sharp blocks look like, and do human agree on sharp/blur blocks. At the time the database was published, there was no sharpness algorithm designed specif-

ically for local prediction. By now, there are a few algorithms can do that, e.g. S_3 [19], $FISH_{bb}$ [20]. It is also interesting to see how well these algorithms, together with modified sharpness metrics, perform in terms of sharpness map prediction.

Currently the JPEG2000-compressed images of common image databases were generated using standard MSE criterion. After that, current state-of-the-art IQA algorithms were tested on these images. On top of these, improving JPEG2000 coding leans on using these IQA algorithms to replace MSE. However, it is possible to generate JPEG2000 images with the same total distortion, but distributed differently across frequency. Therefore, we provide a new JPEG2000 image database, which includes non-MSE-criterion images. Our findings reveal that current IQA algorithms, which perform reasonably well on JPEG2000-compressed images, fail to predict images in our database. Based on the analysis results, and framework of MAD algorithm, we propose a new algorithm, MAD_{DWT}, which designed specifically for JPEG2000 coding which uses local DWT coefficient statistics.

Current state-of-the-art algorithms perform reasonably well in terms of prediction accuracy. However, most of them perform slowly. Some of them require from seconds to minutes to assess quality of one 512×512 image. This running time is far from real time processing. The running time, computation complexity and memory bandwidth have not been focused when developing algorithm. The interaction of IQA algorithms with the underlying hardware remains unexplored. Several questions remain challenges, for example, which is more important, computational complexity or limitations in memory bandwidth? Are there generic implementation techniques or microarchitectural modifications that can be used to accelerate all or at least several IQA algorithms? The study provided in this section (chapter 4) answers these questions. Moreover, the study also provide important insights for (1) designing new IQA algorithms, which are likely to draw on multiple approaches used in several existing IQA algorithms; (2) efficiently implementing multiple IQA algorithms on a given

hardware platform; (3) efficiently applying multiple IQA algorithms to specific applications; and (4) selecting and/or designing specific hardware which can efficiently execute multiple IQA algorithms.

The reduced reference IQA research has been fasten in recent years. Two questions remain challenges for RR task: what is and how much reduced information should we employ to accurately assess quality? In this study, we argue multiscale local sharpness map is a good reduced reference feature. Because it has good characteristics required for RR task: (1) it can efficiently summarize the reference images: the edge and texture areas give us high sharpness values and the blank/smooth areas have the low sharpness values; (2) it is sensitive to a variety of image distortions: the sharpness values increase in white noise images, the sharpness values decrease in Gaussian blurred images, the sharpness values increase for ringing artifacts and decrease for quantized blocks in JPEG/JPEG2000. Moreover, the multiscale maps, which are calculated from sharpness maps of downsampled versions of input image, will provide comprehensive information along with first-scale sharpness map. In this study, we invent a new RR framework, which employs any sharpness algorithm for reduced information.

1.4 Structure of the dissertation

The rest of this chapter briefly highlights the content of all other chapters as following:

- Chapter 2: This chapter first provides background information for topic of local perceived sharpness. The local sharpness experiment, including image, apparatus, subjects, experiment design and experiment process, is provided in detail. The first ever local sharpness database is presented and analyzed. Several sharpness algorithms, which were designed for global sharpness, are modified to work locally. Analysis results of these algorithms applying on the ground truth maps are provided at the end of the chapter.

• Chapter 3: As IQA algorithms are being employed for improved JPEG2000 coding. However, current JPEG2000-compressed images in common image databases are generated using standard MSE criterion, and current IQA algorithms were tested on these images only. This chapter provides a new JPEG2000-compressed image database that does not employ MSE criterion. The study provided in this chapter is trying to clarify that a good-general IQA algorithm often fails in coding context. Based on the database analysis results and based on a current algorithm framework, we propose a new IQA algorithm that is designed specifically for JPEG2000 compression.

• Chapter 4: The interaction of IQA algorithms with the underlying hardware and microarchitectural resources has not been explored. This is the first ever paper that studies IQA algorithms from hardware and software viewpoints. In this chapter, we converted six state-of-the-art IQA algorithms into C++ code to have a base-line comparison. The analysis methodology, architectural concepts, and results of performance analysis of all six algorithms are provided. The discussion section reveals that six algorithms, with different approaches, share the same common algorithmic operation, (e.g., a filtering/transforming step, and a statistical computation step). However, they overwhelm different microarchitectural resources and give rise to different types of bottlenecks, in two main categories: memory bottlenecks and core/computational bottlenecks. Based on these results, microarchitectural-conscious coding techniques and custom hardware recommendations for performance improvement are proposed at the end of the chapter.

• Chapter 5: This chapter starts a new topic on reduced reference IQA framework. The chapter starts with background, related work, RR framework based on sharpness algorithms, and the results of our algorithms, comparing to other RR frameworks and FR algorithms on popular image databases. In this chapter, the multiscale sharpness maps are employed as reduced information. I also examine sharpness feature and

prove that this feature can summarize efficiently the reference image, and it is sensitive to different distortion types. We also examine several sharpness algorithms in our framework and we will demonstrate, the new RR framework, when utilizes around 2% reduced information can outperform all current RR frameworks, which employ from 2% to 3% reduced information. Our best version of the RR framework also outperforms or is competitive to the full-reference IQA algorithms.

- Chapter 6: Final conclusion and remarks, the discussion of future work are provided in this chapter to finish this dissertation.

CHAPTER 2

LOCAL PERCEIVED SHARPNESS

2.1 Background

Although the term *sharpness* lacks a precise technical definition, any human can effortlessly point out the sharp regions in an image or assess the overall sharpness of the image. This is easy for human, however, it remains challenges for computer. Furthermore, the ability to predict sharpness locally is also as important as globally sharpness prediction, because most consumer photographs contain some areas that are perceived to be sharper than others.

Most of the current sharpness metrics, for example, MMZ [21], MDWE [22], ST [23], CPBD [24], and JNB [25], can predict quite well the Gaussian blur subsets of current image databases (LIVE [1], CSIQ [26], and TID [27]). Only S_3 algorithm [28] claimed to be able to predict local sharpness. As they have been focused on global sharpness, a few questions arise: Can they predict local sharpness and point out sharp areas in an image? Is there a based line image database to compare their local abilities?

To answer these questions, we provide a local sharpness experiment to collect human subjective ratings for local blocks in terms of perceived sharpness. We also examine the database to see if people agree with other when taking the experiment.

For current sharpness algorithms that have not been tested on small local blocks, we modify them for them to work each 64×64 blocks with 56 pixels overlap between neighboring blocks. This is not quite fair for all of them. For example, MDWE and JNB operate based on the assumption that there is at least one strong edge in the

image, and they try to measure the spread of this edge. However, this is the only way to compare their ability to predict local sharpness.

2.2 Local Sharpness Experiment

2.2.1 Images

Six color images of size 300×400 shown in Figure 2.1 were chosen to generate subjective sharpness maps. Stimuli were displayed on a LaCie 324 24-inch LCD monitor (1920x1200 at 60 Hz). The display yielded the minimum and maximum luminance of 0.80 and 259 cd/m^2, respectively, with $\gamma = 2.2$. Stimuli were viewed binocularly through natural pupils in a darkened room at a distance of approximately 60 cm. Eleven adult subjects, both male and female whose ages range from 23 to 30, took part in the experiment. All had the normal or corrected-to-normal visual acuity.

2.2.2 Design

The experiment interface is shown in Figure 2.2. Images were displayed in a mid-gray background. The left image was divided into blocks of size 16×16 (shown by grid) and the same image (without grid) was displayed in the right hand side for reference. Subjects rated the sharpness value for each block on a scale from 1 to 3, where 1 means the block is very sharp, 3 means the block is not sharp, and 2 is somewhat in between. There were only two buttons Level 1 and Level 2, which correspond to scale 1 and 2, in the interface of the experiment. A right click on a block would assign the sharpness value of that block to scale 3. There were also options to assign a sharpness level to multiple blocks and undo an assignment, which facilitated the experiment process.

For each image, subjects were asked to perform the experiment twice, where in the second time the grid was offset by 8 pixels both horizontally and vertically. The two resulting maps were then averaged to make a single map and thus provide us a

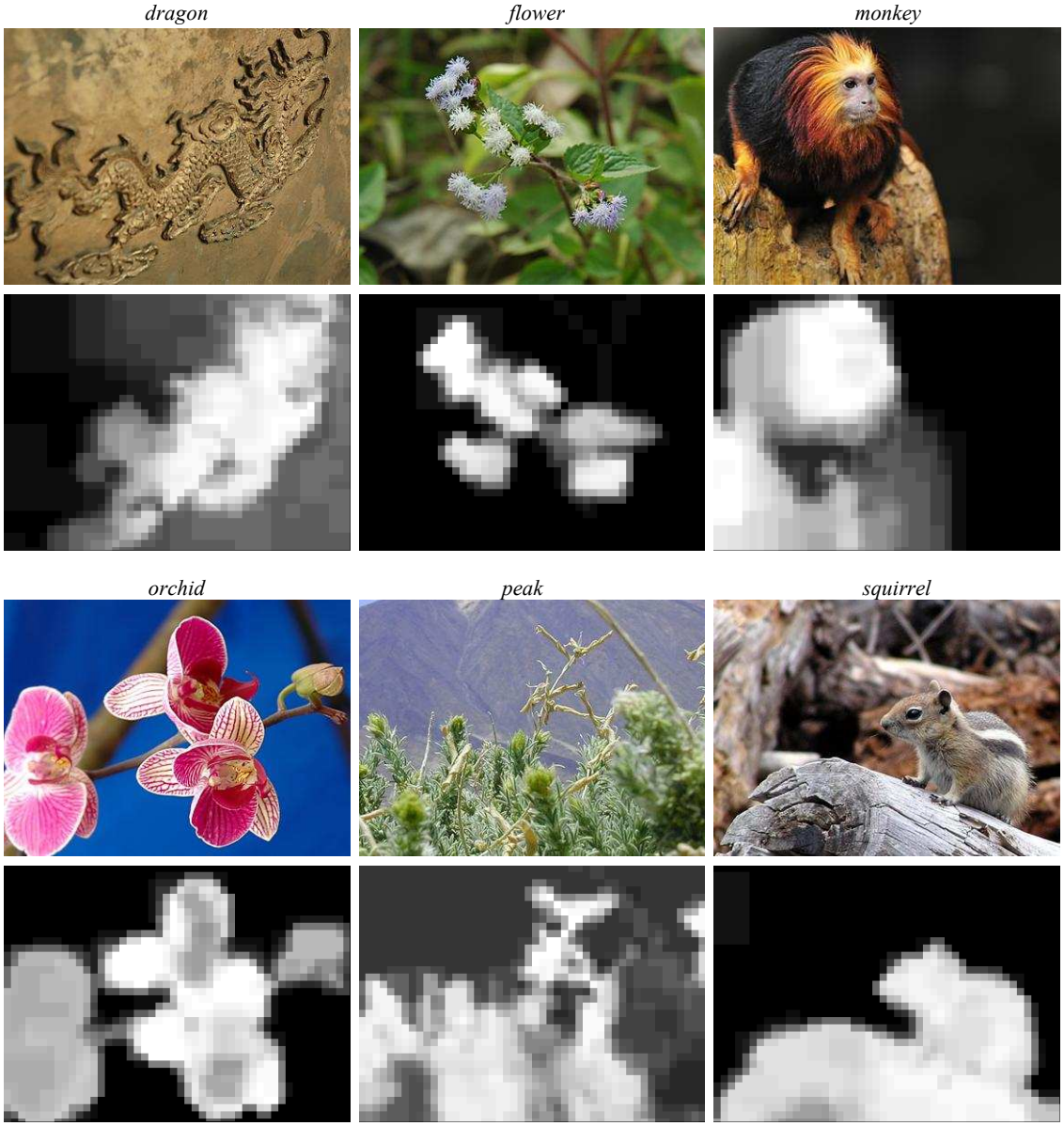


Figure 2.1: Six images in the database and their corresponding average subjective sharpness maps. Eleven subjects highly agreed with each other where the correlation coefficients between average maps and subjects' maps were 0.89 or higher.

sharpness map with block size 8×8. In comparison to the method of rating every 8×8 block, in which there are 1800 blocks for each image, this method helped reducing the number of blocks each subject has to rate by half, while also achieving the resolution (block size) of 8×8 for the final subjective sharpness map. On average, each subject took approximately 10 minutes to finish one map (one session).

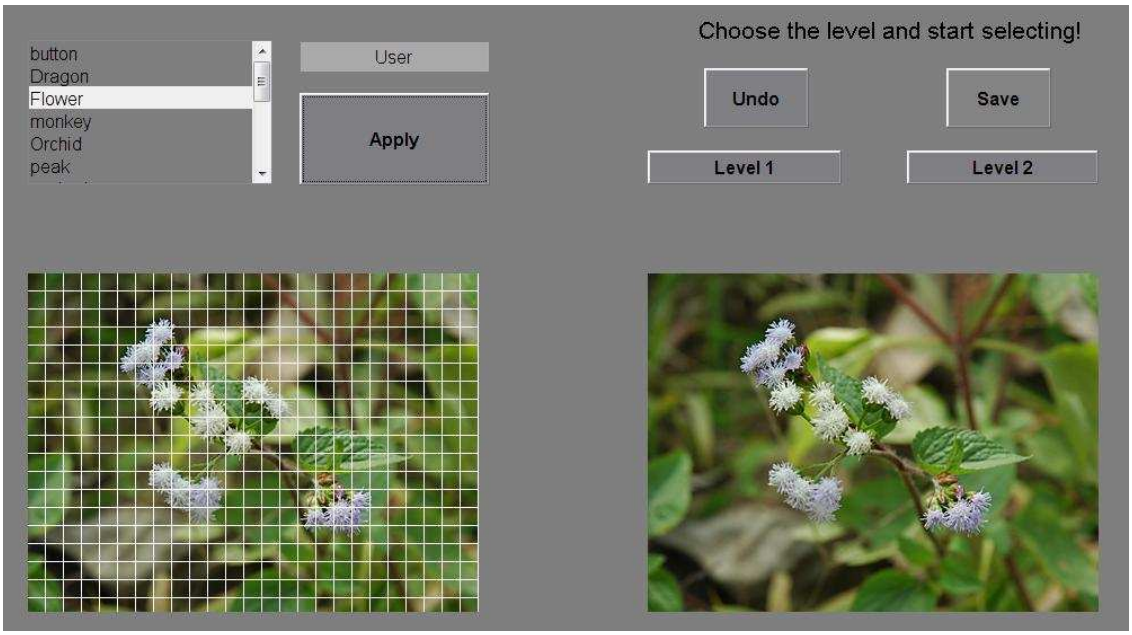


Figure 2.2: Interface of the experiment. Subjects rated the sharpness level for each 16×16 block in the image in the left on a scale from 1 to 3 where 1 means the block is very sharp. Two buttons Level 1 and Level 2 correspond to scale 1 and 2, respectively. A right click on a block would assign the sharpness value of that block to scale 3. The same image was displayed in the right hand side for reference.

2.2.3 Result

For each input image, from the three sharpness levels each subject rated, averaged over 2 sessions (for each image, subjects were asked to perform the experiment twice) and across eleven human subjects, we obtained the sharpness map which contains $206 - 245$ grayscale levels. Figure 2.1 also shows the sharpness maps of the six images in the database. Eleven subjects highly agreed with each other where the correlation coefficients between average maps and subjects' maps were 0.89 or higher.

2.3 Analysis Criteria

Before evaluating the performance of a metric, it is common to apply a logistic transform to the predicted ratings to bring the predictions on the same scale as the MOS or DMOS values. The logistic fitting function is use to nonlinearly map between the predictions and subjective scores. We adopt the logistic transform suggested by the

Video Quality Experts Group [29], which is given by

$$f(x) = \frac{\tau_1 - \tau_2}{1 + exp(\frac{x-\tau_3}{\tau_4})} + \tau_2 \tag{2.1}$$

in which τ_1, τ_2, τ_3, and τ_4 are the model parameters which are chosen to minimize the MSE between the predicted values and the subjective scores.

We use four criteria to compare the performances of different algorithms on the three databases mentioned above. The most common criteria are the Pearson correlation coefficient (CC), which measures how well an algorithm's predictions correlate with the subjective scores, and Spearman rank-order correlation (SROCC), which measures the relative monotonicity between the predictions and subjective scores. The other two criteria are the outlier ratio, OR, and outlier distance, OD. These two criteria attempt to account for the inherent variation in human subjective ratings of quality. This variability is normally quantified using the standard deviation of all subjective ratings for a particular image σ_s. An outlier is defined as a prediction which is outside $2\sigma_s$ of the DMOS or MOS[1]. Let $N_{outlier}$ and N_{total} denote the number of outliers and the total number of predicted ratings, respectively. The outlier ratio is defined as:

$$OR = \frac{N_{outlier}}{N_{total}} \tag{2.2}$$

In addition to the outlier ratio, the outlier distance, which was proposed in Ref.[26], attempts to quantify how far of the error bar ($\pm 2\sigma_s$) the outlier falls. It is defined as:

$$OD = \sum_{x \in X_{false}} \min(|f(x) - (s(x) + \sigma_s)|, |f(x) - (s(x) - \sigma_s)|) \tag{2.3}$$

where X_{false} is the set of all outliers, $s(x)$ is the DMOS or MOS rating of image x,

[1]The range $2\sigma_s$ was chosen because it contains 95% of all subjective quality scores for a given image.

and $f(x)$ is the predicted score after the logistic transform in Equation 2.1.

2.4 Performance Analysis

At the time of writing the paper, there was no algorithm that can generate local sharpness map, excepts S_3 [28] algorithm. Five sharpness measures JNB, CPBD, ST, MMZ, and MDWE are used in this comparison. As none of those algorithms directly outputs a map, we ran each of algorithm in a block-based fashion in order to generate sharpness maps. The input to those algorithms, instead of being the whole image, is now each block of the image.

As described in Ref. [25] and [24], JNB and CPBD algorithm require the smallest block size to be 64×64. We divide the input image into blocks of size 64×64 with 56 pixels overlap between neighboring blocks. The sharpness maps from JNB or CPBD generated by running through all blocks therefore have block size of 8×8. The rest three algorithms, ST, MMZ, and MDWE, are run through each 32×32 block with 24 pixels overlap between neighboring blocks. Their generated sharpness maps thus also have block size of 8×8.

Figure 2.3 shows maps of the six algorithms on images *dragon* and *flower*. Other algorithms do not generate good sharpness maps for image *dragon* but some (e.g., CPBD, MMZ) perform quite well for image *flower*. In general, the maps from S_3 algorithm are the most close to the groundtruth maps. Note that this is not quite a fair comparison as other algorithms are not designed to generate a map

Table 2.1 shows the performance of all algorithms on the entire set of images in our sharpness map database using CC, SROCC, and KullbackLeibler divergence [30], noted as KLD. The KLD measures the difference between two probability density functions p and q. It is not symmetric and is defined as:

$$KLD(p|q) = \sum_x log\left(\frac{p(x)}{q(x)}\right) \tag{2.4}$$

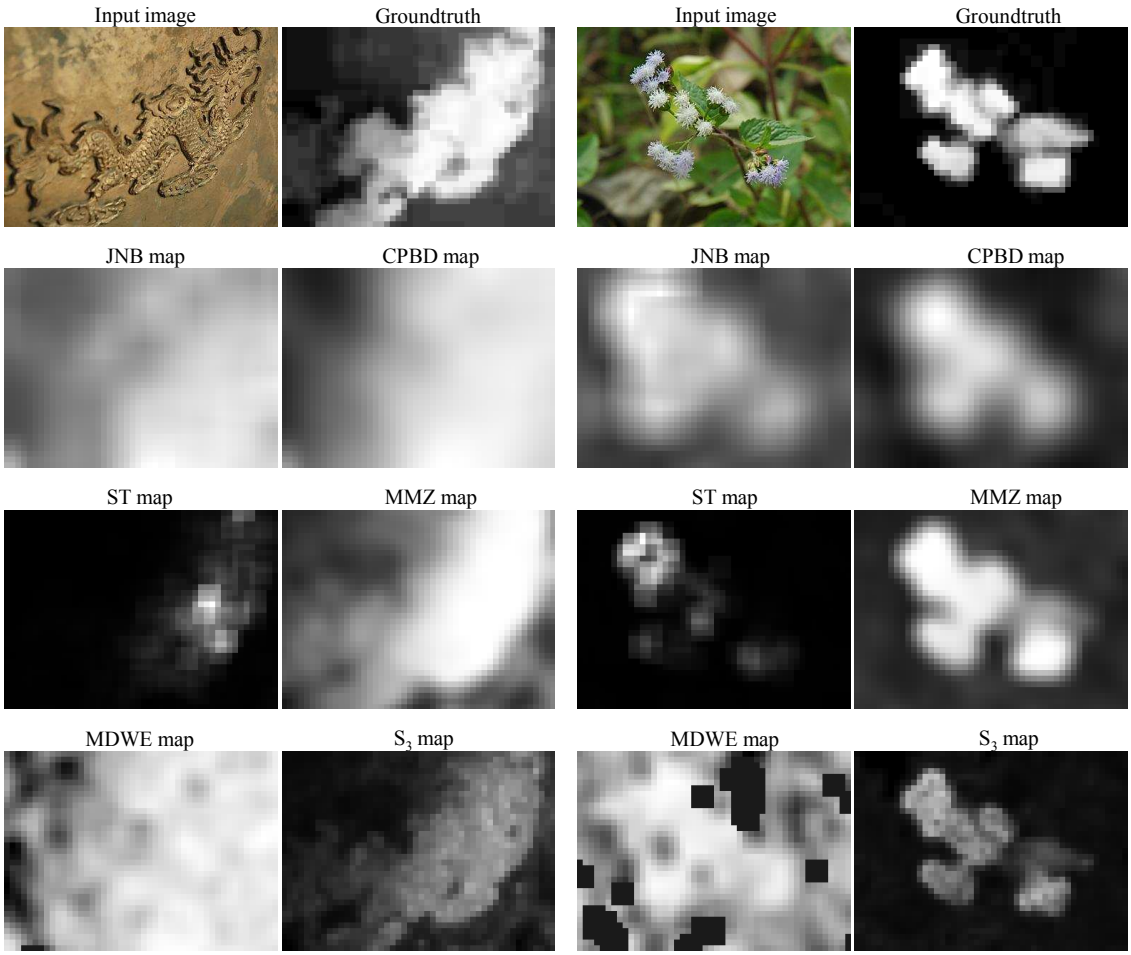


Figure 2.3: Comparison of sharpness maps from different algorithms. In general, the maps from S_3 algorithm are the most close to the groundtruth maps. Note that black blocks in MDWE maps are blocks which the MDWE algorithm failed to run on.

The probability density function deduced from the predicted map is treated as p and the probability density function deduced from groundtruth is treated as q. Note that before evaluating the performance of each algorithm, the logistic transform in Equation 2.1 was also applied to the local sharpness values.

As can be seen from Table 2.1, regarding CC and SROCC, the S_3 algorithm gives best results. The MMZ algorithm generally stands for the second best except image *flower* and *squirrel* where CPBD algorithm is the second best for SROCC. In KLD criterion, our algorithm is the best for image *dragon*, *monkey*, and *orchid*, and the second for image *flower* (the best is from CPBD). These results again demonstrate

that the combination of spectral and spatial measure in a block-based fashion helps build an efficient measure of local sharpness. Again, note that this is not quite a fair comparison as other algorithms are not designed to generate a map.

Table 2.1: Overall performance of all sharpness algorithms on the entire set of images in our sharpness map database. The best two performances are bolded.

		JNB	CPBD	ST	MMZ	MDWE	S3
CC	dragon	0.6077	0.6298	0.7581	**0.8622**	0.5178	**0.9587**
	flower	0.6685	0.8235	0.6676	**0.8728**	0.7183	**0.9470**
	monkey	0.4895	0.8039	0.7941	**0.9398**	0.6007	**0.9597**
	orchid	0.4027	0.4466	0.4565	**0.8678**	0.3471	**0.9329**
	peak	0.4164	0.4411	0.8044	**0.8807**	0.5361	**0.9504**
	squirrel	0.8576	0.9132	0.8965	**0.9415**	0.8179	**0.9696**
SROCC	dragon	0.5352	0.5909	0.8134	**0.8924**	0.5744	**0.9521**
	flower	0.6741	**0.7264**	0.6065	0.5467	0.6030	**0.7744**
	monkey	0.4320	0.8216	0.7499	**0.8848**	0.5143	**0.9606**
	orchid	0.3691	0.4359	0.4752	**0.8412**	0.0909	**0.9423**
	peak	0.4582	0.4274	0.8168	**0.8808**	0.4788	**0.9342**
	squirrel	0.8124	**0.8299**	0.7509	0.8007	0.6896	**0.8536**
KLD	dragon	2.5665	2.6955	2.4673	**2.2691**	2.8435	**2.0327**
	flower	3.9633	**2.4370**	3.9635	3.7239	3.9027	**3.7011**
	monkey	3.3885	**1.2467**	2.7107	1.9998	2.2816	**1.0351**
	orchid	3.4786	2.6319	2.9661	2.7893	**1.3228**	**0.9952**
	peak	**1.7967**	1.8203	2.0881	**1.5784**	2.1353	2.5528
	squirrel	**2.2090**	**2.8211**	3.7357	4.4886	3.6146	3.3488

2.5 Summary

In this chapter, we explored a new area, which is local perceived sharpness. We provided the experiment process to collect subjective rating of local sharpness. The database is made available online for research community.

We performed analysis the database, and discovered that human subjects are highly agreed with other in terms of sharpness. In general, the blocks which include edges or textures often get rated sharper. Blank blocks and smooth blocks are generally rated not as sharp. However, the ratings are also biased by neighboring blocks.

Most of the algorithms were not able to predict the sharpness of local blocks. We modified those algorithms for them to measure small blocks with overlap between neighboring blocks. Only a few algorithms were designed specifically for local sharp-

ness. However, no algorithm could reach 0.87 in terms of SROCC. From this, we can see that the database is truly a representation of human opinions. There is also a lot of room for developing a local sharpness algorithm.

CHAPTER 3

IQA ALGORITHM FOR JPEG2000 CODING

3.1 Background

Various algorithms for *full-reference* IQA have been developed and have been shown to perform very well on current image databases. A common technique involves measuring local pixelwise differences such as the mean-squared error (MSE) or peak signal-to-noise ratio (PSNR). More complete algorithms have employed a wide variety of approaches, for example Structural Similarity Index (SSIM) [5] and Multi-Scale Structural Similarity Index (MS-SSIM) [6] algorithms estimate quality based on image structure, Most Apparent Distortion (MAD) [7] estimates quality based on models of the human visual system.

The SSIM, MS-SSIM, MAD algorithms and other FR IQA algorithms such as NQM [31], VIF & VIFP [15], VSNR and [32], have been shown to be very good at predicting the quality ratings on LIVE, CSIQ, TID2008. Some of them have been employed for improved JPEG2000 coding (e.g., [33], [34]). For example, instead of using standard MSE-optimal coding schemes applied uniformly to all coefficients, Wang *et al.* [33] proposed a new coding scheme which iteratively reallocates the available bits over the image space based on a maximum of minimal structural similarity criterion. Richter *et al.* [34] used MS-SSIM as a criterion to minimize the distortion created by lossy image compression under a rate constraint. These types of approach try to replace the squared-error distortion criterion with a *perceived distortion* measure based on the output of an IQA algorithm.

However, JPEG2000 subsets from these image databases include only standard

MSE-optimal JPEG2000 [35] images, and thus, these FR IQA algorithms have been tested only MSE-optimal images. After that, they would be employed to replace the MSE stage of the coding scheme. Therefore, an IQA algorithm which performs well in predicting quality on an image-quality database (general distortion types or JPEG2000 compression) does not guarantee that the IQA algorithm will succeed in a coding context. During JPEG2000 encoding, it is possible to generate images with vastly different visual qualities depending on how the distortions are allocated to different wavelet subbands. As argued in [36], the way in which distortions are distributed across spatial frequency can significantly affect quality.

This is the first JPEG2000 image database that has a same amount of distortion, but distributed to the subbands differently. As we will demonstrate, most of the IQA algorithms cannot predict the quality rankings in a manner that agrees with subjective rankings. We also provide a new algorithm, MAD_{DWT}, based on the framework of MAD, using local DWT coefficient statistics to predict the rankings of this database. The preliminary results of MAD_{DWT} on this database suggest that it could be a good replacement for MSE in JPEG2000 coding.

The rest of this Chapter is organized as follows: In Section 3.2, we present a new database designed specifically for JPEG2000 images with a fixed amount of total distortion but allocated to different subbands. In this section, we introduce the images and compression method, the experiment process, the apparatus and subjects, and the results of the experiment. In Section 3.3, we analyze a numerous current well-known IQA algorithms on our database. The criteria and results of current IQA algorithms are also provided in this section. Section 3.4 establishes a new algorithm, MAD_{DWT}, designed specifically for JPEG2000 coding which uses local DWT coefficient statistics based on the framework of MAD. A quick review of MAD framework, our modification, and MAD_{DWT}'s performance are also provided. At the end of the chapter, the summary of the whole chapter is provided in the Section

3.5.

3.2 A Database For JPEG2000 Compression

In this section, we provide the experiment methodology to obtain the first ever image database, which contains only non-MSE-standard JPEG2000 images.

3.2.1 Images and compression

We selected 24 color images of size 512×512 pixels, which span a variety of commonplace subject matters in five categories: *animals*, *landscapes*, *people*, *plants*, and *urban*, mostly from the CSIQ database [2] and personal sources. The original images were converted to grayscale for use in the experiment.

A subset of five original images is shown in Figure 3.1. Five images span a huge range of image's contents, for example, the image *geckos* has a sandy texture, the image *log_seaside* and the image *child_swimming* have a different texture in glass, they both have some structure regions in the log and the baby, the image *log_seaside* also contains some very strong edge regions, the images *cactus* and *monument* have some blank/smooth regions, and the image *monument* has some blurred regions.

Let I denote the grayscale 8-bit intensity of one image ($I = 0.2989R + 0.5870G + 0.1140B$, where R, G, and B denote the 8-bit red, green, and blue intensities). First, a Discrete Wavelet Transform (DWT) was applied to the image I using the 9/7 biorthogonal filters (as used in JPEG2000 [37]) and four levels of decomposition. (The center radial spatial frequencies are 1.15, 2.3, 4.6, and 9.2 cycles per degree of visual angle (cpd), respectively.) The DWT subbands were then distorted via scalar quantization. In our experiment, the effects of orientation were not tested; each triplet of subbands at the same level of decomposition was distorted with the same amount.

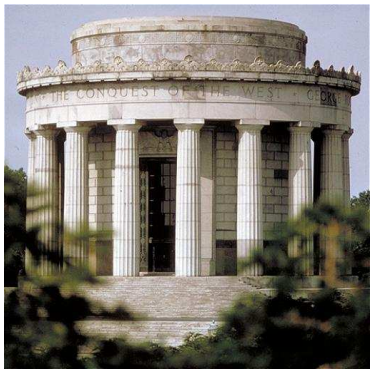
Figure 3.2 shows an example of four levels of DWT decomposition of the image *monument*. At each level $i = 1, 2, 3, 4$, the three subbands (LH, HL, HH) were treated

geckos

log_seaside

child_swimming

cactus



monument

Figure 3.1: Some original images used in the experiment. They span a variety of commonplace subject matters in five categories *animals*, *landscapes*, *people*, *plants*, and *urban*. They also contain different regions, such as texture, edge, structure, blank, smooth, and blurred regions.

equally.

Let $\hat{I}$ denote the reconstructed image. The distortion is given by $E = \hat{I} - I + \mu_I$, where μ_I is the average of I. The Root Mean Square (RMS) contrast C_{TOT} is given by:

$$C_{TOT} = \frac{1}{\mu_{L_I}} \left(\frac{1}{N} \Sigma_k \left[L_E(k) - \mu_{L_E} \right]^2 \right)^{1/2} \tag{3.1}$$

where μ_L denotes the average luminance (μ for the average, L for the luminance domain), $L_E(k)$ is the luminance of k^{th} pixel and N is total number of pixels.

The distortion contrast D_i at level i is calculated from RMS contrast C_{TOT} in the

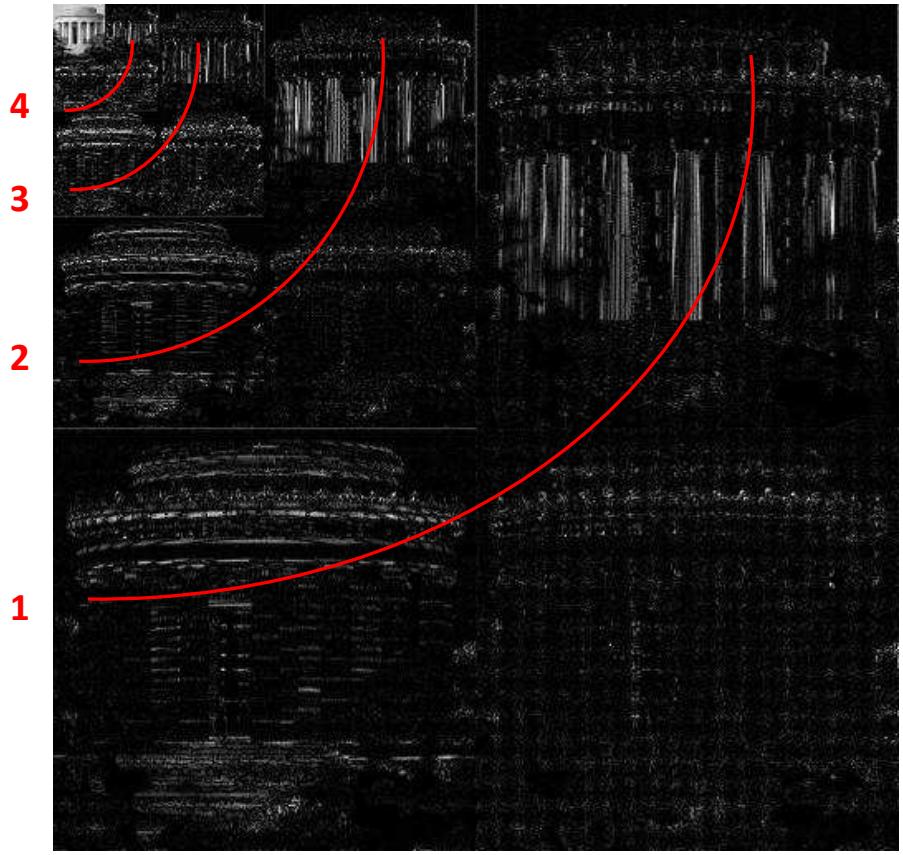


Figure 3.2: An example of four levels of DWT decomposition of the image *monument*. The effects of orientation were not tested; each triplet of subbands (LH, HL, HH) at the same level $i = 1, 2, 3, 4$ of decomposition was distorted with the same amount.

pixel domain by:

$$D_i \approx 2^{2i} \times C_{TOT}^2 \times \zeta \tag{3.2}$$

where ζ is a parameter that takes into account viewing parameters such as display gamma [36].

From equations 3.1 and 3.2, the total distortion contrast D is given by:

$$D = \sum_i (D_i) \tag{3.3}$$

To generate images with the same desired total distortion contrast $D = 0.12$, we chose the quantization step sizes via bisection search to meet the desired distortion contrast of each level D_i. To limit the number of possible proportions, we allowed

only four different amounts of distortion per level (0%, 20%, 40%, or 60% of D): $D_i \in (0\%, 20\%, 40\%, 60\% \times D)$ was chosen for level i such that $\sum_i(D_i) = 100\% \times D$. This scheme resulted in 40 distorted versions of each original image.

Some of images generated from the images *archway* and *log_seaside* are shown in Figure 3.3. The image in Figure 3.3(a) is distorted with 60% amount of distortion on level 1 and 40% distortion on level 2. Figure 3.3(b) has 40% amount of distortion on level 2, 40% distortion on level 3, and 20% on level 4. Figure 3.3(c) has 60% amount of distortion on level 1, 20% distortion on level 2, and 20% distortion on level 3. Figure 3.3(d) has 20% amount of distortion on level 1, 60% distortion on level 2, and 20% amount of distortion on level 4.

3.2.2 Experiment

To obtain the subjective ranking order of 40 distorted versions for each original image, we used a pairwise-comparison paradigm. Subjects were asked to choose the better quality image between each pair of distorted images.

The interface of the experiment to collect ranking order is shown in Figure 3.4. Each pair of distorted images was displayed against a mid-gray background. Subjects chose the better quality image by pushing the corresponding buttons. Subject could also use keyboard shortcut to decide which one is better: left arrow if the left image is higher quality, or right arrow for the image on the right side. Subject was forced to choose a winner, even if two images were at about the same quality level.

A merge-sort method was employed to reduce the number of comparison pairs from $\binom{40}{2} = 780$ to roughly $40\log(40) \approx 150$ pairs. The idea of using merge-sort is if the quality of image $A1$ is lower than image $A2$, and the quality of image $A2$ is lower than image $A3$, we can conclude that image $A1$ is lower quality than image $A3$.

Each subject was asked to perform the experiment with 10 to 20 original images to ensure that each original image was ranked by at least six subjects. All subjects

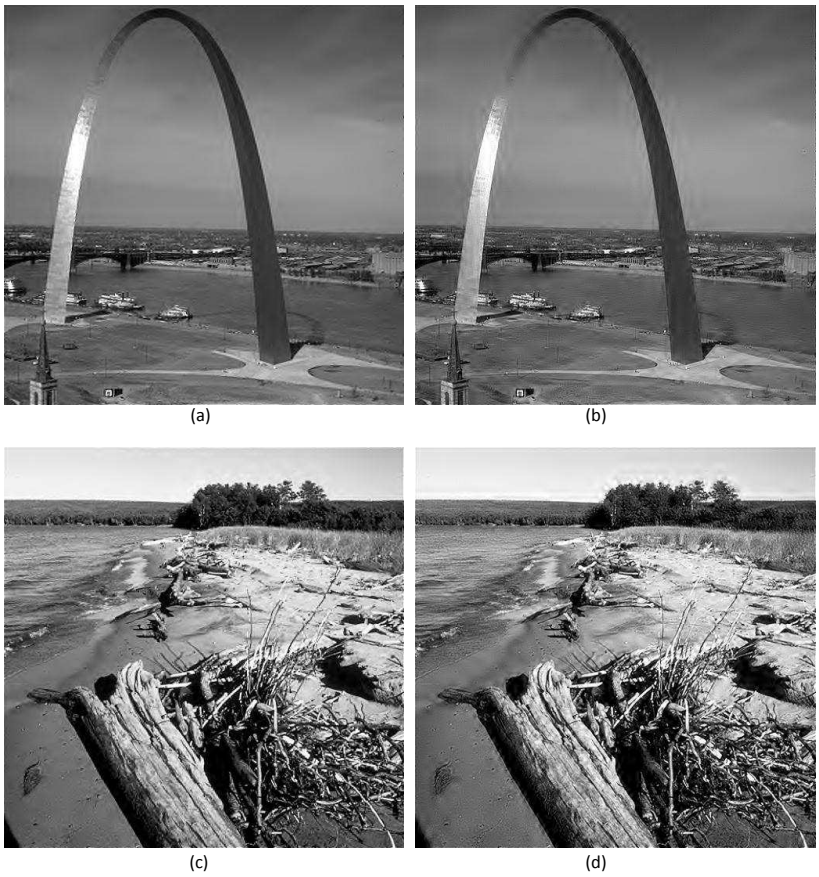


Figure 3.3: Some images used in the experiment. Image (a) is distorted with 60% amount of distortion on level 1, and 40% distortion on level 2. Image (b) has 40% amount of distortion on level 2, 40% distortion on level 3, and 20% on level 4. Image (c) has 60% amount of distortion on level 1, 20% distortion on level 2, and 20% distortion on level 3. Image (d) has 20% amount of distortion on level 1, 60% distortion on level 2, and 20% amount of distortion on level 4.

were able to finish each trial within one hour. Each subject could finish all 10-20 trials in several days, as long as he or she felt good in the doing mood.

3.2.3 Apparatus and Subjects

Images were displayed on a professional-grade, widegamut LaCie 324 24-inch LCD monitor (1920x1200 at 60 Hz; 92% NTSC color gamut). The display yielded minimum and maximum luminances of 0.80 and 259 cd/m^2, respectively, with a luminance gamma of 2.2. Images were viewed binocularly through natural pupils in a darkened

Figure 3.4: The interface of the experiment to collect ranking order.

room at a distance of 60 cm. Nine adult subjects, all with normal or corrected-to-normal vision, took part in the experiment. Subjects ranged from 23 to 34 years in age.

3.2.4 Results And Image Database

By processing results of subjective ratings, we obtained quality rankings of 40 distorted versions for each original image. The five highest-quality images of each original image rated by each subject were taken, and the amounts of distortion for each level of decomposition were computed by averaging those of these highest-quality images.

The amounts of distortion for four levels that generated the highest-quality images for a subset of the original images are shown in Figure 3.5(a). The x-axis represents the spatial frequencies of four levels and the y-axis represents amounts of distortion

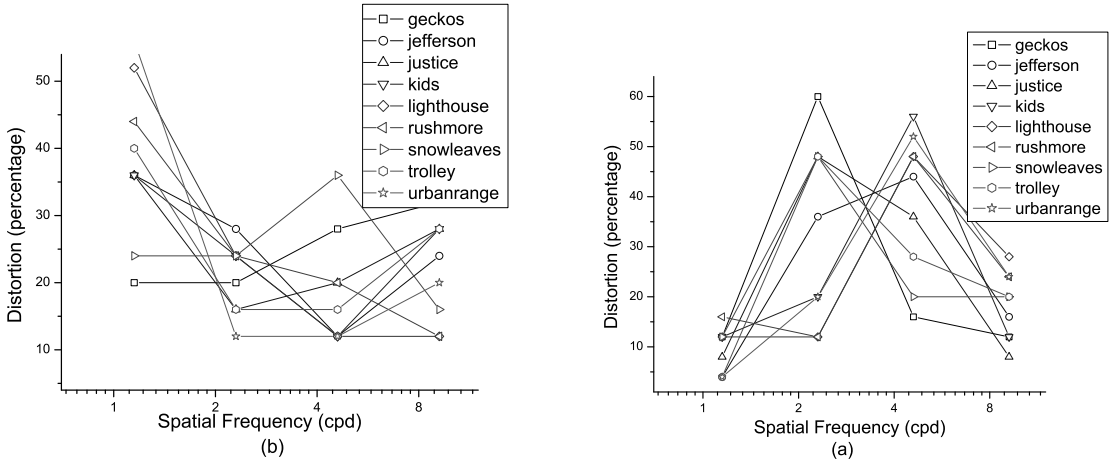


Figure 3.5: The amounts of distortion for four levels that generated highest-quality images (a), and lowest-quality images (b).

(percentage). For example, for the image *justice*, in order to have higher quality images, we should allocate 36% amounts of distortion in level one, 16% in level two, 20% in level three and 28% in level four; for image *rushmore*, we should allocate 44%, 24%, 20%, 12% for level one, two, three, and four, respectively.

The distribution of distortions that created the lowest-quality images was also studied with the same method as for the highest-quality images. The amounts of distortion for four levels that created the lowest-quality images are provided in Figure 3.5(b).

The averages of all highest-quality and lowest-quality distortion-allocations of all 24 images are shown in Figure 3.6. From this figure, we can see that, in general, higher-quality images were achieved by allocating most of the distortion to the finest scale and the least to the coarser scales. On the other hand, the lower-quality images were achieved by allocating the least distortion to finest scale, a little bit more on coarsest scale, and most distortion to mid-frequency bands. However, the distortion-allocation strategy which yielded the highest-ranked and lowest-ranked images showed considerable variation from image to image.

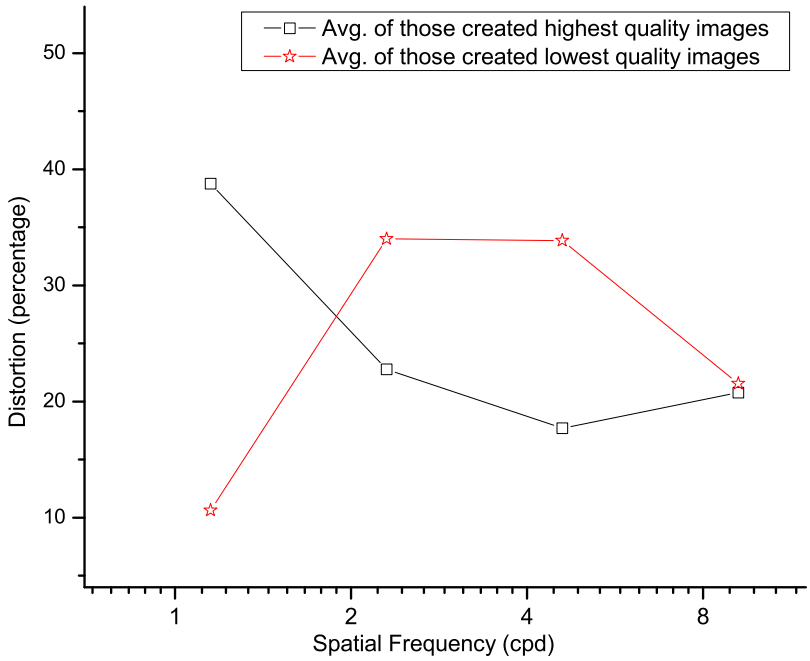


Figure 3.6: The average amounts of distortion for four levels that generated highest-quality images and lowest-quality images.

For each original image, four distorted versions using four distortion-allocation proportions were created. The first proportion that created the best-quality image was obtained from the distorted versions with highest quality, as in Figure 3.5(a). The second and third proportions that created the best-by-average, worst-by-average images were obtained from the average of proportions that created the highest-quality and lowest-quality images, respectively, as in Figure 3.5(c). The worst-quality images are created from fourth proportion obtained from the distorted versions with lowest-quality, as in Figure 3.5(b). Two sets of four distorted images are shown in Figure 3.7. The best, best-by-average, worst-by-average, and worst-quality images of the image *waterside archway* and the image *urban ranger* are located from left to right. The full set of 4×24 images is available online [38].

Figure 3.7: Two sets of four images generated using various distortion-allocation proportions. The images labeled (a)-(d), (e)-(h) are the best, best-by-average, worst-by-average, and worst quality images for image *waterside archway* and *urban ranger*, respectively, based on the rankings from our experiment

3.2.5 Summary

This chapter presented a new JPEG2000 compression image database, which includes 24 reference images and 96 distorted images and their rankings. This is the first ever image database which includes all the non-MSE standard JPEG2000 compression images. The highest-quality, lowest-quality, and average curves (distortion allocations) are also provided. The image database is publically available to download at [38].

3.3 IQA Algorithms on Jpeg2000 Compression

In this section, we first propose two criteria to evaluate an algorithm's performance in predicting the rankings in our database. We then present the analysis results of several well-known IQA algorithms using these criteria.

3.3.1 Proposed Criteria

To evaluate the abilities of IQA algorithms in predicting these rankings, we first converted the algorithms' scores to the ranking order, 1, 2, 3, and 4, and then compared these predicted rankings to the rankings of the image database (i.e., as provided by the experiment). We proposed two criteria to evaluate the performance: the number of correct positions, C, and the precision of prediction, P.

Scalar C is given by:

$$C = \sum_{i=1}^{24} \sum_{j=1}^{4} H(i,j) \tag{3.4}$$

where $H(i,j) = 1$ if $A(i,j) = D(i,j)$, $H(i,j) = 0$ if $A(i,j) \neq D(i,j)$, and $A(i,j)$ and $D(i,j)$ denote the rank order of algorithm and database, respectively, of the image number i and distorted version j.

Scalar P is given by

$$P = \frac{1}{480} \sum_{i=1}^{24} \sum_{j=1}^{4} (A(i,j) - D(i,j))^2 \tag{3.5}$$

where the quantity 480 is used to normalize P to the range of $[0,1]$.

Note that the C score is an integer in the range $[0,96]$; the higher C, the better algorithm; and the P score is a real value in the range $[0,1]$, the lower P, the better the prediction. A perfect prediction would give $C = 96$ and $P = 0$. A higher C score does not guarantee a lower P score. For example, if two algorithms have the same C scores $C = 94$, it means that two algorithms predict 94 positions correctly, and two positions incorrectly. However, the P scores can be different, such as when an algorithm misplaces first and second positions and the other misplaces the first and fourth position. In this case, the P score will indicate that the first algorithm is better than the second algorithm.

3.3.2 Results of Existing IQA Algorithms

We tested the ability of various IQA algorithms in predicting the rankings in this image database (NQM [31], PSNR [39], SSIM [5], MS-SSIM [6], VIF & VIFP [15], VSNR [32], MAD [7]). The performances of these IQA algorithms using the criteria proposed in Section 3.3.1 are shown in Table 3.1 with two best performances bolded.

Table 3.1: The performances of several well-known IQA algorithms. Two best performances are bolded.

	C scores	**P scores**
SSIM	38	0.4250
PSNR	42	0.3135
MS-SSIM	**65**	**0.0833**
VIF	61	0.1250
VIFP	51	0.2167
NQM	60	0.3458
VSNR	**66**	**0.0750**
MAD	54	0.1625

From Table 3.1, two best algorithms are VSNR and MS-SSIM with C scores of 66 and 65, P scores of 0.0750 and 0.0833, respectively. The results of VSNR and MS-SSIM are described in detail in Table 3.2-3.4.

Table 3.2 shows the results of VNSR's quality scores for all 96 images. Note that the VSNR quality scores are in dB, in the range $0 - \inf$; the higher VSNR, the higher quality. These scores are then converted to ranking scores (1, 2, 3, and 4) for each image and they are given in Table 3.3. Table 3.3 shows that VSNR has failed to predict 30 positions ($C = 66$). VSNR has perfectly predicted the fourth positions of all images by pointing out the lowest quality images. For the third positions, VSNR also gives a good results; it only fails on the images *redwood* and *foxy*. VSNR swaps the highest quality image with the second best several times.

The ranking scores of MS-SSIM are also given in Table 3.4. As seen in this table, MS-SSIM is also a good predictor with 65 correct positions. However, for the images *redwood* and *foxy*, MS-SSIM gives the worst images the highest scores. For the images

Table 3.2: VNSR's quality scores for all 96 images

Image name	**Levels**			
	1	2	3	4
child_swimming	27.76	26.34	21.33	20.24
log_seaside	31.22	29.81	24.52	20.03
monument	27.05	28.98	24.15	23.24
redwood	25.97	31.96	26.88	25.55
swarm	33.01	30.50	25.51	22.12
archway	27.28	26.03	21.25	19.81
bridge	30.50	29.05	24.32	22.47
cactus	27.38	29.81	24.23	19.35
child_fishermen	30.30	29.94	25.04	24.45
couple	25.89	28.14	23.01	19.19
deer	23.98	25.17	20.29	16.41
desert_tree	29.94	28.87	23.69	23.49
foxy	25.14	30.80	25.49	21.13
geckos	21.14	25.25	20.19	15.87
jefferson	24.31	25.15	20.21	18.28
justice	29.59	31.00	25.80	23.42
kids	33.61	34.86	29.50	24.91
lighthouse	32.68	30.40	25.33	22.84
rushmore	28.51	27.61	22.65	20.65
snow_leaves	30.34	34.14	28.80	27.19
trolley	28.76	29.74	24.45	22.54
urban_ranger	29.38	27.65	23.86	21.30
veggies	29.33	32.90	27.52	23.62
waterside_archway	26.56	27.82	22.78	20.93

Table 3.3: VNSR's quality rankings for all 96 images

Image name	Database index				Image name	Database index			
	1	2	3	4		1	2	3	4
child_swimming	1	2	3	4	*foxy*	3	1	2	4
log_seaside	1	2	3	4	*geckos*	2	1	3	4
monument	2	1	3	4	*jefferson*	2	1	3	4
redwood	3	1	2	4	*justice*	2	1	3	4
swarm	1	2	3	4	*kids*	2	1	3	4
archway	1	2	3	4	*lighthouse*	1	2	3	4
bridge	1	2	3	4	*rushmore*	1	2	3	4
cactus	2	1	3	4	*snow_leaves*	2	1	3	4
child_fishermen	1	2	3	4	*trolley*	2	1	3	4
couple	2	1	3	4	*urban_ranger*	1	2	3	4
deer	2	1	3	4	*veggies*	2	1	3	4
desert_tree	1	2	3	4	*waterside_archway*	2	1	3	4

geckos and *veggies*, MS-SSIM rates the third-position images as the most beautiful images.

Although all tested algorithms failed to predict the correct rankings on many of the images, the results presented in Table 3.1 are not too surprising. We can see that SSIM performs poorly ($C = 38$ and $P = 0.4250$), but MS-SSIM is much better by considering the multiple scales ($C = 65$ and $P = 0.0833$). Moreover, as reported in [7], MS-SSIM, VIF, and MAD are three best IQA algorithms on the LIVE [1], CSIQ [2], TID [27], Toyama [40] image databases, but their performances are not as good as VSNR and MS-SSIM on our JPEG2000 database. Again, a good algorithm for general type distortion cannot guarantee the same success in JPEG2000 images, especially non-MSE-standard JPEG2000 images.

3.4 A New Algorithm Using DWT

As stated in previous section, predicting quality rankings in this database is challenging for current IQA algorithms because they are not designed to handle the situation where the same distortion is distributed differently across subbands. Here, we pro-

Table 3.4: MS-SSIM's quality rankings for all 96 images

Image name	Database index				Image name	Database index			
	1	2	3	4		1	2	3	4
child_swimming	1	2	3	4	*foxy*	4	1	2	3
log_seaside	2	1	3	4	*geckos*	3	1	2	4
monument	2	1	3	4	*jefferson*	2	1	3	4
redwood	4	1	2	3	*justice*	2	1	3	4
swarm	1	2	3	4	*kids*	2	1	3	4
archway	1	2	3	4	*lighthouse*	1	2	3	4
bridge	1	2	3	4	*rushmore*	1	2	3	4
cactus	2	1	3	4	*snow_leaves*	2	1	3	4
child_fishermen	1	2	3	4	*trolley*	2	1	3	4
couple	2	1	3	4	*urban_ranger*	1	2	3	4
deer	2	1	3	4	*veggies*	3	1	2	4
desert_tree	1	2	3	4	*waterside_archway*	2	1	3	4

pose a new IQA algorithm, MAD_{DWT}, designed specifically for JPEG2000 coding which uses local DWT coefficient statistics based on the framework of Most Apparent Distortion (MAD) [7]. First, we provide a brief overview of MAD, including the idea, block diagrams, and the algorithm's steps. Next, we present the MAD_{DWT} algorithm which is based on MAD's framework. Finally, we conclude the section with MAD_{DWT}'s results on our JPEG2000 database.

3.4.1 Overview of Most Apparent Distortion (MAD)

The Most Apparent Distortion (MAD) algorithm was developed by Larson and Chandler [7] in 2010. MAD uses two strategies to estimate image quality: a detection-based strategy is used for near threshold distortions and an appearance-based strategy is used for clearly visible distortion. The MAD algorithm includes two main 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.

Figure 3.8 shows the two main stage of the original MAD algorithm. In each stage,

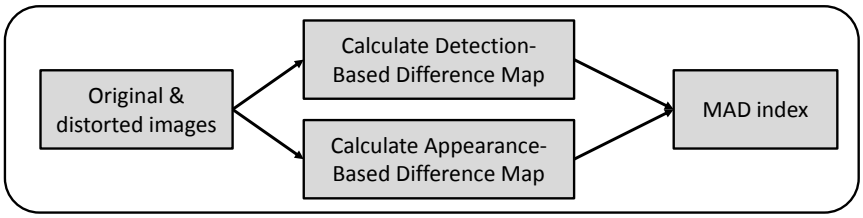


Figure 3.8: Original MADs framework with two stages: a detection-based stage and an appearance-based stage.

MAD operates in a block-based fashion. All small blocks are treated equally and the differences between reference and distorted properties form the detection-based and appearance-based maps. The detail of calculating these maps and the collapsing method are provided in the following subsections.

3.4.2 Detection-based map calculating

Figure 3.9 provides a block diagram of the detection-based map calculating in MAD. The pixel values of the reference and distorted images are first converted to lightness values; then, a model of the human contrast sensitivity function is applied via spatial filtering; then, local RMS contrast values are computed to quantify the masking capability (the ability to hide distortion) of each region in the reference image. This resulting masking map is then used to weight a block-based MSE map computed in the lightness domain to form MAD detection-based map.

This stage requires the grayscale images only, therefore preprocessing step is needed. The pixel values of the reference and distorted images are first converted to grayscale via $I = 0.2989R + 0.5870G + 0.1140B$, where R, G, and B denote the 8-bit red, green, and blue intensities. After that, the grayscale I is converted to lightness values via $L = (a + k\mathrm{I})^{\gamma/3}$ where $a = 0$, $k = 0.02874$, and $\gamma = 2.2$ for 8-bit pixel values of an sRGB display. The division by 3 attempts to take into account the nonlinear HVS response to luminance by converting luminance into perceived luminance.

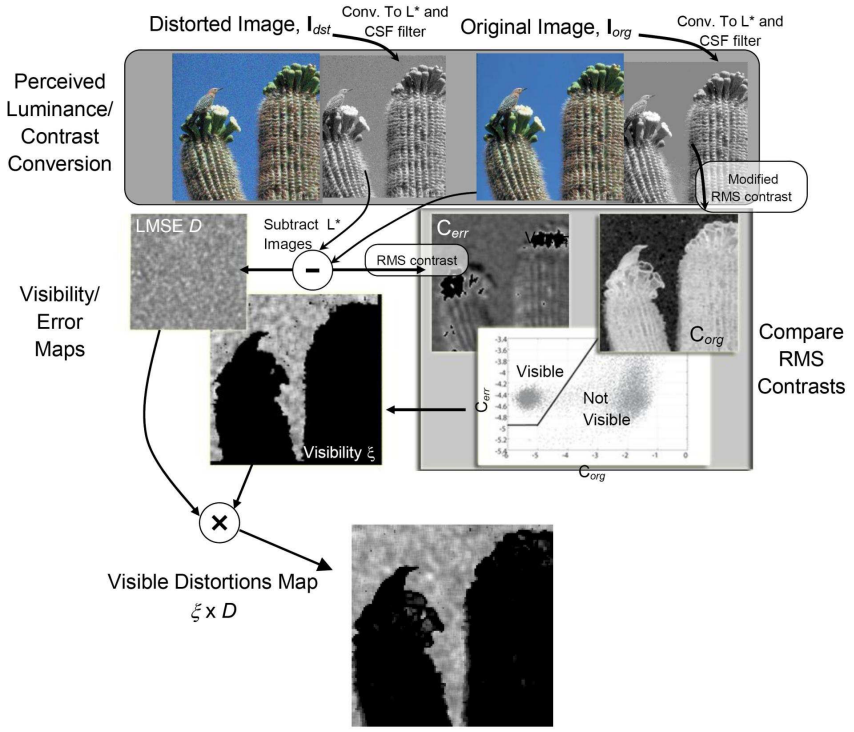


Figure 3.9: Block diagram of the detection-based strategy used to compute a visible distortion map. Both the reference and the distorted image are first converted to perceived luminance, and then filtered by a contrast sensitivity function (CSF). The local distortion visibility map is obtained by comparing the local contrast of the reference image and the error image.

Next, a model of the human contrast sensitivity function is applied via spatial filtering. The contrast sensitivity function (CSF) [41] is applied by filtering both the reference image L and the error image $\Delta L = L - \hat{L}$ via:

$$\tilde{L} = \mathbb{F}^{-1}[H(u,v) \times \mathbb{F}[L]] \tag{3.6}$$

where $\mathbb{F}$ and $\mathbb{F}^{-1}$ denote the DFT and inverse DFT, respectively; $H(u,v)$ is the DFT-based version of the CSF function defined by Equation (3) in Ref. [7].

The ability to hide distortion is different for different image regions. This is so-called *masking capability*. To quantify the masking capability of each region in the reference image, MAD employs a simple spatial-domain measure of contrast masking. The local RMS contrast values are computed for each small block of size 16×16

with 75% overlap between neighboring blocks. Two local RMS contrast maps of the reference and error images are employed to compute a local distortion visibility map. This map reflects how human can see distortion differently in different regions, even thought the distortion is uniform for the whole image.

A block-based MSE map is then computed in the lightness domain in the same block-based fashion from the reference and distorted images. This MSE map is then weighted by the local distortion visibility map via a point-by-point multiplication to form *detection-based map*.

Figure 3.9 illustrates the process of detection-based strategy for the distorted (noise added) and reference images the image *cactus*. The white blocks of final visible distortion map show us that the noise is easily detected in the smooth regions (e.g. sky). The cactus itself can hide distortion, it is not easy to see noise here, and thus creates the darker blocks.

3.4.3 Appearance-based map calculating

Figure 3.10 provides a block diagram of the appearance-based map calculating. MAD employs a computational neural model using a log-Gabor filterbank, which implements both even-symmetric (cosine-phase) and odd-symmetric (sine-phase) filters. The even and odd filter outputs are then collapsed to yield magnitude-only subband values. The variance, skewness, and kurtosis are then computed for each 16×16 block (with 75% overlap between blocks) of each subband of the reference and distorted images. The appearance-based map is then given from the differences between these statistics of the reference and distorted images.

The log-Gabor filterbank employed here is implemented with five scales $s \in \{1, 2, 3, 4, 5\}$ and four orientations $o \in \{1, 2, 3, 4\}$, and thus yields 20 subbands for each image. Each subband is then divided into small blocks of size 16×16 with 75% overlap between neighboring blocks. With each block, the standard deviation, skew-

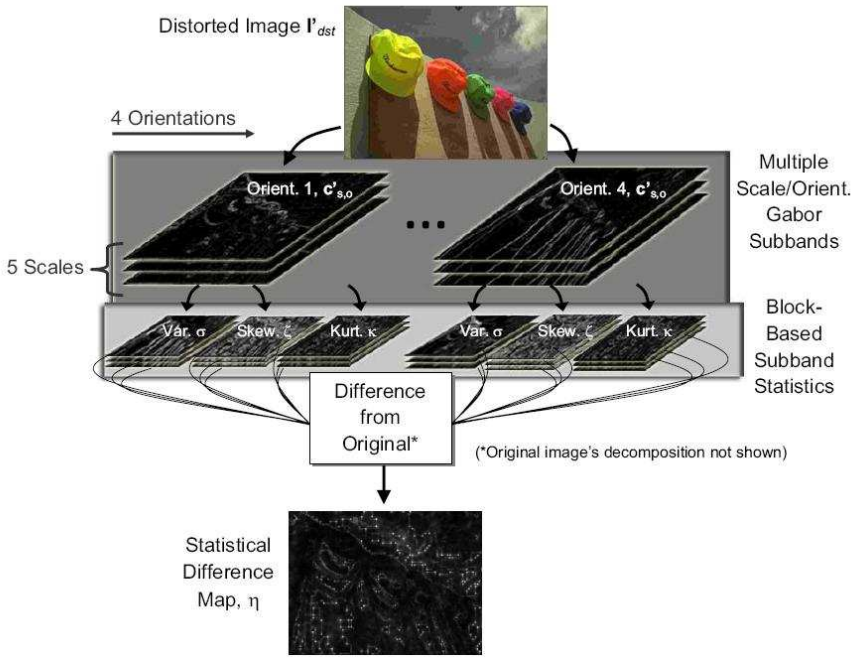


Figure 3.10: Block diagram of the appearance-based strategy used to compute a statistical difference map. A computational neural model using a log-Gabor filterbank is employed. The standard deviation, skewness, and kurtosis are computed for each subband's block. The differences of local standard deviation, skewness, and kurtosis between each pair of reference and distorted subbands are combined into a statistical difference map.

ness, and kurtosis are calculated. Let $\sigma^{s,o}(b)$, $\varsigma^{s,o}(b)$, and $\kappa^{s,o}(b)$ denote the standard deviation, skewness, and kurtosis computed from a block located at b of the subband at s^{th} scale and o^{th} orientation. The appearance-based map, which is statistical difference map, is computed as the weighted combination of the differences in σ, ς, and κ for all subbands. Specifically, appearance-based map, AM, at the location b is given by:

$$AM(b) = \sum_{s=1}^{5}\sum_{o=1}^{4} w_s[|\sigma^{s,o}(b) - \hat{\sigma}^{s,o}(b)| + 2|\varsigma^{s,o}(b) - \hat{\varsigma}^{s,o}(b)| + |\kappa^{s,o}(b) - \hat{\kappa}^{s,o}(b)|]. \quad (3.7)$$

where the weights $w_s = \{0.5, 0.75, 1, 5, 6\}$ (for the finest to coarsest scales, respectively) are used to combine the differences across scales (see Ref. [7] for more details).

3.4.4 Collapsing and combining

From detection-based and appearance-based maps obtained from previous steps, a 2-norm is applied to collapse two maps into two single indices, d_{detect} and d_{appear}, respectively (e.g. $d_{appear} = \left(\frac{1}{P}\sum_p AM(p)^2\right)^{1/2}$). The two scalar values d_{detect} and d_{appear} are combined into an overall distortion value via a weighted geometric mean:

$$\text{MAD} = d_{detect}^{\alpha} \times d_{appear}^{1-\alpha}, \tag{3.8}$$

where the weight α is selected to give greater contribution from d_{detect} for mildly distorted images and greater contribution from d_{appear} for more heavily distorted images (see Equation 13 in [7]).

3.4.5 MAD using Discrete Wavelet Transform

Although MAD performs very well in predicting subjective ratings of image quality for both general distortions and particular distortion including standard JPEG2000 compression distortion, MAD does not perform well on our JPEG2000 database. Using the same two-strategy framework, we propose a new algorithm, MAD_{DWT}, a modification of MAD, using discrete wavelet transform decomposition.

In MAD_{DWT}, the appearance-based stage is performed by using the DWT subbands not the log-Gabor-filtered images as in original MAD. The appearance-based maps of the original MAD and MAD_{DWT} are shown in Figure 3.11(a) and (b), respectively. Original MAD uses log-Gabor filter with four orientations and five scales (20 subbands), while MAD_{DWT} employs discrete wavelet transform decomposition with four scales ($4 \times 3 + 1 = 13$ subbands).

We used the same method to calculate statistical differences. From all 13 pairs of subbands for the reference and distorted images, the local variance, skewness, and kurtosis are computed for each block of the size 16×16 pixels (with 75% overlap

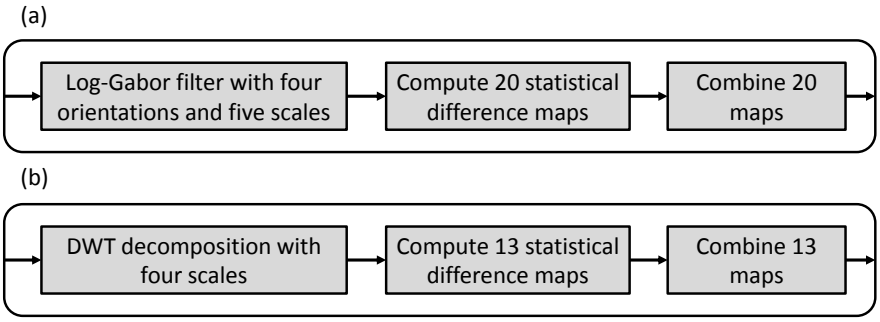


Figure 3.11: Block diagram of the appearance-based maps of the original MAD and MAD_{DWT}. A computational neural model using a log-Gabor filterbank is employed. Original MAD uses log-Gabor filter with four orientations and five scales (20 subbands) in Figure (a), while MAD_{DWT} employs discrete wavelet transform decomposition with four scales ($4 \times 3 + 1 = 13$ subbands) in Figure (b).

between blocks). However, the appearance-based map calculating need a different set of weights to combine across scales. Specifically, The block b of the new appearance-based map is computed as follows:

$$AM(b) = \sum_{i=1}^{4} w_i \sum_{B} \left(|\sigma^{s,o}(b) - \hat{\sigma}^{s,o}(b)| + 2|\varsigma^{s,o}(b) - \hat{\varsigma}^{s,o}(b)| + |\kappa^{s,o}(b) - \hat{\kappa}^{s,o}(b)| \right). \tag{3.9}$$

where $\sigma_{i,B}(b)$, $\xi_{i,B}(b)$, $\kappa_{i,B}(b)$ denote, respectively, the standard deviation, skewness, and kurtosis of the block b corresponding to level i, and subband B. The value $i = 1, 2, 3, 4$ is for the finest to coarsest scales, respectively, and subband B, $B \in (LH, HL, HH)$ for three subbands in each triplet for first three levels, $B \in (LL, LH, HL, HH)$ for level 4. The weight vector $\mathbf{w} = [1\ 40\ 20\ 5]^T$ is chosen for the best fit.

3.4.6 MAD_{DWT} results

The ranking scores of MAD_{DWT} are given in Table 3.5. We can see that MAD_{DWT} performs very well on most of the images. MAD_{DWT} is able to predict the rankings perfectly for the third and fourth positions, except the image *veggies*. For the other 25 images, MAD_{DWT} fails to predict only 22 images on the first and second positions.

Table 3.5: MAD_{DWT}'s ranking scores for all 96 images

Image name	Database index				Image name	Database index			
	1	2	3	4		1	2	3	4
child_swimming	1	2	3	4	*foxy*	2	1	3	4
log_seaside	2	1	3	4	*geckos*	2	1	3	4
monument	2	1	3	4	*jefferson*	2	1	3	4
redwood	2	1	3	4	*justice*	1	2	3	4
swarm	1	2	3	4	*kids*	2	1	3	4
archway	2	1	3	4	*lighthouse*	1	2	3	4
bridge	1	2	3	4	*rushmore*	1	2	3	4
cactus	1	2	3	4	*snow_leaves*	2	1	3	4
child_fishermen	1	2	3	4	*trolley*	1	2	3	4
couple	1	2	3	4	*urban_ranger*	1	2	3	4
deer	2	1	3	4	*veggies*	3	1	2	4
desert_tree	2	1	3	4	*waterside_archway*	1	2	3	4

Table 3.6: The performances of MAD_{DWT} and other IQA algorithms. Two best performances are bolded.

	C scores	**P scores**
SSIM	38	0.4250
PSNR	42	0.3135
MS-SSIM	65	0.0833
VIF	61	0.1250
VIFP	51	0.2167
NQM	60	0.3458
VSNR	**66**	**0.0750**
MAD	54	0.1625
MAD_{DWT}	**71**	**0.0583**

This is much better than the current best algorithm, VSNR, with 30 failed positions.

The performance of MAD_{DWT} is given in Table 3.6 other well-known algorithms. Two best performances are bolded. Table 3.6 shows that MAD_{DWT} outperforms all other IQA algorithms in terms of both C scores and P scores with $C = 71$ and $P = 0.0583$. VSNR and MS-SSIM also have competitive performances with MAD_{DWT}.

3.5 Summary

In this chapter, we presented an image database designed specifically for JPEG2000 images with a fixed amount of total distortion, but in which the distortions were allocated to different frequency bands in different proportions. The result from subjective ranking experiment revealed that, in general, higher-quality images were achieved by allocating most of the distortion to the finest scale and the least to the coarse-to-mid scales.

We also provided an analysis of existing IQA algorithms on this database which revealed that even though the algorithms perform reasonably well on JPEG2000-compressed images in popular image-quality databases, they often fail to predict the correct rankings on the images here.

We also provided a new IQA algorithm, MAD_{DWT}, which designed specifically for JPEG2000 coding. MAD_{DWT} was developed based on MAD algorithm. The MAD_{DWT} employed DWT instead of log-Gabor filtering in original MAD, with different weights to combine across scales. The results demonstrated that MAD_{DWT} outperforms all other IQA algorithms in terms of both C scores and P scores on this database.

CHAPTER 4

MICROARCHITECTURAL ANALYSIS OF IQA ALGORITHMS

4.1 Background

Although a great deal of research on IQA has focused on improving prediction accuracy, much less research has addressed performance issues with respect to algorithmic, microarchitechtural efficiency, and program execution speed. As IQA algorithms move from the research environment into more mainstream applications, issues surrounding efficiency—such as execution speed and memory bandwidth requirements—begin to emerge as equally important performance criteria. Many IQA algorithms which excel in terms of prediction accuracy fall short in terms of efficiency, often requiring relatively large memory footprints and runtimes on the order of seconds for even modest-sized images (e.g., < 1 megapixel). As these algorithms are adapted to process frames of video (e.g., Refs. [42, 43]), or are used during optimization procedures (e.g., during RD optimization in a coding context), efficiency becomes of even greater importance.

From a signal-processing viewpoint, it would seem that the bulk of computation and runtime are likely to occur in two key stages which are employed by most IQA algorithms: (1) local frequency-based decompositions of the input image(s); and (2) local statistical computations on the frequency coefficients. The first of these two stages can potentially require a considerable amount of computation and memory bandwidth, particularly when a large number of frequency bands are analyzed, and when the decomposition must be applied to the image as a whole. The latter of these two stages would seem to require more computation, particularly when multiple statis-

tical computations are computed for each local region of coefficients. For example, in MS-SSIM [14], an image is decomposed into different scales, and local image statistics are computed for each block of coefficients (via a sliding window). In VIF [15], wavelet subband covariances can be computed via a block-based or overlapping block-based approach. In MAD [7], variances, skewness, and kurtosis of log-Gabor coefficients are also computed for overlapping blocks in each subband. These approaches have been argued to mimic the cortical processing in the human visual system (HVS) in which the statistics of local responses of neurons in primary visual cortex (modeled as coefficients) are computed and compared in higher-level visual areas. Yet, unlike the HVS, most modern computing machines lack dedicated hardware for computing the coefficients and their local statistics.

Due to their extensive use in image compression and computer vision, 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 FFT (e.g., Ref. [44]) and by exploiting various algebraic and structural properties of the transform, e.g., via recursion Ref. [45], lifting Ref. [46], matrix factorization Ref. [47], cyclic convolution Ref. [48], and many other techniques (see Ref. [49] 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., Ref. [50–53]). Algorithm- and hardware-based acceleration has also been researched for the discrete wavelet transform (e.g., Refs. [54–56]) and Gabor transform (e.g., Refs. [57–60]).

Techniques for accelerating the computation of local statistics in images have also been researched, though to a much lesser extent than the transforms. One technique, called *integral images*, which was originally developed in the context of computer graphics [61], 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. A similar technique can be used for computing higher-order moments such as the variance, skewness, and kurtosis (see, e.g., Refs. [62, 63]).

In Ref. [64], 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 [1]. 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 Ref. [65], 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.

In Ref. [63], Phan *et al.* presented the results of a performance analysis and techniques for accelerating the MAD algorithm [7]. Although MAD is among the best in predictive performance, it is currently the one of the slowest IQA algorithms when tested on several modern computers (Intel Core 2 and Xeon CPUs; see Ref. [63]). A performance analysis revealed that the main bottleneck in MAD stemmed from its appearance-based stage, which accounted for 98% of the total runtime. Within this appearance-based stage, the computation of the local statistical differences accounted for most of the runtime, and computation of the log-Gabor decomposition accounted for the bulk of the remainder. Phan *et al.* proposed and tested four techniques of acceleration: (1) Using integral images for the local statistical computations; (2) using procedure expansion and strength reduction; (3) using a GPGPU implementation of the log-Gabor decomposition; and (4) precomputation and caching of the log-Gabor filters. The first two of these modifications resulted in an approximately 17x speedup over the original MAD implementation. The latter two resulted in an approximately 47x speedup over the original MAD implementation.

Although these studies have successfully yielded more efficient versions of their

respective algorithms, several larger questions remain unanswered, especially with respect to IQA algorithms: To what extent are the bottlenecks in IQA algorithms attributable to the decomposition and statistical computation stages vs. more algorithm-specific auxiliary computations? To what extent are the bottlenecks attributable to computational complexity vs. limitations in memory bandwidth? Are there generic implementation techniques or microarchitectural modifications that can be used to accelerate all or at least several IQA algorithms? The answers to these questions can provide important insights for (1) designing new IQA algorithms, which are likely to draw on multiple approaches used in several existing IQA algorithms; (2) efficiently implementing multiple IQA algorithms on a given hardware platform; (3) efficiently applying multiple IQA algorithms to specific applications; and (4) selecting and/or designing specific hardware which can efficiently execute multiple IQA algorithms. In this paper, we present the results of a performance analysis designed to examine, compare, and contrast the performances of four popular FR IQA algorithms (MS-SSIM [14] published in 2003 , VIF [15] in 2006, VSNR [66] in 2007, and MAD [7] in 2010) and two NR IQA algorithms (BLIINDS-II [17] and BRISQUE [18] in 2012).

This work draws upon techniques that are standard in the field of performance analysis and software tuning. Listed below, are some similar studies done in the area of multimedia applications. We take a similar approach in this paper.

In Ref. [67], Bhargava *et al.* evaluated the effectiveness of the x86 MMX instructions for DSP and multimedia applications using Intel's Vtune Amplifier XE profiler [68]. Their analysis showed that MMX assembly called within C programs is not an effective strategy to improve performance. They recommend comprehensive hand-coding and restructuring of programs to fully utilize MMX capabilities. They also conclude that parallel processing using the SIMD extensions puts a higher burden on the memory system. Their recommendations have guided developers and compiler writers as well as computer architects over the years.

In Ref. [69], Gordon *et al.* analyzed the performance of model-based video compression for a GPGPU implementation using CUDA, and found that surprisingly, the GPGPU implementation was slower than the native CPU implementation. The authors analyzed data with the help of Intel's Vtune Amplifier XE performance analyzer to gain insight into the specific reasons for the surprising results, and found a high cache miss rate and heavy stalling of the load/store unit of the CPU in the GPGPU version. This led to the discovery that the GPGPU implementation was using the CPU's load store units to access system memory instead of using Direct Memory Access (DMA).

In Ref. [70], Martinez *et al.* analyzed the performance of commercial multimedia workloads on Intel's Pentium 4, focusing on whether these applications make use of the 4-wide out of order superscalar pipeline. They found that the count for instruction per cycle (IPC) is very low, indicating that these applications do not utilize the underlying microarchitecture. They conclude that the low IPC was a result of branch mispredictions and data cache misses, and they recommend static code layout techniques that are aware of cache topology to maximize the utilization of data caches.

Although our examination is limited to six algorithms, we believe that this study is an important first step toward investigating broader performance-related issues in the design and application of IQA algorithms. The findings and recommendations presented in this paper apply broadly to all current-generation Intel IA-32 and Intel 64 based general-purpose computing platforms, whether laptops, servers, or desktops, even though the actual hotspot and bottleneck details might vary. Architectures that are radically different, with hardware accelerators, dedicated image processing cores (such as those found on some tablets and smart phones), and memory shared between GPUs and CPUs (such as AMD's Fusion APUs) are expected to show very different execution characteristics.

The rest of this paper is organized as follows: Section 4.2 provides a brief review of each of the six algorithms, including details of the code implementations and the results of the performance-analysis for each algorithm. Section 4.3 and 4.4 provide the analysis methodology and some architectural concepts. The performance analysis results are presented in Section 4.5. Section 4.6 compares and discusses the differences in performances of all six algorithms. General conclusions of the chapter are provided in Section 4.7.

4.2 Algorithms

This section provides an introduction and a brief overview of all six algorithms. For each algorithm, the *Basic port of the code to C++* subsection presents some techniques that we used when porting the code to C++, for example: computing FFT, calculating matrix's eigenvalues, and code optimizations. The algorithms are ordered here in terms of year of publication.

4.2.1 Multi-Scale Structural Similarity (MS-SSIM)

The multi-scale structural similarity algorithm (MS-SSIM) was developed by Wang *et al.* [14] in 2003. MS-SSIM extends the original SSIM [5] algorithm by applying and combining SSIM for multiple scales, based on the argument that the correct scale depends on the viewing conditions. The SSIM algorithm is derived from a hypothesis that the HVS is highly adapted for extracting structural information. Therefore, measure of structural similarity between the reference and distorted images can be extended to estimate visual quality. The hypothesis also states that one could capture image quality with three aspects of information loss: luminance distortion, contrast distortion, and structural distortion.

A Overview of the algorithm's steps

A block diagram of the MS-SSIM algorithm is shown in Figure 4.1. The algorithm is implemented with five 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, of size 2×2 pixels 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. The luminance, contrast, and structure are

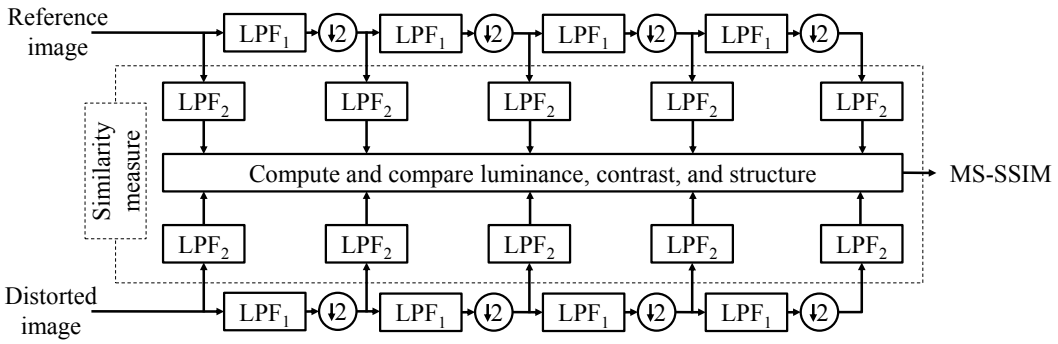


Figure 4.1: Diagram of the MS-SSIM algorithm. LPF_1 is a low-pass filter of size 2×2. $\downarrow 2$ is a downsampling by a factor of two. LPF_2 is a low-pass filter of size 11×11. The reference and distorted images serve as the first scale. The other four scales are obtained by applying LPF_1 and $\downarrow 2$ 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.

computed and compared to yield a different map for each scale. This map is then combined across scales and collapsed to obtain the MS-SSIM quality index.

Specifically, the luminance comparison between two scales is derived from the means of scales' pixel values. The contrast comparison is calculated from the variances, and the structure comparison is computed from both the variances and covariance of the two scales. The single-scale similarity between the original and distorted scales is then calculated from the product of these three comparisons.

Finally, the MS-SSIM index is combined from a weighted geometric mean of contrast and structure comparisons of all five scales and luminance comparison for the last scale. These weights are used to adjust the relative importance of different components.

B Basic port of the code to C++

We implemented MS-SSIM in C++ by porting its Matlab implementation, which is publically available from the authors of the algorithm. The input images are loaded into 1D double arrays, and are accessed as 2D matrices via a thin C++ wrapper class

[71]. The filter LPF_2 was of size 11×11 in Matlab version; in our C++ version, we convolve the image twice with two length 11 1D filters. By taking advantage of this separable convolution, we reduced the number of multiplications for one 512×512 image from $512 \times 512 \times 11 \times 11$ to $512 \times 512 \times 11 \times 2$.

4.2.2 Visual Information Fidelity (VIF)

The visual information fidelity algorithm (VIF) was developed by Sheikh and Bovik [15] in 2006. 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. Specifically, VIF quantifies the information content of the reference image as being the mutual information between the input and output of a modeled HVS channel; this is the information that the brain could ideally extract. Using a similar modeled HVS channel, VIF measures information that the brain would ideally extract from the distorted image. These two information measures are then combined to form the VIF index that correlates with the visual quality.

A Overview of the algorithm's steps

A block diagram of the VIF algorithm is shown in Figure 4.2. First, VIF filters the input images using the Steerable Pyramid [72] to model the image information in wavelet domain. In this step, the Steerable Pyramid is employed with four scales and six orientations, but only eight subbands of interest are used later.

In the second step, the subbands of the reference image are modeled using a Gaussian scale mixtures model. Each subband is modeled as one random field (RF) which is a product of two independent RFs: The first RF is a positive scalar and the second RF is a Gaussian vector with zero-mean and a covariance matrix. For the distorted image, the same idea is applied: signal attenuation by a deterministic

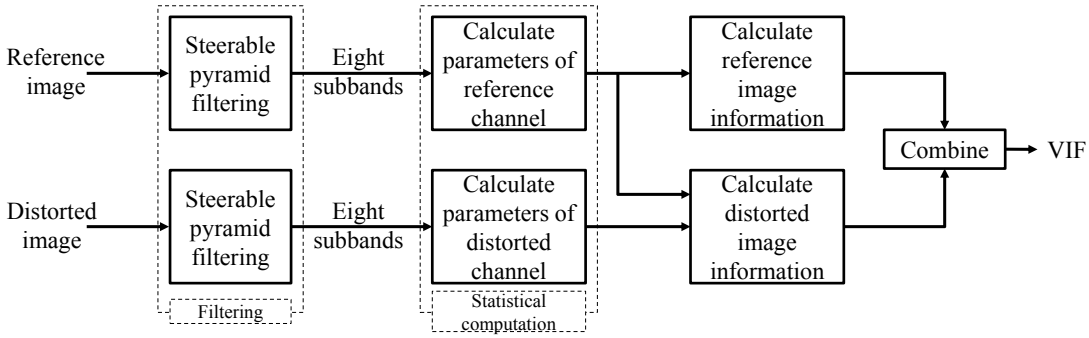


Figure 4.2: The block diagram of our implementation of VIF algorithm. First, two input images are filtered via a six-orientation and four-level Steerable Pyramid which is modified to yield eight subbands for faster computation. The parameters of reference and distorted channels are calculated from the filtered images. Finally, the information of reference and distorted images are calculated and collapsed into a VIF index.

scalar gain field, and a stationary additive zero mean Gaussian noise RF in the same wavelet domain.

In the next step, in order to calculate the reference and distorted image information, VIF also models human visual system noises for two channels as two stationary RFs with zero-means and same covariance, which are uncorrelated multivariate Gaussians with the same dimensionality as the reference image. The image information for each subband is calculated with this noise.

Finally, the image information is summed over eight subbands, and the VIF index is given by the ratio between the distorted image information and reference image information.

B Basic port of the code to C++

We implemented a C++ version of the VIF algorithm by porting the code from its Matlab implementation, which is publically available from the authors of VIF at Ref. [73]. The input images are loaded into 1D double arrays, and are accessed as 2D matrices via a thin C++ wrapper class [71]. The original VIF algorithm uses the Steerable Pyramid toolbox [72] in Matlab; when ported to C++, we used the

Steerable Pyramid C library by the same author. Originally, the Steerable Pyramid library with six orientations and four scales applied the filter 24 times. The VIF algorithm, however, needs only eight subbands; therefore, we modified the library to generate only these eight subbands. This modification reduced the number calls to the filtering function from 24 to 8. To calculate the eigenvalues of covariance matrix, we employed the Newmat C++ matrix library [74].

4.2.3 Visual Signal-to-Noise Ratio (VSNR)

The visual signal-to-noise ratio algorithm (VSNR) was developed by Chandler and Hemami [66] in 2007. VSNR estimates the visual perception of distortions in natural images based on the root-mean-squared (RMS) contrast by computing the contrast thresholds for detection of distortions, the perceived contrast of the distortions, and the degree to which the distortions disrupt global precedence and thereby degrade the image's structure.

A Overview of the algorithm's steps

A block diagram of the VSNR algorithm is shown in Figure 4.3. VSNR is computed via three main steps: A DWT (via filtering with the 9/7 DWT filters), a statistical calculation to compute the perceived contrast, and a final computation for the disruption of global precedence.

In the first step, the reference image is filtered via a five-level 2D DWT to generate 16 subbands. An error image, obtained by subtracting the two input images, is also passed through a 2D DWT transform with the same number of levels. VSNR takes into account the viewing conditions by transforming images into luminance domain via a black-level offset, the pixel-value-to-voltage scaling factor, and a power to the gamma of the display monitor. A set of spatial frequencies, $\mathbf{f} = [f_1, f_2, ..., f_5]$, is used to describe the radial frequency content of visual stimuli expressed in units of

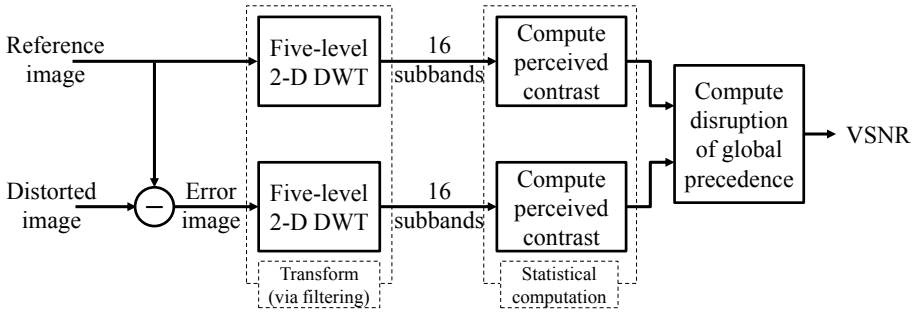


Figure 4.3: The diagram of VSNR algorithm. First two input images are subtracted to generate an error image. The reference and error images are then filtered via a five-scale 2D discrete wavelet transform. Each set of filtered subbands is employed to calculate the perceived contrast. Finally, the VSNR is obtained by computing the disruption of global precedence.

cycles per degree of visual angle (cycles/degree), given by $f_m = 2^{-m} rv \tan(\frac{\pi}{180})$ with $m = 1, 2, ..., 5$; r and v denote the resolution of the display and the viewing distance, respectively. This vector of spatial frequencies is also computed in this first step.

In the next step, the two sets of DWT subbands and the vector of spatial frequencies are employed to assess the detectability of the distortions. The perceived contrast of each image is computed and the contrast thresholds are calculated within each band centered at f_m to determine whether the distortions in the distorted image are visible. The contrasts at each level are calculated from the standard deviations of the three oriented subbands (LH, HL, and HH).

At this point, if the distortions are below the threshold of visual detection for all subbands, the distortions are not visible, and therefore the distorted image is deemed to be of perfect visual quality. If the distortions are suprathreshold, the last step is performed. In this last step, the visual distortion is calculated from a weighted geometric mean of total RMS distortion contrast and a measure of the disruption of global precedence. Finally, the VSNR quality estimate is given as the log of the ratio of the RMS contrast of the reference image and the visual distortion.

B Basic port of the code to C++

The original C++ code of VSNR was obtained from the author's website [71]. The reference image and distorted image are loaded into a 1D float array, and it is accessed as 2D matrix via a thin C++ wrapper class [71]. The five-level 2D DWT decomposition step is implemented based on the lifting scheme (fast DWT [75]) using the default Cohen-Daubechies-Feauveau 9/7 wavelet. In the statistical computation step, in order to obtain the image perceived contrasts, we need to calculate the average luminance of the image, which requires calling power, multiplication, and addition operations for all pixels. We modified this part by using a look-up table technique to obtain a faster implementation that uses those operations only 256 times.

4.2.4 Most Apparent Distortion (MAD)

The most apparent distortion algorithm (MAD) was developed by Larson and Chandler [7] in 2010. MAD uses two strategies to estimate image quality. First, a detection-based strategy is used for near-threshold distortions. In this case, the image is most apparent, and thus the HVS attempts to look past the image and look for the distortions. Second, an appearance-based strategy is used for clearly visible distortions. In this case, the distortions are most apparent, and thus the HVS attempts to look past the distortion and look for the image's subject matter.

A Overview of the algorithm's steps

A block diagram of the MAD algorithm is shown in Figure 4.4. From the input images, the MAD index is computed via two main stages: the detection-based stage and appearance-based stage. Each stage consists of two basic steps: filtering and statistical computations. These two stages yield two quantities indicating the quality of each stage, a detection-based index and an appearance-based index. These two indices are then combined to obtain the overall quality of the distorted image.

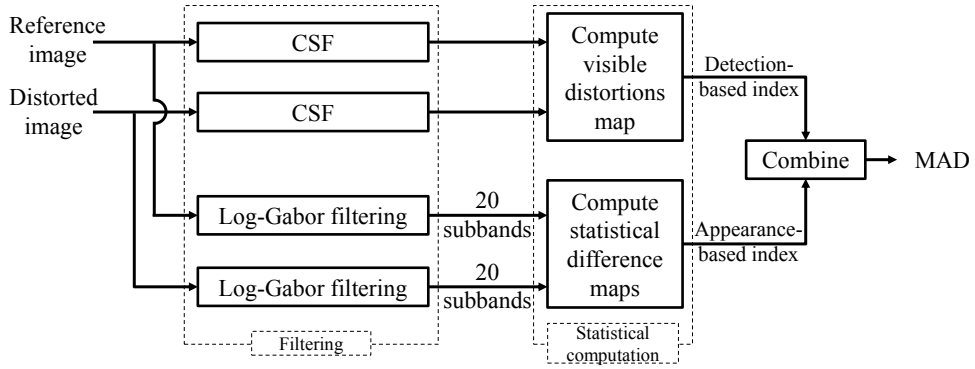


Figure 4.4: The diagram of MAD algorithm. For detection-based stage, reference and distorted images are first filtered using a contrast sensitivity function. The distortion map is then computed from filtered images and collapsed via a MSE measure to obtain a detection-based index. For the appearance-based 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 an appearance-based index. Finally, the MAD index is given by taking a weighted geometric mean of the appearance-based index and detection-based index.

In the detection-based stage, a model of local visual masking which takes into account the contrast sensitivity function (CSF), and luminance and contrast masking, is employed. The two input images are first passed through a contrast sensitivity function filter [76] in the frequency domain using FFT and inverse FFT. Two local contrast maps of the two filtered images are computed in an overlapping block-based fashion with blocks of size 16×16 and a 12-pixel overlap between neighboring blocks. From these two local contrast maps, the visible distortion map is calculated locally for the regions that are deemed to be visibly distorted. This map is then collapsed via MSE measure to obtain a detection-based index.

In the appearance-based stage, the appearance-based difference map is computed from the difference of low-level statistics (mean, variance, skewness, and kurtosis) for all local blocks of the log-Gabor filtered images (subbands). First, the input images are filtered via a log-Gabor filter bank with five scales and four orientations to obtain 20 subbands. The steps to compute those subbands include computing the image's FFT, a product of this image's FFT with a set of 2-D frequency responses,

and an inverse FFT. Each pair of two sets of 20 reference and distorted subbands is then divided into small blocks, each of size 16×16 (and 12 pixels of overlap between neighboring blocks). The standard deviation, skewness, and kurtosis of each block is calculated and compared to generate the statistical difference maps for each scale/orientation. The 20 statistical difference maps are then combined via a weighted mean across scales and collapsed via a 2-norm to obtain the appearance-based index.

Finally, the overall quality of the distorted image is computed by taking a weighted geometric mean of the detection-based index and the appearance-based index, where the weight is chosen based on the detection-based index.

B Basic port of the code to C++

We implemented a C++ version of the MAD code by porting from its Matlab version, which is publically available to download from the authors of MAD at Ref. [77]. The input images are loaded into 1D double arrays, and are accessed as 2D matrices via a thin C++ wrapper class [71]. In the detection-based stage, the images are transformed to the luminance domain by using a look-up table. The Ooura FFT library [78] is employed for calculating FFT and inverse FFT. This Ooura library is also used in the log-Gabor decomposition in the appearance-based stage. The log-Gabor filter was implemented based on Peter Kovesi's work [79]. The statistical difference maps are calculated using integral images for higher orders; details of these modifications can be found in Ref. [63].

4.2.5 Blind Image Integrity Notator using DCT Statistics (BLIINDS-II)

The BLind Image Integrity Notator algorithm (BLIINDS-II) [17] was developed by Saad *et al.* [17] in 2012. BLIINDS-II is a no-reference IQA algorithm using DCT statistics. It inherits the idea from the BLIINDS-I algorithm [80] that the data histograms of specific-domain-transformed natural images share the same shape. One

such domain is local DCT, which utilizes the generalized natural scene statistic based model. The model's parameters are transformed into features, the generalized probabilistic model is then applied to these features to predict the visual quality of the input distorted image. The BLIINDS-II algorithm is trained using features derived directly from a generalized parametric statistical model of natural image DCT coefficients against various perceptual levels of image distortion.

A Overview of the algorithm's steps

A block diagram of the BLIINDS-II algorithm is shown in Figure 4.5. The algorithm is a multi-scale algorithm, similar to the MS-SSIM algorithm. The input image, considered as the first scale, is low-pass filtered and downsampled twice to obtain two more scales. Each scale is then passed through the same procedure: first a block-based DCT, a generalized Gaussian modeling, and then extraction of features. Finally, all features from the three scales are combined to create a BLIINDS index indicating the quality of the distorted image.

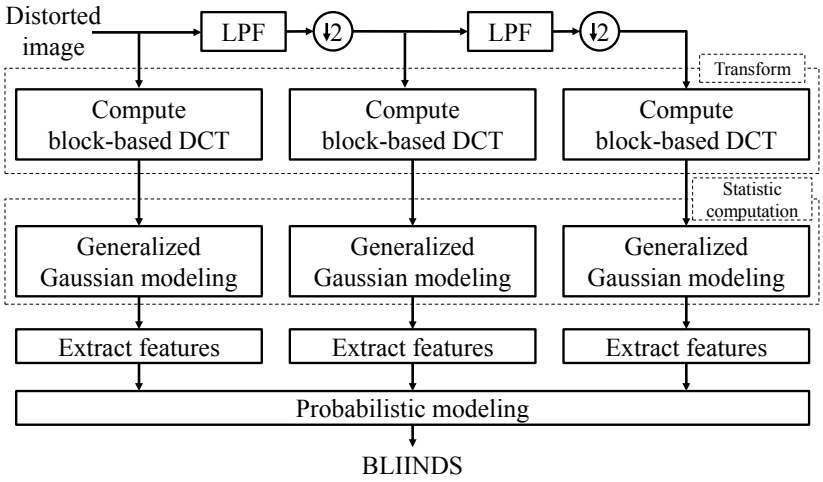


Figure 4.5: BLIINDS-II algorithm. LPF is a low-pass filter of size 3×3. $\downarrow 2$ is a downsampling by a factor of two. The input image serves as the first scale. Two more scales are obtained by the low-pass filter and downsampling. Each of three scales is divided into blocks of size 5×5 so that the DCT can be applied for each block. The transformed coefficients are then modeled using generalized Gaussian to extract features. Finally, a probabilistic modeling is applied to yield the BLIINDS index.

In the first stage of the algorithm, all three scales of the input image are divided into small blocks of size 5×5 with two pixels of overlap between neighboring blocks. Each block is then subjected to a 2D DCT.

In the second stage, a generalized Gaussian model is applied to each block of 25 DCT coefficients, as well as for specific partitions within each block. Here, the DCT histogram is fitted with a multivariate Gaussian model to extract parameters via a line search procedure.

Next, in the feature-extraction stage, eight features are derived from the model parameters of all blocks. All parameters are pooled in two ways: first, a percentile pooling that takes the average of the lowest (or highest) 10^{th} percentile; and second, the ordinary sample mean, which is the average of the 100^{th} percentile. Taking both the averages of 10^{th} and 100^{th} percentile allow the algorithm to determine if the distortions are uniform over space.

The final step employs a probabilistic modeling, where the BLIINDS index (the score for the image quality) is computed by using a simple Bayesian model from a trained parameters set. This model is applied to the 24 features extracted from three scales.

B Basic port of the code to C++

The Matlab code was obtained from the first author of BLIINDS-II via email in early 2012, and it has been confirmed to be the latest version available. We ported this Matlab code to C++. As shown in Figure 4.5, the distorted image is loaded into a 1D double array, and it is accessed as 2D matrix via a thin C++ wrapper class [71]. The low-pass filter step was optimized by using a separable convolution; we convolve the image with two 1D kernels separately, each of size 3×1. The 2D DCT for blocks of size 5×5 was also optimized by using a look-up table for 25 values of the cosine function. In the generalized Gaussian modeling step, some functions are called

repeatedly for each small block. In our C++ code, these functions are pre-calculated out of the main loop. This step includes a fitting process of the DCT data histogram to the model as a line search procedure over 9970 values in the range of $[0 - 10]$. In the feature-extraction step, the algorithm needs to sort values to determine 10^{th} and 100^{th} percentiles. This sorting procedure is performed via a quick sort algorithm.

4.2.6 Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE)

The Blind/Referenceless Image Spatial Quality Evaluator [18] algorithm (BRISQUE) was developed by Mittal *et al.* in 2012. In contrast to BLIINDS-II, which operates in the DCT domain, BRISQUE claims that in the spatial domain, natural images share the same properties: The mean subtracted contrast normalized (MSCN) coefficients of the image and the pairwise products of neighboring MSCN coefficients have the histograms of Gaussian-like appearances. These histograms distribute vary as a function of distortion: The histograms of natural images have the bell shape, while the histograms of distorted images could be of any shape, e.g. Laplacian distributions for blurred images and unusual tails for white-noise images. The generalized Gaussian distribution (GGD) and asymmetric generalized Gaussian distribution (AGGD) models are used for quantifying the features from shape, variance, left variance, and right variance of a histogram. From these features, the final quality score is given via a trained mapping by using a support vector machine (SVM) regressor.

A Overview of the algorithm's steps

A block diagram of the BRISQUE algorithm is shown in Figure 4.6. Similar to previous approaches, this algorithm utilizes two scales by first downsampling the input image to obtain the second scale. The following three stages are then applied.

In the first stage, the locally normalized luminances are computed via local mean subtraction and divisive normalization. This step mainly contains a filtering process

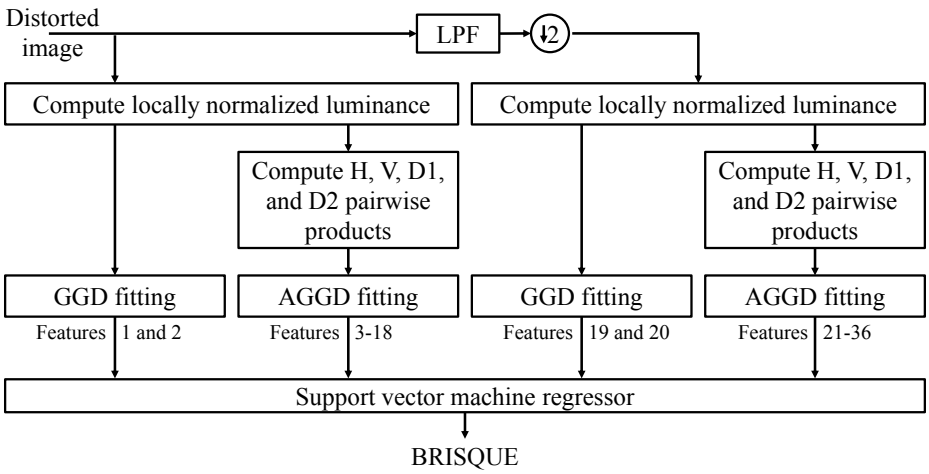


Figure 4.6: BRISQUE algorithm. LPF is a low-pass filter and $\downarrow 2$ is a downsampling by a factor of two; they are utilized to obtain a smaller scale of the input image, which serves as the first scale. Each of two scales is employed to compute locally normalized luminance via local mean subtraction and divisive normalization. The luminances and their pairwise products of neighboring MSCN coefficients along four orientations (H, V, D1, and D2) are fitted with generalized Gaussian distribution (GGD) and asymmetric generalized Gaussian distribution (AGGD) models to extract 36 features. Finally, the support vector machine (SVM) regressor is applied to yield the BRISQUE quality index.

of the image and its divisive normalization by a size 7×7 2D circularly-symmetric weighted Gaussian filter. The statistical relationships between neighboring pixels are also modeled in this stage. Specifically, four orientations, horizontal (H), vertical (V), main-diagonal (D1) and secondary-diagonal (D2) of MSCN coefficients are computed and multiplied point-by-point with the MSCN coefficients.

In the second stage, the generalized Gaussian distribution model is applied to calculate the shape and variance (feature 1 and feature 2 for first scale, feature 19 and 20 for second scale) from histograms of MSCN coefficients. The asymmetric generalized Gaussian distribution model is employed to calculate the shape, mean, left variance, and right variance from histograms of each of the four pairwise products (each scale has 16 features, four features for four products).

In the final stage, all 36 features (two scales, each scale has two features for GGD fitting and four features for AGGD fitting of each orientation) are collapsed into one

index for the score of image quality via a regression module. Here, the BRISQUE algorithm is trained to use the support vector machine regressor to map the 36 features to a quality score.

B Basic port of the code to C++

We implemented a C++ version of the BRISQUE code by porting from its Matlab version, which is publically available from the authors' website [1]. The input image is loaded into 1D double array, and is accessed as a 2D matrix via a thin C++ wrapper class [71]. In the first stage, the Gaussian filter was optimized by using a separable convolution; we convolve the image with two 1D kernels separately, each of size 7×1. In the second stage, the *GGD fitting* and *AGGD fitting* functions contain a fitting process of data histograms to the model as a line search procedure over 9801 gamma values. In our program, the gamma values were pre-calculated and stored for faster speed. In the last stage, the SVM regression module, we used the same the LIBSVM executable [81] as in the Matlab version.

4.3 Analysis Methodology

4.3.1 Algorithms and profiler

We define an experimental framework for performance analysis designed to examine, compare, and contrast the performances of the six algorithms on a typical general-purpose computing platform. To provide a common codebase, we implemented five algorithms in C++ based on the original Matlab code provided by the authors. The C++ implementation for VSNR was directly available from its author and further optimized as described in Section B. An initial code-level profiling was performed in both Matlab and Intel's Vtune Amplifier XE [68] to identify and correct obvious inefficiencies in the baseline implementations.

For performance profiling, we first used Intel's Vtune Amplifier XE to identify segments of the program where most of the execution time was spent. Such sections of the program are called *hotspots*, and they should be targeted for improving the computation performance. After the top hotspot functions were identified, we conducted a microarchitectural analysis to observe the interactions between the hotspot functions and the processor and other microarchitectural sub-systems. Our specific goal was to find architectural bottlenecks and map them to specific execution blocks of the respective algorithm.

4.3.2 Sample images

To get the results shown in this paper, we executed 30 trials of each of the six algorithms on a set of 42 images. These images were taken from the CSIQ database [7], including seven different original images (*I1* to *I7*), each with three different distortion types, additive Gaussian white noise (AWGN), Gaussian blurring, and JPEG compression, with two levels of distortion each, level 1 for low-distorted images

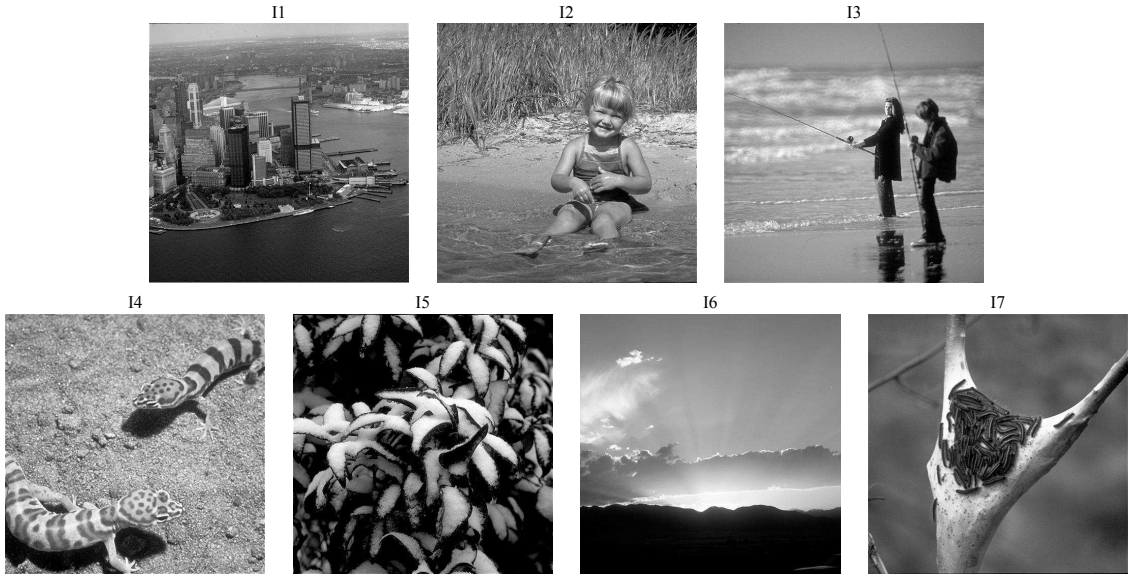


Figure 4.7: Seven original images span a variety of commonplace subject matters in five categories, *animals*, *landscapes*, *people*, *plants*, and *urban*.

(*AWGN1*, *BLUR1*, and *JPEG1*) and level 5 for highly distorted images (*AWGN5*, *BLUR5*, and *JPEG5*). The original images span a variety of commonplace subject matters in five categories, *animals*, *landscapes*, *people*, *plants*, and *urban*. They are shown in Figure 4.7. Nine of the highly distorted versions (AWGN5, BLUR5, and JPEG5) of $I2$, $I3$, and $I7$ are shown in Figure 4.8. A summary of the experiment images is provided in Table 4.1.

Table 4.1: A summary of the experiment images.

Number of images varying in content	7
Types of distortions	AWGN, Blurring, JPEG compression
Levels of distortions	2 (Level 1, Level 5)
Total subject images	42
Image size	512×512

4.3.3 Analysis platform

For this study, we use the 2^{nd}-generation Intel Core i5-2430M processor clocked at 2.4GHz and a system memory (RAM) of 4 GB. The microarchitecture was Sandy Bridge. Further details about the caches and memory hierarchy are provided in Table 4.2.

Figure 4.8: Some distorted versions (AWGN5, BLUR5, and JPEG5) of $I2$, $I3$, and $I7$. Original images are shown in Figure 4.7.

Table 4.2: Processor and system hardware specifications for the experiment

Processor	Intel core i5-2430M
Frequency	2.4 Ghz
Microarchitecture	Sandy Bridge
System memory (RAM)	4 GB
L1 instruction cache	32 KB per core
L1 data cache	32 KB per core
L2 cache (unified instruction and data)	256 KB per core
L3 cache (unified instruction and data)	3 MB shared

4.4 Architectural Concepts

This section provides a brief explanation of the architectural concepts which apply to our results presented in Section 4.5.

4.4.1 Virtual Memory

Virtual memory is an abstraction provided by operating systems to the programmer, so that the programmers do not need to worry about size constraints and the layout of blocks in the actual physical system memory. Without virtual addressing, the programmer would have to explicitly manage physical memory resources shared among multiple programs and multiple users. For example, a program would have to explicitly load and unload sections of the code that correspond to different phases of the program execution, because loading the entire program and the corresponding working data set would overwhelm the physical memory, and consequently, other concurrent programs would starve for memory. Thus, all addresses generated by the program, are virtual addresses, spanning the entire 32-bit or 64-bit address space, and need to be translated to the actual physical locations held by the data. A mapping table called the *page table* handles this process. Pages are typically 4 KB in size, and they facilitate the virtual to physical mapping of addresses, because a single entry in the mapping is required for all the addresses in the 4 KB range instead of individual mappings for each address. The page table is stored in the main system memory, which takes hundreds of clock cycles to access. To be able to translate faster so as to be more compatible with the speed of the processor, a cache (faster and smaller memory) is used to store those translations that are currently in use. This cache is called the Translation Lookaside Buffer (TLB), and there is usually a separate TLB for the program instructions (ITLB) and data (DTLB). As the TLB stores only a

subset of the page table, there are times when the mapping is not found in the TLB. This is called a *TLB miss*. On a TLB miss, the page table entry has to be brought into the TLB from the main memory. This transfer can take hundreds of clock cycles, and thus, frequent TLB misses can lead to a performance loss.

4.4.2 CPU Caches

The main system memory (RAM) is fairly slow and takes hundreds of clock cycles to deliver operands to the processor. Therefore, to deliver operands to the processor every clock cycle, a fast and small cache memory is placed between the processor and system memory. The cache memory closest to the processor is the Level-1 (L1) cache; it is the fastest and smallest cache in the hierarchy. Usually, the L1 is split into an instruction cache (I-cache) to store the program's instructions and the data cache (D-cache) for the operands. The next level of the cache memory is bigger and slower, and is called the Level-2 (L2) cache. The Level-3 (L3) cache is the last level cache (LLC) in the hardware architecture for our analysis platform. More details about cache memories and memory hierarchy can be found in Ref. [82].

4.4.3 Address Calculation

The Load Effective Address (LEA) instruction is an assembly instruction that calculates the effective address of memory operand and places it in a CPU register. Modern superscalar processors have multiple *dispatch ports* to dispatch instructions to execution units. LEA instructions with two operands can be dispatched through Port 1 and Port 5 on the Intel Sandy Bridge microarchitecture. LEA instructions with three operands have a longer latency of three clock cycles and can only be dispatched through Port 1. Thus, a large number of back to back three operand LEA instruction will cause a performance bottleneck. There are some other special situations in which the LEA instructions can take three clock cycles to execute even with two operands.

Details about these cases can be found in Ref. [83]-p3-30 to 3-32.

4.4.4 Speculative Loads

Modern processors internally execute instructions out of the correct program, in order to use the hardware more efficiently, and then commit or retire the instructions in the correct program order. In this process, instructions that load data from the main memory to the CPU are often given a higher priority because the data that they load is to be used in subsequent instructions. The speculative load has to be compared with any pending instructions that might be storing data in the same address. Because this operation could require several comparisons, instead of comparing the entire 32-bit or 64-bit address, typically, only the last 12 bits are compared. If these addresses are 4 KB or multiple of 4096 bytes apart, a false hazard is detected. This is called *4K aliasing* [83] (p11-30), and the speculative load has to be reissued in its correct program order. This reduces the throughput of instructions, and creates a performance slowdown. These false positives from 12-bit comparisons can also cause *machine clears*.

4.5 Results

In this section, we provide results of the microarchitectural analysis for each of the six algorithms. The algorithms' results are presented here in terms of alphabetical order for full-reference algorithms, followed by BLIINDS-II and BRISQUE.

4.5.1 Performance Analysis of MAD

To perform hotspot analysis, the MAD algorithm was applied for 30 iterations for each image. The results of the hotspot analysis are provided in Table 4.3. These results show that the average execution time for all 42 images for MAD is approximately 41.97 seconds. The average execution time for top hotspots functions/blocks is provided in second column with the standard deviation. The hotspot functions are listed in descending order, with the function consuming the highest execution time listed first. The table shows the average execution time for individual functions calculated across the 42 images. We also plot the individual execution time for all 42 images in Figure 4.9.

Table 4.3: Analysis results of MAD. Average execution time for top hotspots functions/blocks of 42 images is presented with the standard deviation. The total execution for each hotspot is calculated from the average. The hardware bottlenecks are also provided.

Function/block	**Execution time (s)**	**Total execution (%)**	**Hardware bottlenecks**
Gabor Convolution	10.19 ± 0.31	24.29%	DTLB overhead
ASDMC	4.06 ± 0.25	9.67%	DTLB overhead
DVDMC	2.83 ± 0.10	7.13%	LLC misses
FFT	2.91 ± 0.19	6.52%	L1D replacement, L2D replacement, LLC misses
Other	21.98 ± 0.44	52.38%	N/A
All	41.97 ± 0.48	100%	N/A

As shown in Table 4.3, the top hotspot functions contribute approximately 48.62% of the total execution time, and the other functions add up to the remaining 52.38%. MAD has minimal variation of total execution time across different image content

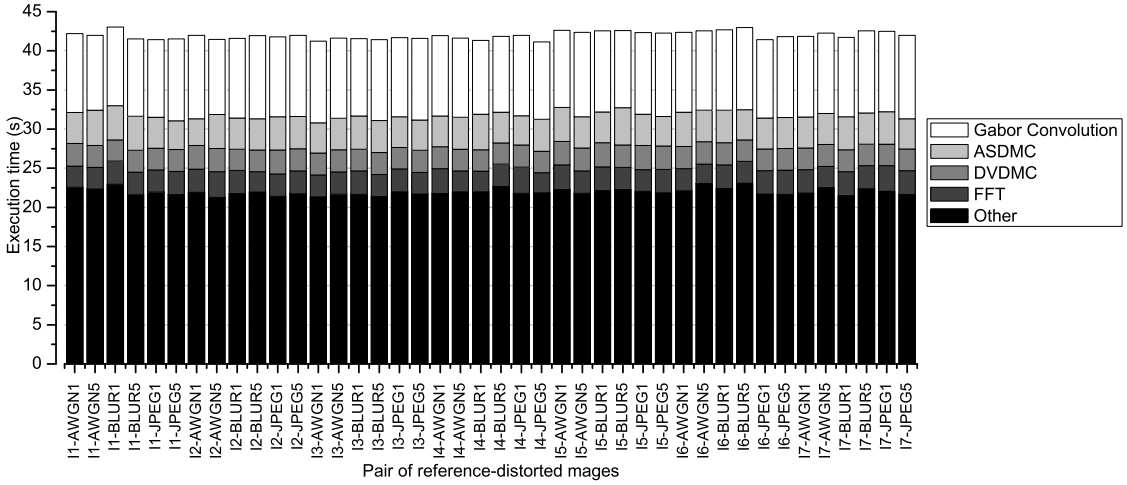


Figure 4.9: The execution time of MAD for each pair of reference and distorted images. The contributions of hotspot functions are stacked together to form the total execution time.

as well as different distortion types. Thus, any optimizations for MAD can be made without any specific consideration of image content or distortion.

Also shown in Table 4.3 are hardware bottlenecks (for each hotspot function) identified via the microarchitechtural analysis. The following subsections describe details of the observed results for each of the hotspot functions, and include explanations of underlying computer architecture concepts whenever necessary.

A Gabor Convolution

The *Gabor Convolution* function is used to decompose input images with five scales and four orientations to yield a set of 20 subbands. This function is called by the Log-Gabor filtering block (as shown in Figure 4.10c). This block includes three main functions: *FFT*, *Gabor Convolution*, and *inverse FFT*. The function takes around 10.19 seconds, which is 24.29% of the total execution time.

By investigating the microarchitectural resources utilized by the *Gabor Convolution* function, we find that the performance bottleneck is in the memory sub-system, specifically with DTLB overheads. The results show that the *Gabor Convolution* function has a high DTLB overhead. The function generates a set of 40 filtered im-

ages (subbands), which is $40 \times 512 \times 512 \times 8$ bytes (80 MB). With a typical page size of 4 KB, this set spans over 20 thousand pages, with each page requiring its own entry in the TLB for translation. The hardware architecture of our analysis platform has 64 entries in the Level-1 DTLB and 512 entries in the Level-2 DTLB. Thus, the 20 thousand translations required for the 40 subbands cause a large number of misses, each of which takes hundreds of clock cycles to service. One technique to overcome the problem of TLB overhead is to use *superpages*. Details on using superpages along with other techniques to reduce penalties due to TLB overhead are discussed in Section 4.5.

B Appearance-Based Statistical Difference Map Computation

The *Appearance-Based Statistical Difference Map Computation* (ASDMC) function calculates the statistical difference map using variance, skewness, and kurtosis of the subbands. This function takes two sets of 20 subbands as the inputs. For each pair of subbands of the same scale and orientation, it calculates the local standard deviation, skewness, and kurtosis difference maps. The combined statistical difference map is then collapsed into an appearance-based index. This function takes 4.06 seconds, which is 9.67% of the total execution time.

The ASDMC function, similar to the *Gabor Convolution* function, suffers from bottlenecks in the memory sub-system and faces penalties due to DTLB overhead. As with the *Gabor Convolution*, the performance penalty is also due to the traversal of a large memory space, as the algorithm calculates statistics (standard deviation, skewness, and kurtosis) for all the 40 subbands.

C Detection-Based Visible Distortion Map Computation

The *Detection-Based Visible Distortion Map Computation* (DVDMC) function calculates the detection-based map and then collapses it into a detection-based index. The

detection-based map is calculated by finding the luminance of the reference and the distorted image, calculating the luminance error image, and then applying a contrast sensitivity function via the FFT to the reference and the error image. The function takes 2.83 seconds, which is 7.13% of the total execution time.

The DVDMC function suffers from LLC misses, which face a high performance penalty because they are serviced from the main memory. DVDMC processes input images, luminance images, and matrices in the Fourier domain. This huge data set cannot fit in the caches, and consequently, the function suffers from a large number of LLC misses. The data has to be fetched from the main memory, which causes a slowdown.

D Fast Fourier Transform

The *Fast Fourier Transform* (FFT) function converts the reference and distorted images into the Fourier domain. The FFT function takes 2.91 seconds, which is 6.52% of the total execution time.

From the microarchitectural analysis, we find that there are misses at the levels of cache, which requires that we investigate the working data set for the FFT function. The output of the FFT operation is a 512×512 complex matrix, including both real and imaginary parts. This matrix uses the data type double to represent floating-point numbers and hence each pixel is 8 bytes. The total data set for the function is $2 \times 512 \times 512 \times 8$ bytes (4 MB). As the total data set is larger than data caches' sizes, data needs to be fetched from the main memory, causing performance degradation.

Whether caches are used effectively depends on spatial and temporal locality. Spatial locality involves accessing memory addresses that are close to each other, and temporal locality involves repeated accesses to the same data. A higher number of misses for L1D and L2D caches suggests that the access pattern lacks locality of reference. Techniques such as cache blocking or loop tiling can be used to improve

locality of reference, thereby improving performance by reducing misses. A further discussion of cache blocking and other techniques to improve cache performance are discussed in Section 4.6.1.

E Mapping algorithmic blocks to hotspots/hardware bottlenecks

Figure 4.10 shows the mapping between hotspots/hardware bottlenecks and the algorithmic blocks of MAD algorithm. The *log-Gabor filtering* block includes three main blocks: FFT (*Fast Fourier Transform* function), *Five-scale and four-orientation decomposition* (*Gabor Convolution* function), and *IFFT* (inverse FFT function) blocks. This block consumes approximately 71% of the execution time. It suffers from L1D and L2D replacements, LLC hits, and DTLB overhead. The *CSF* block endures L1D and L2D replacements, and LLC hits, and consumes approximately 7.5% of the execution time. The *compute visible distortion map* block (including DVDMC function as its main function) faces the LLC misses and consumes approximately 10.5% of the execution time. The *compute statistical difference map* block consumes approximately 11% and experiences DTLB overhead and LLC misses.

4.5.2 Performance Analysis of MS-SSIM

To perform hotspot analysis, the MS-SSIM algorithm was applied for 30 iterations for each image. The results of hotspot analysis are provided in Table 4.4. These results show that the average execution time for all 42 images is approximately 2.51 seconds. The algorithm spends approximately 56.95% of the total time in the top hotspot function, and approximately 74.15% or 80.73% of the total time in the top two or three hotspot functions, while the remaining functions require only 19.27% of the execution time.

We also plot the individual execution time for all 42 images in Figure 4.11. The figure shows that MS-SSIM has minimal variation of total execution time across

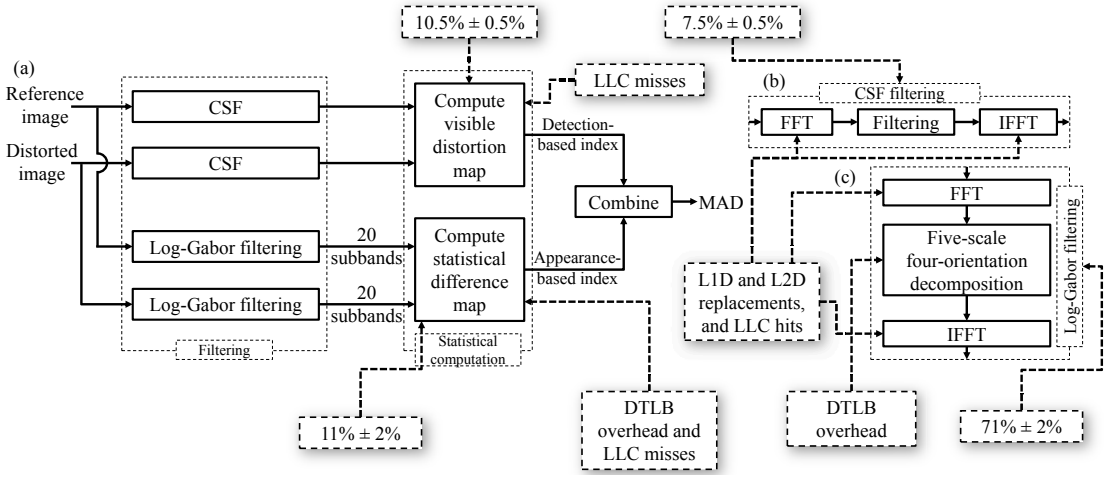


Figure 4.10: Mapping between hotspots/hardware bottlenecks and the algorithmic blocks for MAD. Figure (b) is the detail of CSF block, and figure (c) is the detail of Log-Gabor filtering block. Log-Gabor block suffers from L1D and L2D replacements, LLC misses, and DTLB overhead. FFT and IFFT blocks suffer from L1D and L2D replacement, and LLC misses, the *Appearance-Based Statistical Difference Map Computation* and *Detection-Based Visible Distortion Map Computation* functions are a part of *statistical computation* block, which suffer from DTLB overhead and LLC misses.

different image content as well as different distortion types. Thus, similar to MAD, optimizations can be made without specific consideration of the image content or distortion.

Table 4.4: Analysis results of MS-SSIM. Average execution time for top hotspots functions/blocks of 42 images is presented with the standard deviation. The total execution for each hotspot is calculated from the average. The hardware bottlenecks are also provided.

Function/block	**Execution time (s)**	**Total execution (%)**	**Hardware bottlenecks**
Low Pass Filter 11	1.43 ± 0.10	56.95%	L1D, L2D replacement
Similarity Measures	0.43 ± 0.06	17.20%	L1D, L2D replacement, LLC miss. Assists
LCS Average	0.17 ± 0.03	6.58%	L2D replacement, LLC miss, DTLB overhead. Floating-point divide
Other	0.48 ± 0.09	19.27%	N/A
All	2.51 ± 0.03	100%	N/A

Also shown in Table 4.4 are hardware bottlenecks (for each hotspot function) identified via the microarchitechtural analysis. The following subsections describe details of the observed results for each of the hotspot functions, and include explanations of underlying architectural concepts whenever necessary.

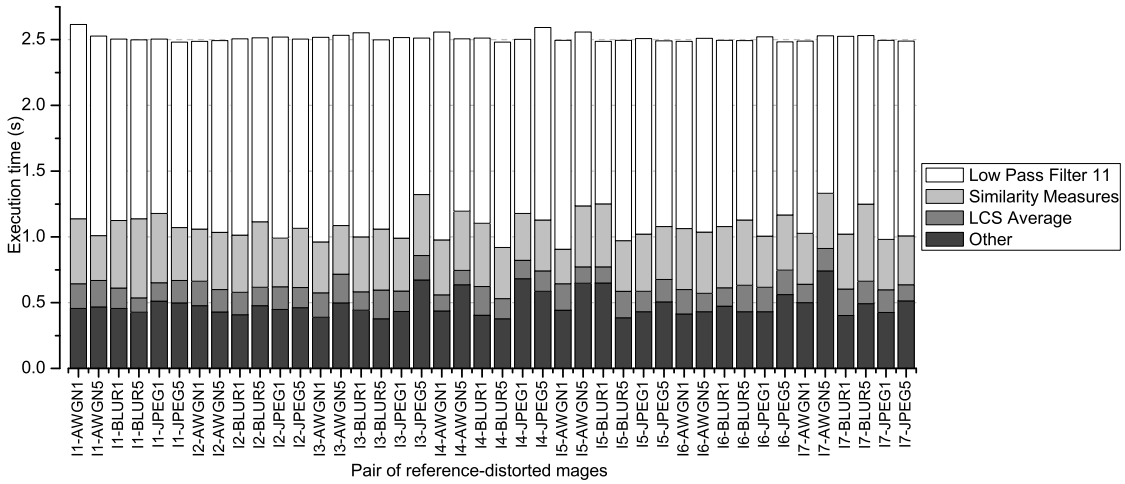


Figure 4.11: The execution time of MS-SSIM for each pair of reference and distorted images. The contributions of hotspot functions are stacked together to form the total execution time.

A Low Pass Filter 11

The *Low Pass Filter 11* function is an implementation of an 11×11 Gaussian low-pass filter over the reference image, the distorted image, and their five different scaled versions. The average execution time is approximately 1.43 seconds, which is 56.95% of the total execution time.

The results show that the associated hardware bottlenecks are in the memory sub-system because of the penalties due to L1D and L2D replacements, similar to the functions in MAD. The filter is initially applied over both the reference and distorted images, and then the filtered images are downsampled to calculate the next scale, after which these downsampled images are again filtered. This process is repeated to achieve total five different scales. We can infer that the filter function demonstrates the temporal locality: the filtered image is used to further downsample the images and filter them again. Thus, once the block of data is brought into the cache, it is accessed repeatedly before it is evicted. Although the function has temporal locality, there are replacements in L1D and L2D cache due to the large working data set which is approximately $4 \times 512 \times 512 \times 8$ bytes (8 MB). This large working data set leads

to cache replacements, which cause a performance penalty and thus higher execution time.

B Similarity Measures

The *Similarity Measures* function calculates the SSIM index for all five scales by using the luminance, contrast and structure maps. The average execution time for *Similarity Measures* function is approximately 0.43 seconds, which is 17.20% of the total execution time.

The microarchitectural analysis indicates that there is a penalty due to L1D replacements, L2D replacements, and LLC misses due to the large working data set. Along with bottlenecks in the memory sub-system, the function also suffers from hardware bottlenecks due to assists [84]. There are instructions in the block which cannot be directly executed by the processor. These instructions are converted into a stream of *microcode* that can be executed by the processor. Each such instruction can generate microcode which can be hundreds of instructions long. Therefore, executing these functions creates a high latency.

Calculation of the SSIM index for each scale requires floating-point operations. Although the IEEE 754 standard is used for implementation of floating-point operations, if the floating-point numbers are very small (denormals), they cannot be directly executed by the processor. Thus, these floating-point operations are converted in a stream of microcode and then inserted in the pipeline of the processor. This microcode is hundreds of instructions long, causing performance degradation. One solution to this problem is to write assembly code directly to set denormals to zero.

C LCS Average

The *LCS Average* function calculates the luminance and contrast maps. The function takes approximately 0.17 seconds, which is 6.58% of the total execution time.

Observing the microarchitectural analysis results, we find that the bottlenecks fall into two categories: memory sub-system and core sub-system. The bottlenecks within the memory sub-system are due to L2D replacements, LLC misses, and DTLB overhead.

The bottleneck within the core sub-system is the floating-point divide unit. The calculation of luminance and contrast requires floating-point operations, which are inherently long-latency operations. Because of the continuous feed of floating-point operations for every pixel and a total of 10 images, the floating-point divide unit is overwhelmed. One solution to improve the performance is to use single-precision floating-point instead of double precision.

D Mapping algorithmic blocks to hotspots/hardware bottlenecks

Figure 4.12 shows the mapping between hotspots/hardware bottlenecks and the algorithmic blocks of MS-SSIM algorithm. The hotspot functions belong to two blocks: LPF_2 and the block corresponding to *computation and comparison of luminance, contrast and structure*. The *Low Pass Filter 11* function belongs to the LPF_2 block (55.5% of the execution time) with L1D and L2D replacements, and LLC misses as the performance bottlenecks. The remaining functions, the *Similarity Measures* and the *LCS Average* function, belong to the block corresponding to *computation and comparison of luminance, contrast and structure*, which suffers from the memory bottlenecks along with a core bottleneck of floating-point divide unit.

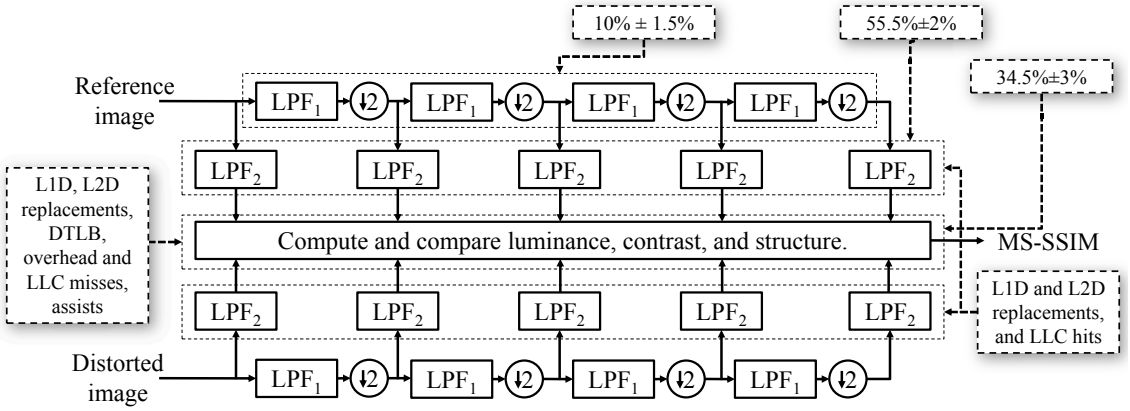


Figure 4.12: Mapping between hotspots/hardware bottlenecks and the algorithmic blocks for MS-SSIM. The LPF_2 block suffers from L1D and L2D replacements, and LLC misses, *Similarity Measures* and *LCS Average* functions, belong to the block corresponding to *computation and comparison of luminance, contrast and structure*, which suffers from L1D replacement, L2D replacement, DTLB overhead, assists, and floating-point divide unit.

4.5.3 Performance Analysis of VIF

To perform hotspot analysis, the VIF algorithm was applied for 30 iterations for each image. The results of the hotspot analysis are shown in Table 4.5. These results show that the average execution time for all 42 images is approximately 12.12 seconds. We also plot the individual execution time for all 42 images in Figure 4.13. The results show that the top two hotspot functions contribute approximately 53.89% of the total execution time. There is minimal variation of the total execution time across different image content as well as different distortion types for VIF, similar to MAD and MS-SSIM. Therefore, neither the image content or distortion needs to be considered when making the optimizations for VIF.

A Pyramid Filtering

The *Pyramid Filtering* function is the main function of the Steerable Pyramid. This function is employed to compute the reference image's subbands, which are used later to compute parameters of the reference channel. It consumes approximately 3.69 seconds, which is 30.47% of the total execution time. The results of the analysis

Table 4.5: Analysis results of VIF. Average execution time for top hotspots functions/blocks of 42 images is presented with the standard deviation. The total execution for each hotspot is calculated from the average. The hardware bottlenecks are also provided.

Function/block	Execution time (s)	Total execution (%)	Hardware bottlenecks
Pyramid Filtering	3.69 ± 0.10	30.47%	Slow LEA stalls
Pyramid Step Filtering	2.84 ± 0.08	23.42%	L1D replacement. Slow LEA stalls
Parameters Calculation	2.15 ± 0.08	17.71%	L1D, L2D replacement, LLC hit, LLC miss, DTLB overhead.
Other	3.44 ± 0.14	28.41%	N/A
All	12.12 ± 0.07	100%	N/A

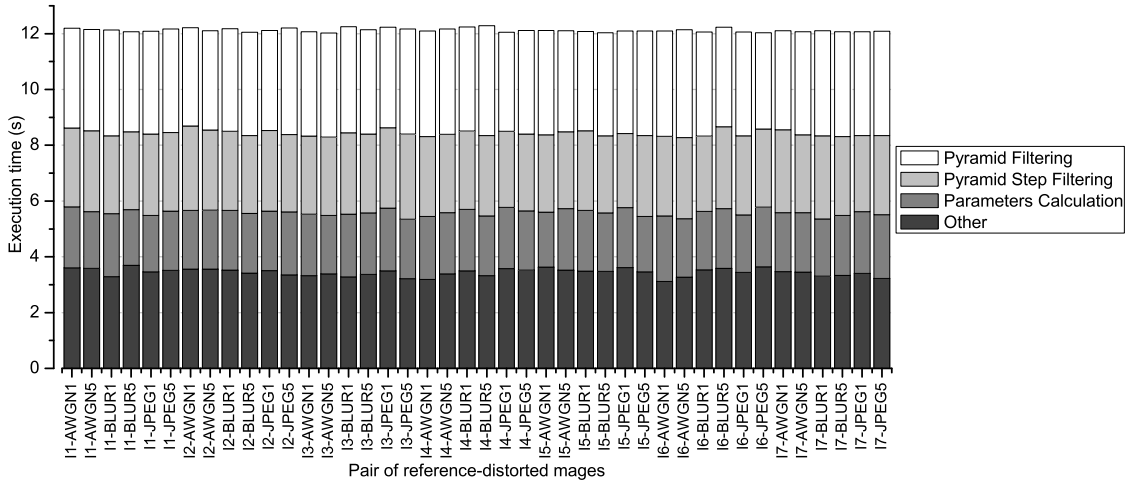


Figure 4.13: The execution time of VIF for each pair of reference and distorted images. The contributions of hotspot functions are stacked together to form the total execution time.

also show that *Pyramid Filtering* has a bottleneck in the core sub-system with stalls due to LEA instructions.

B Pyramid Step Filtering

The *Pyramid Step Filtering* function is similar to *Pyramid Filtering* function. This function takes the subsampling according to the START, STEP, and STOP parameters. This function is employed to compute the distorted image's subbands, which are used later to compute parameters of the channel. It takes approximately 2.84 seconds to execute, which is 23.42% of the total execution time.

The microarachitectural analysis for the blocks shows that the *Pyramid Step Fil-*

tering suffers from bottlenecks in the memory sub-system specifically due to L1D replacements. The results of the analysis also show a bottleneck in the core sub-system with stalls due to LEA instructions.

C Parameters Calculation

The *Parameters Calculation* function computes the parameters of channels from the filtered subbands. This function takes approximately 2.15 seconds to execute, which is 17.71% of the total execution time. The function also suffers from memory bottlenecks caused by LLC hits and LLC misses. The processor has to fetch data from the LLC or the RAM. The penalty for accessing the LLC is approximately 26-31 clock cycles while the access to main memory is hundreds of clock cycles. Consequently, *Parameters Calculation* function is one of the top hotspots.

D Mapping algorithmic blocks to hotspots/hardware bottlenecks

Figure 4.14 shows the mapping between hotspots/hardware bottlenecks and the algorithmic blocks of VIF algorithm. The *Steerable pyramid filtering* block, which includes *Pyramid Filtering* and *Pyramid Step Filtering* functions, consumes approximately 60% and suffers from generation of slow LEA instruction stalls generated by the compiler. The *statistical computation* block (*Parameters Calculation* is the main function of this block) consumes approximately 28.5% and suffers from the major bottleneck in the memory sub-system with L1D and L2D replacements, LLC misses, and DTLB overheads.

4.5.4 Performance Analysis of VSNR

To perform hotspot analysis, the VSNR algorithm was applied for 30 iterations for each image. The results of hotspot for VSNR are provided in Table 4.6. These results show that the average execution time for all 42 images for MAD is approximately 0.72

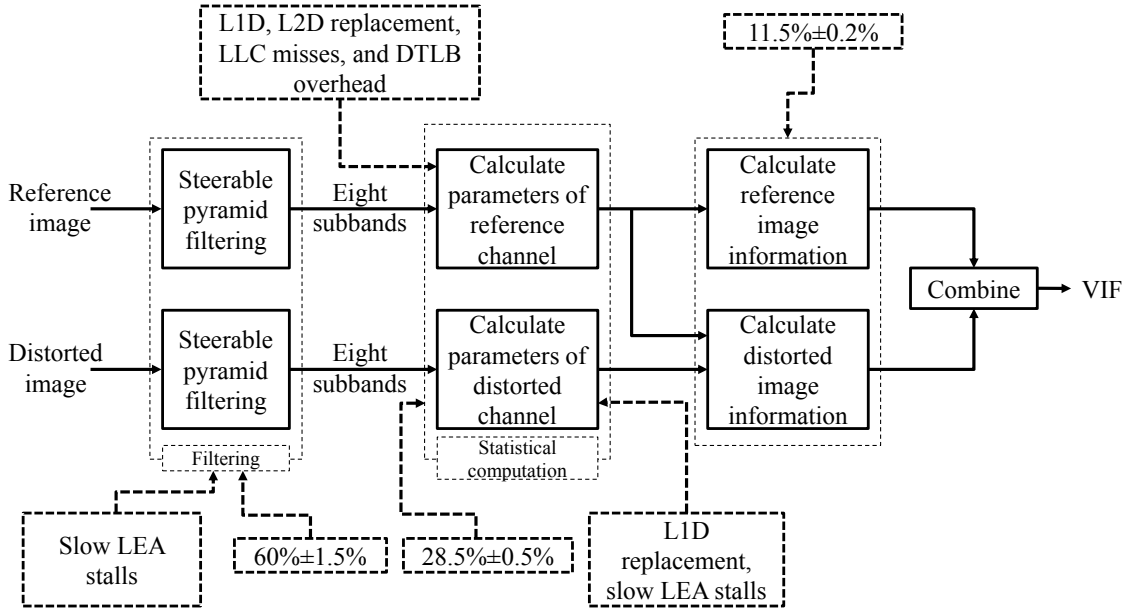


Figure 4.14: Mapping between hotspots/hardware bottlenecks and the algorithmic blocks for VIF. The *Steerable pyramid filtering* block suffers from generation of slow LEA instruction. The *statistical computation* block suffers from memory bottlenecks and generation of LEA instructions.

seconds. We also plot the individual execution time for all 42 images in Figure 4.15. The results show that VSNR has minimal variation of total execution time across different image content as well as different distortion types. Thus, any optimizations for VSNR can be made without any specific consideration of image content or distortion, similar to MAD, MS-SSIM, and VIF.

Table 4.6: Analysis results of VSNR. Average execution time for top hotspots functions/blocks of 42 images is presented with the standard deviation. The total execution for each hotspot is calculated from the average. The hardware bottlenecks are also provided.

Function/block	**Execution time (s)**	**Total execution (%)**	**Hardware bottlenecks**
1D DWT-Columns	0.24 ± 0.02	32.61%	L1, L2 replacement, and LLC hits
Variance	0.12 ± 0.04	16.45%	None
1D DWT-Rows	0.10 ± 0.02	13.68%	4K aliasing except JPEG5, machine clears for JPEG5
Others	0.27 ± 0.05	37.28%	N/A
All	0.72 ± 0.03	100%	N/A

As shown in Table 4.6, the top two hotspot functions contribute approximately 49.06% of the total execution time, while all others account for the remaining 50.94%.

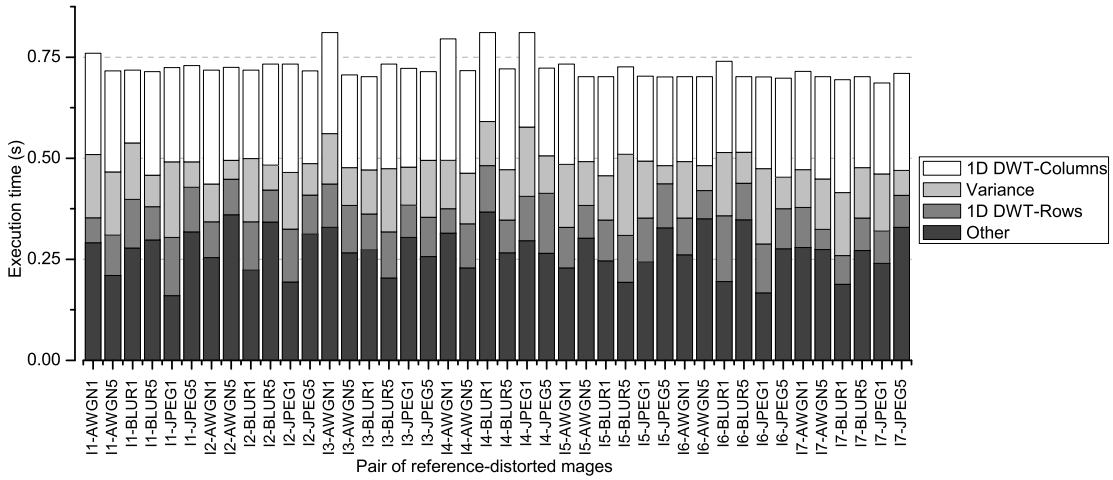


Figure 4.15: The execution time of VSNR for each pair of reference and distorted images. The contributions of hotspot functions are stacked together to form the total execution time.

Also shown in this table are hardware bottlenecks (for each hotspot function) identified via the microarchitechtural analysis. The following subsections describe details of the observed results for each of the hotspot functions, and include explanations of underlying computer architecture concepts whenever necessary.

A 1D DWT-Columns

The *1D DWT-Columns* function computes a 1D Discrete Wavelet Transform (DWT) across the columns of the reference and distorted images. The function takes approximately 0.24 seconds, which is 32.61% of the total execution time.

Investigating the microarchitectural resources utilized by the function, we find that the major penalty for the DWT-columns function is due to LLC hits, which means that the function is accessing LLC frequently. The LLC takes approximately 26-31 clock cycles for a single memory access, which is expensive. Consequently, *1D DWT-Columns* is the top hotspot.

Along with LLC accesses as a bottleneck, we find that there are penalties due to data replacement in L1D and L2D caches. Performance can be improved by reducing the L1D and L2D misses, which will automatically reduce LLC accesses.

B Variance

The *variance* function takes approximately 0.12 seconds, which is 16.45% of the total execution time. The *variance* function is employed to calculate the RMS contrast in the statistical computation block (Figure 4.3). From the microarchitectural analysis, we find that there are no hardware bottlenecks. This means that the function has complex instructions with floating-point numbers that take multiple clock cycles to execute. The analysis does not show a bottleneck because none of the floating-point execution units is overwhelmed, and thus none of the units cause stalls in processor.

C 1D DWT-Rows

The *1D DWT-Rows* function consumes approximately 0.10 seconds, which is 13.68% of the total execution time. The *1D DWT-Rows* function, similar to *1D DWT-Columns*, calculates the DWT coefficients, but across the rows instead of the columns of the reference and distorted images.

The microarchitecture analysis shows that the *1D DWT-Rows* function has performance bottlenecks in the memory sub-system. There are memory reissues because of *4K aliasing* in this function. In addition, for JPEG5 images, our results show that there are also micro operations that get cancelled due to machine clears.

D Mapping algorithmic blocks to hotspots/hardware bottlenecks

Figure 4.16 shows the mapping between hotspots/hardware bottlenecks and the algorithmic blocks of VSNR algorithm. The *five-level 2-D DWT* block, which includes *1D DWT-Columns* and *1D DWT-Rows* functions, consumes approximately 62% of the execution time, and it suffers from memory bottlenecks ranging from cache misses to memory violations. Those bottlenecks, however, are nowhere to be found in the *statistical computation* block (*variance* and some other functions), which is also a hotspot with approximately 28% of the execution time.

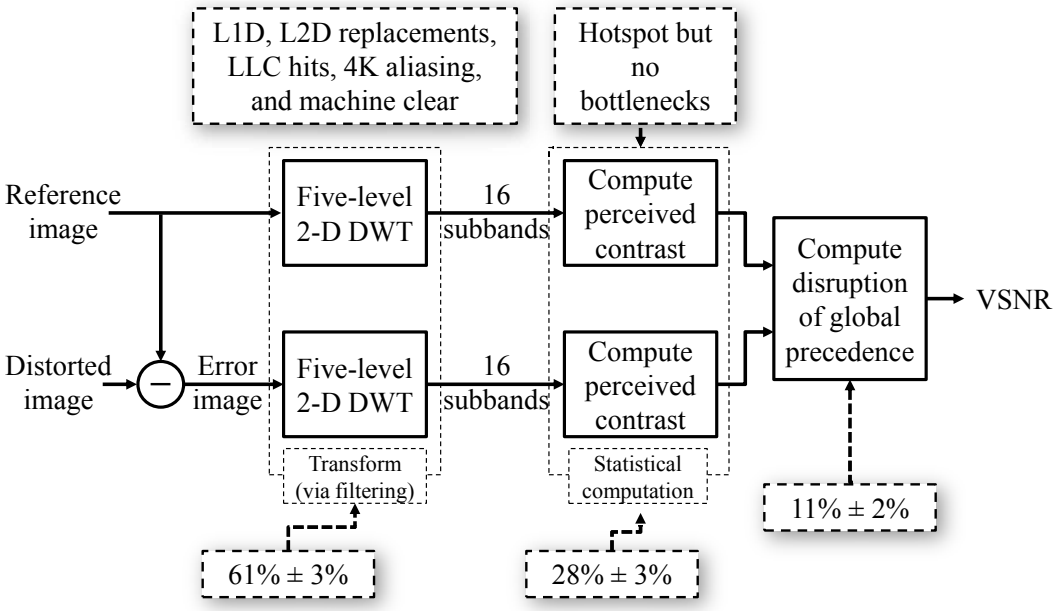


Figure 4.16: Mapping between hotspots/hardware bottlenecks and the algorithmic blocks for VSNR. The *five-level 2-D DWT* block (61% of the execution time) suffers from cache replacements, LLC misses, 4K aliasing, and machine clears. The *statistical computation* block is a hotspot with approximately 28% of the execution time, but there is no bottleneck.

4.5.5 Performance Analysis of BLIINDS-II

To perform hotspot analysis, the BLIINDS-II algorithm was applied for 30 iterations for each image. The results of the hotspot analysis are provided in Table 4.7. These results show that the average execution time for all 42 images for BLIINDS-II is approximately 8.03 seconds. The top hotspot function, *Fast-DCT2*, contributes approximately 47.44% of the total execution time while the second one consumes approximately 23.44%. The other functions add up to the remaining 29.12%.

Figure 4.17 shows the individual execution time for all 42 images. This figure shows that the execution time of BLIINDS-II for JPEG5 images is considerably higher than for the other distortion types. The bottlenecks for JPEG5 images and all the other images are discussed in the later subsections.

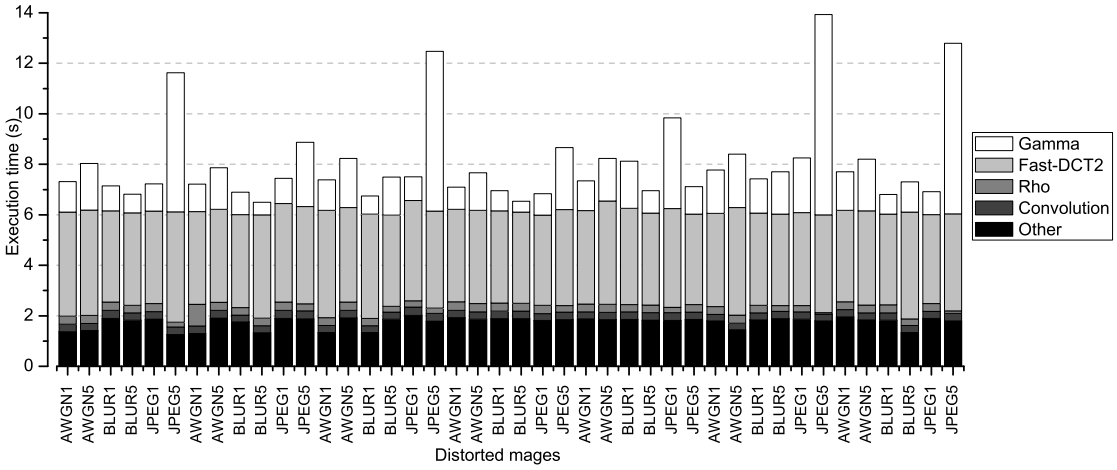


Figure 4.17: The execution time of BLIINDS-II for all distorted images. The contributions of hotspot functions are stacked together to form the total execution time.

A Fast-DCT2

The *Fast-DCT2* function calculates the 5×5 DCT of the image at each of the three scales. The function takes approximately 3.81 seconds, which is 47.44% of the total execution time.

The microarchitectural analysis shows that *Fast-DCT2* function does not have any bottlenecks. Therefore, to gain further insight, we calculate the throughput of the function using the retired pipeline slot metric [84] to investigate if the function traverses the pipeline efficiently. The retired pipeline slot metric for the *Fast-DCT2* function is 0.65, which is greater than the acceptable value of 0.6. The hardware resources are being used optimally in this situation. As mentioned in Section B, the

Table 4.7: Analysis results of BLIINDS-II. Average execution time for top hotspots functions/blocks of 42 images is presented with the standard deviation. The total execution for each hotspot is calculated from the average. The hardware bottlenecks are also provided.

Function/block	**Execution time (s)**	**Total execution (%)**	**Hardware bottlenecks**
Fast-DCT2	3.81 ± 0.23	47.44%	None
Gamma	1.88 ± 1.70	23.44%	None/L1D replacement for JPEG5
Rho	0.30 ± 0.11	3.69%	None
Convolution	0.29 ± 0.02	3.64%	L1D, L2D replacement, LLC hit
Others	1.75 ± 0.22	21.79%	N/A
All	8.03 ± 1.69	100%	N/A

Fast-DCT2 function uses a look-up table to store the cosine values. In addition, we use single loops for first row and column and a nested loop for the remaining pixels. Because the cosine operation is eliminated using the look-up table and all other functions are not relatively inexpensive, the throughput is acceptable, having few stalls and no hardware bottlenecks.

B Gamma

The *Gamma* function performs a fitting process of the DCT data histogram to the Gaussian model as a line search procedure over 9970 values. This function takes approximately 1.88 seconds to operate, which is 23.44% of the total execution time.

The microarachitectural analysis shows that the bottleneck for this function only occurs for JPEG5 images. For these images, the *Gamma* function becomes the top hotspot. The hardware bottleneck is caused by the data being replaced in the L1D cache, which means the processor fetches data from the L2 cache, which has higher latency. During the line search process, the function traverses an array of 9970 values. If there is a match, the traversal stops. We observe that for JPEG5 images, the function has to traverse to the end of the array. This 9970-value array needs approximately 9970×8 bytes (80 KB). However, the L1 cache is 32 KB and cannot hold all of the data. Therefore, some of the values in the array are stored in the next level of cache, and for JPEG5 images, the function has to fetch data from the next level of cache (L2), resulting in higher latency and higher execution time. To improve the performance for the *Gamma* function, one suggestion would be to traverse the array based on the input image's profile. For JPEG5 images, traversing the array from the end would match the value in fewer iterations and improve performance.

C Rho

The *Rho* function is a sorting function used for feature extraction, which takes the 10^{th} percentile of the sorted array. Because it is employed for multiple features, it is called multiple times in the code. The results of analysis show that the execution time for *Rho* function is approximately 0.30 seconds, which is 3.69% of the total execution time.

No bottlenecks were shown in the microarchitectural analysis, we therefore computed retired pipeline slot metric to find the throughput of the function. The retired pipeline slot metric is 0.2, which is much lower than the acceptable minimum of 0.6. This finding suggests that the *Rho* function inherently has complex computations which require higher number of clock cycles, but the Vtune Amplifier XE is unable to identify the exact cause.

D Convolution

The *Convolution* function performs convolution across the image and is employed to perform low-pass filtering. Its average execution time is close to the *Rho* function's, at approximately 0.29 seconds, which is 3.64% of the total execution time. From the microarchitectural analysis, we find that there are L1D as well as L2D replacement penalties because the function accesses the LLC to fetch its operands.

E Mapping algorithmic blocks to hotspots/hardware bottlenecks

Figure 4.18 shows the mapping between hotspots/hardware bottlenecks and the algorithmic blocks of BLIINDS-II algorithm. The *block-based DCT* block is a hotspot with approximately 66% of the execution time. However, this block has no bottlenecks with the optimized *Fast-DCT2* function. The *Gamma* function is the main function of *generalized Gaussian modeling* block (24% of the execution time), and has no bottlenecks except L1D replacements for JPEG5 images. The *Convolution*

function, one of the *LPF* block's functions suffers from memory bottlenecks, L1D and L2D replacements, and LLC hits.

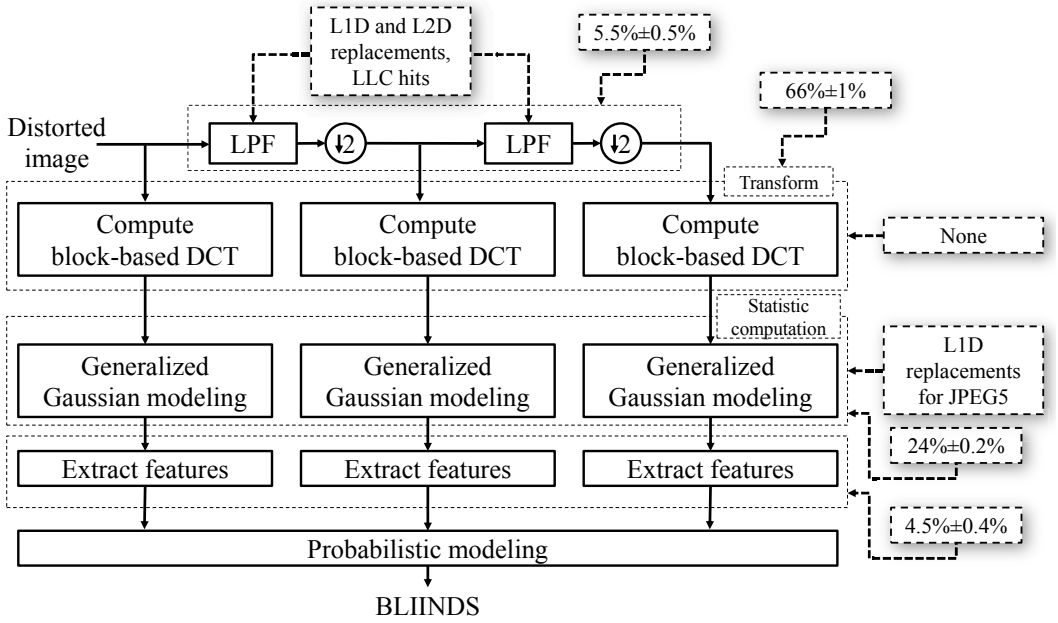


Figure 4.18: Mapping between hotspots/hardware bottlenecks and the algorithmic blocks for BLIINDS-II. The *block-based DCT* is a hotspot with approximately 66% of the execution time. However, this block has no bottlenecks. The *Gamma* function is the main function of *generalized Gaussian modeling* block has no bottlenecks except L1D replacements for JPEG5 images. The *Convolution*, one of the *LPF* block's functions suffer from memory bottlenecks, L1D and L2D replacements, and LLC hits.

4.5.6 Performance Analysis of BRISQUE

To perform hotspot analysis, the BRISQUE algorithm was applied for 30 iterations for each image. The results of the hotspot analysis are provided in Table 4.8. These results show that the average execution time for all 42 images for BRISQUE is approximately 2.65 seconds. The top hotspot function, *Circularly Shifting*, contributes approximately 23.02% of the total execution time. The second hotspot consumes approximately 20.75%. The others add up to the remaining 56.23%.

Figure 4.19 shows the individual execution time for all 42 images. This figure shows that the execution time of BRISQUE for JPEG5 images is faster than for the

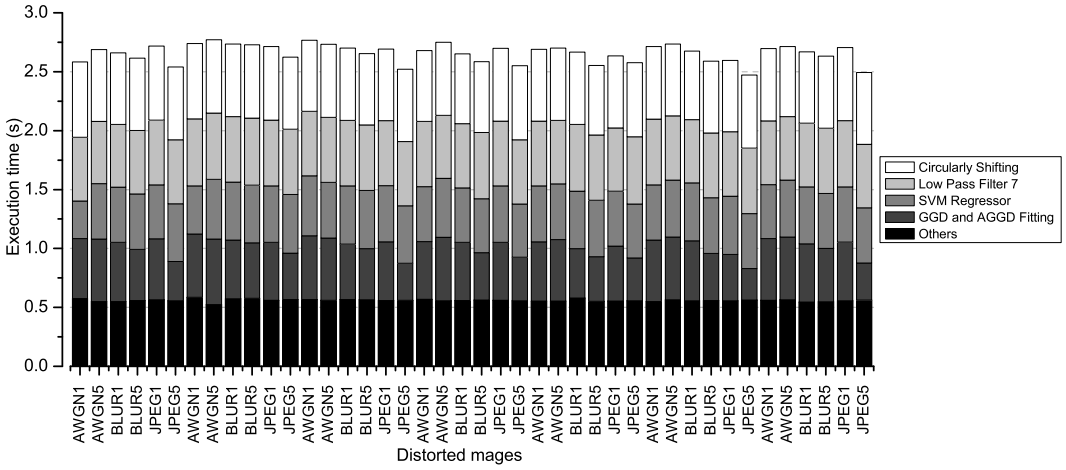


Figure 4.19: The execution time of BRISQUE for all distorted images. The contributions of hotspot functions are stacked together to form the total execution time.

other distortion types. Among the top hotspots, the *GGD fitting* and *AGGD fitting* blocks exhibit the most variation; this variation is discussed in Section D.

A Circularly Shifting

The *Circularly Shifting* function circularly shifts the MSCN coefficients one pixel to four orientations to obtain horizontal (H), vertical (V), main-diagonal (D1), and secondary-diagonal (D2) orientated versions. This function is employed to take into account the statistical relationships between neighboring pixels. This block takes approximately 0.61 seconds to operate, which is 23.02% of the total execution time.

The *Circularly Shifting* function clearly shows a loss of performance due to the

Table 4.8: Analysis results of BRISQUE. Average execution time for top hotspots functions/blocks of 42 images is presented with the standard deviation. The total execution for each hotspot is calculated from the average. The hardware bottlenecks are also provided.

Function/block	**Execution time (s)**	**Total execution (%)**	**Hardware bottlenecks**
Circularly Shifting	0.61 ± 0.01	23.02%	LLC misses, and L1D, L2D replacements
Low Pass Filter 7	0.55 ± 0.01	20.75%	L1 Cache misses, L2D replacements, and LLC misses
SVM Regressor	0.47 ± 0.03	17.74%	Branch mispredict, front end (I-Cache)
GGD and AGGD Fitting	0.46 ± 0.07	17.36%	LLC misses, L1D, L2D replacements, and DTLB overhead
Others	0.56 ± 0.01	21.13%	N/A
All	2.65 ± 0.08	100%	N/A

CPU caches. The data caches at both the level one and level two caches, and the combined last level cache are all overwhelmed by this function. This is because computing the pairwise products of the MSCN coefficients with their four circularly shifted versions requires large amounts of memory.

B Low Pass Filter 7

The *Low Pass Filter 7* function is an implementation of a 7×7 circularly-symmetric Gaussian filter. This function is called four times total for two scales. The average execution time is approximately 0.55 seconds, which is 20.75% of the total execution time.

Observing the microarchitectural analysis results, we find that the bottlenecks are again purely in the memory sub-system. Specifically, the bottlenecks are caused due to L1 Cache misses, L2D replacements and LLC misses. The *Low Pass Filter 7* function was optimized to work with two 7×1 windows instead of one 7×7 window (see Section B), and thus there is no core bottlenecks found here. It needs memory for the MSCN coefficients, squared MSCN coefficients, and Gaussian filtered images. Consequently, this function also has a large memory footprint causing bottlenecks at all levels of caches in the memory sub-system.

C SVM Regressor

The *SVM Regressor* block is basically a function call to the LIBSVM executable via the *system* function, similar to the Matlab version. It collapses 36 features into one single quality index. The time for this function is approximately 0.47 seconds, which is 17.74% of the total execution time.

The code currently makes calls to the executable file to collapse the previously calculated 36 features into a single BRISQUE index. Since, this executable is not compiled as a part of the C++ implementation, there is no prefetching of the in-

structions (not fetched into the I-cache), which causes the I-cache misses and branch mispredictions. Similar issues when linking to external files have been previously reported (See Ref. [85]).

D GGD and AGGD fitting

The *GGD fitting* and *AGGD fitting* blocks perform a fitting of the MSCN coefficients histogram to the generalized Gaussian model, and four pairwise products with an asymmetric generalized Gaussian model as a search procedure over 9801 values. Similar to the *Gamma* function in BLIINDS-II, *GGD and AGGD fitting* employs a line search to fit shape, mean, left variance, right variance to the values of a *gamma* function. However, in contrast to BLIINDS-II where the fitting process is applied for every small 5×5 DCT block, for BRISQUE, the fitting process is employed only 10 times. During the line search process, there could be an early stop when a match found. Therefore, the *GGD and AGGD fitting* block's execution times have a larger variation.

This block suffers from memory bottlenecks. Specifically, the bottlenecks are LLC misses, and L1D, L2D replacements. The block operates on a total of 10 images (eight are generated from the *Circularly Shifting* function and two MSCN coefficients). Because the block operates separately on two different data sets for two scales, there are replacements in the L1 and L2 cache and misses in the LLC. Since the block operates on two different data sets, the mapping of the virtual to physical addresses for both the data sets cannot fit into the small DTLB, which causes the DTLB overhead.

E Mapping algorithmic blocks to hotspots/hardware bottlenecks

Figure 4.20 shows the mapping between hotspots/hardware bottlenecks and the algorithmic blocks of the BRISQUE algorithm. The *Compute locally normalized lumi-*

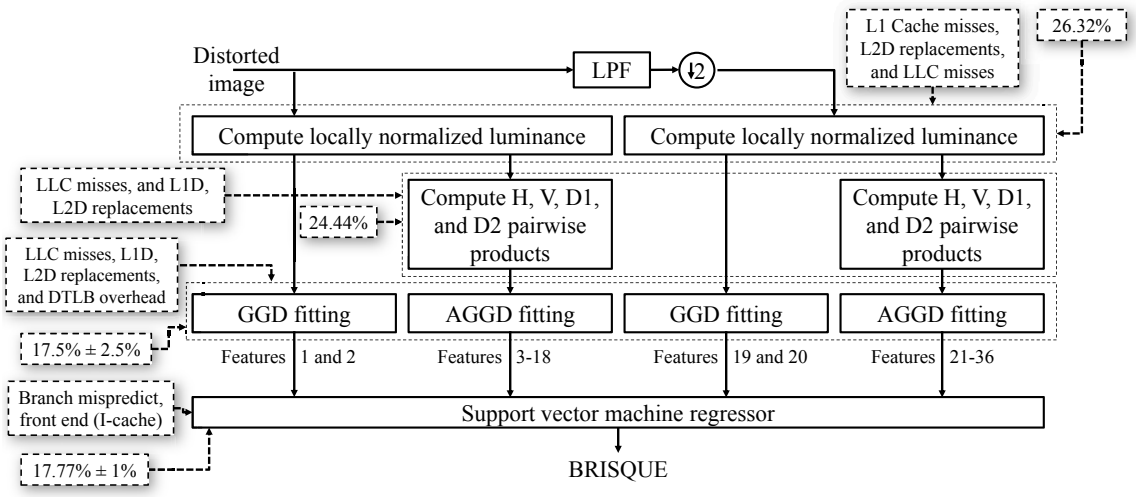


Figure 4.20: Mapping between hotspots/hardware bottlenecks and the algorithmic blocks for BRISQUE. The *Compute locally normalized luminance* is a hotspot with approximately 26.32% of the execution time and suffers from L1 Cache misses, L2D replacements, and LLC misses. In the next stage, the *Compute H, V, D1, and D2 pairwise products* block contain the *Circularly Shifting* function with point-by-point multiplications between the MSCN coefficients and their circularly shifted versions. This block takes approximately 0.65 seconds to operate, which is 24.44% of the total execution time. The *GGD fitting* and *AGGD fitting* blocks consume 17.5% of the running time and suffer from LLC misses, L1D, L2D replacements, and DTLB overhead bottlenecks.

nance block contains mainly the *Low Pass Filter 7* function. It is a hotspot with approximately 26.32% of the execution time and suffers from L1 Cache misses, L2D replacements, and LLC misses. In the next stage, the *Compute H, V, D1, and D2 pairwise products* block contain the *Circularly Shifting* function with point-by-point multiplications between the MSCN coefficients and their circularly shifted versions. This block takes approximately 0.65 seconds to operate, which is 24.44% of the total execution time. The *GGD fitting* and *AGGD fitting* blocks consume 17.5% of the running time and suffer from LLC misses, L1D, L2D replacements, and DTLB overhead bottlenecks.

4.6 Discussion

In this section, we discuss the bottlenecks and provide an insight into them across all the six analyzed algorithms. We first discuss the most common memory bottlenecks and microarchitectural conscious coding techniques to gain better performance. Following the memory bottlenecks, we discuss the core bottlenecks and the techniques to boost performance. We also propose a custom hardware framework, which can be used as a platform to design engines for IQA algorithms and image processing algorithms in general.

4.6.1 Memory Bottlenecks

The results of the microarchitectural analysis show that all of the IQA algorithms have a backend-bound memory bottleneck, but the amount of performance degradation due to these bottlenecks vary greatly for individual algorithms. A memory bottleneck essentially means that the hotspot functions spend a significant amount of time accessing image data. This is usually because they have read/write access patterns that result in misses in the CPU caches. These cache misses have to be serviced from lower levels of the memory hierarchy that are slower to access, which is why we see the algorithms spending more time in these functions and causing them to be hotspots. The large number of cache misses is due to the large working data set for these algorithms. All of the algorithms produce intermediate matrices and process them to assess the image quality. Accessing these multidimensional arrays multiple times causes most of the performance bottlenecks for all of the IQA algorithms tested here.

Even though all the IQA algorithms experience memory bottlenecks, the root cause of the bottleneck as well as the extent of performance degradation due to

these bottlenecks varies across algorithms. For example, all the hotspot functions in MAD are backend memory bound, whereas VIF has a memory bottleneck for just the *Parameters Calculation* function. It should be noted that the *Parameters Calculation* function is the last hotspot for VIF, which means that it is not the major cause for slowdown of the algorithm; its impact on the performance of the algorithm is not high as other functions, and these other functions are not memory bound. Thus, it is important to note that even if all the algorithms at some point have a memory bottleneck, the impact of this memory bottleneck on the speed or throughput of the complete algorithm depends on the rank of the hotspot, which consequently would decide the priority for optimization.

Another observation from the analysis is that even though at an abstract level, all the algorithms show memory as a bottleneck, the actual physical microarchitectural bottleneck is different for different algorithms. For example, the top two hotspot functions in MAD have poor performance because of the DTLB overhead, while the top two hotspots for MS-SSIM and BRISQUE have a higher execution time because of the L1D replacements, L2D replacements, and LLC misses. Different microarchitectural resources are overwhelmed by different functions and algorithms.

Apart from issues with the usual suspects in the memory hierarchy (CPU caches at different levels), there are hotspot functions which show penalties associated with machine clears and 4K aliasing (as in case of VSNR). Thus, our analysis has revealed some interesting performance bottlenecks that would otherwise have gone undetected.

One obvious way to reduce cache misses is to change the hardware platform to a processor with larger caches. This will allow more data to reside in the caches, and thereby reduce the number of cache misses, and will consequently improve performance. However, there is a limit to how big each level of cache in the memory hierarchy can be, and the L1 caches are usually kept under 64 KB for fast access. Thus, it is important that the implementation of an IQA algorithm respects the prin-

ciple of locality of reference to maximize temporal and spatial locality. Locality of reference is the tendency of a program to access the same (temporal locality) or nearby (spatial locality) memory locations repeatedly and frequently. Thus, caching these memory locations can reduce misses. Coding an IQA algorithm with more/better locality can significantly improve performance.

Another technique for improving locality, which is commonly used especially in image processing is called *cache blocking*. It works by dividing larger data chunks into smaller ones that fit in the cache, and making sure that once the small block is brought into the cache, all the operations to be performed are performed before moving on to the next block. Thus rather than traversing a whole image to perform one operation and then reading in the whole image to perform another operation (thus effectively wiping out the cache repeatedly), parts of the image are read and processed at a time while that part is still in the cache.

Accesses to the main memory are very expensive, usually hundreds of clock cycles. For algorithms such as MAD, MS-SSIM, VIF, and BRISQUE where the bottlenecks are due to LLC misses/replacements, using software instructions that can pre-fetch the data into the caches from the main memory effectively masks the memory latency, and can thus increase performance. Such pre-fetch instructions are suitable when memory accesses are predictable and when there are CPU stalls for data being unavailable for processing.

The final memory bottleneck is the DTLB overhead. The TLB is a small cache, which stores a section of the page table (a page table stores a mapping between virtual and physical addresses). When the working data set is large, as in the case of MAD, the TLB is not able to cache all the active mappings and this causes DTLB misses. One solution to this problem is to use a larger page sizes or superpages. With a larger page size, a TLB of the same size can keep track of larger amounts of memory, which avoids the costly TLB misses, reducing the pressure on the TLB.

As an example of applying these techniques, we made slight modifications to the *FFT* and *Gabor Convolution* functions for MAD. We chose these functions because they are in the top hotspot list. Based on our analysis, we knew that the architectural bottleneck was related to the memory hierarchy, as the DTLB, LLC misses, and L1D and L2D replacement feature prominently for MAD. We modified the nested loop in the *FFT* function to improve the locality and data reuse by removing one layer of nesting. This had the effect of accessing a cache block multiple times, as each cache block brought in 8 elements of the array assuming a 64 Byte block size, and they were processed sequentially resulting in 7 hits for every miss in the cache for that loop. The original nested loop processed only one array element per cache block, which resulted in a 100% cache miss rate for that particular nested loop. The second modification we made to the *Gabor Convolution* function was to eliminate the memory required for additional matrices of the same size of the input image. Essentially, the optimization was equivalent to changing the statement $C = A + B$ to $A = A + B$, thus reusing the memory allocated to A. These two modifications resulted in a 9% improvement in terms of running time.

While these examples seem somewhat obvious in retrospect, most codecs are written with many such opportunities overlooked because the focus may not have been on efficient use of the underlying hardware. It is also not possible to anticipate the exact hardware bottlenecks without conducting the kind of analysis illustrated in this paper. The analysis clearly helps pinpoint the functions that take up the largest share of the execution time, and highlights the architectural resource being stressed. Thus, a programmer knows where to look, and what changes to make.

4.6.2 Core Bottlenecks

The next category of bottlenecks is the core bottlenecks or the bottlenecks caused due to manipulation of data. First, we discuss the performance degradation caused

due to floating-point operations, because they have the most significant impact on performance in the category of core bottlenecks.

The floating-point operations inherently have a longer latency and thus have a large impact on performance. The floating-point unit is a bottleneck for MS-SSIM for the *LCS Average* function. We discuss some generic guidelines to improve performance for floating-point units.

Operations carried on single-precision floating-point numbers execute faster than double precision numbers and consume less memory. For example, the *LCS Average* function calculates the luminance and contrast of the reference and distorted images for multiple scales. The mean of pixel values is used to calculate luminance while the variance is used to calculate contrast. All of the algorithms currently use an 8-bit grayscale image, which implies that the resultant luminance and contrast would never exceed the range supported by 32-bit single-precision floating-point numbers. Since calculation for luminance and contrast is calculated for multiple scales, using single-precision floating-point numbers to calculate and represent luminance and contrast can significantly improve performance. On our test platform, single-precision floating point can be set through the precision control (PC) field in the x87 FPU.

Another simple solution is to use integers if possible. For example, if a particular range varies from 0 to 1, the programmer can estimate the degree of precision required, and choose to express the range from 0 to 10, 0 to 100, 0 to 1000, and so on (depending on whether one, two or three digit precision is required).

The developer of an IQA algorithm should keep in mind that the operations should remain in range, i.e., ensure that there are no overflows, underflows, or denormals (extremely small values) for the results. Denormal values and underflow can cause high penalties as they require microcode assists. To improve performance for such a situation, one solution is to write the code in such a way that denormals are not generated, which means that the developer should keep track of the range of the results

produced during the floating-point operations. In addition, by enabling the Flush-To-Zero (FTZ) and Denormals-Are-Zero (DAZ) modes [86], there can be significant improvement in performance. Note that the FTZ and DAZ modes are applicable only for SSE instructions.

The next bottleneck is caused due to generation of slow LEA instructions. LEA instruction is an X86 assembly instruction generated during compilation. On an Intel processor, one solution is to use an Intel compiler, which would produce assembly optimized for Intel's microarchitectures. For example, in a loop, the LEA instructions are generated for the index access. If the index access is reduced, it will decrease the number of LEA instructions to be executed, and thus reduce the overhead.

4.6.3 Summary and Recommendations for a Framework for Custom Image Quality Assessment Hardware

From the microarchitectural analysis, we found that there are two categories of bottlenecks in the IQA algorithms tested here: execution/core bottlenecks and memory bottlenecks. From the analysis, we found that the majority of the algorithms show performance degradation because of the memory bottlenecks. Other studies also show that memory is usually a major bottleneck for image processing algorithms.

Within the memory bottlenecks, the most common issue was due to L1D and L2D replacements and LLC misses. To improve performance in such cases, having larger caches is recommended. A suitable size for the caches and its configuration along with defining a memory hierarchy can be decided by performing cache simulations with various cache sizes and configurations. A combination of a particular cache size and configuration that surpasses a predefined threshold for hit/miss rates should be used. Creating models for a cache configuration that best suits the performance, cost, and other requirements of IQA and related algorithms are topics of future research.

The next common memory bottleneck was the DTLB overhead. MAD, VIF, and

MS-SSIM show performance degradation due to DTLB overhead. A DTLB is special cache which stores a sub-set of translations from virtual memory to physical memory. For a custom IQA engine, there is no requirement of a virtual memory system. Hence, this eliminates the requirement of using a TLB.

Another bottleneck was 4K aliasing. VSNR is the only algorithm with 4K aliasing bottleneck. This problem is caused due to out-of-order execution of memory instructions in the processor. If the custom hardware engine design is an in-order machine, we eliminate the possibility of 4K aliasing.

The most common core bottleneck was the overwhelming of the floating-point unit. All the image transform, image filtering, and statistic calculation require floating-point operations and consequently a floating-point execution unit. Operating on a logarithmic number system would improve the performance. The overhead for converting to log domain and then transforming back is negligible if the number of floating-point operations is very large. In a logarithmic system, the multiply and divide operations change to add and subtract operations respectively, which are less expensive and can save many clock cycles. A custom IQA engine would have multiple floating-point units to exploit parallelism. In addition, the unit will be pipelined to hide/overlap the latency of the instructions for getting data to the IQA execution engine.

The next core bottleneck was due to the slow LEA instructions. These instructions are an outcome of the complex addressing mode of the CISC Intel architecture. Therefore, if the memory control hardware is designed as a load store machine, the performance degradation is automatically eliminated. In addition, the proposed custom engine is hardcoded, which eliminates issues due to generation of such instructions by the compiler.

The final core bottleneck was the generation of micro assists. Floating-point micro assists occurred because the operands or results of an operation were denomals. If

the precision can be traded-off, these denormals are directly converted to zero and if precision is required, a custom hardware just to process denormals can be designed. A special port can be designated to dispatch the denormals to this unit. If there are no denormals, this unit can work as a normal floating-point unit.

In general, all tested IQA algorithms contain the same operation at an abstract level: an image transform or filtering and a statistical computing. For example, MS-SSIM and BRISQUE use low-pass filters, whereas VIF, VSNR, MAD, and BLIINDS-II transform the image into frequency domains via Steerable Pyramid, DWT, FFT, and DCT, respectively. These algorithms also perform a statistical computation. For example, MS-SSIM needs the mean and variance to calculate the structure similarity; MAD calculates the standard deviation, skewness, and kurtosis to form the statistical difference maps. Therefore, a generic IQA engine would have a transform engine, a filtering engine, and a statistical computation engine.

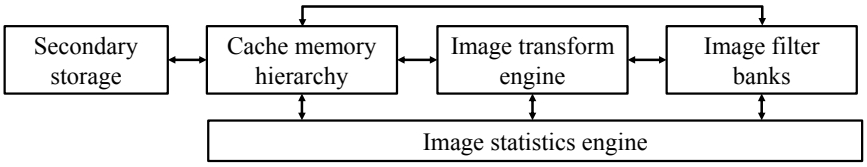


Figure 4.21: Blocks for custom IQA engine framework. It constitutes of three basic computational blocks generally used in IQA algorithms: image transform engine, the filter banks and the image statistics engine.

A block diagram for a general IQA hardware engine is shown in Figure 4.21. We have a secondary storage to store the input images. These images are brought into the fast memory, the caches. From the caches, the images act as operands to one or more of the three engines depending on the sequence of operations performed by the specific IQA algorithm. In addition, there are instances where an operation is performed multiple times. Therefore, we propose a fully connected internetwork. Such an interconnected network helps to feed data directly to the respective engine, which leads to reuse of the existing hardware and saves chip area and cost.

The different execution engines are designed as follows:

1: The transform block can be a general purpose floating-point unit or a transform specific custom design such as a DWT unit in VSNR.

2: The filter blocks can be implemented as a general-purpose filter if multiple filter banks are used or can be a specific implementation, such as a log-Gabor filter unit in MAD.

3: The image statistics block can be implemented as a general purpose engine or a custom engine. However, if we observe the algorithms, they comprise of multiple statistical computations. Therefore, it is not recommended to create an engine for all, but rather to utilize a general purpose floating-point unit. In order to define which operation needs to be performed, control signals can be generated. It is suggested to pipeline these engines. Pipelining would hide some latency for accessing memory and also improve the throughput of the hardware. Further details about pipelining, tradeoffs for a pipeline and designing a pipelined hardware can be found in Ref. [82].

4.7 Summary

This chapter presented performance analyses of six popular image quality assessment algorithms. Even though the approaches to the six IQA algorithms are different, the algorithms shared the same main stages: a filtering (transforming), and a statistical computation. Our results revealed that different IQA algorithms overwhelm different microarchitectural resources and give rise to different types of bottlenecks in two main categories: memory bottlenecks and core/computational bottlenecks. Specific microarchitectural bottlenecks for each function/block of each algorithm were pointed out. We also proposed the hardware/microarchitectural conscious coding techniques for optimization and performance improvement. The findings and recommendations presented in this paper apply broadly to all current-generation Intel IA-32 and Intel 64 based general-purpose computing platforms, whether laptops, servers, or desktops, even though the actual hotspot and bottleneck details might vary. Architectures that are radically different, with hardware accelerators, dedicated image processing cores (such as those found on some tablets and smart phones), and memory shared between GPUs and CPUs (such as AMD's Fusion APUs) are expected to show very different execution characteristics. Further studies using a similar methodology are recommended to analyze the performances of IQA algorithms on these specialized architectures.

CHAPTER 5

A RR IQA FRAMEWORK BASED ON SHARPNESS MAPS

5.1 Introduction

The RR IQA approaches can be classified based on blockiness/edge quantification in either spatial domain or frequency domain, analysis in a transformed domain, using natural-scene statistics, or developing RR IQA based on an existing FR idea. In section 5.2, we provide a review of existing approaches to RR quality assessment.

Current works on RR are mainly limited by two factors: They are either limited by the number of distortion types (up to four common distortion types: AGWN, Gaussian blur, JPEG, and JPEG2000) or limited in their ability to achieve good performance because of the small number of scalars for reduced information. These distortion-specific RR and some other NR algorithms ([16, 87]) were trying to be generic by combining few distortion types. However, they have never been tested with all 17 distortion types in TID2008[3] or 24 distortion types in TID2013 [4] image databases. The others employed from a few to a hundred scalars for reduced information, and could only achieve performance at the level of about the same as MSE/PSNR. More recently, Soundararajan and Bovik [88], and Liu *et al.* [89] have provided two RR frameworks, RRED and SPCRM, respectively. These frameworks can achieve better performance with increase in data rate. When utilizing 2.8% and 3.1% reduced information (for RRED and SPCRM, respectively), these algorithms are competitive to current state-of-the-art FR algorithms such as MS-SSIM[14], VIF [15], MAD [7], or FSIM [90] on all big image databases (TID2008 [3], LIVE [1], CSIQ [2]). Therefore, what we desire in this paper is a fully generic distortion types RR

algorithm (by utilizing distortion families), which is better than above-mentioned FR algorithms with around 2% of reduced information.

In this paper, we develop a new RR IQA framework based on multi-scale image sharpness maps and seven distortion families. Previous work on image sharpness has been focused mainly on developing sharpness index, a single value, which quantifies the overall sharpness of an image [19, 20, 91–94], only a few of them (S_1, S_2, S_3 in [19] and FISH in [20]) can generate a sharpness map, which quantifies the perceived sharpness of different areas within an image. The image sharpness map has been employed in NR IQA of blurred images [19, 20], and NR IQA of JPEG2000-compressed images [95]. Now, we employ sharpness map in our RR framework, named S4RR (Sharpness for Reduced Reference task). In S4RR framework, any sharpness maps (S_1, S_2, S_3, FISH, or local standard deviation) can be used as reduced reference features. The amount of reduced information depends on which sharpness algorithm has been used, and the block-size of the algorithm.

We stated that the local standard deviation could generate sharpness map, because local standard deviation could yield a meaningful sharpness maps comparing to our local sharpness map database, as presented in Chapter 2.

For RR task, Wang and Simoncelli [96] suggested that the appropriate RR features should provide an efficient summary of the reference images, be sensitive to a variety of image distortions, and be relevant to visual perception of image quality. We argue that the multi-scale sharpness maps are reasonable to assess image quality because of two main reasons. First, they provide an efficient summary of the reference images, i.e., smooth areas, texture areas, or sharp areas. Second, they are sensitive to a variety of image distortions; for example, the sharpness values increase in a noise-added image (AWGN, ColorNoise, PinkNoise); the sharpness values decrease in a blurred image (Gaussian blurring, Contrast distortion, Denoised image); or both increasing and decreasing (JPEG: block artifact and block quantization. JPEG2000:

aliasing/ringing and blurring).

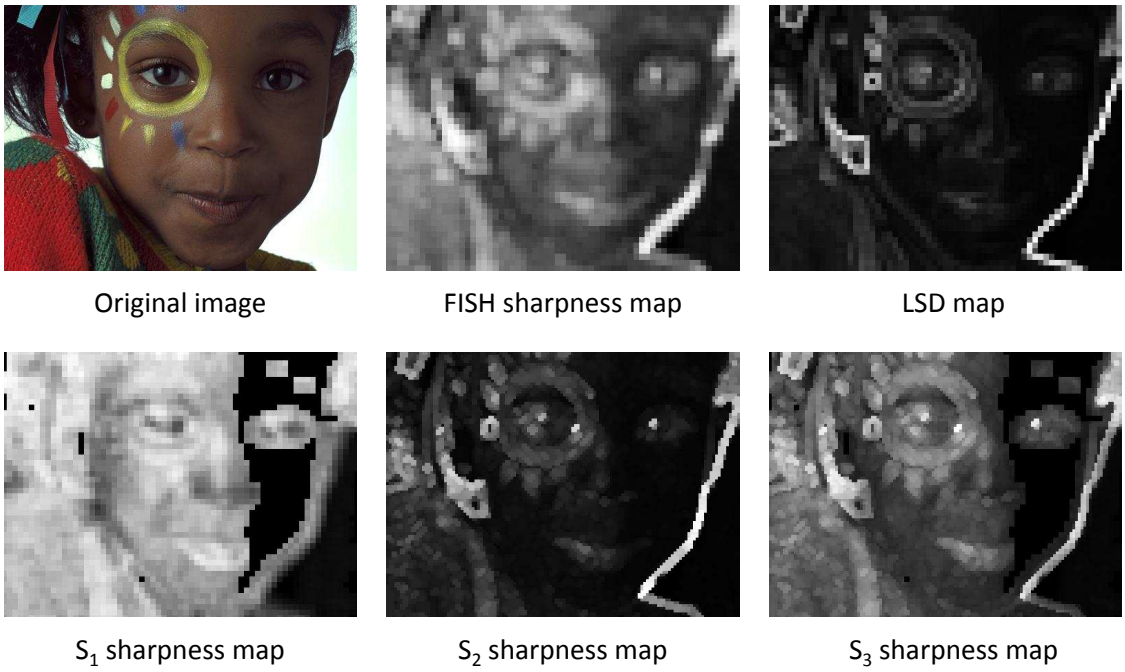


Figure 5.1: Original images and five different sharpness maps.

Figure 5.1 shows an original image and its five level-one sharpness maps: FISH sharpness map, local standard deviation (LSD) maps, and S_1, S_2, and S_3 sharpness map. The maps are generally much smaller than the input image, with different ranges. Here, we scale and normalize them for showing purposes. We can see that the maps can summarize efficiently the input image. For example, smooth areas on the left side on the background and under the girl's eye are captured by dark-gray areas in all the maps. The strong edges between the girl's face and the background are highlighted by the most whitest areas in the maps. The texture/busy area on the girl's shoulder is stressed out by mid-gray with different values in the maps.

From Figure 5.1, we can see that all types of sharpness maps can capture well the characteristics of the input image. The S_2 and LSD maps look similarly and darker than the other three maps, because of sharpness values' distribution. The S_1 map has a large black area because of the algorithm's quantization. However, as we will show, they can replace each other in S4RR framework. From this point, we employ

LSD map as sharpness map for demonstrating purposes.

Figure 5.2 demonstrates that sharpness maps can capture both distortion types and distortion intensities. Showing in first column is original image, its first-scale sharpness map, and scatterplot of the map versus itself. Showing in a2-a8 and d2-d8 are 14 distorted images of seven distortion types and two distorted levels. Rows *b*, *e* and *c*, *f* are first-scale sharpness maps and scatterplots of the distorted maps versus original map, respectively. We can see that different distortion types can be captured by the change in the scatterplots' shapes. For example, when the image is original (figure 5.2(*c1*)), the shape (in green) is fit perfectly on the red line, which is a linear matching. In *c2* and *f2*, the scatterplot spreads both sides of the red line, especially the left-bottom area. On columns *3* and *4*, the white noise and high-frequency noise modify mostly the lower areas of the shapes (*c3*, *f3* and *c4*, *f4*). On columns *5* and *6*, contrast-changing images produce very well-correlated scatterplots, with the contrast reduction shapes fall under the red line (*c5* and *f5*), while the contrast enhancement shapes mostly above the red line (*c6* and *f6*). The blurred image scatterplots in *c7* and *f7* have a large drop-down in the higher area. For localized distorted images, figure 5.2(*c8* and *f8*) has most of the green dots on the red line, and some sparse dots everywhere else. Distortion intensities can also be captured by sharpness map if we compare row *c* and row *f*.

Our framework, S4RR (Sharpness maps for Reduced Reference task), is a generic RR framework via employing seven distortion families. It also employs two measurements on a multi-scale sharpness to assess quality. Two employed measurements are the average of three maxima of the differences (for uniform distortion types), and the average of three L_2-norms of the differences (for the nonuniform distortion types). Seven distortion families employed in our algorithm are: (1) Spatially correlated broadband distortions, (2) Uniform White Noise, (3) High-frequency Noise, (4) Contrast reduction, (5) Contrast enhancement, (6) Blurring, and (7) Spatially local-

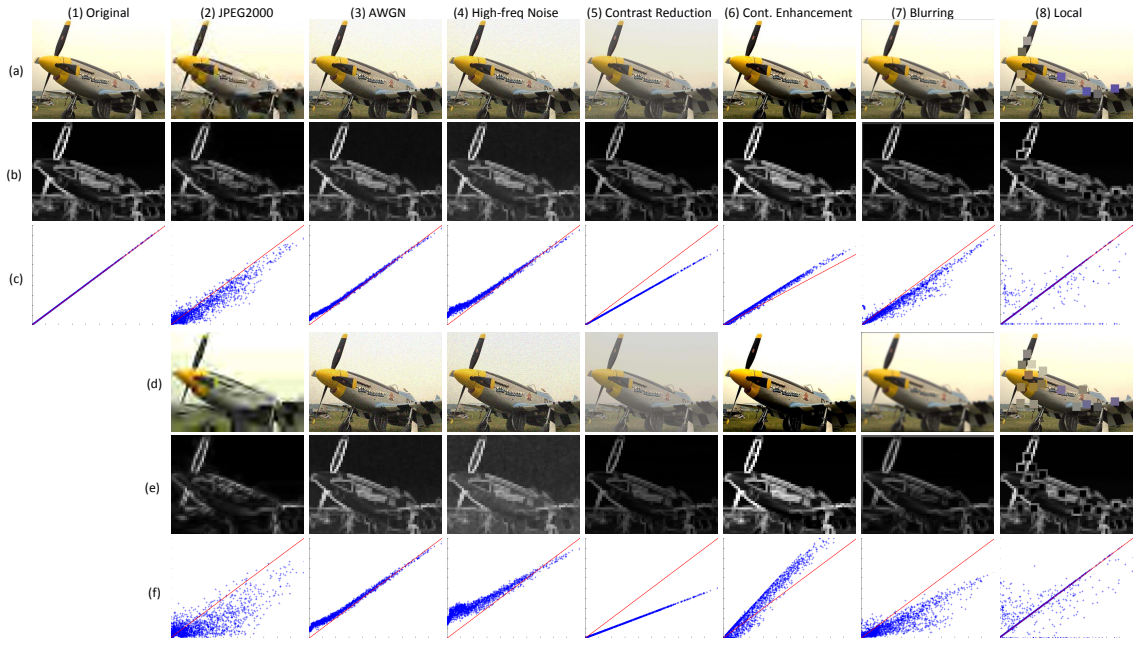


Figure 5.2: The demonstration that sharpness maps can capture both distortion types and distortion intensities. Showing in first column is original image, its first-scale sharpness map, and scatterplot of the map versus itself. Showing in a2-a8 and d2-d8 are 14 distorted images of seven distortion types and two distorted levels. Rows b, e and c, f are first-scale sharpness maps and scatterplots of the distorted maps versus original map, respectively. Different distortion types and distortion intensities can be captured by the change in the scatterplots' shapes.

ized distortions. Namely, S4RR adaptively selects and switches among seven scalars and two measurements to estimate quality. The framework provided in this paper is also useful for other feature maps, as long as these feature maps can capture the effectively reference information, change differently for different distortion types/families, and capture the distortion intensities. As we will demonstrate on four largest image databases, our method - S4RR, when utilizing approximately 2.08% of reduced information, is better than current RR frameworks, which employed around 3% reduced information. S4RR also outperforms current state-of-the-art FR algorithms on most of the criteria.

The rest of this Chapter is organized as follows. Section 5.2 summarizes existing methods of reduced-reference image quality assessment. Section 5.3 provides the detail steps of the S4RR framework, including the reduced information extraction,

collecting 19 multiscale sharpness features to classify an image into seven distortion families, the average of three maxima and average of three L-2 norms measurements, and combining measures and distortion families process. The results and discussion of the our algorithm in predicting a subjective quality rating on TID2008 [3], LIVE [1], CSIQ [2], and TID2013 [4] image databases are provided in Section 5.4. This section provides prediction performance on training database, TID2008, and the classification accuracy and the prediction performances on all other databases. General summaries are presented in Section 5.5.

5.2 Related work

Even though what and how much reduced information remain challenging questions, a lot of relevant progress has been made in RR area [88, 89, 96–114]. It is difficult to classify RR IQA algorithms into separated classes. However, there are a few noticeable trends to calculate reduced information and to estimate image quality. Some RR IQA works employ the natural scene statistic (NSS) models to extract RR features [96, 102, 106]. Some other works extract the statistics of image in some transformed domains [100, 101, 103, 105, 108]. On another trend, some works focus on developing RR algorithms, which employ more than one strategy for as many distortion types as possible [97, 104, 109]. More recently, some researchers were trying to find the relationship between current FR models and their RR algorithms [88, 107, 111–113]. In this section, we provide a brief review of current RR methods based on these main trends.

5.2.1 Methods based on natural scene statistic

Natural scene statistics were explored early in image processing research [115, 116]. NSS has been also employed widely to develop RR IQA algorithms. For example, in [96], Wang and Simoncelli used the Kullback-Leibler divergence between the marginal

probability distribution of wavelet coefficients based on a wavelet-domain statistical model to estimate the quality. In [102], Li and Wang employed a Gaussian-scale mixture-based statistical model of wavelet coefficients to quantify image quality. In [106], Ma *et al.* developed an RR algorithm based on a generalized Gaussian distribution model on DCT domain.

5.2.2 Methods in a transformed domain

Besides wavelet and DCT to calculate NSS features, calculating the differences on grouplet, curvelet, and contourlet transforms is also a common RR approach. For example, in [100], Maalouf *et al.* calculated information regarding textures and gradients of the images via grouplet transform and employed a CSF filtering and thresholding to estimate quality. In [101], Gao *et al.* performed multiscale geometric analysis including curvelets, bandlets, wavelets, and contourlets to develop the RR IQA algorithm. In [103], Tao *et al.* employed the city-block distance to measure the quantity differences of the visual sensitive coefficients in the contourlet domain. In [105], Xue and Mou developed the algorithm based on Weibull distribution of wavelet coefficients in wavelet domain. In [108], Lin*et al.* used the directional information of the image obtained from the complex wavelet domain to measure image quality.

5.2.3 More distortion types, closer to the general distortion

In contrast to using one single strategy to model HVS [98, 99], many algorithms advocate that the HVS uses multiple strategies to determine image quality. For example, in [97], Gunawan and Ghanbari proposed *gain* and *loss* idea. The algorithm first applied Sobel edge detection for original and distorted images and computed local harmonic amplitude information of edge maps as the reduced information. By performing analysis of the local harmonic amplitude information, the authors divided the local harmonic amplitude maps into gain (positive) and loss (negative) informa-

tion areas, and combined two separated measurements to estimate image quality. In [104], Engelke *et al.* calculated five measures for four artifacts in wireless imaging: blocking, blur, ringing, and intensity masking and lost blocks. In [109], Bordevic *et al.* refined four quality features to qualify image quality in three distortion types, Gaussian blur, JPEG, and JPEG2000.

5.2.4 Methods based on current FR algorithms

More recently, another RR approach inspired by the ideas of the state-of-the-art FR IQA models has been developed. For example, in [107], Rehman and Wang developed the RR-SSIM algorithm, which is reduced information estimation of the SSIM [5] algorithm. From the same SSIM idea, Bhateja *et al.* [112, 113] measured the structural dissimilarity to estimate the quality. Visual information fidelity [15] also inspired RR algorithms [88, 111]. Especially, in [88], Soundararajan and Bovik used to develop a framework for RR IQA based on information theoretical measures of differences between the reference and distorted images by using the entropic of wavelet coefficients. When the reduced reference information (the number of scalars required) is around 2.8% of the image size, the algorithm performs nearly as good as the best performing FR IQA algorithms.

5.2.5 Summary of existing methods

In summary, current methods share a common thread: using one or a few strategies to model the way the HVS operates, e.g., measuring how unnatural an image is, measuring the difference in some transformed domains, or a combination of different distortion measures. Some researchers have been focused on using more than one strategy and tried to be generic at the same time. However, they archived only good results for a few distortion types [97, 104, 109].

In the following section, we describe our approach to RR IQA, the S4RR frame-

work, considering a distorted image in distortion families, not distortion types. Via dividing common distortion types (five distortion types in LIVE, six distortion types in CSIQ, and 24 types in TID2013 image databases) into seven distortion families, we demonstrate that S4RR is truly generic, and with approximately 2.05% reduced information, S4RR can perform as good as the state-of-the-art FR IQA algorithms.

5.3 New Reduced-Reference IQA Framework

This section describes the details of S4RR framework. The block diagram of S4RR is shown in Figure 5.3 with five main stages. We will go through each stage in detail in each subsection (Section 5.3.1 to 5.3.5). In general, the reduced reference information is the three-scale sharpness maps. From three pairs of sharpness maps, we compute 19 futures to class the distorted image using seven-class classification. Two measurements, the average of three maxima and average of three L_2-norms, are calculated from reduced information. Finally, the soft-classification results with probabilities are employed to combine a class-weighted average of two measurements to yield the final S4RR quality index.

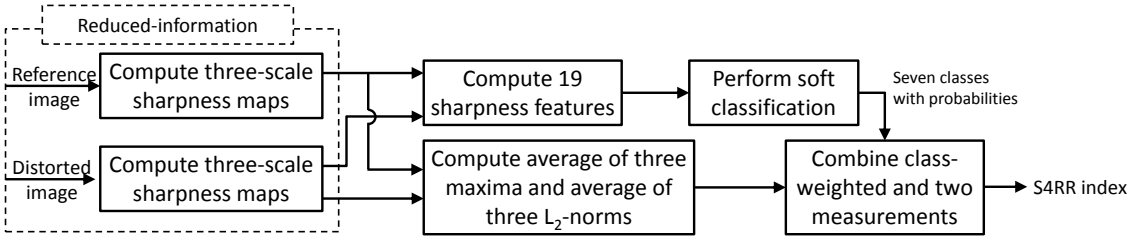


Figure 5.3: Block diagram of the S4RR framework. Multiscale sharpness maps are the reduced reference information for the S4RR framework. Two measurements, the maximum and average differences, are calculated for latter use to assess image quality. From the three-scale sharpness maps, we compute 19 features to classify all distortion types into seven distortion families. The soft-classification results with probabilities are then employed to combine with two calculated measurements to yield the final S4RR quality index.

5.3.1 Reduced-reference information

This section first provides the method to collect the reduced-reference information from multi-scale sharpness maps of original and distorted images. We analyze the amount of reduced reference information (number of scalars versus image size) based on sharpness algorithm, block size, block overlapping, and image size. We also demon-

strate that the three-scale sharpness maps can provide an efficient summary of the original image and they are sensitive to a variety of distortion types via some examples. Two more scales of sharpness maps are obtained by applying sharpness algorithm to filtered and down-sampled image by a factor of two (via *imresize* in Matlab, which includes anti-aliasing filter with bicubic option).

A S_1, S_2, and S_3 sharpness maps

The S_3 algorithm was first presented in [93] and then further developed in [19]. S_3 claimed to be the first algorithm that could yield a sharpness map in which greater values denote sharper regions. Sharpness map S_3 is a combination of a spectral measure based on the slope of the local magnitude spectrum (S_1 map) and a spatial measure based on local maximum total variation (S_2 map).

In S_1 map, block of size 32 pixels and 24 pixels overlapping between neighboring block were applied. Therefore, this sharpness map has the sizes of 1/8 input image's. Using this sharpness map, the amount of reduced information for three scales is: $1/8^2 + 1/2^2/8^2 + 1/4^2/8^2 \approx 2.05\%$.

In S_2 map, block of size 8 pixels and 4 pixels overlapping between neighboring block were applied. Therefore, S_2 and S_3 sharpness maps have the sizes of 1/4 input image's. Using S_2 or S_3 sharpness maps, the amount of reduced information for all three scales is: $1/4^2 + 1/2^2/4^2 + 1/4^2/4^2 \approx 8.20\%$.

B FISH sharpness map

FISH sharpness map is a block-based version of Fast wavelet-based Image SHarpness algorithm [20]. A three-level discrete wavelet transform (DWT) were employed in FISH. The sharpness index for each block of size 16×16 is then given by a combination of the log-energies of blocks of size 8×8,4×4, and 2×2, respectively for three triplet subbands. FISH employs 50% overlap between neighboring blocks to

generate sharpness map. Therefore, FISH sharpness map is 1/8 size of the input image. Consequently, the amount of reduced information for all three scaled maps is: $1/8^2 + 1/2^2/8^2 + 1/4^2/8^2 \approx 2.05\%$.

C LSD map

The local standard deviation (LSD) has been employed previously in noise estimation [117], contrast enhancement [118], or texture analysis [119]. Here, we argue that LSD map is also a good candidate for sharpness map. The LSD map is formed locally by calculating the standard deviation for each block of size 12×12 (4 pixel overlapping between neighboring blocks) of the input image. As seen in Figure 5.1, LSD map is similar to S_2 map. The main difference between the two maps is on the texture areas such as the shoulder of the girl. Moreover, the correlation coefficient between LSD maps applied on the sharpness image database provided in [19] is high (CC > 0.90).

The LSD map has the size of $1/(12-4) = 1/8$ the input image. Therefore, the amount of reduced information for all three scaled maps is: $1/8^2+1/2^2/8^2+1/4^2/8^2 \approx 2.05\%$.

Figure 5.4 shows an example reduced information (sharpness maps) collecting process, in which maps are scaled for showing purpose and S, D denote the sharpness calculating and downsampling operations, respectively. We can see that three sharpness maps provide an efficient summary of the reference image. For example, the dark areas in the level 1, level 2, and level 3 maps indicate a smooth area in the sky of the input image. The brightest areas in the three maps point out the sharp edges around the building and the horse statue. Three maps also compensate each other for texture areas. For example, the texture in the flowers is captured by the corresponding dark-bright-mixed area in level 1 map, and gray-dark areas in level 2 and level 3 maps.

Figure 5.5 demonstrates that sharpness maps are sensitive to a variety of distortion

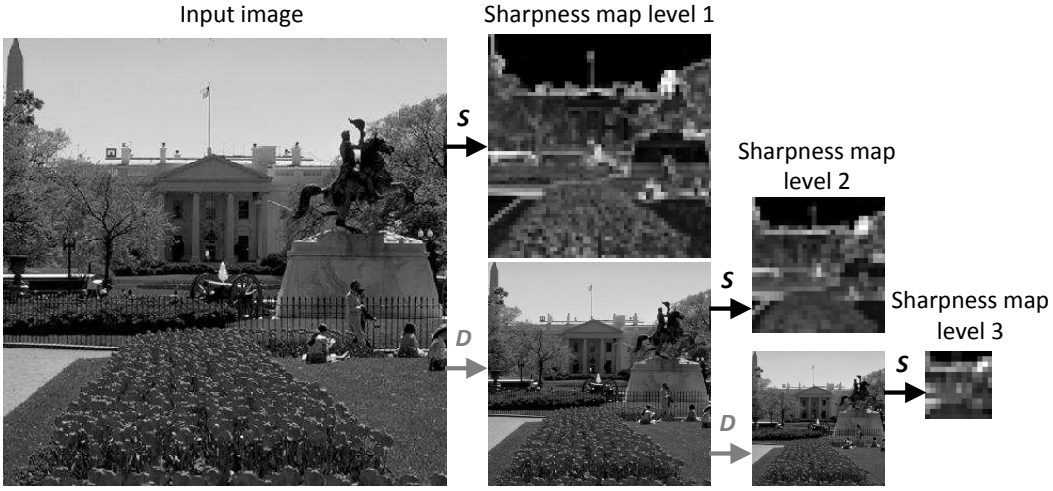


Figure 5.4: An example reduced information (sharpness maps) collecting process. (Maps are scaled for showing purpose and S, D denote the sharpness calculating and downsampling operations, respectively.) Three sharpness maps provide an efficient summary of the reference image. For example, the dark areas in the level 1, level 2 and level 3 maps indicate a smooth area in the sky of the input image. The brightest areas in the three maps point out the sharp edges around the building and the horse statue. Three maps also compensate each other for texture areas. For example, the texture in the flowers is captured by the corresponding dark-bright-mixed area in level 1 map, and gray-dark areas in level 2 and level 3 maps.

types. Four distorted images are shown in row a: AWGN, contrast-reduced, JPEG image, and JPEG2000 images, respectively. Four original and distorted LSD maps are shown in rows b and c for comparison (first scale only). We can clearly see that, for the AWGN image in column 1, the AWGN sharpness map in (c,1) is brighter than the original sharpness map in (b,1), due to the fact that white-noise makes sharpness values higher. Especially, the most change in the sharpness map is in the sky (the smoothest area). For contrast-reduced image in column 2, the distorted map (c,2) is darker than the original map (b,2) because sharpness values have been decreased. The distortion is captured mostly at the edges around the hat and wall. For the JPEG image in (a,1), the sky has been changed from one smooth area to several same-color areas. This is captured in original and distorted maps, (b,2) and (c,2): The dark area corresponding to the sky in (b,2) is changed either to black (due to quantization) or brighter (due to blocking artifact) in (c,2). The same pattern happens for the

JPEG2000 image in (a,4): The sea area is smooth (low frequency), and it is distorted to be either blurring or ringing (lower-area of (a,1)). This is also captured in the maps (b,2) and (c,2): The middle area corresponding to the sea is darker (due to blurring artifact) and the left area is whiter (due to ringing artifact).

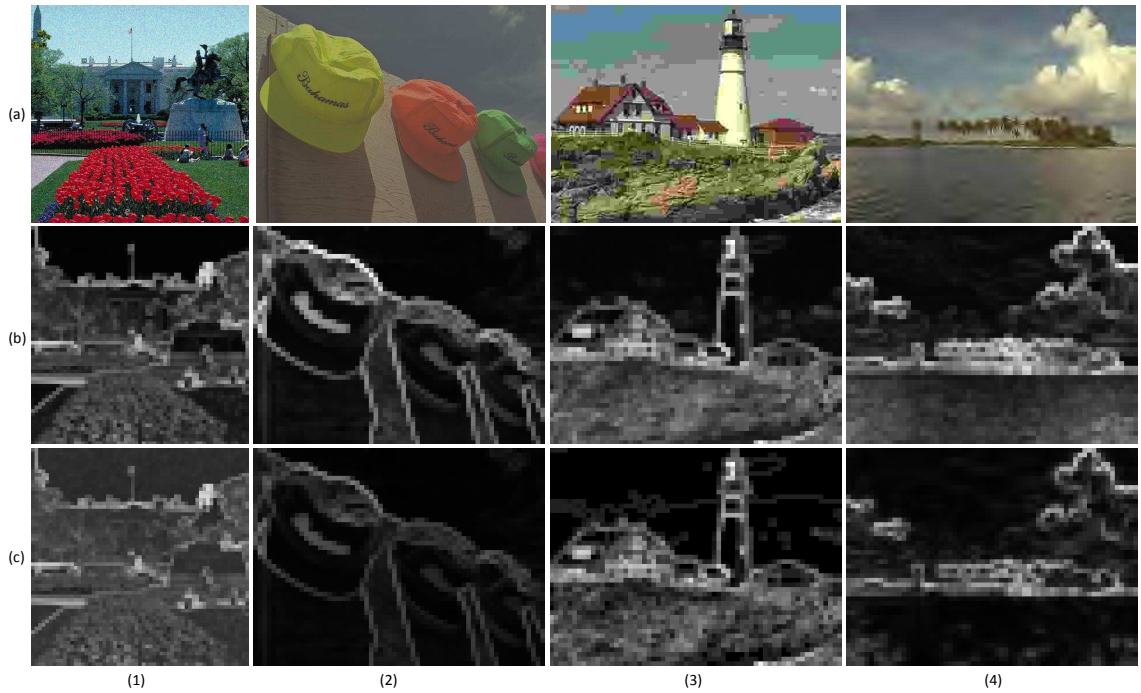


Figure 5.5: Four distorted images [row (a)]: AWGN, contrast-reduced, JPEG, and JPEG2000 images; first-scale original and distorted sharpness maps [rows (b) and (c)]. For the AWGN image in column 1, the AWGN sharpness map in (c,1) is brighter than the original sharpness map in (b,1), due to the fact that white-noise makes sharpness values higher. Especially, the most change in the sharpness map is in the sky (the smoothest area). For contrast-reduced image in column 2, the distorted map (c,2) is darker than the original map (b,2) because sharpness values have been decreased. The distortion is captured mostly at the edges around the hat and wall. For the JPEG image in (a,1), the sky has been changed from one smooth area to several same-color areas. This is captured in original and distorted maps, (b,2) and (c,2): The dark area corresponding to the sky in (b,2) is changed either to black or brightened (JPEG blocking artifact) in (c,2). The same pattern happens for the JPEG2000 image in (a,4): The sea area is smooth (low frequency), and it is distorted to be either blurring or ringing [lower-area of (a,1)]. This is also captured in the maps (b,2) and (c,2): The middle area corresponding to the sea is darker (blurring) and the left area is whiter (ringing).

5.3.2 Sharpness features

As shown in Section 5.3.1, sharpness maps are sensitive to a variety of distortion types. This section describes what the sharpness features are, and how to extract sharpness features to classify images into different distortion families.

The sharpness maps (reduced information) are calculated for both reference and distorted images. Let S^{ref} and S^{dst} denote the first scale sharpness map of reference and distorted images, respectively. As seen in figure 5.2, the shapes of scatterplots between distorted and original maps can capture the distortion types/intensities. Perfect quality image would have the shape lines perfectly on the red line as seen in figure 5.2 (*1c*). Looking at the scatterplots, the changes that distortions make to the shapes can be seen clearly in four areas of the x-axis: low values, mid values, high values, and very high values. Therefore, we divide each of sharpness maps S^{ref} and S^{dst} into four sub-maps based on the values of S^{ref}: low values, mid values, high values, and very high values maps (L^{ref}, M^{ref}, H^{ref}, and V^{ref} maps for S^{ref}, and L^{dst}, M^{dst}, H^{dst}, and V^{dst} maps for S^{dst}, respectively). The eight maps are given by:

$$L^{ref}(l) = S^{ref}(l), \tag{5.1}$$

$$L^{dst}(l) = S^{dst}(l), \qquad \forall l | S^{ref}(l) < T/6 \tag{5.2}$$

$$M^{ref}(m) = S^{ref}(m), \tag{5.3}$$

$$M^{dst}(m) = S^{dst}(m), \qquad \forall m | T/6 \leq S^{ref}(m) < T/3 \tag{5.4}$$

$$H^{ref}(h) = S^{ref}(h), \tag{5.5}$$

$$H^{dst}(h) = D^{dst}(h), \qquad \forall h | T/3 \leq S^{ref}(h) < T/2 \tag{5.6}$$

$$V^{ref}(v) = S^{ref}(v), \tag{5.7}$$

$$V^{dst}(v) = D^{ref}(v), \qquad \forall v | T/2 \leq S^{ref}(v) \tag{5.8}$$

where l, m, h, and v denote the indices of the S^{ref} map corresponding to low, mid,

high, and very high values, and T denote the maximum value that a sharpness algorithm can generate.

All 19 features, $f(1)$-$f(19)$, and their purposes are given in Table 5.1, where l, m, h, and v indices are obtained in eq. 5.1-5.8 . These 19 features are useful to classify an image into any distortion type. For example, an AWGN image will have a large $f(3)$ and $f(5)$, while $f(4)$ is close to zero; a JPEG image will have a large $f(10)$ and $f(13)$, $f(11) \times f(12) < 0$, and $f(13) \times f(14) < 0$; a JPEG2000 image will have the opposite sign of $f(8)$ and $f(9)$, high $f(15)$ and $f(18)$.

5.3.3 Distortion family classification

In this section, we provide the training process in TID2008 and results of soft-classification when classifying a distorted image into seven distortion families: (1) Spatially correlated broadband distortions, (2) Uniform White Noise, (3) High-frequency Noise, (4) Contrast reduction, (5) Contrast enhancement, (6) Blurring, and (7) Spatially localized distortions. These seven distortion families are chosen based on the tendency of the 19 features. For example, the Spatially correlated broadband distortions family contains both gain and lost of sharpness values; Uniform White Noise contains mainly the incremental in low frequency regions; the Spatially localized distortions family's sharpness maps are locally distorted; Contrast families include well correlated sharpness maps; and Blurring includes mainly the decrement in high frequency regions.

The training process requires only the distortion types in TID2008. The mappings between seven distortion families and 17 distortion types in TID2008 are provided in Table 5.2. Here, TID2008 contrast subset includes both contrast reduction and contrast enhancement images; therefore, we need two families for contrast distortion.

All 19 features are calculated for each image in TID2008 to prepare for the training procedure. We divide the database into two randomly no-overlapped subsets –

Table 5.1: All 19 features, their formulas and purposes.

Feature	**Formulas**	**Purpose/Good for:**
Linear correlation between two maps	$f(1) = corr(S^{ref}, S^{dst})$	High value of $f(1)$ suggests a high quality image or a contrast distorted image
Linear correlation between reference and error maps	$f(2) = corr(S^{ref}, (S^{dst} - S^{ref}))$	High values $f(1)$, $f(2)$ suggests contrast enhanced images
Maximum of the error map	$f(3) = max(S^{dst} - S^{ref})$	Sharpened if $f(3) + f(4) > 0$ Blurred if $f(3) + f(4) < 0$
Minimum of the error map	$f(4) = min(S^{dst} - S^{ref})$	
Average of the error map	$f(5) = mean(S^{dst} - S^{ref})$	Image appears to be sharpened if $f(5) > 0$ and appears to be blurred if $f(5) < 0$
Average of the increment in S^{ref} map	$f(6) = mean(S^{dst}(s) - S^{ref}(s))$ for all $S^{dst}(s) > R(s)$	(1) $f(6) \approx 0$ and $f(7) << 0$: blurred family. (2) $f(6) >> 0$ and $f(7) \approx 0$: Noise family.
Average of the decrement in S^{ref} map	$f(7) = mean(S^{dst}(s) - S^{ref}(s))$ for all $S^{dst}(s) < S^{ref}(s)$	(3) $f(6) >> 0$ and $f(7) << 0$: two-sided distortions
Average of the increment in L^{ref} map	$f(8) = mean(L^{dst}(l) - L^{ref}(l))$ for all $L^{dst}(l) > L(l)$	High value suggests a noise-added image.
Average of the decrement in L^{ref} map	$f(9) = mean(L^{dst}(l) - L(l))$ for all $L^{dst}(l) < L^{ref}(l)$	Low value for quantization in pixel domain in smooth area.
Number of zeros in L^{dst} map	$f(10) = count(L^{dst}(l) == 0)$	High value suggests JPEG, local intensity, or highly distorted image.
Average of the increment in M^{ref} map	$f(11) = mean(M^{dst}(m) - M^{ref}(m))$ for all $M^{dst}(m) > M^{ref}(m)$	Similar to $f(8)$
Average of the decrement in M^{ref} map	$f(12) = mean(M^{dst}(m) - M^{ref}(m))$ for all $M^{dst}(m) < M^{ref}(m)$	Similar to $f(9)$
Number of zeros in M^{dst} map	$f(13) = count(M^{dst}(m) == 0)$	Similar to $f(10)$
Average of the increment in H^{ref} map	$f(14) = mean(H^{dst}(h) - H^{ref}(h))$ for all $H^{dst}(h) > H^{ref}(h)$	Similar to $f(8)$
Average of the decrement in H^{ref} map	$f(15) = mean(H^{dst}(h) - H^{ref}(h))$ for all $H^{dst}(h) < H^{ref}(h)$	Similar to $f(9)$
Number of zeros in H^{dst} map	$f(16) = count(H^{dst}(h) == 0)$	Similar to $f(10)$
Average of the increment in V^{ref} map	$f(17) = mean(V^{dst}(v) - V^{ref}(v))$ for all $V^{dst}(v) > V^{ref}(v)$	Similar to $f(8)$
Average of the decrement in V^{ref} map	$f(18) = mean(V^{dst}(v) - V^{ref}(v))$ for all $V^{dst}(v) < V^{ref}(v)$	Similar to $f(9)$
Number of zeros in V^{dst} map	$f(19) = count(V^{dst}(v) == 0)$	Similar to $f(10)$

80% training and 20% testing. We employed a support vector machine regressor to perform this training process by utilizing the LIBSVM package [81]. Furthermore, this randomizing train-test procedure is repeated 1000 times to avoid performance bias.

Table 5.2: Mappings between seven distortion families and 17 distortion types in the TID2008 database

Id	Distortion families	Distortion types
1	Spatially Correlated Broadband Distortions	JPEG, JPEG2000, CorrNoise
2	Uniform White Noise	AWGN, ColorNoise
3	High-frequency Noise	HiNoise
4	Contrast reduction	CNT_dst, MeanShift
5	Contrast enhancement	CNT_enh
6	Blurring	BLUR, Denoise
7	Spatially localized distortions	JPPackageErr, JP2KPackageErr, ImpulseNoise, QuanNoise, Local, MaskedNoise, NEPNoise

Overall, the median classification accuracy reaches 97.58% for 1700 images in the TID2008 database if we employ LSD sharpness. For all other sharpness algorithms, the classification accuracies are from 95% to 98%. Figure 5.6 provides the accuracy (%) of LSD sharpness, where the sum of each row in the matrix is 100%, and each value represents the average of 1000 train-test trials. From the table, we can see that White Noise and High-Frequency Noise are the most confused distortion types: 9.41% White Noise images were classified to High-Frequency Noise, and 13.03% High-Frequency Noise images were classified to Uniform White Noise. All the other distortion types have correction rates greater than 95%, including two contrast subsets, which are 49.96% and 49.83%, respectively.

5.3.4 Maximum and average difference measurements

From three pairs of sharpness maps, we employ a simple masking model using a sigmoid to calculate the different maps. This sigmoid takes into account the low-value ($0-T/6$), mid-value ($T/6-T/3$), and high-value ($\geq T/3$) of reference sharpness maps,

	Broadband	White noise	Hi-noise	Cnt_dec	Cnt_inc	Blurring	Local
AWGN	0.29	**90.30**	9.41	0.00	0.00	0.00	0.00
ColorNoise	0.24	**98.25**	1.35	0.00	0.00	0.00	0.16
CorrNoise	**99.07**	0.50	0.02	0.00	0.00	0.00	0.42
MarkedNoise	0.05	0.02	0.00	0.00	0.00	0.01	**99.93**
HiNoise	0.20	13.03	86.74	0.00	0.00	0.00	0.03
ImpulseNoise	0.00	0.34	0.01	0.00	0.00	0.00	**99.65**
QuanNoise	0.86	0.01	0.00	0.00	0.00	0.05	**99.08**
Blur	1.10	0.00	0.00	0.00	0.00	**98.84**	0.06
Denoise	3.31	0.00	0.00	0.01	0.00	**95.96**	0.72
JP	**98.52**	0.00	0.00	0.00	0.00	0.21	1.27
JP2K	**95.91**	0.00	0.00	0.00	0.00	3.59	0.51
JPpackageErr	0.13	0.00	0.00	0.00	0.00	0.05	**99.82**
JP2KPackageErr	0.82	0.06	0.00	0.00	0.00	0.00	**99.12**
NEPNoise	0.60	0.00	0.00	0.00	0.00	0.04	**99.36**
Local	0.40	0.00	0.00	0.00	0.00	0.01	**99.59**
MeanShift	0.19	0.19	0.08	0.01	0.00	0.64	**98.89**
CNT	0.00	0.00	0.00	49.96	49.83	0.04	0.17

Figure 5.6: TID2008 accuracy matrix (%) for classifier across 1000 trials.

where the differences in lower sharpness values are more sensitive than the difference in higher values. Specifically, the sigmoid is given by:

$$\begin{aligned} Sig(x) = 1 + g_1 - g_1/(1 + \exp(t_1 \times (x - t_2))) \\ + g_2 - g2/(1 + \exp(t_3 \times (x - t_4))) \end{aligned} \tag{5.9}$$

where $t_1 = -0.8$, $t_3 = -0.5$ are two slopes, $t_2 = 20$, $t_4 = 40$ are two cut-offs, and $g_1 = g_2 = 10$ are the gains of the sigmoid. The values $-0.8, -0.5, 10, 20, 40$ are chosen empirically, and for the best fit to the TID2008 image database [3], assuming sharpness value x is in the range of $0 - 120$.

The differences between the reference and distorted maps, P_i, after applying sigmoid are then given for three scales, $i = 1, 2, 3$, by:

$$P_i = (R_i - D_i) \times Sig(R_i) \tag{5.10}$$

where the sigmoid function is applied for the reference maps to highlight the difference

in areas that have the lower sharpness values.

From three different maps, P_i, we calculate two measurements for latter use to assess quality: The average of three maxima, and the average of L_2-norms of three different maps. Specifically, two measurements are given by:

$$Me_1 = \frac{1}{3} \times (\max|P_1| + \max|P_2| + \max|P_3|) \tag{5.11}$$

$$Me_2 = \frac{1}{3} \times \left(\sum_{i=1}^{3} \left(\frac{1}{N_i} \sum_{N_i} P_i^2 \right)^{1/2} \right) \tag{5.12}$$

where the summation is over all i^{th} map's values and N_i is the number of scalars in i^{th} map.

The average of three maxima, Me_1, is reasonable for uniform-distortion families, such as Spatially correlated broadband distortion and Uniform White Noise families. Whereas the average of three L_2-norms, Me_2, is mainly used for nonuniform-distortion families, such as blurring and spatially localized distortions. Figure 5.7 demonstrates Me_1 and Me_2 for a variety of image distortions. Four distortion types are AWGN, contrast-reduction, JPEG package error, and JPEG2000 package error. For uniform distorted images [i.e. images (a) and (b)], both Me_1 and Me_2 are reasonable. However, for locally distributed distortion [i.e. image (c) and (d)], Me_1 values, which are the average of three maxima of three scales, are too large. The Me_2 values, which are the average of three L_2-norm, are better representations of image qualities.

5.3.5 Combining measures and distortion families adaptively

This section provides the last stage of S4RR framework (as shown in Section 5.3, Figure 5.3), which adaptively combines the maximum and average differences into a final quality index based on the distortion family classification. One scalar is selected to combine with one measurement based on the distortion family. Since it is soft classification, the probability of each distortion family is provided. The final quality

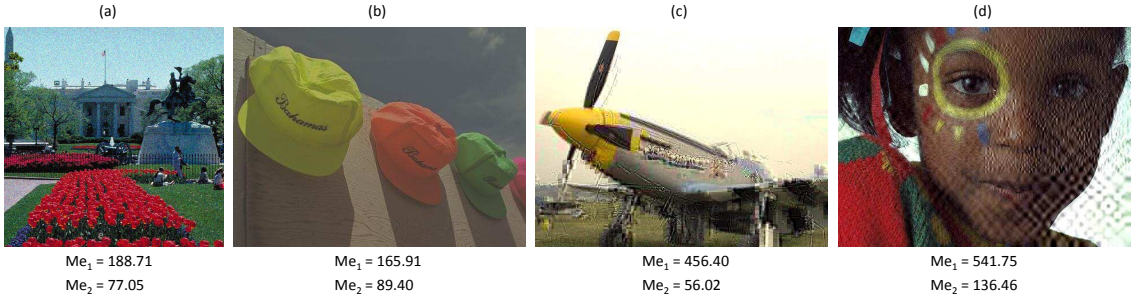


Figure 5.7: Me_1 Me_2 demonstration. For uniform distorted images [i.e. images (a) and (b)], both Me_1 and Me_2 are reasonable. However, for locally distributed distortion [i.e. image (c) and (d)], Me_1 values, which are the average of three maxima of three scales, are too large. The Me_2 values, which are the average of three L_2-norm, are better representations of image qualities.

index, $\mathrm{S4RR}_X$ (where $X \in \{S_1, S_2, S_3, F, L\}$ for S_1, S_2, S_3, FISH, or LSD sharpness algorithms, respectively), is the weighted summation between probabilities, scalars, and measurements. Specifically, $\mathrm{S4RR}_X$ is given by:

$$\mathrm{S4RR}_X = \sum_{i=1}^{7} Pr(i) \times [k_1(i) \times Me_1 + k_2(i) \times Me_2] \tag{5.13}$$

where $Pr(i)$ is the probability that the distorted image belongs to the i^{th} distortion family. Two scalar vectors k_1 and k_2 of length seven are adaptively selected based on the distortion family. The seven values of k_1, k_2 are provided in Table 5.3.

Table 5.3: k_1 and k_2 vectors. The algorithm adaptively selects scalar based on distortion families.

Id	Distortion families	k_1	k_2
1	Spatially correlated broadband distortions	1	0
2	Uniform White Noise	1/1.25	0
3	High-frequency noise	1/1.75	0
4	Contrast reduction	1/2.5	0
5	Contrast enhancement	-1/5	0
6	Blurring	0	2.25
7	Spatially localized distortions	0	2.5

The values of k_1 and k_2 in Table 5.3 are chosen empirically, and for the best fit to the TID2008 image database [3]. However, it is reasonable to come up with

these values for two main reasons: First, from the properties of L_2-norm and the maximum operation, Me_2 is expected to be 2.5 times less than Me_1, and thus k_2 is expected to be 2.5 times greater than k_1. Second, within the uniform distortion families, $Id = 1 - 4$, distortion families are sorted based on their effects on the high frequency areas. Since the distortions in the smooth areas (low frequency) are more sensitive than in the edge areas (high frequency), the smaller k_1 values are chosen for the greater group index. The algorithm can detect a contrast-enhanced image, therefore $k_1(5)$ is negative. A negative quality index implies an enhanced image, and zero means perfect quality image.

5.4 Results and discussion

5.4.1 Correlation with human opinions

We tested the performance of the S4RR$_X$ algorithm using distortion-family-training database, TID2008 [3], which consists of 25 reference images and 1700 distorted images spanning 17 distortion types (*JPEG*, *JPEG2000*, *CorrNoise*, *BLUR*, *Denoised*, *AWGN*, *ColorNoise*, *MaskedNoise*, *JPpackageErr*, *ImpulseNoise*, *QuanNoise*, *Local*, *JP2KPackageErr*, *HiNoise*, *Contrast*, *MeanShift*, and *NEPNoise*). The results reported here were the average numbers of 1000 trials. Each trial, with 80% training and 20% testing data points, has been randomly selected to eliminate performance bias.

The Spearman rank ordered correlation coefficient (SROCC) and Pearson correlation coefficient (CC) between the predicted score from the algorithm and MOS values were employed to access performance. A logistic transform was employed before computing SROCC and CC scores (see [19]). Scores are in the range of [0-1], where higher values denote better performance and ones are for perfect prediction.

For comparison, we also calculated the performances of one classic FR IQA algorithm: peak signal-to-noise ratio (PSNR); four state-of-the-art FR IQA algorithms:

Table 5.4: CC and SROCC scores of S4RR and other algorithms in the TID2008 database. Best performances are bolded, italic entries are second-best performances, and italic algorithms are full-reference.

	CC	SROCC
PSNR	0.5355	0.5245
MS-SSIM	0.8390	0.8528
VIF	0.8055	0.7496
MAD	0.8306	0.8340
FSIM	*0.8710*	*0.8805*
RRED	0.8252	0.8237
SPCRM	0.8509	0.8325
S4RR$_{S_1}$	0.7312	0.7467
S4RR$_{S_2}$	0.8584	0.8601
S4RR$_{S_3}$	0.7970	0.8046
S4RR$_F$	0.6937	0.7048
S4RR$_L$	**0.8809**	**0.8915**

multi-scale structural similarity index (MS-SSIM) [14], visual information fidelity (VIF) [15], most apparent distortion (MAD) [7], and feature similarity index (FSIM - grayscale version) [90]; and two state-of-the-art RR IQA frameworks: reduced reference entropic differencing (RRED) [88] and regularity of phase congruency metric (SPCRM) [89]. Notice that, in terms of *reduced reference*, S4RR$_X$ is using 2.05% ($X = S_1, F$, or L), or 8.20% number of scalars ($X = S_2$ or S_3), whereas RRED and SPCRM use 2.8% and 3.1% number of scalars, respectively.

Table 5.4 summarizes the performance of S4RR and other algorithms in the TID2008 database. In terms of both CC and SROCC, S4RR$_L$ outperformed all other algorithms with $CC = 0.8809$ and $SROCC = 0.8915$. The second-best algorithm, FSIM, which is a FR IQA algorithm, could reach $CC = 0.8710$ and $SROCC = 0.8805$ only. S4RR$_{S_2}$ and SPCRM are two RR algorithms, which are better than other FR IQA algorithms, except FSIM.

In addition, scatterplots of TID2008 MOS versus four algorithms' linearized scores, MS-SSIM, MAD, FSIM, and S4RR$_L$-the best version of S4RR framework, are provided in Figure 5.8. We can see that all four scatterplots have outliers in the low-range

MOS values (low quality images - $MOS < 1$). However, the main differences are in the high-quality range, $MOS > 6$. Because the TID2008 image database has 50 contrast enhancement images, these images appear to be high quality. MS-SSIM, MAD, and FSIM failed to predict these images' scores. Only S4RR$_L$ classified these images into the right distortion family, and obtained a better result. However, S4RR framework does not take into account the over-increased images and color changing (see [120]); therefore, there are still a few outliers in the high-quality range. To conclude, these scatterplots show that S4RR$_L$ outperformed the other state-of-the-art FR-IQA metrics.

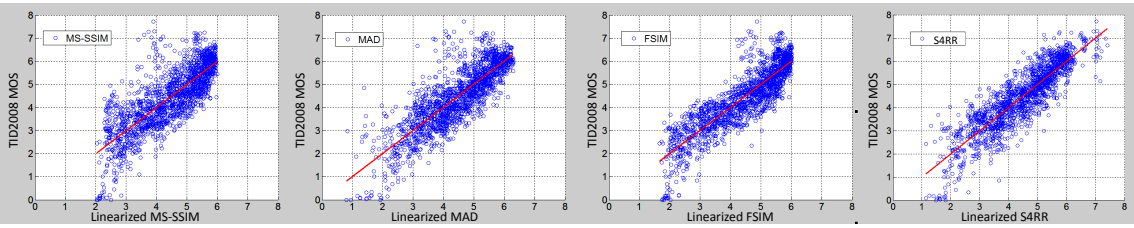


Figure 5.8: Scatterplots of TID2008 MOS versus four algorithms' linearized scores, MS-SSIM, MAD, FSIM, and S4RR$_L$. S4RR$_L$ outperformed the other FR-IQA metrics with less outliers in the high-quality range, $MOS > 6$.

The SROCC scores of S4RR$_X$ and other algorithms on 17 individual distortion types are given in Table 5.5, where bold entries denote the best performances. The last row shows the number of times the SROCC was above 0.92. We can see that among 17 distortion types, S4RR$_L$ has the highest SROCC scores four times, while the others have three times or less. On the last row, S4RR$_L$'s SROCC was above 0.92 six times, while the others were four times or less. Again, S4RR$_L$ shows a very good performance not only on the whole database of 1700 images, but also on each individual distortion type.

5.4.2 Classification Accuracy

Although classification accuracy is not our main goal, it is useful to know how good S4RR$_L$, the best version of S4RR framework, can classify different distortion types in

Table 5.5: The SROCC scores of S4RR$_X$ and other algorithms on 17 distortion types of the TID2008 database. Bold entries denote the best performance. The last row shows the number of times the SROCC was above 0.92.

	PSNR	MSSSIM	VIF	MAD	FSIM	RRED	SPCRM	S4RR$_L$
AGWN	0.911	0.809	0.880	0.863	0.857	0.820	0.815	**0.930**
JPEG	0.901	0.935	0.917	**0.941**	0.928	0.933	0.917	0.927
JPEG2000	0.830	0.974	0.971	0.972	**0.977**	0.968	0.959	0.956
Gaussian Blur	0.868	**0.961**	0.955	0.914	0.947	0.957	0.912	0.848
Contrast	0.613	0.640	0.819	0.492	0.648	0.679	0.076	**0.834**
JPEG Package Err	0.766	0.874	0.858	0.851	0.871	0.871	**0.887**	0.872
JPEG2000 Pkg. Err	0.777	0.852	0.851	0.840	**0.854**	0.742	0.848	0.836
Quantization Noise	0.870	0.854	0.796	0.850	0.856	0.831	0.855	**0.876**
Denoised	0.938	0.957	0.919	0.945	**0.960**	0.949	0.948	0.937
Color Noise	**0.907**	0.806	0.878	0.839	0.853	0.850	0.799	0.886
Corr Noise	0.923	0.820	0.870	0.898	0.848	0.842	0.829	**0.923**
Masked Noise	0.849	0.816	**0.870**	0.736	0.803	0.833	0.760	0.782
High Freq. Noise	**0.932**	0.868	0.907	0.897	0.909	0.909	0.883	0.920
Impulse Noise	**0.918**	0.687	0.883	0.512	0.745	0.741	0.676	0.677
NEP Noise	0.593	0.734	0.761	**0.838**	0.749	0.713	0.819	0.777
Local Block	0.585	0.762	0.832	0.161	0.849	0.824	**0.849**	0.088
Mean Shift	0.697	**0.737**	0.513	0.589	0.672	0.538	0.714	0.587
#>0.92	3	4	2	3	4	4	2	6

	Broadband	White noise	Hi-noise	Cnt_dec	Cnt_inc	Blurring	Local
LIVE/JP	**85.09**	0.01	0.00	0.00	0.00	0.20	14.70
LIVE/JP2K	**85.60**	0.00	0.00	0.00	0.00	7.63	6.77
LIVE/Blur	6.22	0.00	0.00	0.00	0.00	**93.65**	0.12
LIVE/Noise	13.98	45.51	26.94	0.00	0.46	0.00	13.11
LIVE/Ray	**71.95**	0.02	0.00	0.00	0.00	10.41	17.62
CSIQ/Contrast	0.18	0.02	0.00	**87.70**	0.00	5.61	6.49
CSIQ/JP	**70.68**	2.64	0.05	0.00	0.00	0.23	26.41
CSIQ/JP2k	**83.60**	0.11	0.00	0.00	0.00	7.25	9.04
CSIQ/Blur	14.98	0.00	0.00	0.00	0.00	**84.40**	0.62
CSIQ/AWGN	18.26	**65.40**	8.39	0.00	0.00	0.00	7.94
CSIQ/Fnoise	67.11	1.13	0.00	0.00	0.00	0.15	31.61
TID2013/Color Saturation	3.02	0.32	1.22	3.63	3.75	0.34	**87.72**
TID2013/Multiplicative Gaussian noise	16.40	2.24	34.25	4.94	15.38	24.78	2.01
TID2013/Comfort noise	26.73	0.29	1.11	4.57	18.09	36.86	12.35
TID2013/Lossy compression of noisy images	25.31	0.92	3.58	11.61	18.43	33.62	6.52
TID2013/Quantization with dither	12.46	2.21	10.20	10.55	18.12	18.98	27.47
TID2013/Chromatic aberrations	8.60	2.39	9.23	29.75	23.77	24.84	1.42
TID2013/Sparse sampling and reconstruction	17.53	1.27	4.89	19.82	13.24	41.43	1.82

Figure 5.9: The accuracy (%) of classifying images in LIVE and CSIQ, and seven new distortion types in TID2013 into distortion families across 1000 trials.

the LIVE [1], CSIQ [2], and TID2013 [4] image databases into the distortion families. Besides nine new distortion types in LIVE and CSIQ (RAY - fast fading channel [121] in LIVE, PinkNoise -1/Fnoise in CSIQ [2], and seven new distortion types in TID2013 [4]), it is also helpful to understand how classification accuracy affects the prediction performances on these databases (see Section 5.4.3). The accuracy (%) of classifying images in LIVE and CSIQ, and seven new distortion types in TID2013 (since TID2013 has seven new distortion types comparing to TID2008), are provided in Figure 5.9, where the sum of each row in the confusion matrix is 100 (%), and each value represents the average of 1000 trials.

From Figure 5.9, for LIVE subsets, we can see most of the time, the algorithm classified correctly, e.g., 85.09%, 85.60%, and 93.65% for JPEG, JPEG2000, and Blur

subsets, respectively. For of RAY (fast fading) images, 71.95% were classified to the *Spatially Correlated Broadband Distortions* family, 10.41% to *Blurring*, and 17.62% to *Local*. This is reasonable because the algorithm has not been trained on Rayleigh fading channel distortion, which is based on the JPEG2000 scheme (JPEG Bit Error), and plus, highly compressed JPEG2000 can introduce blurring.

Figure 5.9 also shows two failure cases on two AWGN subsets: only 45.51% of Live/Noise and 65.40% of CSIQ/AWGN were classified correctly to *Uniform White Noise*. This is explainable: because the distortion level for the AWGN images in TID2008 is not as wide as in LIVE and CSIQ. Specifically, the AWGN subset of TID2008 has the PSNR in range of $24.44 - 34.29$ dB. Whereas in the LIVE and CSIQ databases, PSNR of AWGN images are in the range of $7.87 - 41.93$ dB and $23.01 - 41.45$ dB, respectively.

For seven new distortion types in TID2013 showed in Figure 5.9, 87.72% of *Color Saturation* were classified to the *Local* family. This is acceptable, because the S4RR does not take into account color information, and thus S4RR treated all *Color Saturation* images as high quality images. The other six distortion types span all seven distortion families. Except the *Color Saturation* images, the performances of S4RR$_L$ on these six distortion types are still very good as will be demonstrated in Section 5.4.3.

5.4.3 Database Independence

The S4RR$_L$ algorithm, the best version of S4RR framework, has been trained using the TID2008 distortion types, and S4RR$_L$ has shown an excellent performance in this database. We now demonstrate that the performance of S4RR$_L$ is not bound by the trained database. The LIVE [1], CSIQ [2], and TID2013 [4] databases are employed to evaluate the S4RR$_L$'s prediction performance.

There were some limitations in the distortion-family classification results of LIVE

Table 5.6: CC and SROCC of S4RR$_L$ and other algorithms in LIVE, CSIQ, and TID2013, and the average weighted by the number of images each database. Bold entries denote the best performance for each row. Italic algorithms are full-reference algorithms. There are 779, 866, and 3000 images in LIVE, CSIQ, and TID2013, respectively. CC scores are in the first haft of the table.

CC	*PSNR*	*MSSSIM*	*VIF*	*MAD*	*FSIM*	RRED	SPCR	S4RR$_L$
LIVE	0.8707	0.9330	0.9595	**0.9683**	0.9539	0.9337	0.9412	0.9468
CSIQ	0.7998	0.8972	0.9252	0.9502	0.9095	0.9079	0.9464	**0.9557**
TID2013	0.6726	0.8341	0.7468	0.8221	**0.8568**	0.8164	0.8180	0.8506
Average	0.7295	0.8625	0.8157	0.8705	0.8829	0.8531	0.8626	**0.8863**
SROCC	*PSNR*	*MSSSIM*	*VIF*	*MAD*	*FSIM*	RRED	SPCR	S4RR$_L$
LIVE	0.8763	0.9437	0.9633	**0.9675**	0.9634	0.9429	0.9444	0.9484
CSIQ	0.8056	0.9137	0.9192	0.9466	0.9242	0.9184	0.9410	**0.9567**
TID2013	0.6395	0.7872	0.6085	0.7807	0.8007	0.7632	0.7559	**0.8155**
Average	0.7102	0.8370	0.7259	0.8430	0.8510	0.8223	0.8220	**0.8641**

and CSIQ (presented in Section 5.4.2). This is because some subsets TID2008 have limited distortion levels and distortion types, on which the algorithm has not been trained. To build the classification accuracy matrix, we had to hard classify into distortion families. However, the algorithm actually does not need to perform the hard classification, and therefore, the overall performance does not drop dramatically, as shown in Table 5.6 and 5.7.

Table 5.6 shows the performances of S4RR$_L$ and other algorithms on the entire set of images from the LIVE, CSIQ, and TID2013 databases using CC and SROCC. Also shown in Table 5.6 is the average performance of three databases weighted by the number of images in each database. Bold entries denote the best performance for each row. Italic algorithms are FR algorithms. We can see that in LIVE, MAD is the best algorithm with $CC = 0.9683$ and $SROCC = 0.9675$; and FSIM is the best for CC score on TID2013. For all the other criteria of the other databases and on average, S4RR$_L$ shows the best performances with $CC/CSIQ = 0.9557$, $CC/Average = 0.8506$, $SROCC/CSIQ = 0.9567$, $SROCC/TID2013 = 0.8155$, and $SROCC/Average = 0.8641$.

Table 5.7 provides SROCC scores of S4RR$_L$ and other algorithms for each indi-

Table 5.7: SROCC scores of S4RR$_L$ and other algorithms for six distortion types in CSIQ, five distortion type in LIVE, and 24 distortion types in TID2013 with bold entries for the best performance. Italic algorithms are full-reference algorithms. The last row of the table also shows the number of times that the SROCC was above 0.95.

		PSNR	*MSSSIM*	*VIF*	*MAD*	*FSIM*	RRED	SPCRM	S4RR$_L$	# imgs
Awgn	CSIQ	0.936	0.947	0.957	**0.960**	0.926	0.935	0.919	0.958	150
	LIVE	**0.985**	0.973	0.985	0.971	0.965	0.978	0.973	0.983	145
	TID2013	0.929	0.874	0.849	0.884	0.902	0.852	0.874	**0.951**	125
Blur	CSIQ	0.929	0.972	**0.975**	0.966	0.973	0.963	0.969	0.963	150
	LIVE	0.781	0.959	**0.973**	0.899	0.971	0.968	0.959	0.922	145
	TID2013	0.915	**0.971**	0.964	0.932	0.957	0.967	0.941	0.909	125
Jpeg	CSIQ	0.888	0.962	**0.970**	0.966	0.965	0.952	0.956	0.967	150
	LIVE	0.881	0.979	**0.984**	0.949	0.983	0.976	0.975	0.973	175
	TID2013	0.919	0.931	0.916	0.922	0.930	0.927	0.916	**0.935**	125
Jpeg2000	CSIQ	0.936	0.969	0.967	**0.977**	0.969	0.963	0.958	0.971	150
	LIVE	0.895	0.965	0.969	0.938	**0.972**	0.958	0.957	0.965	169
	TID2013	0.884	0.950	0.941	0.951	**0.958**	0.954	0.946	0.955	125
Contrast	CSIQ	0.862	0.952	0.936	0.917	0.942	0.938	0.924	**0.952**	116
	TID2013	0.441	0.468	0.833	0.407	0.472	0.643	0.245	**0.874**	125
JP2KBitError	LIVE	0.893	0.930	**0.965**	0.883	0.950	0.943	0.928	0.927	145
JP2KPackageErr	TID2013	0.888	0.889	0.873	0.879	**0.895**	0.790	0.879	0.891	125
1/fnoise	CSIQ	0.934	0.933	0.951	**0.954**	0.923	0.936	0.923	0.926	150
ColorNoise	TID2013	**0.898**	0.781	0.794	0.802	0.823	0.789	0.783	0.871	125
CorrNoise	TID2013	0.920	0.866	0.862	0.891	0.879	0.852	0.865	**0.956**	125
MarkedNoise	TID2013	0.831	0.813	**0.833**	0.738	0.800	0.804	0.763	0.780	125
HiNoise	TID2013	0.914	0.869	0.868	0.888	0.898	0.896	0.884	**0.916**	125
ImpulseNoise	TID2013	**0.897**	0.775	0.834	0.277	0.811	0.793	0.749	0.753	125
QuanNoise	TID2013	**0.878**	0.874	0.761	0.851	0.872	0.828	0.875	**0.878**	125
Denoise	TID2013	**0.948**	0.929	0.900	0.925	0.931	0.923	0.923	0.932	125
JPpackageErr	TID2013	0.768	0.849	0.830	0.828	0.845	0.847	**0.854**	0.842	125
NEPNoise	TID2013	0.686	0.794	0.770	**0.832**	0.789	0.781	0.811	0.773	125
Local	TID2013	0.154	0.479	0.534	0.510	0.555	0.549	**0.565**	0.190	125
MeanShift	TID2013	0.766	0.775	0.591	0.645	0.730	0.614	**0.781**	0.600	125
Color Saturation	TID2013	**0.545**	0.515	0.534	0.362	0.515	0.046	0.325	0.463	125
MGaussian Noise	TID2013	**0.890**	0.800	0.785	0.841	0.854	0.795	0.842	0.846	125
Comfort noise	TID2013	0.841	0.862	0.834	0.906	0.913	0.904	0.906	**0.925**	125
Compressed of Noisy	TID2013	0.914	0.914	0.893	0.944	0.948	0.920	0.938	**0.962**	125
Color quantization	TID2013	**0.927**	0.865	0.781	0.874	0.877	0.865	0.846	0.857	125
Chromatic aberration	TID2013	0.888	0.885	**0.892**	0.831	0.874	0.891	0.852	0.852	125
Sparse sample/rec	TID2013	0.904	0.950	0.925	**0.957**	**0.957**	0.954	0.954	0.939	125
	#>0.95	1	11	11	8	10	10	8	12	

vidual distortion type with bold entries for the best performance. The last row of the table also shows the number of times that the SROCC was above 0.95. Again, the S4RR$_L$ algorithm, using 2.05% of the reduced information, shows an good performance with nine bold entries and 12 times SROCC at above 0.95, whereas RRED (2.8% number of scalars), SPCRM (3.1% number of scalars), and other FR metrics reaches eight bold entries and 11 times SROCC at above 0.95 or less.

Scatterplots of DMOS/MOS versus linearized S4RR$_L$ can be seen in Figure 5.10

for LIVE, CSIQ, and TID2013. We can see that there are several outliers on the LIVE database in the low DMOS range, on the CSIQ database in the high DMOS range, and on TID2013 low MOS range. This indicates that S4RR$_L$ receives significant influence from the mis-classification results (described in Section 5.4.2). We believe if there is a better database to train on distortion types/families, S4RR$_L$ can perform better.

There are a fair number of outliers on the scatterplot of TID2013 in Figure 5.10 for the S4RR$_L$ scores around 5.8. These outliers arise from *Color Saturation* subset in the database, because *Color Saturation* distortion images, when converted to grayscale, appear to be high quality images.

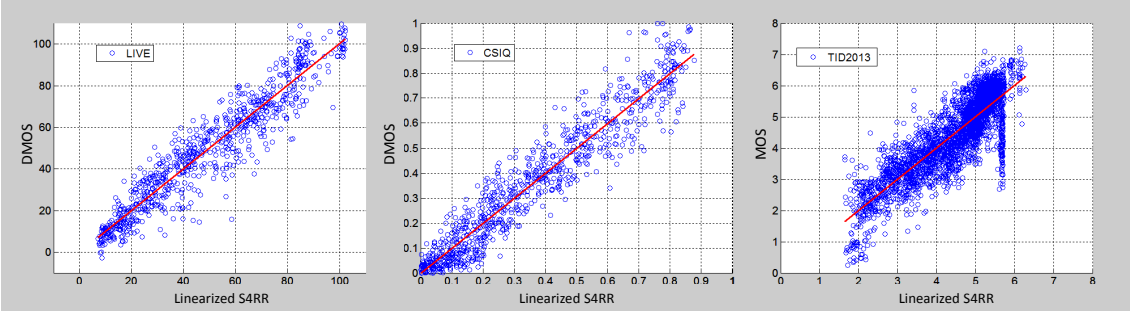


Figure 5.10: Scatterplots of LIVE, CSIQ, and TID2013 databases DMOS versus linearized S4RR$_L$ scores.

5.5 Summary

This work presented a new reduced-reference IQA framework, S4RR, based on multiscale sharpness maps. S4RR extracts 19 features from three-scale sharpness maps to classify the distorted image into seven distortion families. S4RR then adaptively switches among seven scalars, combining with the average of three maxima and three L_2-norms to assess quality. We provided the training process on TID2008's (17 distortion types into seven distortion families) and the excellent performances on this database. S4RR$_L$, the best version of S4RR with 2.05% reduced information, was able to outperform all state-of-the-art algorithms, including full-reference and other

reduced-reference frameworks (2.8%-3.1% reduced information). We also demonstrated that the chosen seven distortion families were efficient to assess quality by performing analysis on LIVE, CSIQ, and TID2013 databases, which include new distortion types (six distortion types in CSIQ, including one new distortion type; five distortion types in LIVE, including one new distortion type; and 24 distortion types in TID2013, including seven new distortion types.). The new distortion types raised some small issues with classification accuracy. However, the results on LIVE, CSIQ, and TID2013 showed that S4RR was able to outperform all other metrics on most of the criteria.

CHAPTER 6

CONCLUSIONS

Two main sections will be provided in this chapter. The section *Summary and contribution* briefly summarizes the work and its contributions. The section *Future work* provides some directions that other researchers and I can go on.

6.1 Summary and Contributions

We have explored four areas in IQA research: local quality (local sharpness), IQA for improved JPEG2000 coding, microarchitectural analysis of IQA algorithms, and a reduced reference IQA framework.

The study on local sharpness suggested that a sharp block does not need to be an edge block. (Many algorithms tried to calculate the edge's characteristics to measure sharpness). A local sharpness map database was provided in the report. The analysis results claimed that an algorithm, which was good at global sharpness prediction, could fail to predict local sharpness on many blocks. The database was made available online for research community.

We performed analysis the database, and discovered that human subjects are highly agreed with other in terms of sharpness. In general, the blocks which include edges or textures often get rated sharper. Blank blocks and smooth blocks are generally rated not as sharp. However, the ratings are also biased by neighboring blocks.

Most of the algorithms were not able to predict the sharpness of local blocks. We modified those algorithms for them to measure small blocks with overlap between

neighboring blocks. Only a few algorithms were designed specifically for local sharpness. However, no algorithm could reach over 0.87 in terms of SROCC. From the analysis results, we could see that the database is truly a representation of human opinions.

The study on IQA for improved JPEG2000 coding revealed that an IQA algorithm which performed reasonably well on standard-MSE JPEG2000-compressed images in popular image-quality databases (e.g., LIVE, CSIQ, TID2008, TID2013) does not guaranty a good result in coding context. A new database for JPEG2000 images was provided with a new algorithm on DWT domain. The new algorithm outperformed all other IQA algorithms on predicting the ranking of the image in our database.

We presented an image database designed specifically for JPEG2000 compression with a fixed amount of total distortion, but in which the distortions were allocated to different frequency bands in different proportions. The result from subjective ranking experiment revealed that, in general, higher-quality images were achieved by allocating most of the distortion to the finest scale and the least to the coarse-to-mid scales.

We also provided an analysis of existing IQA algorithms on this database which revealed that even though the algorithms perform reasonably well on JPEG2000-compressed images in popular image-quality databases, they often fail to predict the correct rankings on the images here.

We proposed a new IQA algorithm, MAD_{DWT}, which designed specifically for JPEG2000 coding. MAD_{DWT} was developed based on MAD algorithm. The MAD_{DWT} employed DWT instead of log-Gabor filtering in original MAD, with different weights to combine across scales. The results demonstrated that MAD_{DWT} outperforms all other IQA algorithms in terms of both C scores and P scores on this database.

The study on IQA algorithms' efficiency, which was reported in Chapter 4, was the first ever to examine IQA algorithms to determine microarchitecture bottlenecks.

The results showed us that different algorithms, with different approaches, shared the same algorithmic operation (filtering/transforming and statistical computation), but with different memory bottlenecks and core/computational bottlenecks. In this study, we also proposed microarchitectural-conscious coding techniques and custom hardware recommendations for performance improvement.

A performance analysis of six popular image quality assessment algorithms was presented. Even though the approaches to the six IQA algorithms are different, the algorithms shared the same main stages: a filtering (transforming), and a statistical computation. Our results revealed that different IQA algorithms overwhelm different microarchitectural resources and give rise to different types of bottlenecks in two main categories: memory bottlenecks and core/computational bottlenecks. Specific microarchitectural bottlenecks for each function/block of each algorithm were pointed out. We also proposed the hardware/microarchitectural conscious coding techniques for optimization and performance improvement. The findings and recommendations presented in this study apply broadly to all current-generation Intel IA-32 and Intel 64 based general-purpose computing platforms, whether laptops, servers, or desktops, even though the actual hotspot and bottleneck details might vary. Architectures that are radically different, with hardware accelerators, dedicated image processing cores (such as those found on some tablets and smart phones), and memory shared between GPUs and CPUs (such as AMD's Fusion APUs) are expected to show very different execution characteristics. Further studies using a similar methodology are recommended to analyze the performances of IQA algorithms on these specialized architectures.

The study on reduced-reference IQA framework, presented in Chapter 5, provided a family of RR algorithms that are based on multiscale sharpness maps. The amount of reduced information was depended on the employed sharpness map algorithm and the algorithm's block sizes. However, when utilizing approximately 2% of

reduced information, our framework provided the best version, which outperformed all other RR frameworks that employed around 3% of reduced information on predicting the DMOS/MOS scores on several popular image databases (LIVE, CSIQ, TID2008, TID2013). Our framework was also competitive or better than current state-of-the-art full-reference IQA algorithms.

Our framework, S4RR, extracts 19 features from three-scale sharpness maps to classify the distorted image into seven distortion families. S4RR then adaptively switches among seven scalars, combining with the average of three maxima and three L_2-norms to assess quality. We provided the training process on TID2008's (17 distortion types into seven distortion families) and the excellent performances on this database. S4RR$_L$, the best version of S4RR with 2.05% reduced information, was able to outperform all state-of-the-art algorithms, including full-reference and other reduced-reference frameworks (2.8%-3.1% reduced information). We also demonstrated that the chosen seven distortion families were efficient to assess quality by performing analysis on LIVE, CSIQ, and TID2013 databases, which include new distortion types (six distortion types in CSIQ, including one new distortion type; five distortion types in LIVE, including one new distortion type; and 24 distortion types in TID2013, including seven new distortion types.). The new distortion types raised some small issues with classification accuracy. However, the results on LIVE, CSIQ, and TID2013 showed that S4RR was able to outperform all other metrics on most of the criteria.

The contributions of this dissertation are summarized as following:

• A local sharpness database, and database analysis. This sharpness map is the first ever local quality database at the time it is presented. By now, several algorithms aim to, and are designed to predict local sharpness. Their ability of local sharpness prediction is verified using the ground-truth in the database.

• A new JPEG2000-compressed image database and new algorithm designed

specifically for JPEG2000 coding, which uses local DWT coefficient statistics. The proposed algorithm outperformed current state-of-the-art IQA algorithms on predicting ranking order of this database.

• Results of current IQA algorithms' microarchitecture analysis, and hardware/microarchitectural conscious coding techniques for optimization and performance improvement. The results revealed that different IQA algorithms with different approaches share the same algorithmic operations (such as filtering/transforming and statistical computation). However, they overwhelmed different microarchitectural resources and give rise to different types of bottlenecks in two main categories: memory bottlenecks and core/computational bottlenecks. Hardware/microarchitectural conscious coding techniques for optimization and performance improvement were provided, not only for improving these algorithms, but also for any algorithm in the future.

• A foundation for custom IQA engine hardware. In the near future, if any company wants to design a custom hardware for one of these algorithms, for a new algorithm, or for several algorithms, this should be in their starter kit, if they want the algorithms run efficiently.

• A family of reduced-reference IQA algorithms - a RR framework based on local sharpness maps. This framework utilizes any sharpness map algorithm as reduced reference features. It then divides any distortion type into seven distortion families via a soft classification technique with probabilities. The framework switches between seven scalar (for seven distortion families) and two measurements adaptively to assess quality. The proposed framework shows the best results testing on several popular image databases.

6.2 Future work

The next stage of the algorithm designed specifically for JPEG2000 coding would be applying this algorithm to develop a new compression algorithm, which can yield a better perceived quality with the same compression ratio, comparing to current JPEG2000. However, at this point the running speed is the main concern. In order to be widely used application, it needs to respond in a real-time or at least very short time. The main reason that no algorithm could replace mean square error (MSE) in JPEG2000 is MSE is too fast. MSE calculates the difference between original and compressed (distorted) image with a matrix subtraction operation (pixel-wise). Whereas modern IQA algorithms normally require at least one stage of decomposition (filtering/transforming), and one stage of statistical computation (mean, variance, skewness, or kurtosis).

Two main things that need to be done to obtain a new JPEG2000 coding are: (1) optimizing current MAD_{DWT} (proposed in Section 3.4) to run as fast as possible, and (2) applying this algorithm into JPEG2000 coding efficiently. In this MAD-based JPEG2000 coding, the MAD_{DWT} would be called iteratively. Therefore, it will require new techniques to optimize, such as caching to save memory or pre-computing to save computation. The analysis methodology and optimization techniques provided in Chapter 4 would be useful again in this case. It can show us where, when, and how to speed up this MAD-based JPEG2000 coding.

The RR IQA framework can be extended to reduced-reference video quality assessment. As any other image quality algorithm, this can also have an extension for videos. Because human vision system could treat a video differently to a set of frames (images), video quality assessment is a different task. The work to extend to video, if applicable, would be useful and it would have a wide application. Reduced reference video quality assessment is a promising area to explore.

Current RR framework, S4RR, has a main limitation: it converts color images

into grayscale before performing quality assessment, therefore it does not keep color information. Among popular image databases (LIVE, CSIQ, TID2008, TID2013), only TID2013 has color distortions. We could see that these color-distorted subsets of TID2013 made many algorithms confused. Special attention needs to be paid to color distortions, not only for S4RR, but also for other RR frameworks and FR algorithms.

Another potential work to extend my research is developing a hardware designed specifically for IQA algorithms and/or two common stages (transforming and statistical computation) of IQA algorithms. Such hardware will be useful for testing IQA algorithms on frames of videos, other types of images such as medical/radiation/thermal images efficiently. It is also useful in current imaging devices, such as speeding cameras of police, information-collecting cameras in unmanned aerial vehicle.

REFERENCES

[1] 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/.

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

[3] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti, "TID2008 - a database for evaluation of full-reference visual quality assessment metrics," *Advances of Modern Radioelectronics*, vol. 10, pp. 30–45, 2009.

[4] N. Ponomarenko, O. Ieremeiev, V. Lukin, K. Egiazarian, L. Jin, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, *et al.*, "Color image database TID2013: Peculiarities and preliminary results," in *Visual Information Processing (EUVIP), 2013 4th European Workshop on*, pp. 106–111, IEEE, 2013.

[5] 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.

[6] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," 37th IEEE Asilomar Conference on Signals, Systems and Computers 2003.

[7] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference im-

age quality assessment and the role of strategy," *Journal of Electronic Imaging*, vol. 19, no. 1, p. 011006, 2010.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] D. S. Swamy, K. J. Butler, D. M. Chandler, and S. S. Hemami, "Parametric quality assessment of synthesized textures," in *IS&T/SPIE Electronic Imaging*, pp. 78650B–78650B, International Society for Optics and Photonics, 2011.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] A. Mittal, A. Moorthy, and A. Bovik, "No-reference image quality assessment in the spatial domain," *Image Processing, IEEE Transactions on*, vol. 21, no. 12, pp. 4695–4708, 2012.

[19] C. T. Vu, T. D. Phan, and D. M. Chandler, "S3: A spectral and spatial measure of local perceived sharpness in natural images," *Image Processing, IEEE Transactions on*, vol. 21, no. 3, pp. 934–945, 2012.

[20] P. Vu and D. Chandler, "A fast wavelet-based algorithm for global and local image sharpness estimation," *Signal Processing Letters, IEEE*, vol. 19, pp. 423 –426, july 2012.

[21] X. Marichal, W.-Y. Ma, and H. Zhang, "Blur determination in the compressed domain using dct information," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 2, pp. 386 –390 vol.2, 1999.

[22] P. Marziliano, F. Dufaux, S. Winkler, T. Ebrahimi, and G. Sa, "A no-reference

perceptual blur metric," in *IEEE 2002 International Conference on Image Processing*, pp. 57–60, 2002.

[23] D. Shaked and I. Tastl, "Sharpness measure: towards automatic image enhancement," *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, pp. I–937–40, Sept. 2005.

[24] N. Narvekar and L. Karam, "A no-reference perceptual image sharpness metric based on a cumulative probability of blur detection," in *Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on*, pp. 87 –91, 2009.

[25] R. Ferzli and L. Karam, "A no-reference objective image sharpness metric based on the notion of just noticeable blur (jnb)," *Image Processing, IEEE Transactions on*, vol. 18, no. 4, pp. 717 –728, 2009.

[26] 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.

[27] A. Z. K. E. M. C. N. Ponomarenko, V. Lukin and F. Battisti, "TID2008 a database for evaluation of full reference visual quality assessment metrics," *Advances of Modern Radioelectrionics*, vol. 10, pp. 30–45, 2009.

[28] C. T. Vu, T. D. Phan, and D. M. Chandler, "S3: A spectral and spatial measure of local perceived sharpness in natural images," *Image Processing, IEEE Transactions on*, vol. 21, no. 3, pp. 934–945, 2012.

[29] VQEG, "Final report from the video quality experts group on the validation of objective models of video quality assessment, phase ii," August 2003. http://www.vqeg.org.

[30] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 99th ed., August 1991.

[31] N. Damera-Venkata, T. D. Kite, W. S. Geisler, B. L. Evans, and A. C. Bovik, "Image quality assessment based on a degradation model," *IEEE Trans. Image Process.*, vol. 9, 2000.

[32] D. M. Chandler and S. S. Hemami, "VSNR: A wavelet-based visual signal-to-noise ratio for natural images," *IEEE Trans. Image Processing*, vol. 16, pp. 600–612, September 2007.

[33] Z. Wang, Q. Li, and X. Shang, "Perceptual image coding based on a maximum of minimal structural similarity criterion," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 2, pp. II –121 –II –124, 16 2007-oct. 19 2007.

[34] T. Richter and K. J. Kim, "A ms-ssim optimal jpeg 2000 encoder," in *Data Compression Conference, 2009. DCC '09.*, pp. 401 –410, march 2009.

[35] M. B. (Ed.), "Information technology the JPEG2000 image coding system: Part 1," *ISO/IEC IS 15444-1*, 2000.

[36] D. M. Chandler and S. S. Hemami, "Dynamic contrast-based quantization for lossy wavelet image compression," *IEEE Transactions on Image Processing*, vol. 14, pp. 397–410, April 2005.

[37] "Information technology–jpeg 2000 image coding system: Core coding system," Tech. Rep. ISO/IEC FDIS15444-1: 2000, International Organization for Standardization, Geneva, Switzerland, August 2000.

[38] O. supplement, "On the use of image quality estimators for improved jpeg2000 coding," 2012. http://vision.okstate.edu/mad_dwt/.

[39] A. T1.TR.74-2001, *Objective Video Quality Measurement Using a Peak-Signal-to-Noise-Ratio (PSNR) Full Reference Technique*, 2001.

[40] Y. K. Z. M. P. S. Yuukou Horita, Keiji Shibata, "Subjective quality assessment Toyama database," 2008. http://mict.eng.u-toyama.ac.jp/mict/.

[41] S. Daly, "Subroutine for the generation of a two dimensional human visual contrast sensitivity function," *Eastman Kodak*, 1987.

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

[43] P. Vu, C. Vu, and D. Chandler, "A spatiotemporal most-apparent-distortion model for video quality assessment," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 2505 –2508, sept. 2011.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

[55] 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.

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

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

[62] 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.

[63] T. Phan, S. Sohoni, D. Chandler, and E. Larson, "Performance-analysis-based acceleration of image quality assessment," in *IEEE Southwest Symposium on Image Analysis and Interpretation*, 2012.

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

[65] 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.

[66] D. M. Chandler and S. S. Hemai, "Vsnr: A wavelet-based visual signal-to-noise ratio for natural images," *IEEE Transactions on Image Processing*, vol. 16, no. 9, pp. 2284–2298, 2007.

[67] R. Bhargava, L. K. John, B. L. Evans, and R. Radhakrishnan, "Evaluating MMX technology using DSP and multimedia applications," in *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*, pp. 37–46, IEEE Computer Society Press, 1998.

[68] Intel, "Available online." http://developer.intel.com/design/perftool/vtcd/.

[69] 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.

[70] C. Martinez, M. Pinnamaneni, and E. B. John, "Performance of commercial multimedia workloads on the Intel Pentium 4: A case study," *Computers & Electrical Engineering*, vol. 35, no. 1, pp. 18–32, 2009.

[71] D. M. Chandler and S. S. Hemami, "Available online." http://foulard.ece.cornell.edu/dmc27/vsnr/vsnr.html.

[72] Internet, "Available online." http://www.cns.nyu.edu/ eero/steerpyr/.

[73] H. R. Sheikh and A. C. Bovik, "Available online." http://live.ece.utexas.edu/research/quality/VIF.htm.

[74] R. B. Davies, "Available online." http://www.robertnz.net/nm_intro.htm, 2006.

[75] J. Villasenor, B. Belzer, and J. Liao, "Wavelet filter evaluation for image compression," *IEEE Trans. Image Process.*, vol. 4, pp. 1053–1060, 1995.

[76] J. L. Mannos and D. J. Sakrison, "The effects of a visual fidelity criterion on the encoding of image," *IEEE Trans. Info. Theory*, vol. 20, pp. 525–535, 1974.

[77] E. C. Larson and D. M. Chandler, "Online supplement." http://vision.okstate.edu/MAD, 2010.

[78] T. Ooura, "Available online." http://www.kurims.kyoto-u.ac.jp/ ooura/, 2006.

[79] 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 online:http://www.csse.uwa.edu.au/∼pk/research/matlabfns/.

[80] M. Saad, A. Bovik, and C. Charrier, "A DCT statistics-based blind image quality index," *IEEE Signal Process. Lett.*, vol. 17, p. 583 586, June 2010.

[81] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[82] J. P. Shen and M. H. Lipasti, *Modern processor design: fundamentals of superscalar processors*, vol. 2, ch 3. McGraw-Hill Higher Education, 2005.

[83] Internet, "Available online." http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf.

[84] Internet, "Available online." http://software.intel.com/sites/products/documentation/doclib/st
2013/amplifierxe/win/win_ug/index.htm.

[85] S. Vlaovic, E. S. Davidson, and G. S. Tyson, "Improving BTB performance in the presence of DLLs," in *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, pp. 77–86, ACM, 2000.

[86] Internet, "Available online." http://software.intel.com/en-us/articles/x87-and-sse-floating-point-assists-in-ia-32-flush-to-zero-ftz-and-denormals-are-zero-daz/.

[87] Y. Zhang and D. M. Chandler, "No-reference image quality assessment based on log-derivative statistics of natural scenes," *Journal of Electronic Imaging*, vol. 22, no. 4, pp. 043025–043025, 2013.

[88] R. Soundararajan and A. C. Bovik, "Rred indices: Reduced reference entropic differencing for image quality assessment," *Image Processing, IEEE Transactions on*, vol. 21, no. 2, pp. 517–526, 2012.

[89] D. Liu, Y. Xu, Y. Quan, and P. L. Callet, "Reduced reference image quality assessment using regularity of phase congruency," *Signal Processing: Image Communication*, 2014.

[90] L. Zhang, D. Zhang, and X. Mou, "Fsim: a feature similarity index for image quality assessment," *Image Processing, IEEE Transactions on*, vol. 20, no. 8, pp. 2378–2386, 2011.

[91] J. Caviedes and S. Gurbuz, "No-reference sharpness metric based on local edge kurtosis," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 3, pp. III–53 – III–56 vol.3, 2002.

[92] J. Dijk, M. van Ginkel, R. J. van Asselt, L. J. van Vliet, and P. W. Verbeek, "A new sharpness measure based on gaussian lines and edges," in *CAIP*, pp. 149–156, 2003.

[93] C. Vu and D. M. Chandler, "S3: A spectral and spatial sharpness measure," in *2009 First International Conference on Advances in Multimedia*, 2009.

[94] Z. W. Rania Hassen and M. Salama, "No-reference image sharpness assessment based on local phase coherence measurment," March 2010.

[95] P. V. Vu and D. M. Chandler, "A no-reference quality assessment algorithm for jpeg2000-compressed images based on local sharpness," in *IS&T/SPIE Electronic Imaging*, pp. 865302–865302, International Society for Optics and Photonics, 2013.

[96] Z. Wang and E. P. Simoncelli, "Reduced-reference image quality assessment using a wavelet-domain natural image statistic model," in *Electronic Imaging 2005*, pp. 149–159, International Society for Optics and Photonics, 2005.

[97] I. P. Gunawan and M. Ghanbari, "Reduced-reference picture quality estimation by using local harmonic amplitude information," in *London Communications Symposium*, vol. 2003, 2003.

[98] M. Carnec, P. Le Callet, and D. Barba, "Visual features for image quality assessment with reduced reference," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, pp. I–421, IEEE, 2005.

[99] M. Carnec, P. Le Callet, and D. Barba, "Objective quality assessment of color images based on a generic perceptual reduced reference," *Signal Processing: Image Communication*, vol. 23, no. 4, pp. 239–256, 2008.

[100] A. Maalouf, M.-C. Larabi, and C. Fernandez-Maloigne, "A grouplet-based reduced reference image quality assessment," in *Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on*, pp. 59–63, IEEE, 2009.

[101] X. Gao, W. Lu, D. Tao, and X. Li, "Image quality assessment based on multiscale geometric analysis," *Image Processing, IEEE Transactions on*, vol. 18, no. 7, pp. 1409–1423, 2009.

[102] Q. Li and Z. Wang, "Reduced-reference image quality assessment using divisive normalization-based image representation," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, no. 2, pp. 202–211, 2009.

[103] D. Tao, X. Li, W. Lu, and X. Gao, "Reduced-reference IQA in contourlet domain," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 6, pp. 1623–1627, 2009.

[104] U. Engelke, M. Kusuma, H.-J. Zepernick, and M. Caldera, "Reduced-reference metric design for objective perceptual quality assessment in wireless imaging," *Signal Processing: Image Communication*, vol. 24, no. 7, pp. 525–547, 2009.

[105] W. Xue and X. Mou, "Reduced reference image quality assessment based on weibull statistics," in *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*, pp. 1–6, IEEE, 2010.

[106] L. Ma, S. Li, F. Zhang, and K. N. Ngan, "Reduced-reference image quality assessment using reorganized dct-based image representation," *Multimedia, IEEE Transactions on*, vol. 13, no. 4, pp. 824–829, 2011.

[107] A. Rehman and Z. Wang, "Reduced-reference image quality assessment by structural similarity estimation," *Image Processing, IEEE Transactions on*, vol. 21, no. 8, pp. 3378–3389, 2012.

[108] Z. Lin, J. Tao, and Z. Zheng, "Reduced-reference image quality assessment based on average directional information," in *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, vol. 2, pp. 787–791, IEEE, 2012.

[109] D. Bordevic, D. Kukolj, M. Pokric, and I. Ostojic, "Image quality assessment using reduced-reference nonlinear model," in *Intelligent Systems and Informatics (SISY), 2013 IEEE 11th International Symposium on*, pp. 167–170, IEEE, 2013.

[110] M. Zhang, X. Mou, H. Fujita, L. Zhang, X. Zhou, and W. Xue, "Local binary pattern statistics feature for reduced reference image quality assessment," in *IS&T/SPIE Electronic Imaging*, pp. 86600L–86600L, International Society for Optics and Photonics, 2013.

[111] J. Wu, W. Lin, G. Shi, and A. Liu, "Reduced-reference image quality assessment with visual information fidelity," 2013.

[112] V. Bhateja, A. Srivastava, and A. Kalsi, "A reduced reference distortion estimation measure for color images," in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pp. 699–704, IEEE, 2014.

[113] V. Bhateja, A. Kalsi, and A. Srivastava, "Reduced reference IQA based on structural dissimilarity," in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pp. 63–68, IEEE, 2014.

[114] J. Farah, M.-R. Hojeij, J. Chrabieh, and F. Dufaux, "Full-reference and reduced-reference quality metrics based on sift," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 161–165, IEEE, 2014.

[115] D. L. Ruderman, "The statistics of natural images," *Network: computation in neural systems*, vol. 5, no. 4, pp. 517–548, 1994.

[116] P. Reinagel and A. M. Zador, "Natural scene statistics at the centre of gaze," *Network: Computation in Neural Systems*, vol. 10, no. 4, pp. 341–350, 1999.

[117] L. Beaurepaire, K. Chehdi, and B. Vozel, "Identification of the nature of noise and estimation of its statistical parameters by analysis of local histograms," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 4, pp. 2805–2808, IEEE, 1997.

[118] D.-C. Chang and W.-R. Wu, "Image contrast enhancement based on a histogram transformation of local standard deviation," *Medical Imaging, IEEE Transactions on*, vol. 17, no. 4, pp. 518–531, 1998.

[119] R. Wang, A. Hanson, and E. Riseman, "Texture analysis based on local standard deviation of intensity," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 482–488, 1986.

[120] C. T. Vu, T. D. Phan, P. S. Banga, and D. M. Chandler, "On the quality assessment of enhanced images: a database, analysis, and strategies for augmenting existing methods," in *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*, pp. 181–184, IEEE, 2012.

[121] H. R. Sheikh, Z. Wang, A. C. Bovik, , and L. K. Cormack, "LIVE multiply distorted image quality database." Online. http://live.ece.utexas.edu/research/quality/live_multidistortedimage.html.

VITA

THIEN DUC PHAN

Candidate for the Degree of

Doctor of Philosophy

Dissertation: STRATEGIES FOR IMPROVING EFFICIENCY AND EFFICACY OF IMAGE QUALITY ASSESSMENT ALGORITHMS

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, in Information Technology, 2008.
Received the Master of Science degree from Oklahoma State University,
Oklahoma, USA, in Electrical Engineering Oklahoma State University in December, 2014.
Completed the requirements for the degree of Doctor of Philosophy with a major in Electrical Engineering Oklahoma State University in December, 2015.

Experience:
Research Assistant, Oklahoma State University 2010-2015.

Selected Journals:
T. D. Phan, S. K. Shah, D. M. Chandler, and S. Sohoni, "Microarchitectural analysis of image quality assessment algorithms," *Journal of Electronic Imaging (JEI)*, 23(1), 013030, February 2014. doi: 10.1117/1.JEI.23.1.013030.

C. T. Vu, T. D. Phan, and D. M. Chandler, "S3: A Spectral and Spatial Measure of Local Perceived Sharpness in Natural Images," *IEEE Transactions on Image Processing*, Vol. 21, No. 3, March 2012.