Perceptual Image Similarity Metrics and Applications

by

Yuanhao Zhai

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in the University of Michigan
2015

To my parents, Jianhua Zhai and Yongcai Pang;
and Yun (Joy) Xu

Acknowledgments

At this point, I am about to close an unforgettable chapter of my life. Looking back into these five years' PhD study on this lovely campus, there are so many people to whom I want to express my sincere gratitude.

First, I want to thank my parents, Mr. Jianhua Zhai and Ms. Yongcai Pang, for everything. Although most of the time they are at the other half of the earth, I can always feel their love and support. I can still remember every small details during the time we spent together when they visited Ann Arbor. They cooked my favorite food every day and we usually took a nice walk after dinner. They made a lot of dumplings by hand and stored them in the refrigerator, so that I could still taste the food from hometown after they left. All these small details are my most valuable memory. Every time I feel helpless, I can always find support through video chatting with them. No matter what happened, I know my family will always be there for me.

Next, I have to give my biggest thanks to Dave. Dave is not only my advisor, but also my friend. Without his guidance and support, I could not be writing this dissertation now. As an advisor, he is always so approachable and patient. I can still remember our countless meetings during which we brainstormed new research ideas and edited papers together. Those meetings were not only productive, but also enjoyable. Although in this dissertation, sometimes I mention "I proposed this new metric" or "I conducted this experiment", readers should always be aware that what I actually mean is "Dave and I". Dave's help to me goes far beyond research and study. My academic writing and presentation skills would still be at entry level without Dave's guidance. Every time I finished a paper draft, Dave would edit it sentence by sentence, or even word by word, side by side with me so that I can learn how to improve. His rigorous spirit also influences me a lot. Thrasos used to say that "Dave doesn't tolerate typos". And that is totally true. In addition, Dave always tries to read our papers from readers' perspective to see whether our description is self-explanatory and clear enough. That is the reason why his papers are always of high quality.

As a friend, Dave is humorous, considerate, and a man of great integrity. I enjoyed all the sailing trips with Dave, especially the one time we raced against other sailing boats. Our boat was stuck in the middle of Lake Erie for hours due to the lack of wind. By the time we finished the race, it was almost 9:30 at night, but it was a beautiful summer evening and everyone on the boat had a wonderful time with each other. I also want to thank Dave for hosting a great party for me the next day after my defense. I didn't know that Dave was such a good party planner and BBQ cook. See, there is always something new about Dave for me to discover even after so many years.

Besides Dave, Thrasos has also been very helpful and supportive to me throughout my PhD study. He is the second professor in this world that know my research so well and his sense of humor always makes our discussion a blast. I can still remember that night he hurt his finger when setting up the sofa bed for me at his house. After he went to the emergency room, I felt so sorry and moved. Thank god he has not only a strong interest in image processing, but also a strong finger bone. He will always be a mentor and friend to me. Also I want to thank my collaborators and friends from Thrasos' research team: Guoxin, Shengxin and Jana. Through discussions with them, a lot of great ideas came up. I miss my trips to Orlando and Evanston together with them.

I also want to thank the rest of my committee for their suggestions, comments and guidance. I knew Jeff from his image processing class. His lectures were always so interesting and inspiring. My interest in image processing started from there. I still remember the course project we did, "The Mystery of Stereograms". Last but not least, Jun also provided me a lot of valuable comments. As a researcher in psychology and mathematics, his inputs to my research always come from a different point of view and his questions can always open my eyes.

There are also many other faculties and staffs who influenced me from various aspects. I have to thank Demos for introducing me to this lovely campus and guiding me through my first semester. He is such a nice mentor and educator. Students are his first priority and he is always there when students need him. His stochastic process course is the best one in this area and I would like to recommend it to anyone who is interested in this field. I also want to thank Sandeep and Wayne. The experience to work with them as a graduate student instructor was unforgettable. Through the work, I not only refreshed my memory on the course materials, but also practiced my teaching and presentation skills. I discovered the feeling of loving to mentor people and help them succeed. I want to give my thanks and best wishes to Becky. She has been so nice and helpful. I can still remember the orientation party she organized for all new graduate students five years ago. She made my life much easier by helping me deal with all those paperworks. I wish her a happy retired life in Seattle.

I owe my thanks to my mentors and friends during my internships: Zhengyu, Alex, Manu, Savo, Qi, Misha, Dan and many more. I learnt so much from these industrial experiences. I'm so lucky to have the opportunity to work with these smart minds and learn from them.

I would like to thank Joy for her love and support. I am so lucky to have her together with me to go through these years. I cannot imagine my life without her companionship. She is always so optimistic and has the ability to put smile on my face. There are just too many words I want to put here to thank her.

Also during these five years, I harvested many great friendships. There are so many good friends that enriched my life in Ann Arbor: Ruihao (Terence) Zhu, Tianpei Xie, Yang Liu, Matt Prelee, Zhaoshi (Josh) Meng, Hao Sun, Yingze (Sid) Bao, Guanyu Zhou, Jiangfeng Wu, Chongruo Wu, Yi Chen, Zhen Wu, Awlok Josan, Qingsi Wang, and so many more. Additionally, I would like to thank all family and friends from outside Ann Arbor who have been supportive.

Finally, I want to give my thanks to this lovely town and university. Five years ago, I came here with some vague dreams and nothing else. Today, I'm going to leave with my PhD degree and so many memories. I'll always remember all the good times I spent here.

I left my footprints in Diag and I'm so proud to be a wolverine. Go Blue!

Yuanhao Zhai

Ann Arbor, Michigan

May 20, 2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Perceptual Image Similarity Metrics and Applications

by

Yuanhao Zhai

Chair: David L. Neuhoff

This dissertation presents research in perceptual image similarity metrics and applications, *e.g.*, content-based image retrieval, perceptual image compression, image similarity assessment and texture analysis.

The first part aims to design texture similarity metrics consistent with human perception. A new family of statistical texture similarity features, called Local Radius Index (LRI), and corresponding similarity metrics are proposed. Compared to state-of-the-art metrics in the STSIM family, LRI-based metrics achieve better texture retrieval performance with much less computation. When applied to the recently developed perceptual image coder, Matched Texture Coding (MTC), they enable similar performance while significantly accelerating encoding. Additionally, in photographic paper classification, LRI-based metrics also outperform pre-existing metrics. To fulfill the needs of texture classification and other applications, a rotation-invariant version of LRI, called Rotation-Invariant Local Radius Index (RI-LRI), is proposed. RI-LRI is also grayscale and illuminance insensitive. The corresponding similarity metric achieves texture classification accuracy comparable to state-of-the-art metrics. Moreover, its much lower dimensional feature vector requires substantially less computation and storage than other state-of-the-art texture features.

The second part of the dissertation focuses on bilevel images, which are images whose pixels are either black or white. The contributions include new objective similarity metrics intended to quantify similarity consistent with human perception, and a subjective experiment to obtain ground truth for judging the performance of objective metrics. Several similarity metrics are proposed that outperform existing ones in the sense of attaining significantly higher Pearson and Spearman-rank correlations with the ground truth. The new metrics include Adjusted Percentage Error, Bilevel Gradient Histogram, Connected Components Comparison and combinations of such.

Another portion of the dissertation focuses on the aforementioned MTC, which is a block-based image coder that uses texture similarity metrics to decide if blocks of the image can be encoded by pointing to perceptually similar ones in the already coded region. The key to its success is an effective texture similarity metric, such as an LRI-based metric, and an effective search strategy. Compared to traditional image compression algorithms, *e.g.*, JPEG, MTC achieves similar coding rate with higher reconstruction quality. And the advantage of MTC becomes larger as coding rate decreases.

CHAPTER 1

Introduction

In the field of image processing, the study of image quality and similarity have been greatly developed during the past few decades. The goal is to design objective metrics to predict human judgments on image quality and similarity. Although *quality* sounds like *similarity*, there exist subtle differences between them. Generally speaking, human observers can judge the *quality* of an image without any reference. However, *similarity* is not a private property of any single image. Instead, one can only judge an image's similarity to another image, *e.g.*, a reference image. Two images with high similarity can both have high or low quality. An image similar to a high quality image should also have high quality. Sometimes, people mean *similarity* when they say *quality*. For example, in image quality assessment society, *quality* has the meaning of the fidelity of a distorted image relative to its original. Actually, in our discussion, this is a special case of the *similarity* of two images with one being an original image with perfect *quality* and the other one being a distorted version of the original itself. Another issue to notice is that for any image with meaningful content, it is usually feasible to judge its quality. On the other hand, it is not always feasible to judge the similarity of two images. For example, consider two images with non-homogeneous contents, *i.e.*, not purely smooth or textured with homogeneous patterns, that agree with each other in most areas except for a very small region, it is very hard to say how similar they are. Different human observers may have different opinions on their similarity. And different applications, *e.g.*, image compression and image retrieval, may also

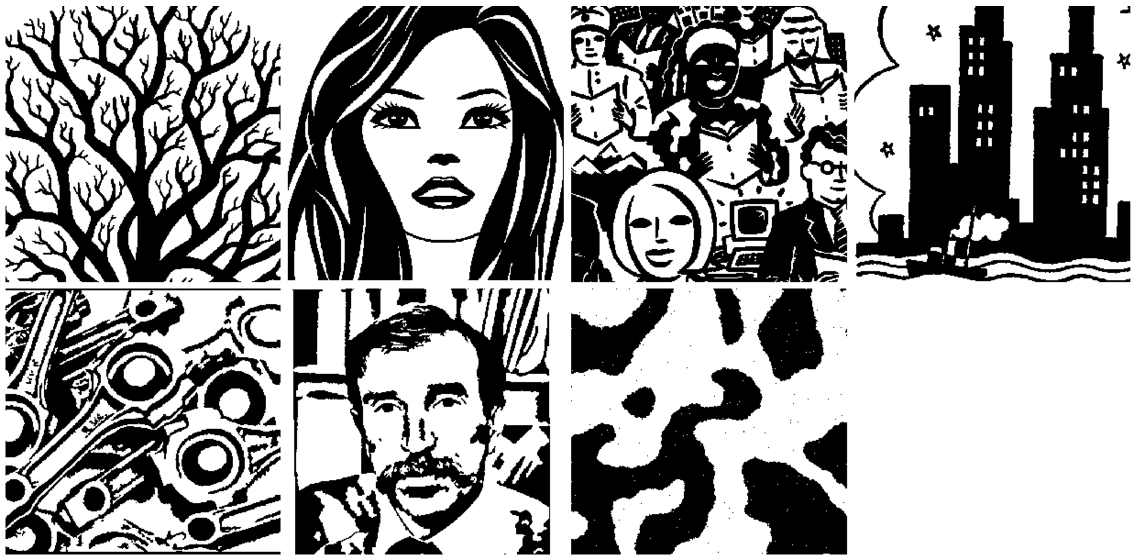


Figure 1.1: Seven scenic images: 'tree', 'woman', 'people', 'boat', 'tools', 'Alc', 'MRF'.

have different demands and tolerance on this kind of dissimilarity. However, when images are homogeneous, then making a similarity judgment is a much easier task. Hence, in image similarity study, people usually focus only on images (or image patches, *e.g.*, 32×32 patches) with homogeneous content on which human observer opinions tend to be consistent. Among the many possible kinds of homogeneous content, texture is very unique and has been studied a great deal. Texture is a unique kind of visual signal and does not have a widely agreed definition. However, Portilla and Simoncelli [1] provide a general definition with which we agree: *"Loosely speaking, texture images are spatially homogeneous and consist of repeated elements, often subject to some randomization in their location, size, color, orientation, etc."* Texture similarity is one main focus of this research. Texture similarity metrics are reviewed in detail in Section 1.1, and new texture similarity metrics are proposed in Chapters 3 and 4.

Note that the discussion above is oriented to grayscale and color images. Things are different for bilevel images. In bilevel images, each pixel can only have intensity of either 0 (black) or 1 (white). Different from text, line drawings and silhouettes, the bilevel images in which we are interested are scenic images, which are complex images containing natural

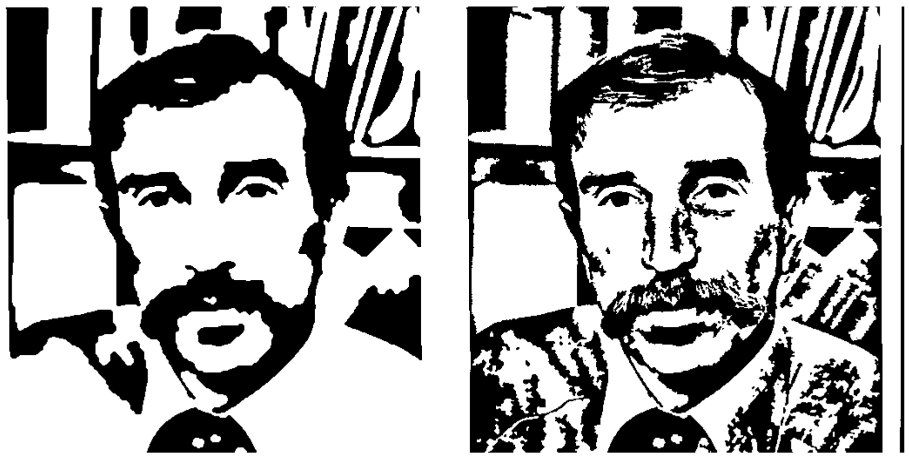


Figure 1.2: Two original bilevel images.

scenes, *e.g.*, landscapes and portraits, but are not halftoned. Seven scenic bilevel images are shown in Fig. 1.1. Unlike grayscale and color images, whose quality can usually be evaluated by human observers without any reference, we assert that it can often be difficult or even impossible to judge the quality of a bilevel image without a reference, due to the fact that many scenic bilevel images are man-made or man-processed and that artists and image processors have different stylistic intentions which to some may appear as distortion, but not to others. As an example, the image on the left of Fig. 1.2 might appear to be overly smoothed, while the image on the right might appear to be overly noisy, despite each being the intended result of the ACA segmentation algorithm [2] applied to a grayscale image with parameters set differently due to different intentions. Hence, for bilevel images, there is even more motivation for studying similarity metrics (than quality metrics) than for grayscale or color images. In particular, if one image is a distorted copy of the other, then the similarity can be thought as the fidelity of the distorted image to its original. As another focus of this research, in Section 1.2, an introduction of bilevel image similarity metrics will be given, and new bilevel similarity metrics are proposed in Chapter 8.

As discussed previously, this research focuses on image similarity metric design and applications. The objective opinion given by similarity metrics should agree with human perception, especially at or near the high end of the similarity scale, meaning two images being compared are sufficiently similar to each other. Moreover, a similarity metric should not only provide high metric scores to indicate their high similarity, but also have a monotonic relationship with human judgments. For example, if a certain kind of minor distortion is added to an original image with different amounts, a good similarity metric should produce monotonic metric scores that are consistent with the amounts of distortion added. Also at the high end of the similarity scale, in some applications, *e.g.*, identical texture retrieval [3, 4], the ability to distinguish "identical" images from "non-identical" ones is desired.

Such a metric is needed in many image processing applications, *e.g.*, content-based image retrieval, perceptual image compression, image similarity assessment and texture analysis. For example, they can be used to assess overall performance of such algorithms, *e.g.*, compression algorithms, or they can play a role in the operation of algorithms, *e.g.*, compression and retrieval algorithms. As applications become increasingly more sophisticated and demanding, so do the requirements for image similarity metrics.

In particular, this research mainly focuses on two specific branches of image similarity: grayscale homogeneous texture similarity and bilevel image similarity. In the next two subsections, we review these two areas, respectively.

1.1 Grayscale Texture Similarity

Texture analysis, including representation, modeling and similarity metrics, has been actively studied over the past few decades, due to its importance in many research areas, including image processing, computer vision and pattern recognition. To fulfill the demands of different applications, various similarity features and metrics have been proposed to represent, classify and compare the similarity of texture patterns. The goal is to make predictions about texture that are consistent with human perception.

Due to the busyness of typical textures and the insensitivity of humans to texture de-

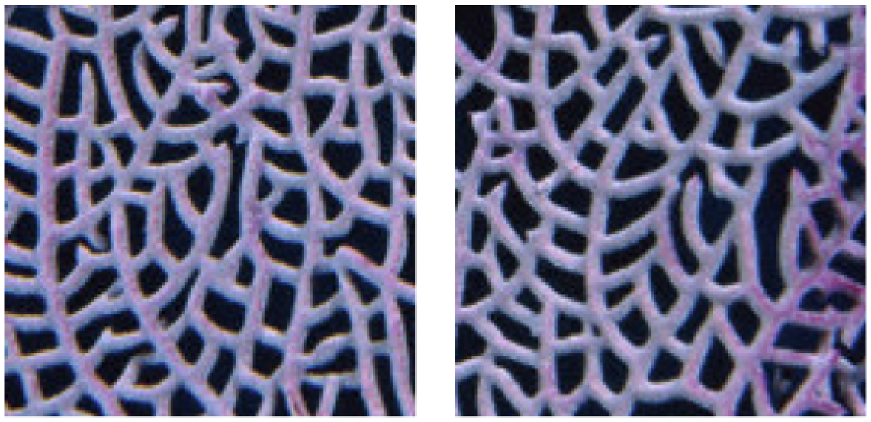


Figure 1.3: Two similar textures with substantial pixel differences.

tails, traditional point-by-point similarity metrics, such as Mean-Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and metrics that apply MSE or PSNR in a transform domain, usually fail, for they are overly sensitive to small differences between texture patterns, such as small shifts and rotations. For example, the two images in Fig. 1.3 look very alike to human observers even with substantial pixel differences. Hence, successful metrics are generally based on statistical features, either in spatial or transform domain.

For different applications, statistic-based texture similarity metrics need to possess different properties. For example, for perceptual image compression and identical texture retrieval, such metrics should tolerate small spatial shifts and small rotations, but penalize sizable rotations monotonically [5]. Representative methods that apply in this domain include some based on co-occurrence matrices [6–8], which are matrices count the numbers of pairs of pixels with various displacements and intensities. Such co-occurrence matrices are closely related to second-order image statistics [9, p. 220]. Others are based on Markov Random Fields (MRF) [10, 11], in which each pixel in the image is assumed to depend only on its neighbors. Currently, state-of-the-art performance in this application domain is attained by metrics in the family of Structural Similarity Metrics (SSIM), as originated in [12–14]. Specifically, it is the Structural Texture Similarity Metrics (STSIM) [3,4,15–17] that are best at measuring texture similarity, as evidenced by their success in applications including identical texture retrieval and the newly proposed Matched Texture Coding (MTC) [18], which is a block-based image coder that uses a texture similarity metric to decide if an image block can be encoded by pointing to a similar one that has already been coded.

Another important application domain is texture classification, which is an active area of research whose goal is to develop systems that accurately classify textures, regardless of their orientation. Thus, metrics for this application must be rotation invariant, or at least, rotation insensitive. Being rotation sensitive, the methods and metrics mentioned previously are not appropriate. However, there are rotation-invariant methods such as Voronoi tessellations [19, 20] and structural displacement rules [21–23], which assume textures are composed of repetitive texture elements with certain geometrical displacement rules. In computer vision tasks, such as pattern recognition, sometimes scale-invariance is also desired, *e.g.*, [24–26].

Most importantly, the Local Binary Patterns (LBP) feature, originally proposed in 2002 [27], has drawn much recent attention. It is a rotation-invariant histogram of angular pattern information, based on simple computations in the spatial domain. No training is required. Metrics based on LBP have achieved good texture classification performance. Subsequently, a number of additional rotation-invariant texture features and metrics have been motivated by LBP, such as Dominant LBP (DLBP) [28] and Completed LBP (CLBP) [29]. In addition, a number of complementary texture features have been combined with LBP to attain better performance, including Dominant Neighborhood Structure (DNS) [30] and Enhanced Binary Coding (LEBC) [31].

1.1.1 Contributions

The goal of my research is to achieve equal or better performance in the applications mentioned above, namely identical texture retrieval, perceptual image compression and texture classification, with less complexity.

First, we focus on the need for texture similarity metrics that assess the similarity of homogeneous textures in the broad domain that includes perceptual image compression and retrieval where, as mentioned earlier, changes in rotations should be penalized monotonically. This is the domain for which the STSIMs have been developed [3, 4, 15–17, 32, 33]. With this in mind, we propose a new statistical texture feature, called Local Radius Index (LRI) [34], and new texture similar metrics based on this feature in combination with other statistical features, such as LBP and a newly proposed feature called Subband Contrast Distribution (SCD), which will be described later. These new metrics are computationally much simpler than STSIM metrics (by an order of magnitude). We test these new metrics on the problem of identical texture retrieval, as recently considered in [3, 4], and find that they significantly outperform all previous metrics, while being computationally much simpler than the best of such. For example, the retrieval experiment shows that the newly proposed metrics achieve Precision @ 1 (the frequency that the first retrieved image is correct) as high as 99%, compared to state-of-the-art performance of 96% by a pre-existing metric. We also test one of the new metrics in MTC. The results show that it attains comparable compression and decoded image quality to that attained with STSIM2, but significantly accelerates the encoding algorithm (by a factor of 10). In addition, when applied to photographic paper classification, the proposed metrics also outperform existing metrics significantly. For example, experiments on two datasets show that one proposed metric achieves Precision @ 1 as high as 90% and 99%, respectively, compared to state-of-the-art performances of 88% and 86% by pre-existing metrics.

Second we focus on the situations that need rotation invariance, such as texture classification. As an extension, a rotation-invariant version of LRI, called Rotation-Invariant Local Radius Index (RI-LRI) [35], is proposed for such situations. Although its name only emphasizes rotation invariance, the proposed RI-LRI feature is also gray-scale and illuminance insensitive. The RI-LRI texture feature and the corresponding texture similarity metrics are tested on the Outex [36] and CUReT databases [37] in the standard way and are compared to the state-of-the-art metric, LEBC [31], and other recently proposed metrics, including DLBP [28], CLBP [29], DNS [30], VZ-MR8 [38] and VZ-Joint [39]. Results show that the newly proposed metric achieves performance comparable to the best of such. Moreover, it has a much more compact feature vector with 139 dimensions, compared to the 1280-dimensional LEBC feature vector and the 960-dimensional VA-MR8 feature vector. Consequently, it requires substantially less computation and storage space than pre-existing state-of-the-art metrics.

1.2 Bilevel Image Similarity

As mentioned earlier, bilevel images have only two intensity levels: 0 (black) and 1 (white). The bilevel images in which we are primarily interested are *scenic* bilevel images, such as those illustrated in Fig. 1.1, which are complex bilevel images, typically containing natural or hand-drawn scenes, *e.g.*, landscapes and portraits, but which do not include text, line drawings or halftoned images. Silhouettes generally have a simpler form than scenic images.

While a number of objective similarity metrics have been developed for grayscale and color images, with the goal of consistency with human perception, and while a number of bilevel similarity metrics have been developed, there has been almost no development of objective similarity metrics for bilevel images consistent with human perception.

The most common objective similarity metric for bilevel images, is *percentage error* (PE), which for bilevel images is the same as mean-squared error (MSE). Unfortunately, this metric is not always so consistent with human perception, as images with similar percentage error often appear very different to viewers.

With applications other than perceptual similarity in mind, many *intensity-based overlap* metrics have been proposed, as reviewed in [14, 40, 41]. Generally speaking, like PE, these penalize pixel-level disagreements, based on different assumptions about what is important in specific applications. Examples of this kind of metric include those developed by Jaccard [42], Kulczynski [43], Braun-Blanquet [44], Dice [45], and Ochiai [46]. These metrics were first widely used in biology related disciplines to group biotal communities [42] or ecologically related species [47]. Additionally, metrics like Dice [45] were used to quantify bilevel image similarity for medical image processing applications [48, 49]. While these metrics may be good for their intended applications, they were not designed to reflect human judgments of similarity. Hence, it is natural to try to design metrics that better reflect human perception. To the authors' knowledge, the only bilevel metric that attempts to reflect human perception is the SmSIM metric [50] which is based on a Markov random field model. Unlike previous metrics, SmSIM makes use of dependencies among adjacent pixels and measures the similarity of the "smoothness"/"roughness" of two images, as well as their pixel-level similarity.

As mentioned earlier, for color and grayscale images, many perceptual similarity metrics have been developed, *e.g.*, LBP [27], STSIM [3, 16, 17] and LRI [34, 35]. Such grayscale metrics can provide templates and insight for designing bilevel similarity metrics. Indeed, in some cases, they can be directly applied to bilevel images. In this dissertation, we propose several new bilevel similarity metrics based on hypotheses about human perception in Chapter 8.

In order to assess the performance of objective image similarity metrics – indeed, to enable their development – it is essential to have ground truth, *i.e.*, a set of distorted images whose perceptual similarity to the corresponding original images (perceived distortion) have been subjectively rated by human viewers. While there has been considerable work in developing subjective experiments to obtain ground truth for grayscale and color images and videos, there are none for bilevel images. In ITU-R BT.500-11 [51], a thorough study of subjective experiment methodologies has been conducted for videos, *e.g.*, television pictures. Several methods were suggested for different assessment tasks, including double-stimulus continuous quality-scale (DSCQS), double-stimulus impairment scale (DSIS), single-stimulus (SS), and simultaneous double stimulus for continuous evaluation (SDSCE). Some of the proposed methodologies have been applied in practice by researchers. For example, the DSCQS method has been applied by the Video Quality Expert Group (VQEG) [52] and has been claimed to have the least contextual effects. However, due to the large number of sample images to be viewed, the DSCQS method may be too complex, in that observers might not have time to rate enough images. Hence, the SS method is often used instead. For example, it was used in [53] to judge the quality of color images. Motivated by previous work, in Chapter 8, a subjective experiment using a modified version of the SDSCE method is designed and conducted to obtain ground truth for bilevel images.

1.2.1 Contributions

In this dissertation, we design a subjective experiment to obtain ground truth for bilevel images and propose new objective metrics to quantify bilevel image similarity consistent with human perception.

In the subjective experiment, seven scenic images are each distorted in forty-four ways, including random bit flipping, dilation, erosion and lossy compression. To produce subjective rating scores, the distorted images are each viewed side-by-side with the corresponding original by 77 subjects. The ratings from each subject are normalized. Several screening tests are applied to rule out subjects whose ratings are not sufficiently good. The normalized ratings are then analyzed on the basis of rating time, contextual effects and standard deviation. The result is a set of 264 subjectively rated pairs of images to use as ground truth for testing metrics and other applications.

The original and distorted images used in the subjective experiments, along with the subjective rating data obtained can be found in the "Bilevel Image Similarity Ground Truth Archive" at University of Michigan Deep Blue[1].

[1]http://deepblue.lib.umich.edu/handle/2027.42/111059.

Table 1.1: Publication details.

Chapter	Publication
3	ICASSP 2013 [34], submitted to TIP [54]
4	ICIP 2014 [35], submitted to TIP [54]
5	ICIP 2015 [55]
6	ICIP 2015 [56]
7	ICIP 2012 [18], ICASSP 2013 [34], submitted to TIP [54]
8	ICASSP 2014 [57,58], submitted to TIP [59]

Based on hypotheses about human perception of bilevel images, we propose several new objective bilevel image similarity metrics. These include Adjusted Percentage Error (APE), Bilevel Gradient Histogram (GH), Connected Components Comparison (CC) and combinations of such. The performance of these and pre-existing metrics is then assessed in terms of Pearson and Spearman-rank correlation with the ground truth obtained in the subjective experiment. It is found that the GH method outperforms all previous methods, and also, that the overall best performance is achieved by the combination of APE and GH, attaining Pearson and Spearman-rank correlation coefficients as high as 0.95 and 0.94, respectively. These are significantly better than the best of the pre-existing metrics, namely, 0.90 for LBP, and 0.84 for LBP or LRI, respectively.

The ground truth and the best new metric (APE + GH) are then used to compare the performance of four compression algorithms, and to assess the severity of the various kinds of distortion.

We anticipate that the proposed bilevel similarity metrics will be useful in a number of other applications, either to judge the performance of some method, or to be used as part of the system, for example in a retrieval or segmentation algorithm.

1.3 Organization of Dissertation

The remainder of the dissertation is organized as follows. In Chapter 2, we provide a background review of the existing similarity metrics for grayscale textures (including similarity metrics for grayscale images) and bilevel images, respectively. Chapter 3 proposes the new texture similarity metric, Local Radius Index (LRI), for grayscale homogeneous textures. The rotation-invariant version of LRI, Rotation-Invariant Local Radius Index (RI-LRI), is proposed in Chapter 4 for applications that demand rotation invariance. In Chapter 5, a theoretical analysis of LRI is provided using periodic tessellations. Chapters 6 and 7 discuss two applications of LRI, namely, photographic paper classification and Matched Texture Coding (MTC), respectively. Chapter 8 describes a study of bilevel image similarity, including a subjective experiment to obtain ground truth and development of new objective metrics to quantify bilevel image similarity consistent with human perception. Finally, Chapter 9 summarizes the dissertation and suggests future work. Portions of this dissertation have already been submitted or published. Table 1.1 provides a summary.

CHAPTER 2

Background

In this chapter, we provide a background review of existing similarity metrics for grayscale textures (including similarity metrics for grayscale images) and bilevel images, respectively.

2.1 Review of Grayscale Image Similarity Metrics

Section 2.1.1 reviews traditional image similarity metrics. Texture similarity metrics are discussed in general in Section 2.1.2. SSIM metrics [12–14] are presented in Section 2.1.3 and Section 2.1.4. After that, Section 2.1.5 and Section 2.1.6 describe the STSIM metrics [3, 4, 15–17]. LBP [27] is reviewed in Section 2.1.7. Finally, LEBC [31] is introduced in Section 2.1.8.

2.1.1 Traditional Image Similarity Metrics

Traditionally, image similarity metrics are used to measure *similarity* of a distorted image to its original. Since a distorted image can be thought as the summation of its original and a noise image, the most straightforward way to measure their similarity is to assess the visibility of the noise image. The simplest and the most popular similarity metric based on this philosophy is the Mean-Squared Error (MSE), in which the noise image itself, *i.e.*, pixel value differences, are squared and averaged to give a similarity measurement.

However, without any consideration of the Human Visual System (HVS), MSE often fails to predict human perception of image similarity.

To understand how the HVS processes visual signals, pioneered by J. Mannos and D. Sakrison [60], many psychophysical and psychological experiments have been conducted in order to model the early stage low-level properties of the HVS. On the one hand, many metrics derived from MSE are proposed based on different perceptual models, including Laplacian Mean Square Error (LMSE), Peak Mean Square Error (PMSE), N. Mean Square Error (NMSE), N. Absolute Error, Weighted Distance *etc*. A review can be found in [61]. On the other hand, since it is well known that the HVS can be thought of as a spatial frequency filter-bank with octave spacing of subbands in radial frequency, and angular bands of roughly 30 degree spacing [62], it makes sense to consider frequency-domain-based metrics that extract features from subband coefficients instead of pixel values in the spatial domain. Based on this knowledge, many transformations, most are linear, have been proposed to mimic the HVS and decompose images into multiple subbands (or *channels* in the psychophysics literature) in frequency domain. Examples include the Cortex transform [63] that is related to the neural responses in the primary visual cortex, Laplacian pyramid [64] and steerable pyramid decomposition [65, 66]. Each of these transforms separate images into subbands in frequency domain with different scales and orientations. Perceptual models based on these transforms include the "Visible Differences Predictor" proposed by Daly [67], the model proposed by Lubin [68] and the "Perceptual Image Distortion" proposed by Teo and Heeger [69]. After decomposition, the errors between the distorted image and its original in each subband are calculated and normalized into units of just-noticeable-difference (JND), where the error definition varies for different metrics and JND is determined by the contrast sensitivity function of the HVS [60] and other metric-dependent factors. The final metric score is usually obtained by pooling these subband errors. Other than those transformations that are based on psychophysical study of the HVS, some simpler decompositions are also used in some image similarity metrics, including the discrete cosine transform (DCT) [70, 71] and separable wavelet transforms [72–74]. Reviews of such image similarity metrics can be found in [12, 62, 75]. Most of these metrics, no matter whether implemented in the spatial or transform domain, use pixel-by-pixel or coefficient-by-coefficient comparison weighted according to the sensitivity of the HVS to changes in each pixel or coefficient.

Although these traditional metrics are widely accepted and have been applied to many image processing applications, they still suffer from many limitations. For example, in some applications, pixel-by-pixel comparison is overly sensitive to image shifts and rotations and may result in inconsistency with human perception, especially for assessing texture similarity. A detailed discussion can be found in [3, 12]. To overcome such limitations, SSIM metrics have been proposed are they are reviewed in Section 2.1.3 and Section 2.1.4.

2.1.2 Texture Similarity Metrics

Similarity metrics discussed in the previous section are designed mainly for image quality assessments, which as mentioned before, are applied to assess the fidelity of a distorted image to its original. In this section and the next several sections, we review similarity metrics that are designed specifically for textures.

The first class of methods to measure texture similarity includes statistical methods in the spatial domain. One of the most popular methods is based on co-occurrence matrices [6–8], which count the number of intensity pairs for pixels apart from each other by some particular displacement vectors. The co-occurrence matrix is closely related to second-order image statistics [9, p. 220]. Another famous statistical method is based on the autocorrelation function of an image [76]. However, Julesz *et al.* have proven that the second-order statistics by themselves are not adequate to measure texture similarity [77, 78]. Recently, methods based on Local Binary Patterns (LBP) [27] have been widely studied. Details of LBP will be discussed in Section 2.1.7. These statistical methods are usually very simple to compute and hence easy to implement in practice. Although the HVS does a frequency analysis, Varma *et al.* showed that filter-banks are not necessary in texture classification and information provided in spatial domain is enough for assessing texture similarity [79].

The second class of texture similarity metrics includes the geometrical methods in the spatial domain, including those relying on Voronoi tessellations [19, 20] and structural displacement rules [21–23]. These methods assume textures are composed of texture elements with certain geometrical displacement rules. As a result, if the assumption fails, the methods are limited in power. In practice, they can only deal with very regular textures.

Another class of mothods is based on texture models. One widely used model is the Markov Random Field (MRF) model, in which each pixel in the image is assumed to depend only on its neighborhood. One example of the MRF model can be found in Section 2.2.2. In texture classification, each texture pattern is assigned to the most probable class based on the MRF model [10, 11]. However, not every texture pattern can be accurately represented using an MRF model, which is one of the main limitations of this method. Generally speaking, model-based metrics are restricted by the generality of their specific texture models.

The fourth class of texture similarity metrics uses transformations and is based on features in transform domain. As discussed in Section 2.1.1, the HVS does a frequency analysis, especially for textures that contain plenty of high frequency components. Metrics based on Gabor filters [80, 81] and steerable pyramid subband decomposition [3, 14, 16, 17] are proposed to measure texture similarity and have been proven to be effective in tasks like texture classification and perceptual image compression. STSIM metrics [3, 16, 17] belong to this class and have state-of-the-art performance in identical texture retrieval field. We will review them in Section 2.1.5 and Section 2.1.6. Recall Varma's opinion that filter-banks are not necessary in texture classification [79]. Since there exist well-designed metrics in both spatial domain and transform domain, there is still no clear conclusion on this debate.

Reviews of texture similarity metrics can be found in [9, p. 207-248] and [3].

2.1.3 Structural Similarity (SSIM) Metric

Instead of trying to explicitly model the functional properties of early stages of the HVS as traditional perceptual image similarity metrics do, *e.g.*, quantifying pixel (or coefficient) error visibility, Structural Similarity (SSIM) metrics attempt to simulate the hypothesized functionality of the overall HVS. The design of SSIM is based on the assumption that "*the human visual system is highly adapted to extract structural information from the viewing field*" [12]. Structural information characterizes dependencies among image pixels that "*carry important information about the structure of the objects in the visual scene*" [12]. Hence by measuring the change of structural information from one image to another, the perceptual image distortion can be assessed accordingly. The main challenge is to find good measurements of structural information using image statistics that can incorporate the overall functionality of the HVS, *e.g.*, local mean and variance of pixel values. SSIM tries to incorporate local image variance, mean and cross-correlation in image comparison, and can be applied in both the spatial domain (SSIM) [12] and complex wavelet domain (CW-SSIM) [14]. In the following we review SSIM, and then in the next subsection we introduce CW-SSIM. Note that SSIM metrics are not designed specifically for textures. Even though, they provide valuable insights for texture similarity study.

Given two images X and Y to be compared, SSIM first computes certain statistics in the spatial domain, namely, mean and variance, for both images within small sliding windows, *e.g.*, 7×7 or 11×11. We get to choose an appropriate window sliding step size, *e.g.*, 1 or 4 (both horizontally and vertically). And then for each window location, a window similarity score is computed by assessing and combining the similarity of the corresponding statistics within that window. The final metric score is the combination of all window similarity scores.

For each small sliding window x (with N pixels) in image X, the mean μ_x and variance

σ_x^2 statistics are:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \quad \sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2.$$

Compute the same statistics for the corresponding window y in image Y. Then the following formulas are used to assess similarities of mean and variance in this window location, which are called the luminance term $l(x, y)$ and the contrast term $c(x, y)$, respectively.

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\,, \quad c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\,.$$

where C_1 and C_2 denote small constants to avoid the denominator from being zero. Notice that both $l(x, y)$ and $c(x, y)$ are defined to be the ratio between the geometric mean and the algebraic mean, which gives values between 0 and 1, with 1 meaning identical.[1] In other words, $l(x, y)$ and $c(x, y)$ terms will produce values close to 1 for pairs of images with similar local means and local variances. Another issue to mention about the small constants C_1 and C_2 is that they increase robustness when the numerator and the denominator are very small. For example, suppose in $c(x, y)$, $2\sigma_x\sigma_y$ and $(\sigma_x^2 + \sigma_y^2)$ are small. Then C_2 dominates the numerator and denominator so that $c(x, y)$ is close to 1. This is desired because people tend to think that two smooth images, both with small variances, are similar to each other.

Besides the luminance term and the contrast term, in SSIM, there is another term $s(x, y)$, called the structure term, for each window location, which is actually not a comparison of statistics:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}\,,$$

where C_3 is a small constant similar to C_1 and C_2, and

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)\,.$$

[1]One may argue that the ratio of the geometric mean over the algebraic mean does not satisfy triangle inequality, which is an important property of distance measures. However, texture similarity metrics, *e.g.*, SSIM, are not rigorous metrics mathematically. As long as they can predict human perception of image similarity, triangle inequality is not a requirement for these metrics.

For any window location, the final score SSIM(x, y) is the combination of the three terms:

$$\text{SSIM}(x, y) = l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x.y)^{\gamma} ,$$

for some specified nonnegative values of α, β, γ, typically all chosen to be one. The SSIM metric score for two entire images X and Y is the combination of all window scores. Usually, the weighted algebraic mean of all window scores is applied.

Although SSIM outperforms many traditional image similarity metrics, such as MSE and PSNR, it still does not fully embrace the philosophy of comparison of statistics. In particular, the structure term is still a pixel-by-pixel comparison operator and restricts SSIM from measuring only structural information. Moreover, since SSIM is implemented in the spatial domain, the existence of the structure term makes it sensitive to small image shifts and rotations, which is inappropriate in some applications. To overcome such drawbacks, CW-SSIM has been proposed.

2.1.4 Complex Wavelet Structural Similarity (CW-SSIM) Metric

CW-SSIM [13, 14] is an SSIM-type metric implemented in the complex wavelet domain using the steerable pyramid subband decomposition [1, 66], which is a type of redundant wavelet transform that avoids aliasing in subbands [14]. Each subband captures image information at a certain scale and orientation, which mimics the early stage low-level properties of the HVS. Figure 2.1 shows a steerable pyramid subband decomposition with three scales and four orientations in complex wavelet domain.

Basically, it applies the SSIM metric to each subband in the complex wavelet domain. However, since the mean value of each subband is theoretically zero, the luminance term is not considered in the metric. Subband variances are computed within small sliding windows for each subband. For each window location c_x in image X (with N_c complex coefficients) and the corresponding window location c_y in image Y, the window metric score for

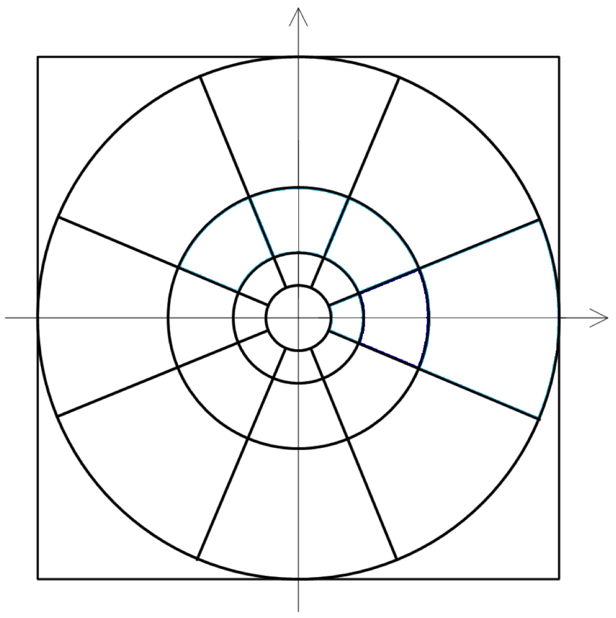


Figure 2.1: A three-scale and four-orientation steerable pyramid subband decomposition.

subband m, CW-SSIM$^m(c_x, c_y)$, is defined to be

$$\text{CW-SSIM}^m(c_x, c_y) = \frac{2\left|\sum_{i=1}^{N_c} c_{x,i} c_{y,i}^*\right| + C}{\sum_{i=1}^{N} |c_{x,i}|^2 + \sum_{i=1}^{N} |c_{y,i}|^2 + C},$$

$$c_x = \{c_{x,i} | i = 1, 2, \ldots, N_c\}, \quad c_y = \{c_{y,i} | i = 1, 2, \ldots, N_c\}.$$

where c_x and c_y are complex subband coefficients within window locations x and y, respectively. C is a small constant similar to the small constant in the luminance term in SSIM metric. Actually, the formula above is a combination of the contrast term and the structure term in SSIM. The final metrics score of CW-SSIM is a weighted summation of all these window metric scores.

CW-SSIM is simultaneously insensitive to small luminance change, contrast change, and geometric translation, scaling and rotation. This is because when such changes are small, they only cause small phase shift of subband coefficients, which will not influence the metric value very much [13]. However, the numerator of CW-SSIM formula is still a pixel-by-pixel operator. When applied to quantify image similarity, CW-SSIM is sometimes inconsistent with human perception, since structurally similar textures do not need to be pixel-by-pixel matched, even in the complex wavelet domain.

2.1.5 Structural Texture Similarity Metric (STSIM)

To quantify homogeneous texture similarity, X. Zhao *et al.* proposed the Structural Texture Similarity Metric (STSIM) [16]. Without any terms relying on pixel-by-pixel comparison like the structure term in SSIM, STSIM is truly a statistic-based similarity metric.

STSIM shares the same complex wavelet domain with CW-SSIM. However, different from CW-SSIM, the luminance term is included in STSIM, since although subbands (except the low-frequency band) are *zero-mean* over the *whole* image, this may not be true for small windows [3]. Additionally, STSIM includes two new terms involving the first-order horizontal and vertical autocorrelation coefficients of each image that will be described later. In short, local mean, variance, first-order horizontal and vertical autocorrelation coefficients are used in STSIM. Statistics are computed within local windows, sliding across each subband in the complex wavelet domain.

In particular, to compare the similarity of two images X and Y, for each window location c_x and c_y within subband m, first compute the luminance term $l^m(c_x, c_y)$ and the contrast term $c^m(c_x, c_y)$ as in SSIM. Notice that now these two terms are computed in the complex wavelet domain instead of the spatial domain. Second, compute the two new terms involving the first-order horizontal and vertical autocorrelation coefficients:

$$c^m_{0,1}(c_x, c_y) = 1 - 0.5|\rho^m_{c_x}(0,1) - \rho^m_{c_y}(0,1)|^p \,,$$

$$c^m_{1,0}(c_x, c_y) = 1 - 0.5|\rho^m_{c_x}(1,0) - \rho^m_{c_y}(1,0)|^p \,,$$

where $\rho^m_{c_x}(0,1)$ and $\rho^m_{c_x}(1,0)$ are the first-order horizontal and vertical autocorrelation coefficients for window location c_x within subband m:

$$\rho^m_{c_x}(0,1) = \frac{E[(c^m_x(i,j) - \mu^m_{c_x})(c^m_x(i,j+1) - \mu^m_{c_x})]}{(\sigma^m_{c_x})^2},$$

$$\rho^m_{c_x}(1,0) = \frac{E[(c^m_x(i,j) - \mu^m_{c_x})(c^m_x(i+1,j) - \mu^m_{c_x})]}{(\sigma^m_{c_x})^2}.$$

Actually, to avoid singularity, a small constant should be added to both the numerator and the denominator when $\rho^m_{c_x}(0,1)$ and $\rho^m_{c_x}(1,0)$ are computed. Notice that the two new terms, $c^m_{0,1}(c_x, c_y)$ and $c^m_{1,0}(c_x, c_y)$, only rely on image statistics.

The STSIM score for the window location c_x and c_y in subband m is computed by multiplicatively combining these four terms:

$$\text{STSIM}^m(c_x, c_y) = \left[l^m(c_x, c_y) \cdot c^m(c_x, c_y) \cdot c^m_{0,1}(c_x, c_y) \cdot c^m_{1,0}(c_x, c_y)\right]^{\frac{1}{4}}.$$

The STSIM metric score for two entire images is the combination of all window metrics scores in all subbands. Both algebraic and multiplicative combinations are proposed. Notice that since all statistics are computed within one subband, STSIM is an intra-subband metric. In the next subsection, we review an enhanced version of STSIM, called STSIM2, that relies on not only intra-subband statistics, but also inter-subband statistics.

2.1.6 Structural Texture Similarity Metric 2 (STSIM2)

Based on STSIM, J. Zujovic *et al.* proposed an inter-subband metric, called STSIM2 [3, 17]. STSIM2 shares the same complex wavelet domain with STSIM and contains all the terms in STSIM. In addition, STSIM2 adds new terms relying on the cross-correlations between different subbands, based on the belief that similar images should have similar inter-subband dependencies, and vice versa. In an identical texture retrieval application, STSIM2 has exhibited state-of-the-art performance. In Chapter 7, we introduce Matched Texture Coding (MTC) [18], where we apply STSIM2 to perceptual image compression.

And in Chapter 3, we propose a new texture similarity metric that outperforms STSIM2 in the identical texture retrieval application. Since STSIM2 plays such an important role in the following sections, we now discuss it in detail.

As mentioned earlier, STSIM2 contains all the terms in STSIM. In addition, it adds several new terms involving the cross-correlations between subbands. In particular, for two images X and Y, the new term in STSIM2 between subbands m and n at window locations c_x and c_y is

$$c_{0,0}^{m,n}(c_x, c_y) = 1 - 0.5|\rho_{c_x}^{m,n}(0,0) - \rho_{c_y}^{m,n}(0,0)|^p ,$$

where $\rho_{c_x}^{m,n}(0,0)$ and $\rho_{c_y}^{m,n}(0,0)$ denote the cross-correlations between subbands m and n at window locations c_x and c_y, respectively. Specifically, $\rho_{c_x}^{m,n}(0,0)$ is defined as

$$\rho_{c_x}^{m,n}(0,0) = \frac{E[(|c_x^m(i,j)| - \mu_{|c_x|}^m)(|c_x^n(i,j)| - \mu_{|c_x|}^n)]}{\sigma_{|c_x|}^m \sigma_{|c_x|}^n} ,$$

where the $(0,0)$ has the meaning that the cross-correlation between subbands m and n has no horizontal or vertical shifts, which is different from the first-order horizontal and vertical autocorrelations described in the previous subsection. Notice that in this formula, the magnitudes of subband coefficients are used instead of the raw subband coefficients. The cross-correlation term is computed between all adjacent subbands in the same orientation and all subbands in the same scale. This is based on the justification in [1] that although raw subband coefficients may be uncorrelated, their magnitudes may not; and in natural images, large magnitudes often occur at the same location in subbands at adjacent scales and orientations.

In summary, for each window location, STSIM2 first computes the luminance term, the contrast term and the two terms involving first-order correlation coefficients in the horizontal and vertical directions for each subband. For a three-scale and four-orientation steerable pyramid subband decomposition (see Fig. 2.1), there are 14 subbands (one lowpass subband, one highpass subband and 12 oriented subbands). Hence $14 \times 4 = 56$ terms are calculated. In addition, for each orientation, STSIM2 computes the terms involving cross-correlations between the magnitudes of subband coefficients at adjacent scales (two terms per orientation for a three-scale decomposition). And for each scale, STSIM2 computes the terms involving cross-correlations between the magnitudes of subband coefficients at all orientations (six terms per scale for a four-orientation decomposition). Thus, for a three-scale and four-orientation decomposition, totally there are $2 \times 4 + 6 \times 3 = 26$ additional terms involving the cross-correlations. So for each window location, totally $56 + 26 = 82$ terms are needed in STSIM2.

Finally, for each window location, one first multiplicatively combines the intra-subband terms for each subband as in STSIM, and then takes the algebraic mean of the 14 combined values and the 26 inter-subband terms as the window metric score. The final STSIM2 metric score for the two images is the weighted algebraic mean of all window metric scores.As can be seen, STSIM2 is computationally expensive. In Section 3.4.1, we compare the computational complexity of STSIM2 with a newly proposed structural texture similarity metric.

2.1.7 Local Binary Patterns (LBP)

Local Binary Patterns (LBP) [27] is a widely used feature in texture analysis. Texture similarity metrics based on LBP have been applied to many applications, *e.g.*, texture classification. Different from the SSIM and STSIM metrics, metrics based on LBP are *rotation-invariant*. In the following, we review the LBP feature first, followed by texture similarity metrics based on it.

First, LBP defines a pixel level circular texture operator. At each pixel c with value x_c, this operator extracts a rotation-invariant local feature and labels it with an integer pattern index. Specifically, given an integer R, and m locations uniformly located along a circle of radius R centered at c, *e.g.*, $R = 2$ and $m = 16$, the operator takes the current pixel value x_c and the m pixel values x_i at these locations if they fall into pixel grid, or interpolated values if not, and computes the pattern index between 0 and $2^m - 1$ as follows. For each neighboring pixel (or interpolated value) x_i, if its value is no less than the current pixel x_c, then a 1 is generated; otherwise a 0 is generated. Hence, at each pixel, an m bit binary representation is generated. After considering all circularly-shifted versions of this representation, the one corresponding to the smallest integer is chosen as the pattern index, $\text{LBP}_{x_c}^{m,R}$, for the center pixel x_c. The computation of $\text{LBP}_{x_c}^{m,R}$ is summarized as follows:

$$\text{LBP}_{x_c}^{m,R} = \min_{0 \leq n < m} \left[\sum_{i=1}^{m} u(x_i - x_c) 2^{[(i+n) \bmod m]} \right], \quad u(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} .$$

This index is invariant to circular rotation. For example, both “11100110 and “11001101 will give the same index of 55. $\text{LBP}_{x_c}^{m,R}$ represents the rotation-invariant pattern information in the vicinity of the current pixel x_c.

One can easily develop a texture similarity metric based on the operator described above. After calculating the pattern index for every pixel in an image, a histogram feature could be formed based on these indices. The basic idea is that similar histograms indicate similar texture patterns. Given two images, the similarity metric value is defined to be the similarity of the histogram features for these two images measured by some distance measurement, *e.g.*, the Kullback-Leibler divergence [82] and Earth Mover’s Distance (EMD) [83]. Specifically, in [27], the log-likelihood statistic is applied.

Usually, in practice, only the *uniform* LBP patterns are considered. A m bit binary representation is called a uniform pattern if there are no more than two 0/1 transitions. For example, “00000000” and “00000001” both represent uniform patterns, for they have zero and two 0/1 transitions, respectively. On the contrary, “00110011” is not a uniform pattern since there are four 0/1 transitions. The motivation to only consider the uniform patterns is that they are fundamental properties of texture [27]. In the method that only considers the uniform LBP patterns, all non-uniform patterns are thrown into a miscellaneous bin in the histogram feature. As a result, the histogram feature has much fewer bins and is therefore

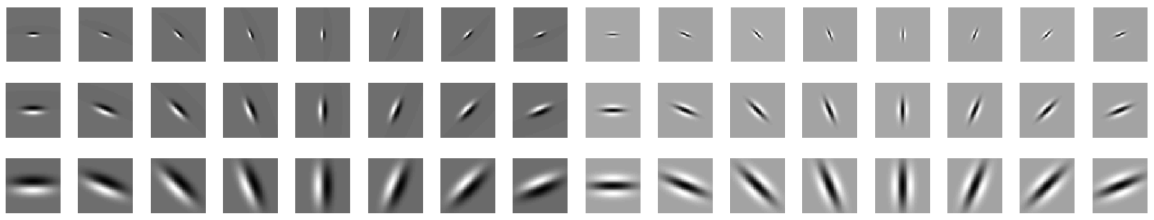


Figure 2.2: A 3-scale, 8-orientation subband decomposition using RFS filter bank. The left eight columns are the 24 edge filters and the right eight columns are the 24 bar filters. Filters in each row belong to the same scale in the subband decomposition.

much denser. In Section 3.4, we combine a similarity metric based on the LBP feature with only uniform patterns with a newly proposed texture similarity metric.

2.1.8 Locally Enhanced Binary Coding (LEBC)

Locally Enhanced Binary Coding (LEBC) [31] is a recently proposed texture similarity feature motivated by LBP [27] and CLBP [29]. LEBC is invariant to rotation and illumination changes in texture patterns. The proposed texture similarity metric based on LEBC achieves state-of-the-art performance in texture classification tasks as tested on the Outex [36] and CUReT [37] databases. In the following, we review the LEBC feature first, followed by texture similarity metrics based on it.

LEBC works in transform domain using the edge (first derivative) and bar (second derivative) filters from the Root Filter Set (RFS) [38, 39]. Specifically, a 3-scale, 8-orientation subband decomposition is applied, for both the edge and the bar filters, respectively. (In the notation of [38], the three scales are $\{(1,3),(2,6),(4,12)\}$.) Figure 2.2 shows the corresponding 24 edge and 24 bar filters. Each subband has the same size as the texture image in the spatial domain. For example, if the texture image has $N \times M$ pixels, each subband has $N \times M$ coefficients.

At each pixel location, we compute an LEBC pattern. Then the LEBC feature vector of the entire image can be derived from these LEBC patterns.

The LEBC pattern at each pixel location can be computed using the following four

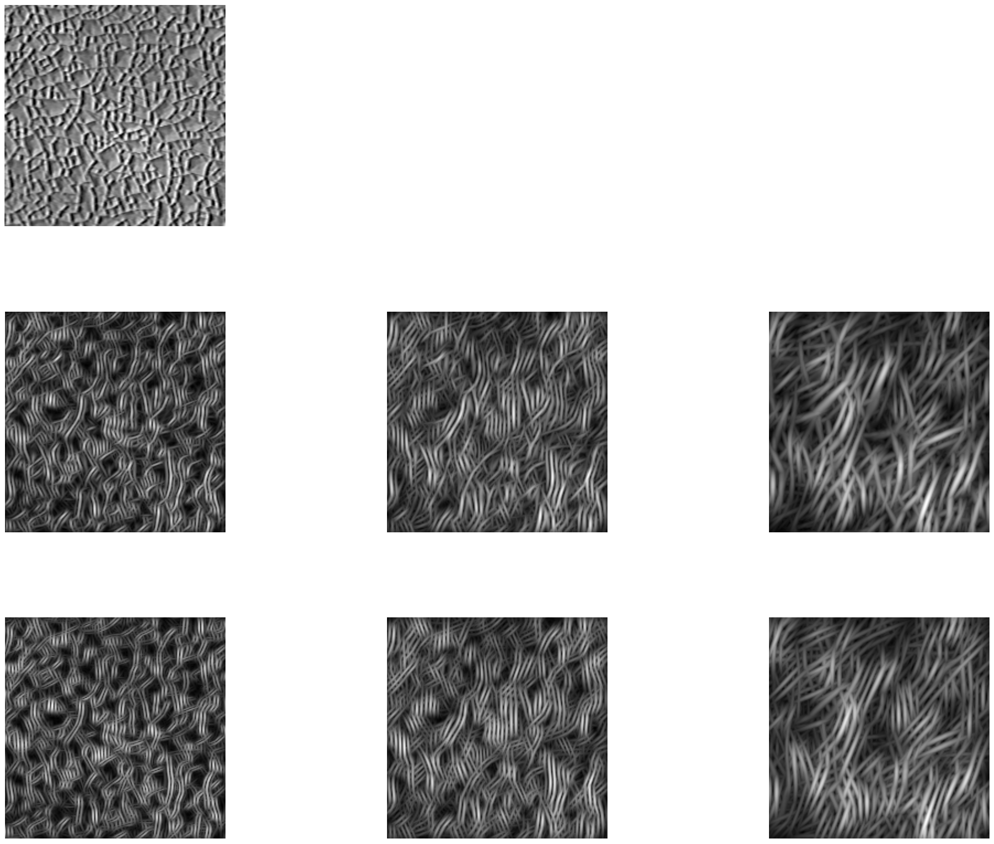


Figure 2.3: An example of a normalized texture image (first row) together with the six chosen subbands (second and third rows).

steps. First, one normalizes the texture image to have zero mean and unit variance. This step makes the LEBC pattern invariant to global affine illumination changes. Second, one applies the aforementioned subband decomposition to the normalized image, resulting in 48 subbands. In each scale of the edge or the bar filters, there are eight subbands each corresponding to a different orientation. To achieve rotation invariance, after comparing the variances of the coefficients in each of these eight subbands, choose the subband with the largest variance and discard the remaining seven. After this step, we end up having six subbands, each from a different scale of the edge or the bar filters. Figure 2.3 shows an example of the six chosen subbands of a normalized texture image. Third, the resulting 6-dimensional *filter response vector* $R(x)$ at pixel location x is normalized by the Weber's law [84]:

$$R(x) \leftarrow R(x)\big[\log(1 + ||R(x)||_2/0.03)\big]/||R(x)||_2 \,.$$

This results in a 7-dimensional vector F_x at pixel location x, which consists of the normalized pixel value at x plus the normalized 6-dimensional filter response vector $R(x)$, as shown in Fig. 2.3. Finally, at each pixel location x, define a binary operator similar to CLBP [29] using $F_x(i)$, $i = 1, 2, \ldots, 7$:

$$b_i(x) = \begin{cases} 1, & F_x(i) \geq \text{th}_i \\ 0, & \text{otherwise} \end{cases} \quad ,$$

where th_i is a threshold determined by the mean value of $F_x(i)$ over all pixel location x. With $\{b_1(x), b_2(x), \ldots, b_7(x)\}$, the LEBC pattern at pixel location x is:

$$L(x) = \sum_{i=1}^{7} b_i(x) 2^{i-1} \,.$$

$L(x)$ takes integer values between 0 and 127.

To form the LEBC feature of the entire image, first determine $L(x)$ and $\text{LBP}_x^{8,1}$ at each pixel location x. $\text{LBP}_x^{8,1}$ is computed on the original texture image and considers only *uniform* patterns. Specifically, $\text{LBP}_x^{8,1}$ takes integer value between 0 and 9. Then form a 128×10 2-D histogram H, with $H[m, n]$ equaling the number of pixel location x with $L(x) = m$ and $\text{LBP}_x^{8,1} = n$. H is the LEBC feature vector for the texture image.

A texture similarity metric based on the LEBC feature can be easily derived. Given two images, the similarity metric value is defined to be the similarity of the LEBC features for these two images measured by the chi-square distance measure.

In Section 4.4, we compare the performance of the texture similarity metric based on LEBC with a newly proposed rotation-invariant texture similarity metric in texture classification tasks.

Table 2.1: List of intensity-based overlap metrics.

Reference	Metric
Jaccard, 1912 [42]	$\frac{a}{a+b+c}$
Kulczynski, 1928 [43]	$\frac{a}{b+c}$
Kulczynski, 1928 [43]	$\frac{1}{2} \times (\frac{a}{a+b} + \frac{a}{a+c})$
Braun-Blanquet, 1932 [44]	$\frac{a}{\max(a+b, a+c)}$
Dice, 1945 [45]	$\frac{2a}{2a+b+c}$
Ochiai, 1957 [46]	$\frac{a}{\sqrt{(a+b)(a+c)}}$
Sokal & Michener, 1958 [85]	$\frac{a+d}{a+b+c+d}$
Simpson, 1960 [86]	$\frac{a}{\min(a+b, a+c)}$
Rogers & Tanimoto, 1960 [87]	$\frac{a+d}{a+d+2(b+c)}$
Sokal & Sneath, 1963 [88]	$\frac{2(a+d)}{2(a+d)+b+c}$
Sokal & Sneath, 1963 [88]	$\frac{a}{a+2b+2c}$

2.2 Review of Bilevel Image Similarity Metrics

In the following two subsections, a review of existing bilevel image similarity metrics, including intensity-based overlap metrics and SmSIM [50], will be given.

2.2.1 Intensity-based Overlap Metrics

The most commonly used objective metric to quantify bilevel image similarity is the percentage error, which is equivalent to mean-squared error (MSE) in the bilevel case. Even though percentage error is not always consistent with human perception, its simplicity and clear interpretation still make it the most popular metric. Motivated by trying to improve on percentage error, many intensity-based overlap metrics have been developed. These metrics are based on the overlap of the 1 and 0 regions in one image with those of the other image. They give nonnegative values with 0 representing little or no similarity and large values indicating high similarity. For all but the second Kulczynski metric, they assign 1 to identical images. A comprehensive review can be found in [14, 40, 41]. Given two bilevel images, the value assigned by any of these metrics can be expressed in terms of the following four overlap counts: a, respectively d, denotes the number of pixels that have intensity of 1, respectively 0, in both images; b, respectively c, denotes the number of pixels that have intensity 1 only in the first, respectively second image. The most popular of the many metrics of this sort are shown in Table 2.1. One can see that all metrics are symmetric with respect to the two input images, which may not be suitable for applications that focus more on one image than the other, such as compression, where one image is the original and the other a distorted reproduction. Among these metrics, the Dice [45] is most commonly used, especially in medical image processing. Its value is easily seen to equal

$$\frac{2}{1 + \frac{|B_1 \cup B_2|}{|B_1 \cap B_2|}},$$

where B_i is the set of pixels where image i has intensity 1 and $|C|$ denotes the number of pixels in set C. One can see from this that the metric depends strongly on the overlap of the regions with intensity 1 in both images; metric value 1 implies a perfect match, and 0 implies total mismatch. Note that like one or two other metrics, it does not depend on d, which implicitly presumes that 1's are more important than 0's.

2.2.2 SmSIM

Compared to PE, SmSIM [50] focuses more on the smoothness of the edges in a bilevel image. It measures the smoothness via the formalism of a pairwise Markov Random Field (MRF) model [50]. In a pairwise MRF model, the probability of a specific bilevel image x

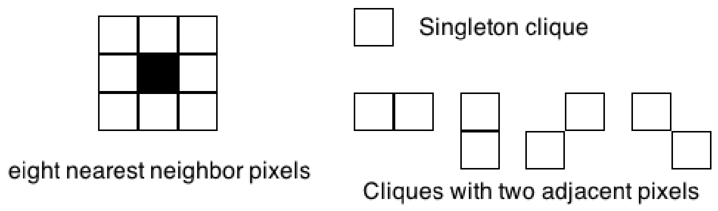

Singleton clique
eight nearest neighbor pixels
Cliques with two adjacent pixels

Figure 2.4: Clique examples.

is given by

$$p(x) = \frac{1}{Z}\exp\Big\{ - \sum_{c \in C} \Phi_c(x) \Big\} ,$$

where C denotes the collection of all cliques and $\Phi_c(x)$ is the clique potential function. Z is a normalizing constant to make $p(x)$ a valid probability measurement. A clique c is a subset of *adjacent* pixels in an image such that all pairs of pixels in c are considered to be neighbors. Figure 2.4 shows one pixel with its eight nearest pixels. Cliques with one and two pixels are also presented. In [50], for cliques consisting of a pair of adjacent pixels, the potential function assigns $-\beta$ if the pixels have the same value, and assigns $+\beta$ if they have different values. Hence, the MRF model penalizes images with plenty of 0/1 transitions by giving them low probability and favors smooth images by giving them high probability. Figure 2.5 presents two image patches, with the left one being smoother and the right one being rougher. Based on the MRF model described above, the left image should have higher probability than the right one.

SmSIM compares the MRF probability of the two image at every pixel location. In particular, for two images X and Y, at any pixel location s, obtain the conditional probability $p_X(s|n_s)$ and $p_Y(s|n_s)$, where n_s is the set consisting of the neighbors of s. Based on the MRF model described above, $p_X(s|n_s)$ is determined by the number of 0/1 transitions between s and each of its neighbors, which actually measures the local smoothness at pixel s. The more 0/1 transitions there are, the lower $p_X(s|n_s)$ will be. The similarity in smoothness between the two images X and Y at pixel location s is measured by the geometric

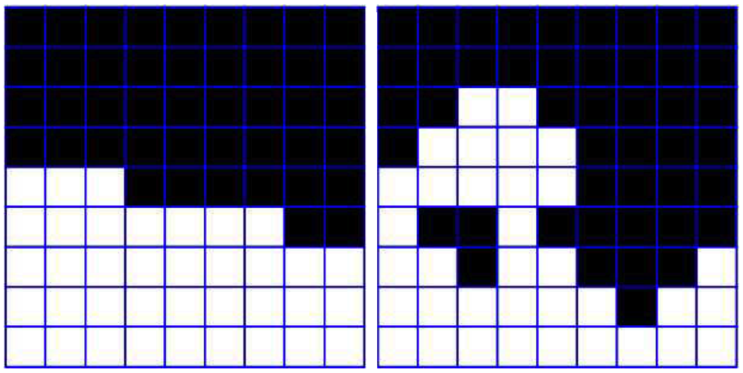


Figure 2.5: Left image: a high probability image patch. Right image: a low probability image patch.

mean over the algebraic mean of their MRF probabilities $p_X(s|n_s)$ and $p_Y(s|n_s)$:

$$\text{SmSIM}(X, Y, s) = \frac{2\sqrt{p_X(s|n_s)p_Y(s|n_s)}}{p_X(s|n_s) + p_Y(s|n_s)} ,$$

which ranges between 0 and 1, with 1 meaning identical. This metric can be calculated at each pixel location in the image. Then the algebraic average over the entire image, SmSIM(X, Y), gives the overall image similarity on smoothness. To further refine the metric, one can consider the horizontal and vertical 0/1 transitions separately from the diagonal ones. In particular, SmSIM$_h(X, Y)$ only considers cliques with two pixels next to each other in the horizontal direction. Similarly, SmSIM$_v(X, Y)$, SmSIM$_d(X, Y)$ and SmSIM$_{d'}(X, Y)$ only consider cliques with two pixels next to each other in the vertical, diagonal and anti-diagonal directions, respectively.

However, for images to be similar, having similar smoothnesses alone is not enough. The pixel values of the images at the same location must also agree with each other. In order to present pixel level accuracy, singleton cliques, which are cliques contain only one pixel (see Fig. 2.4), are added with potential functions that assign $-\alpha$ when the pixels of the two images agree with each other, and assign $+\alpha$ when they do not. The metric only considers singleton cliques, SmSIM$_\alpha(X, Y)$, actually measures the pixel level accuracy and gives high probability to images with low percentage error (PE).

Thus, separate MRF models, each with one type of clique, are considered and multiplicatively combined to get the final metric score:

$$\text{SmSIM}(X, Y) = \text{SmSIM}_\alpha(X, Y) \cdot \text{SmSIM}_h(X, Y) \cdot \text{SmSIM}_v(X, Y)$$

$$\cdot\, \text{SmSIM}_d(X, Y) \cdot \text{SmSIM}_{d'}(X, Y)\,.$$

In this way, the SmSIM metric score is the combination of smoothness similarity in four different directions along with the pixel level accuracy. By multiplicatively combining the five terms, it requires all five terms to be close to 1 in order to have a high SmSIM score.

In Chapter 8, the performance of SmSIM is evaluated. And results show that SmSIM cannot reflect human perception of bilevel image similarity especially well. However, we believe that the idea that similar images should have not only similar smoothnesses, but also pixel level accuracy, is a good one.

CHAPTER 3

Local Radius Index (LRI)

As mentioned in Section 1.1, we focus on the need for texture metrics that assess the similarity of homogeneous textures in the context of image compression and retrieval where, as mentioned earlier, changes in rotations, scalings and viewpoints should be penalized monotonically. This is the domain for which STSIM type metrics have been developed [3, 4, 89, 90]. With this in mind, we propose a new type of statistical texture feature, called Local Radius Index (LRI), and new structural texture similarity metrics based on this feature in combination with other texture features, such as LBP [27], the newly proposed Subband Contrast Distribution (SCD) and Intensity Penalty (IP). These new metrics are computationally much simpler than STSIM metrics.

3.1 Local Radius Index (LRI)

Texture is a unique kind of visual signal and does not have a widely agreed definition. However, Portilla and Simoncelli [1] provide a general definition with which we agree: *"Loosely speaking, texture images are spatially homogeneous and consist of repeated elements, often subject to some randomization in their location, size, color, orientation, etc."* Texture features aim at representing texture characteristics in order to compare texture similarity. When projected to feature space, perceptually similar textures should be "closer" than dissimilar ones according to some pre-defined distance measure. Generally speaking, a texture contains repetitive smooth regions and transition regions between them, *i.e.*, edges. The sizes of the smooth regions influence the repetitive structure of the texture elements and can be captured by considering edges around them. The distance between two adjacent edges, which we will refer to as an *inter-edge distance*, is a good feature. Due to stochasticity, the inter-edge distances that measure the widths of the smooth region are not constant, but there is a distribution of such. Also, the inter-edge distance distributions may vary with angle and such variations can be key to characterizing the texture. Accordingly, one version of LRI (LRI-A) is designed to capture inter-edge distance distributions in different directions by focusing on pixels adjacent to edges. Similarly, another version of LRI (LRI-D) is designed to capture the distributions of distances-to-nearest-edges as a key texture feature/statistic. In particular, we propose two *LRI operators* that for each pixel output eight integer *directional indices*, each corresponding to one direction. Then for each direction, a histogram of the corresponding directional indices is computed. The collection of all eight histograms is considered an *LRI feature vector* or *statistic* for the image.

3.1.1 LRI-A

As mentioned above, the first version of LRI, called LRI-A, is designed to capture the inter-edge distance distributions by measuring the width of the Adjacent smooth regions for pixels next to edges. Intuitively, one could measure the inter-edge distance distribution at a given angle by passing a line at that angle through the image and recording lengths of regions and then compiling statistics for all lines of this angle. What we do is different but equivalent.

For the ith pixel, the *LRI operator* produces eight directional indices $I_{i,1}, \ldots, I_{i,8}$, each corresponding to the direction of one of its eight nearest neighbors. In particular, $I_{i,1}$ corresponds to the nearest neighbor to the right of i, $I_{i,2}, \ldots, I_{i,8}$ correspond to the other seven nearest neighbors in counterclockwise order, and given an *intensity change threshold* T,

1. $I_{i,d} = 0$ if the absolute difference between the current pixel and the adjacent one in direction d is less than T.

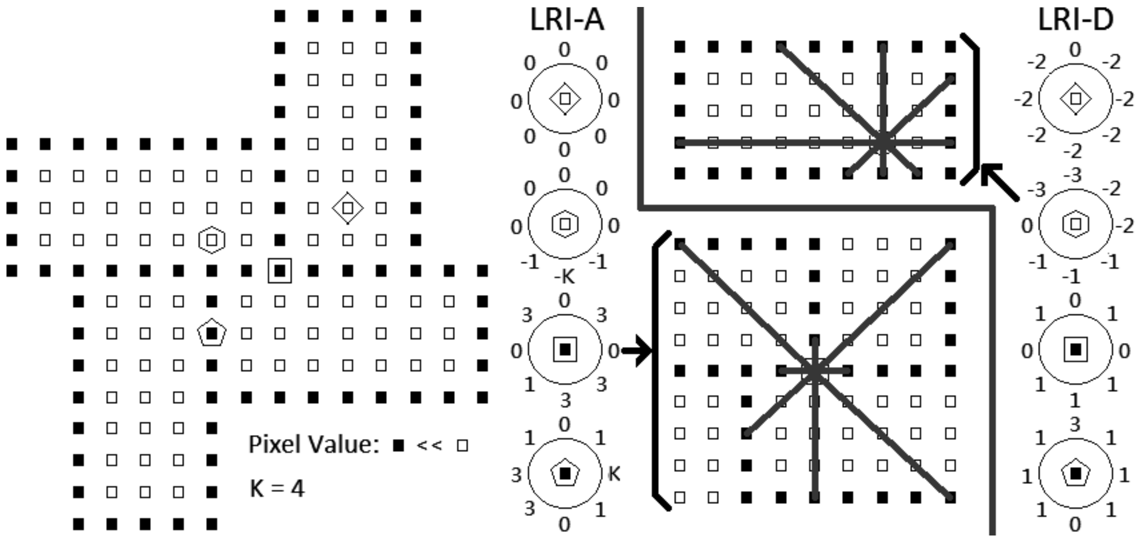


Figure 3.1: Examples of LRI-A and LRI-D directional indices.

2. $I_{i,d} = \min\{j, K\}$ if $j > 0$ successive pixels in direction d are greater than the current pixel by at least T, and the $(j+1)$th pixel is not.
3. $I_{i,d} = \max\{-j, -K\}$ if $j > 0$ successive pixels in direction d are smaller than the current pixel by at least T, and the $(j+1)$th pixel is not.

Figure 3.1 illustrates LRI-A directional indices for several pixels. Note that $I_{i,d} = 0$ means pixel i is not adjacent to an edge in direction d, whereas $I_{i,d} \neq 0$ indicates the presence of an adjacent edge in direction d and $|I_{i,d}|$ represents the size of the corresponding adjacent texture element. In addition, the sign of $I_{i,d}$ indicates whether the edge is an increasing or decreasing one.

As suggested earlier, the inter-edge distance distributions in each direction are captured by computing histograms of the directional indices in each direction. The *LRI feature vector* F_x for an image x then consists of these eight histograms.

Of the two parameters involved in the operator (and feature vector), the intensity change threshold T determines what is considered an edge and also controls noise sensitivity. Large T makes the operator noise insensitive and only sharp edges are detected. Small T has the opposite effect. T should be image dependent, and we have found that for homogeneous textures, choosing it to be proportional to the standard deviation of the image works well. In this chapter we choose it to be the standard deviation divided by 2. With this choice, the LRI operator is invariant to any affine transformation of gray-scale, *i.e.*, gray-scale or contrast invariant. The size parameter K limits the maximum size of texture elements detected by LRI-A. By fixing K, LRI cannot distinguish texture elements with larger sizes. In particular, the LRI feature vector in direction d contains the following information:

i. $\Pr(I_{i,d} = 0)$ indicates the percentage of pixels not adjacent to an edge in direction d. It measures the "busyness" of the texture pattern in direction d.

ii. $\Pr(I_{i,d} = k)$, $k = -K+1, \ldots, -1, 0, 1, \ldots, K-1$, indicates the percentage of pixels adjacent to a texture element with size $|k|$ in direction d. The sign of k distinguishes increasing edges from decreasing ones.

iii. $\Pr(I_{i,d} = k)$, $k = \pm K$, indicates the percentage of pixels adjacent to a texture element with size greater than or equal to $|k|$ in direction d. The sign of k distinguishes increasing edges from decreasing ones.

From this analysis one can see that larger K is preferred than smaller K. However, it also increases the computational complexity of LRI. From experimental results, we found that $K = 4$ is usually large enough for most applications. This choice of K is very small relative to the sizes of many texture elements. However, the fact that it works well suggests that the distributions of inter-edge distances are so powerful that only a small fraction of them is capable of distinguishing most texture patterns. Another viewpoint is that most of the distinctive information of texture patterns is contained in the inter-edge distance distributions with short lengths. Note that the number of directions considered in LRI also helps improve its distinguishing power, since it is highly unlikely for two distinct texture patterns to have similar inter-edge distance distributions in all the eight directions.

One may wonder why the LRI-A operator (and the LRI-D operator in Section 3.1.2) treats an increasing edge differently from a decreasing one. To see why, consider the his-

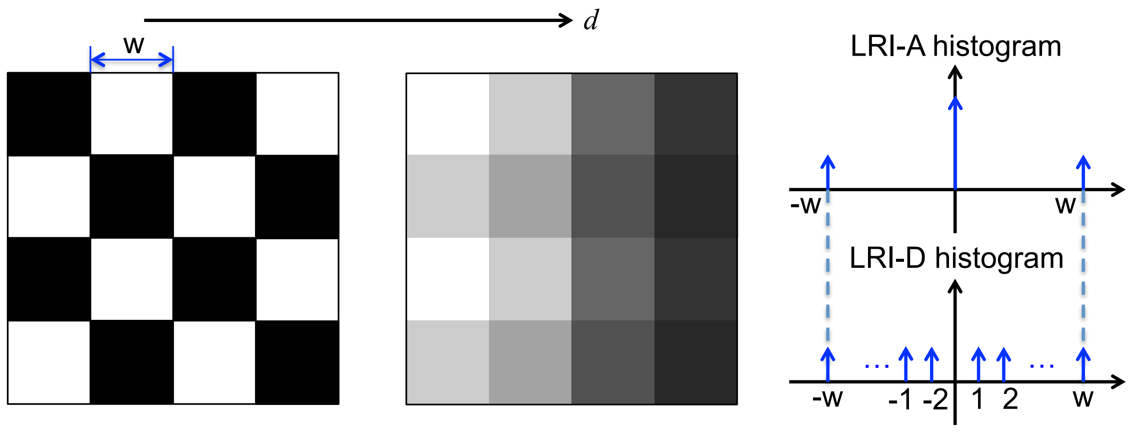


Figure 3.2: Two texture patterns. LRI-A and LRI-D histograms in horizontal direction d shown for the checkerboard pattern.

tograms of LRI horizontal indices for the two patterns shown in Fig. 3.2. If the indices were unsigned, they would be the same, for each would entirely concentrate at the width of a square. However, with signed indices, the histogram for the left pattern concentrates at both positive and negative values, while that for the right concentrates on negative values only.

3.1.2 LRI-D

Whereas LRI-A captures the width of the Adjacent smooth region, LRI-D captures the Distance to the nearest edge from every pixel, *i.e.*, to the boundary of the next smooth region, again based on an intensity change threshold T and a size limit K. Specifically, the LRI-D operator calculates directional indices $I_{i,d}$ for the ith pixel in direction $d = 1$ to 8 as follows.

$I_{i,d} = \pm \min(j, K) \bmod K$, if for some $j \geq 1$, $j-1$ successive pixels in direction d have absolute difference with the current pixel less than T, and the jth pixel is greater/smaller than the current pixel by at least T.

Note that $I_{i,d} = 0$ means the distance from the current pixel to the nearest edge in direction d is more than K, or there is no such edge. Otherwise $|I_{i,d}|$ indicates the distance to such an edge and the sign of $I_{i,d}$ indicates whether the nearest edge is an increasing or decreasing one. Figure 3.1 illustrates the LRI-D directional indices for several pixels. The roles of T and K in LRI-D are similar to those in LRI-A. And as with LRI-A, the feature vector F_x for an image x is the collection of directional index histograms.

To accentuate the difference between LRI-A and LRI-D, consider their respective indices in direction d as one traverses a smooth region along a line with direction d from one of its boundaries to the next. Notice that the LRI-A indices are all zero, except for the last, corresponding to the pixel adjacent to the second boundary, whereas the LRI-D indices count down the distance to the second boundary. As a result, LRI-A indices are sparser, and LRI-D indices contain nonzero redundant information. Figure 3.2 illustrates this idea. The LRI-A and LRI-D histograms in horizontal direction of a checkerboard texture pattern are shown, with the LRI-A histogram being much sparser. Another viewpoint is that LRI-A focuses on representing size information of texture elements, whereas LRI-D concentrates on distance to adjacent texture elements.

3.1.3 Nontrivial Edge Detectors

In both LRI-A and LRI-D, an intensity change threshold T is used to determine the existence of boundaries between texture elements. This can be seen as a kind of trivial edge detector, which leads one to wonder if a nontrivial edge detector would improve LRI. Accordingly, we experimented with two edge detectors: Canny [91] and ACA (Adaptive Clustering Algorithm) [2]. Figure 3.3 shows results for two examples. The first and third columns display (from top to bottom): the original images, edges produced by Canny edge detection, and 4-level ACA segmentation, respectively. One can see that in textured regions, both Canny and ACA, due to continuity and robustness constraints, tend to miss some edges that are important for texture characterization. For example, several squares are either missing or combined. Such inaccurate edge profiles will negatively influence the resulting feature vector. Moreover, in this application, edge continuity is not so important.

Table 3.1: Identical texture retrieval experiment using LRI-A (K = 4) on the Corbis-based database.

Metric	P@1	MRR	MAP
LRI-A on original image	91.8%	93.5%	86.9%
LRI-A on Canny edge profile	88.3%	90.9%	83.3%
LRI-A on ACA segmented image	88.5%	91.2%	83.6%

The second and fourth columns display the sum of LRI-A directional indices for each pixel when the LRI-A operator is applied to the adjacent left image. Clearly, the LRI-A plots at the top are more informative. One should also keep in mind that conventional edge detectors are designed with the goals of producing smooth and connected edges, neither of which is needed for LRI. Indeed, as a statistical feature vector, it is rather insensitive to these properties.

To verify that the trivial edge detector results in a better LRI metric, we ran one of the identical texture retrieval experiments described in Section 3.5 with the trivial edge detector replaced by Canny and ACA. The results in Table 3.1 confirm that LRI works best with the trivial edge detector.

3.1.4 Computing LRI Indices

Computing the output of either LRI operator by directly using the definitions in Sections 3.1.1 and 3.1.2 is fairly simple, as it only involves comparing the current pixel with its eight neighbors. Specifically, it requires up to $8K$ such comparisons per pixel. Nevertheless, it can be reduced to approximately $4K$ per pixel by exploiting the fact that the comparison between any pair of pixels is used twice, once for each pixel. In particular, for each of the four unsigned directions d (horizontal, vertical, diagonal and anti-diagonal) and for each k from 1 to K, one can precalculate differences between all pairs of pixels separated by k in direction d and store them as a *pixel difference image*. For instance, the pixel difference

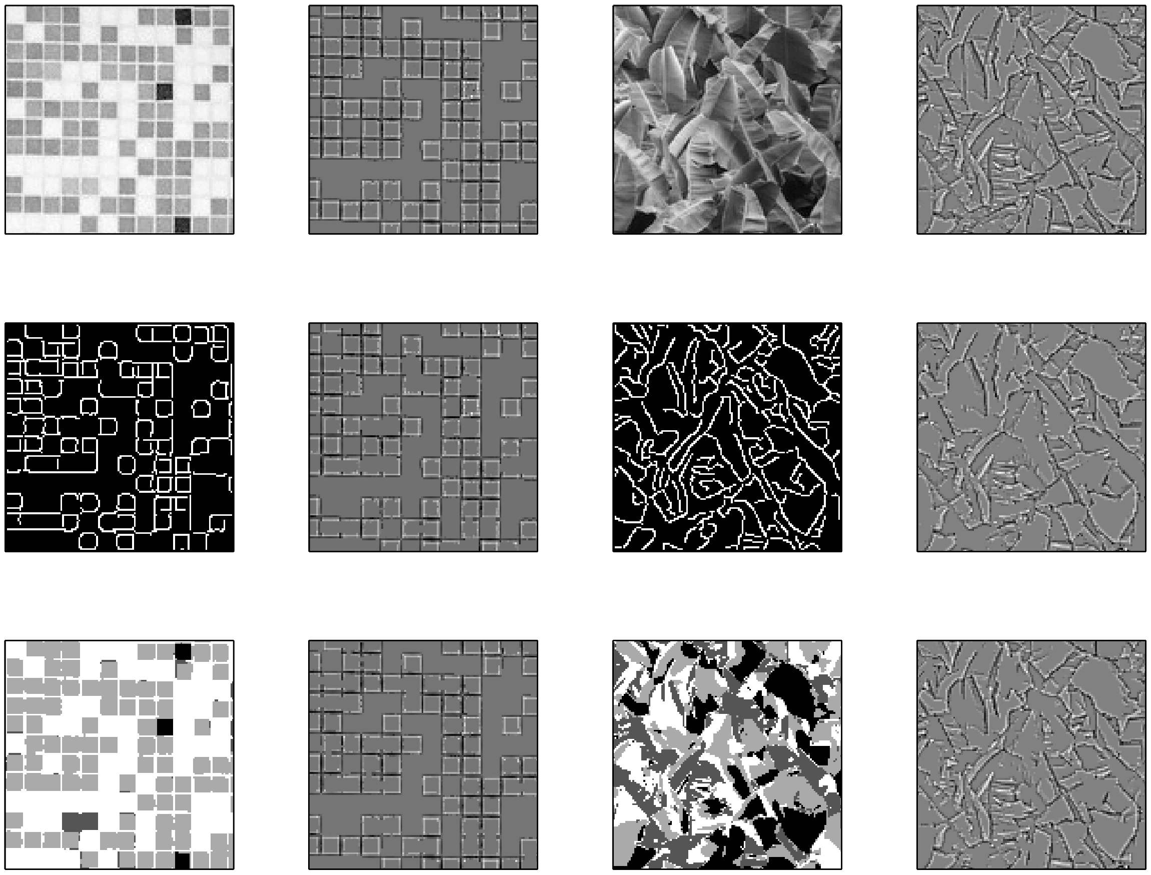


Figure 3.3: Comparing different edge detectors.

image $\Delta^{h,k}$ for pixels separated horizontally by k has pixels $\Delta^{h,k}_{m,n} = x_{m,n} - x_{m,n+k}$, where for an $N \times N$ image, $1 \leq m \leq N$ and $1 \leq n \leq N - k$. Such computations can be viewed as filtering operations, with a total of $4K$ filters, namely, K in each of the four unsigned directions. For example, the magnitude of the frequency response for the k^{th} filter in the horizontal direction has the form:

$$|H^{h,k}(\omega_X, \omega_Y)| = 2 - 2\cos(k\omega_Y)\,.$$

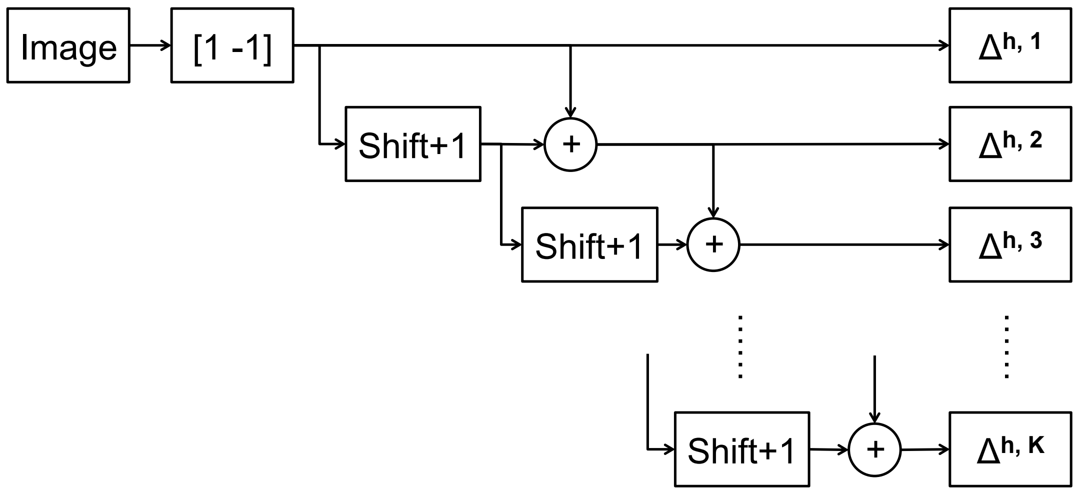


Figure 3.4: Computing horizontal difference images. “Shift+1” indicates a horizontal shift of one pixel.

Such filters differ in orientation and scale, and though not orthogonal, they can be viewed as producing a subband decomposition. As discussed later, their outputs can also be used to speed the computation of another useful feature.

As another perspective, consider the horizontal difference filters. Once $\Delta^{h,1}$ is obtained using a filter with impulse response $[1, -1]$ (as in the Haar wavelet), other horizontal pixel difference images can be produced as shown in Fig. 3.4.

The precise complexity, measured in operations per pixel (opp), of computing the pixel difference images is:

$$4K - \frac{3K(K+1)}{N} + \frac{K(K+1)(2K+1)}{3N^2}.$$

This number takes into account boundary conditions and is usually very close to $4K$. For example, when $N = 128$ and $K = 1, 2, 3, 4$, the complexities of computing the pixel difference images are 3.95, 7.86, 11.72 and 15.53 opp, respectively.

3.1.5 A Purely LRI-based Similarity Metric

A texture similarity metric based purely on LRI can be easily derived. First, for each of two images, compute and concatenate the eight histograms to form the $8 \times (2K + 1)$ dimensional feature vector and normalize it to unit length. Then, compare the similarity of the vectors obtained using some measure of histogram similarity, which here we choose to be Kullback-Leibler divergence [82]. The resulting metric has non-negative values, with 0 meaning identical, and a small value indicating high similarity. Because a small metric value suggests that all eight histograms are similar and because each histogram represents the distribution of inter-edge-distances in some direction (or distances-to-nearest-edge), it provides strong support for the hypothesis that the two images have similar texture.

Note that divergence $D(X||Y)$ is asymmetric in its arguments. In texture retrieval and classification, X will usually be the query image and Y a candidate in the database. Similarly, in image processing and compression, X will usually be the original and Y a reproduction.

3.2 Subband Contrast Distribution (SCD)

Subband decompositions oriented to visual perception have been central to a number of perceptually oriented similarity metrics including those based on Gabor filters [80, 81], cortex transforms [63, 92], steerable pyramid subband decompositions [3, 13, 14, 16, 17], and others [68, 93]. We find that the variances in each subband, which indicate subband contrasts, provide useful complementary statistics to LRI, which by itself is contrast invariant.

Specifically, we propose two closely related features that we collectively refer to as *Subband Contrast Distribution* (SCD). The first applies the *real* steerable pyramid decomposition [1] with three scales and four orientations, similar to the *complex* steerable pyramid decomposition used in CW-SSIM and STSIM. Subband coefficients from a real steerable pyramid decomposition are two times the real parts of subband coefficients from a complex one. The real and imaginary parts of subband coefficients from a complex steerable pyramid decomposition form Hilbert pairs. If we do not care about phase information, the imaginary parts of subband coefficients are unnecessary. Since we only need contrast information for each subband, a real decomposition, which is faster than a complex decomposition, is enough.

The resulting feature for image x is the SCD vector $F_x = (\sigma^2_{x,1}, \ldots, \sigma^2_{x,12})$ of 12 subband variances. The similarity of the SCD vectors from two different images, x and y, is assessed using the approach taken in SSIM type metrics [12]:

$$\text{SCD}(x,y) = \prod_{i=1}^{12} \frac{2\sigma_{x,i}\sigma_{y,i} + C}{\sigma^2_{x,i} + \sigma^2_{y,i} + C},$$

where $\sigma^2_{x,i}, \sigma^2_{y,i}$ denote the variances of the i^{th} subband of x and y, respectively. As such, SCD(x,y) ranges from 0 to 1, with 1 indicating identical. Multiplicatively combining all terms has the interpretation that only when all subbands have similar contrasts can SCD have a high value. C is a small positive constant that prevents the denominator from being zero and improves robustness when subbands have small energy. Typically, $C = 10$, when images take values from 0 to 255.

The second approach makes use of the $4K$ pixel difference images produced when computing LRI directional indices. As mentioned previously, these are filtered versions of an image, with differing orientations and scales. As such, they can be used in place of the steerable pyramid decomposition. The statistic formation and similarity scoring formula are the same as before. To distinguish this approach from the previous, it will be called estimated SCD or SCD$_{\text{EST}}$.

The advantage of the first approach is that every σ^2_i has a clear interpretation: it represents the energy at a certain scale and orientation. The advantage of the second approach is that it needs essentially no additional computations after LRI is computed. Experimental results in Section 3.5 show that although methods using SCD work a little better than those

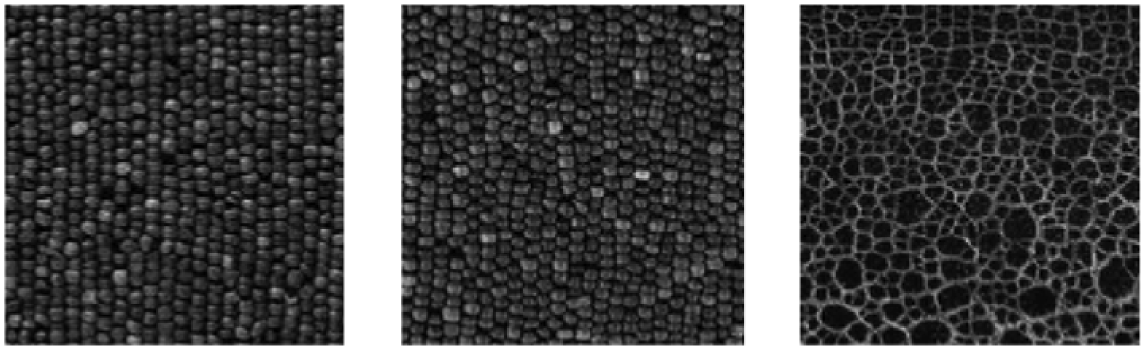


Figure 3.5: Retrieval failure example using SCD.

using estimated SCD, the difference is not significant.

SCD by itself penalizes image transformations that cause energy shifts among different subbands, such as size scaling and rotation. Note that small scalings and rotations are tolerated since the steerable pyramid has only three scales and four orientations. In perceptual image compression, we need scale- and rotation-sensitive metrics like SCD. However, in identical texture retrieval, SCD is sometimes too sensitive and not so consistent with human perception. Figure 3.5 shows an example of failure in retrieval of identical texture using (only) SCD as the similarity metric. The image in the middle is the query image and the right image is the first retrieved image. Human observers would agree that the left image is more similar to the query than the right one. However, notice that the texture pattern in the left image is more vertically aligned than the query image. As a result, the energy in the left image is more concentrated in subband that characterizes horizontal changes. Since generally the most dissimilar subband dominates the SCD score, the left image is not likely to yield the smallest metric value to the query in the whole database. Even though sometimes SCD is not so consistent with human perception, when combined with LRI and other features with complementary information, much better retrieval performance can be achieved, as can be seen in Section 3.5.

3.3 Intensity Penalty (IP)

Besides texture pattern and contrast, intensity can also influence human similarity judgment. To penalize large intensity differences, we propose the following penalty function:

$$\mathrm{IP}(x,y) = \left[\frac{\max\left(T_L, \left|m_x - m_y\right|\right)}{255}\right]^{p_L}, \; T_L = 10\,, \; p_L = 2\,,$$

where m_x and m_y are the mean values of images x and y, T_L serves as a kind of just noticeable intensity difference, and p_L determines the severity of the penalty. The denominator 255 is chosen presuming images have pixel intensities ranging from 0 to 255. The value of IP(x,y) ranges from $(T_L/255)^{p_L}$ to 1 with small values, *i.e.*, small penalties, indicating similar intensities. This metric is not needed for applications in which penalizing intensity differences is not important.

3.4 LRI-based Texture Similarity Metric

After experimenting with a number of ways of combining LRI with complementary features, we propose the following metric combining LRI with LBP, SCD (or SCD$_{\mathrm{EST}}$) and IP:

$$\mathrm{LRI}^{+} = \mathrm{LRI}^{p_1} \times \mathrm{LBP}^{p_2} \times f(1-\mathrm{SCD})^{p_3} \times \mathrm{IP}\,,$$

where LRI^{+} is the resulting metric value and $f(t)$ is an increasing function of t on $[0,1]$. We use $f(t) = \tan(\frac{\pi}{2}t)$, but also experimented with $f(t) = t$. Parameters p_1, p_2, p_3 weight the different features, with typical values 1, 1.1 and 1.2, respectively. Actually, metric performance is not very sensitive to the choice of these three parameters.

LRI^{+} is nonnegative, with 0 meaning identical, and a small metric value meaning similar. Since the values of LRI^{+} are often close to 0, for convenience, we usually report the logarithm of LRI^{+}. As a benchmark, we have found that $\log_{10}$ LRI$^{+} = -6$ indicates very similar textures. In Section 3.5, LRI^{+} is tested on its ability to perform identical

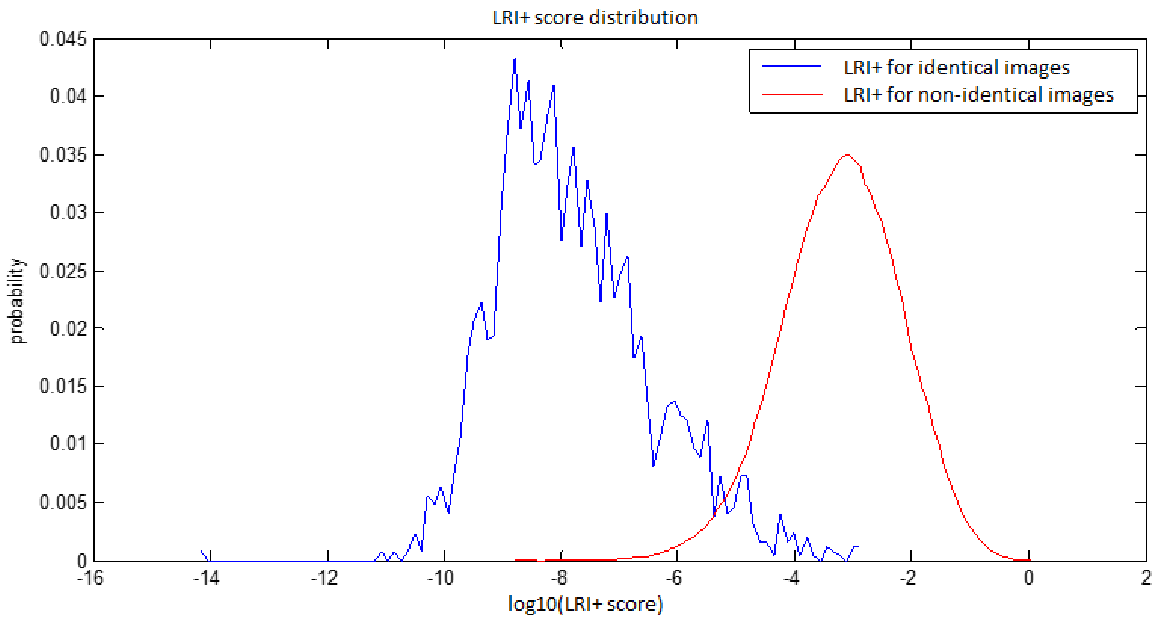


Figure 3.6: $\log_{10}(\mathrm{LRI}^{+})$ scores between all identical and non-identical image pairs in the Corbis-based database.

texture retrieval on the Corbis-based database [3] having 1181 images with each pair labeled as "identical" or "not identical". Figure 3.6 shows the normalized distributions of $\log_{10}(\mathrm{LRI}^{+})$ scores between all identical and non-identical image pairs in the Corbis-based database. Since there are more non-identical image pairs than identical ones, its distribution is smoother. The fact that the overlapping of the two distributions is small support the hypothesis that the proposed LRI^{+} is a distinctive metric that can distinguish identical texture image pairs from non-identical ones.

As another demonstration of how the metric behaves, Figures 3.7, 3.8 and 3.9 show the distance from three hand-picked target blocks (one has homogenous texture, one is smooth, and one is a mixture of the two), each to eight hand-picked candidate blocks, respectively. One can see that as image similarity decreases, the LRI^{+} metric produces higher scores. In addition, the LRI^{+} metric works well not only in homogeneous texture regions, but also in inhomogeneous texture regions and smooth regions. This is useful in applications, such as Matched Texture Coding (MTC) discussed in Chapter 7, where LRI^{+} is called upon to assess the similarity of candidates for textured, smooth and mixed blocks.

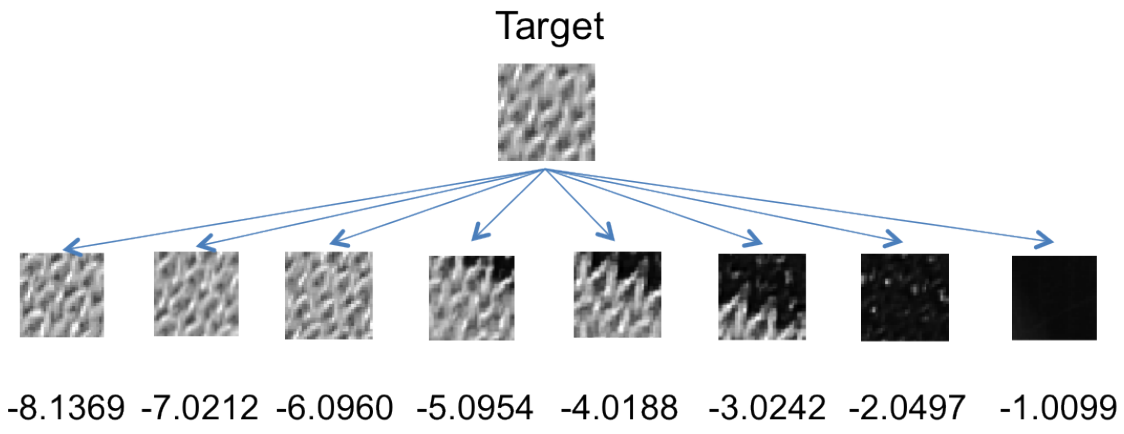


Figure 3.7: Typical candidate blocks and the their corresponding $\log_{10}$ LRI$^+$ metric scores with a target block with homogeneous texture.

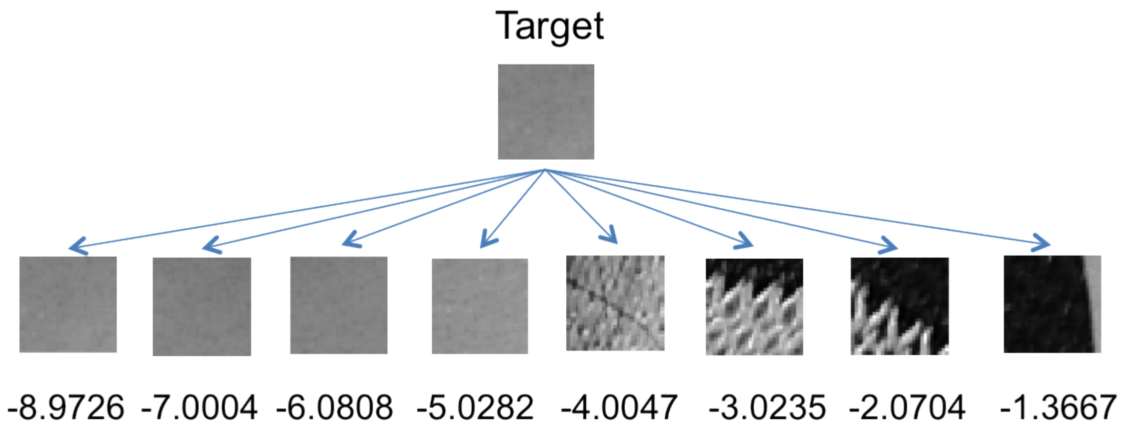


Figure 3.8: Typical candidate blocks and the their corresponding $\log_{10}$ LRI$^+$ metric scores with a smooth target block.

3.4.1 Computational Complexity

An LRI-based metric requires many fewer computations than STSIM2. Leaving aside transform costs, STSIM2 (with a global window) requires approximately 500 operations per pixel (with parameters as shown in [3]), whereas LRI and SCD require 16 ($4K$ with $K = 4$) and 36, respectively. Of course both STSIM2 and SCD require the steerable pyramid decomposition, the computation of which requires 14 FFT, each requiring $O(\log N)$ operations per pixel for images with N pixels [94]. However, when SCD$_{\text{EST}}$ is applied instead of SCD, essentially no additional computations are needed beyond those needed for

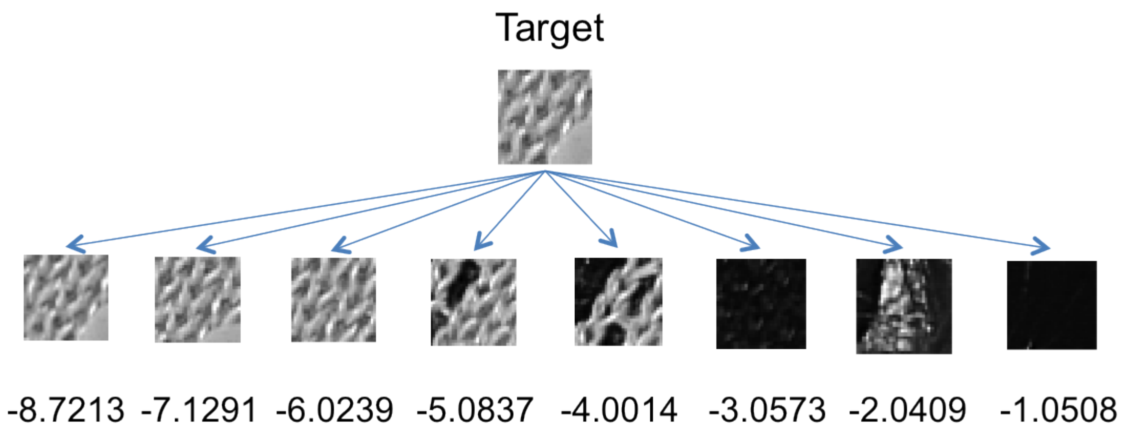


Figure 3.9: Typical candidate blocks and the their corresponding $\log_{10}$ LRI$^+$ metric scores with a target block with inhomogeneous texture.

LRI.

3.5 Metric Performance Evaluation

To assess the performance of LRI$^+$, we consider how well it works in identical texture retrieval, as formulated in [3, 4, 15] and in Matched Texture Coding (MTC) [18]. In both experiments, STSIMs have state-of-the-art performance, and our goal is to achieve similar or better performance, with lower computational complexity. In addition, we also test the noise sensitivity of LRI$^+$ in this section. The motivation of this experiment is to test the robustness of LRI$^+$ under noise.

We experimented with different values of parameter K when computing the LRI feature vectors and found $K = 4$ worked well. Larger K, *e.g.*, $K = 10$, did not improve performance very much, but increased computational complexity. Smaller K showed a noticeable decrease in performance. Hence, in this section, without further explanation, $K = 4$.

3.5.1 Experiment 1: Retrieval Test on Corbis-based Database

In an identical texture retrieval experiment [3,4,15], one has a database of textured images such that for each image, some number of other images in the database are classified as "identical", where variations in intensity, rotation, and scale are not permissible. Then for each image in the database, one computes the metric between this image and all others in the database, and then orders the other images according to ascending metric values. Ideally, the other images classified as identical yield the smallest metric values. A suitable database can be formed by taking a number of homogeneous large texture images, choosing smaller sub-images from each and classifying all the resulting smaller images derived from one image as identical, and not identical to those derived from other large images. Accordingly, the large images need to have textures that are distinct. One benefit of this setup is that subjective experiments are not needed to establish the ground truth.

As one such database, we chose the one used in [3,4,15] from Corbis images [95]. In the next subsection, we will discuss another database derived from the CUReT database [37]. We call them the Corbis-based and CUReT-based databases.

The Corbis-based database includes 1181 images, all with size 128×128, carefully cropped from 425 photographs [95], chosen to have homogeneous textures. Some sample images are shown in Fig. 3.10. As described above, images are considered to be identical if and only if they were cropped from the same photograph. This database is challenging due to the large number (425) of different textures.

The goodness of the metric in identical texture retrieval can be judged in several ways. As in [3,4,15], we considered the following four retrieval performance measures:

i. Precision at 1 (P@1): The frequency with which the first retrieved image is identical to the query.

ii. Mean Reciprocal Rank (MRR): The average value of the inverse rank of the first image that is identical to the query [96].

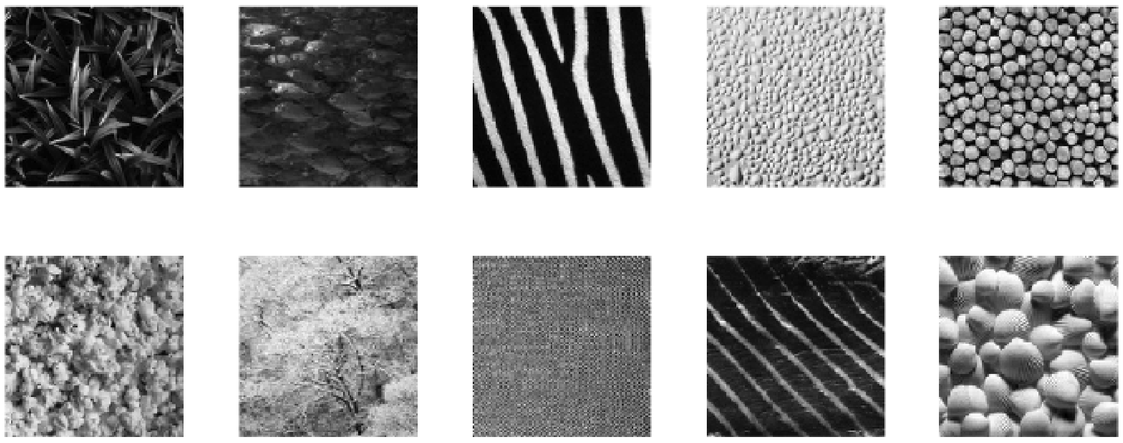


Figure 3.10: Samples images from the Corbis-based database.

iii. Mean Average Precision (MAP): For most images in the database, there exist more than one other images coming from the same texture class in the database. In such cases, usually MAP is reported. The average over all queries in the database of the average of the fractions of the first r retrieved images that are identical to the query, averaged over all r such that the r-th retrieved image is identical to the query [97]. More specifically, the MAP criterion is computed as follows:

   For image i, denote the number of other images coming from the same texture class as n_i. Let indicator function $Q_i(j) = 1$ when the j^{th} retrieved image is the correct one and $Q_i(j) = 0$ otherwise. If we define $P_i(r)$ as the precision of retrieval when we retrieve r images, then,

$$P_i(r) = \sum_{j=1}^{r} \frac{Q_i(j)}{r} \, .$$

   The average precision is defined as

$$P_{\text{ave},i} = \frac{1}{n_i} \sum_{r=1}^{N-1} P_i(r) Q_i(r) \, ,$$

   where N is the total number of images in the database. In our case, $N = 1181$.

Table 3.2: Identical texture retrieval - Corbis-based database.

Metric	P@1	MRR	MAP	AUROC
PSNR [3]	4%	7%	6%	0.75
SSIM [3]	9%	11%	6%	0.45
CW-SSIM [3]	39%	46%	40%	0.92
LBP$^{riu}_{8,1}$ & LBP$^{riu}_{24,3}$ [3]	90%	92%	86%	0.59
STSIM2 [3]	93%	95%	89%	0.98
STSIM-M [3]	96%	97%	92%	0.98
LRI-A	91.8%	93.5%	86.9%	0.982
LRI-D	83.2%	86.9%	78.8%	0.974
SCD	83.7%	88.0%	80.5%	0.984
LRI$^{+}_{a}$	**98.7%**	**99.2%**	**96.4%**	**0.994**
LRI$^{+}_{b}$	**98.1%**	**98.7%**	**95.4%**	**0.994**
LRI$^{+}_{c}$	**99.0%**	**99.2%**	**96.3%**	**0.994**

Now, we can describe MAP as the mean value of the average precision for the whole database:

$$\text{MAP} = \frac{1}{N}\sum_{i=1}^{N} P_{\text{ave},i} \,.$$

iv. Area Under the Receiver Operating Curve (AUROC): When a threshold τ on metric value is used to decide if pairs of images in the database are identical or not, the well known Receiver Operative Curve (ROC) can be applied to produce a plot of the fraction of identical image pairs that are correctly classified as identical vs. the fraction of nonidentical image pairs that are incorrectly classified as identical, as τ changes in the whole range of metric scores. The Area Under the ROC (AUROC), which is a well known measure of the goodness of detection performance, has been used to judge the goodness of metrics in identical texture retrieval [3].

While the first three measure the ability of the metric to correctly rank similarity of other images to a query, they do not test whether a metric value indicating identicality of a texture y to a texture x is the same for all x. The latter is tested by AUROC. For perceptual

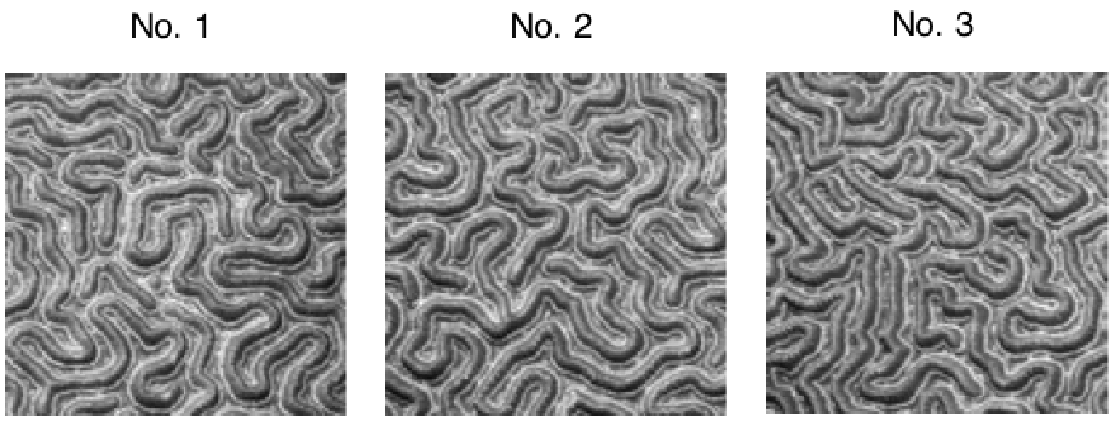


Figure 3.11: A retrieval example using LRI_a^+ in the Corbis-based database. No. 1 is the query image. No. 2 and No. 3 are the first two retrieved images using LRI_a^+.

Table 3.3: Metric value details of the retrieval example shown in Fig. 3.11.

Query image	First retrieved image	Second retrieved image
No. 1	No. 2	No. 3
LRI_a^+	1.9267×10^{-9}	1.3625×10^{-7}
LRI-A	0.0010	0.0041
LBP	7.0252×10^{-4}	0.0056
SCD	0.1467	0.3329
IP	0.0180	0.0180

image coding, P@1 is the most important criterion, as can be seen from Chapter 7.

Experimental results are shown in Table 3.2 for several LRI-based metrics, as well as the results reported in [3] for the following metrics: PSNR, SSIM [12], CW-SSIM [14], LBP [27] and STSIMs [3, 17]. The LRI-based metrics include, LRI-A, LRI-D, LRI_a^+ (LRI-A, LBP, SCD, IP), LRI_b^+ (LRI-A, LBP, SCD_{EST}, IP) and LRI_c^+ (LRI-D, $\text{LBP}_{8,1}^{riu}$, SCD, IP). The results show that LRI-A performs better than all previous metrics except STSIM2 and STSIM-M, and the three versions of LRI^+ perform better than all other metrics, though STSIM2 and STSIM-M are not that far behind.[1] It is worth mentioning that STSIM-M, the most advanced metric in the STSIM family, needs a training phase to determine weights for

[1]While some might view that an improvement from, say, 96% to 99% is small, another viewpoint is that when methods work well, one should compare error rates rather than success rates. From this viewpoint, a decrease in error rate from 4% to 1% is significant.

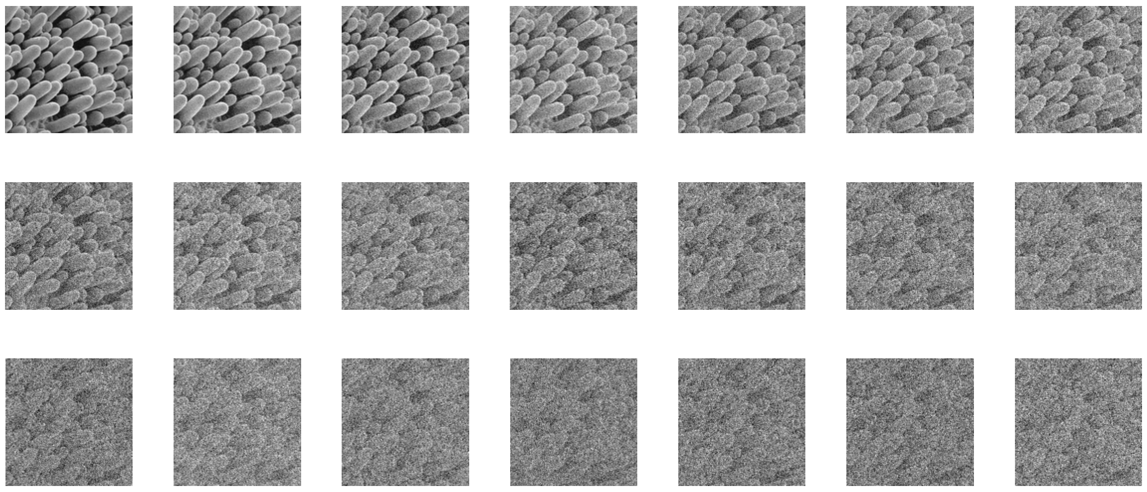


Figure 3.12: An image from the Corbis-based database together with the 20 noisy versions of it, each at a different noise level. Upper left image is the original.

the 82 components of its feature vectors whenever it is applied to a new database. On the other hand, unlike STSIM-M, both LRI^+ and STSIM2 are training-free, which can be an advantage in some situations. It is interesting that while LRI-D by itself is not as good as LRI-A by itself, when combined with the other features, it works just as well, as the other metric components compensate for its shortcomings. It is also interesting that the performance of the low complexity LRI_b^+ is so similar to that of LRI_a^+ and LRI_c^+. This shows that substituting the low complexity SCD_{EST} for SCD has little effect on performance. Another interesting thing to note is how well LRI-A, LRI-D and SCD do in terms of AUROC, whereas they do not work nearly as well in terms of the other three performance criteria.

In Fig. 3.11, a retrieval example using LRI_a^+ is shown. Image No. 1 is the query image. Images No. 2 and No. 3 are the first two retrieved images, both are correct. The LRI_a^+ values between the query image and the first two retrieved images and the individual metric values for all the four components in LRI_a^+ are reported in Table 3.3.

In Chapter 6, the low complexity LRI_b^+ is applied to photographic paper classification and achieves state-of-the-art performance. In Chapter 7, LRI_b^+ is applied to a newly proposed perceptual image compression algorithm, Matched Texture Coding (MTC) [18], to

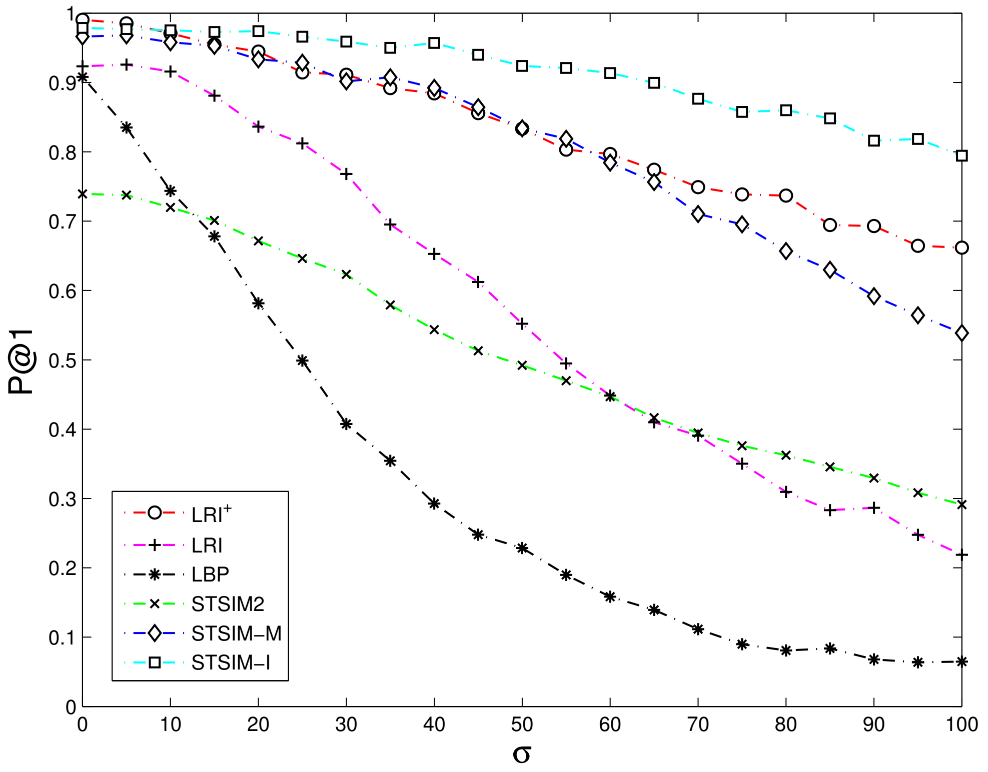


Figure 3.13: P@1 under different noise levels.

substitute STSIM2, with satisfactory coding results.

3.5.2 Experiment 2: Noise Sensitivity Test on Corbis-based Database

In this subsection, we use the aforementioned Corbis-based database to test the noise sensitivity of the proposed LRI$^+$ metric. The motivation of this experiment is to test the robustness of LRI$^+$ under noise. In other words, we want to find out how fast the performance of LRI$^+$ declines with the existence of noise. In particular, we test LRI$_b^+$ and two of its components, namely LRI-A (with $K = 4$) and LBP (with $R = 1$ and $m = 8$ without interpolation). The test procedure follows the experiment setup in [98].

For each image X_i, $i = 1, 2, \ldots, 1181$, in the Corbis-based database, we generate 20

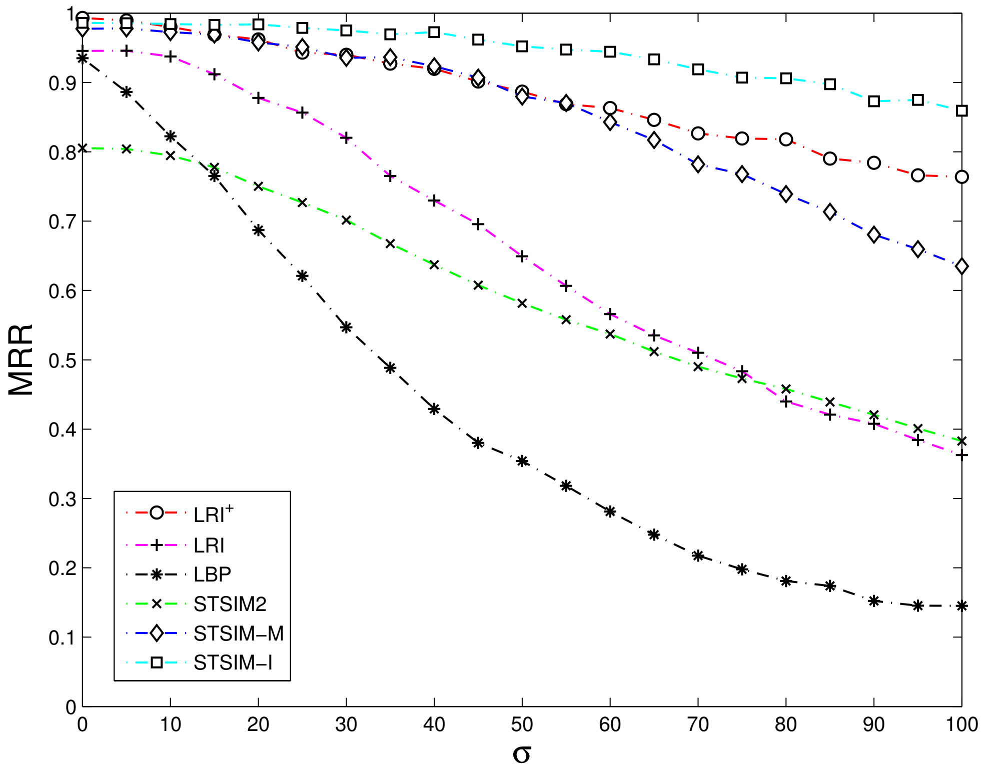


Figure 3.14: MRR under different noise levels.

noisy versions $Y_i(j)$, $j = 1, 2, \ldots, 20$, as follows:

$$Y_i(j) = X_i + N_i(j)\,,$$

where $N_i(j) \cong \mathcal{N}(0, \sigma_j^2)$ is i.i.d. Gaussian noise with variance σ_j^2. $\sigma_j = 5 \times (j-1)$, $j = 1, 2, \ldots, 20$. All noisy images are considered to have pixel intensity from 0 to 255. Figure 3.12 shows an image from the Corbis-based database together with the 20 noisy versions of it, each at a different noise level. One can see that images at high noise levels are severely damaged by the additive Gaussian noise and the texture pattern can barely be seen.

The test procedure is as follows. First, divide the 425 classes of textures into five groups with equal number of classes, resulting in 85 classes per group. Second, for each noise level and each group, conduct the identical texture retrieval experiment as described in

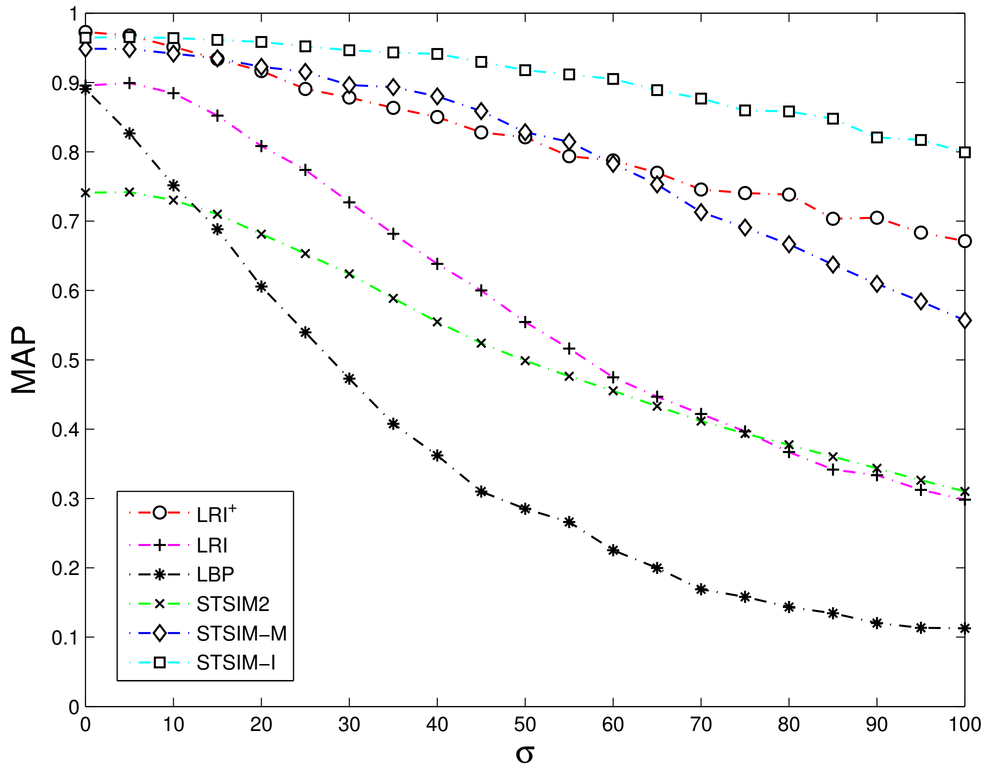


Figure 3.15: MAP under different noise levels.

Section 3.5.1. Note that for the experiment in each noise level, both the query image and the remaining images in the database are damaged by the Gaussian noise at that level. Metric performance is measured by P@1, MRR [96] and MAP [97]. Third, for each noise level, average each of the three performance measurements over the five groups.

Figure 3.13 shows the P@1 for different metrics under all the 20 noise levels. The results of STSIM-I, STSIM-M and STSIM2 are from [98]. Note that STSIM-I and STSIM-M are two improved versions of STSIM2, both of which need a training phase to determine weights for the 82 components of their feature vectors whenever they are applied to a new database. Hence, these two metrics naturally have an advantage over all the other metrics shown in Fig. 3.13. Not surprisingly, STSIM-I is the most robust one under additive Gaussian noise, with the proposed LRI_b^+ not far behind. It is quite noticeable that LRI_b^+ outperforms not only STSIM2, but also STSIM-M. Compared to LBP, the newly proposed

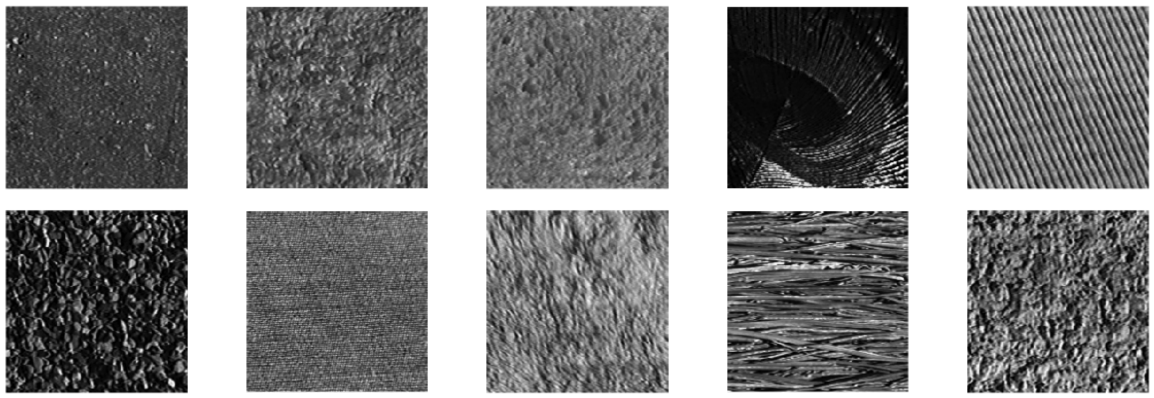


Figure 3.16: Sample images from the CUReT-based database.

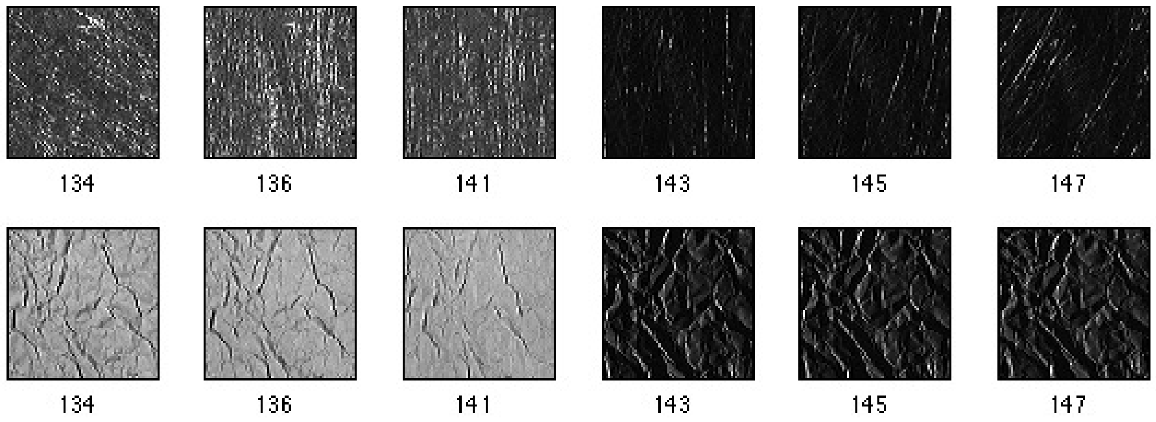


Figure 3.17: CUReT images from six lighting and illumination conditions. Images in the same row are from the same class.

LRI-A is significantly more robust to noise. This is because in LRI-A, the threshold T can control noise sensitivity. (Recall that in LBP, there is no such threshold.) Figures 3.14 and 3.15 show the MRR and MAP for the same metrics under the same 20 noise levels. One can see that the results for MRR and MAP are very similar to the results for P@1.

From this experiment, we claim that the proposed LRI$^+$ metric is very robust to additive Gaussian noise. In addition, through the study of a newly proposed perceptual image compression algorithm, MTC, that will be discussed in Chapter 7, we also found that LRI$^+$ is very robust to distortions caused by JPEG compression. (Actually LRI$^+$ is more robust to distortions caused by JPEG compression than STSIM2.) This is very useful in image compression applications.

Table 3.4: Identical texture retrieval - CUReT-based database.

Metric	P@1	MRR	MAP
PSNR [15]	11%	17%	17%
SSIM [15]	6%	11%	10%
CW-SSIM [15]	69%	77%	72%
CW-SSIM global [15]	31%	45%	35%
STSIM [15]	81%	85%	80%
STSIM global [15]	93%	94%	90%
STSIM2 [15]	81%	86%	81%
STSIM2 global [15]	97%	97%	95%
STSIM2-M [15]	96%	97%	95%
LRI-A	95.6%	96.7%	95.1%
LRI$_a^+$	**100%**	**100%**	**99.8%**
LRI$_b^+$	**100%**	**100%**	**99.8%**

3.5.3 Experiment 3: Retrieval Test on CUReT-based Database

The CUReT-based database [37] contains images of 61 real-world textures each taken at 92 different viewing and illumination directions. Some sample texture images are shown in Fig. 3.16. Note that many texture classes are quite similar.

To test metrics, we adopt the same experiment setup as in [3] and the exact same database. For each of the 61 texture classes, three 128×128 patches were cut from lighting and viewing condition 122. Hence the database includes $61 \times 3 = 183$ images. The retrieval results are shown in Table 3.4, for LRI-A, LRI$_a^+$, LRI$_b^+$, as well for several other metrics, as reported in [3]. Compared to STSIM2, the proposed LRI$^+$ achieves better performance in all three criteria, with both P@1 and MRR being 100% and MAP being very close to 100%. In addition, even LRI-A itself performs almost as well as STSIM2.

The near perfect performance of LRI$_a^+$ andLRI$_b^+$ might be attributed to the fact that images in the same class are taken with the same lighting and viewing condition. As a stiffer test, we also ran our proposed metrics on a larger CUReT-based database formed as follows. For each of the 61 texture classes, choose six images from lighting and viewing conditions 134, 136, 141, 143, 145 and 147. Several sample images from two classes in this database are shown in Fig. 3.17. As can be seen, this database with 366 images is more challenging due to its larger variations in lighting and viewing conditions. In this experiment, LRI-A and LRI$_b^+$ metrics give P@1 of 85.0% and 94.3%, respectively, which though not as good as before, are still quite respectable.

CHAPTER 4

Rotation-Invariant Local Radius Index (RI-LRI)

4.1 Rotation-Invariant Local Radius Index (RI-LRI) Features

To make LRI rotation invariant, and thereby suitable for texture classification, we propose to estimate a dominant direction for each pixel based on the eight LRI-A directional indices, and then circularly rotate the collection of directional indices so that the index corresponding to the direction closest to the dominant direction is now the first index.

When at least one of the LRI-A indices $I_{i,1}, \ldots, I_{i,8}$ at pixel i is not zero, the *dominant direction* D_i at pixel i is

$$D_i = \text{angle}\Big(\sum_{d=1}^{8} I_{i,d} \times \phi_d \Big) \,,$$

where ϕ_d is the d^{th} component of the complex vector

$$\Phi = [\phi_1, \ldots, \phi_8] = [1, 1+j, j, -1+j, -1, -1-j, -j, 1-j] \,,$$

angle() returns the angle of a complex number, and $0 \le D_i < 2\pi$. As can be seen, the dominant direction D_i is formed by using the directional indices to weight complex numbers representing the usual eight directions. In the horizontal and vertical directions, the norms of the complex numbers are 1; and in diagonal and anti-diagonal directions, the norms are $\sqrt{2}$. This is consistent with the locations of the pixels on which LRI-A indices are based. (Recall that, in LRI, interpolation is not used to estimate pixel values on the circle centered at pixel i, which saves considerable computation and does not decrease performance.) As a result, the scaling factor $\sqrt{2}$ naturally applies.

The LRI index corresponding to the direction closest to D_i is I_{i,s_i}, where $s_i = 1 + [\frac{D_i}{\pi/4}] \bmod 8$, and $1 \leq s_i \leq 8$. Finally, the RI-LRI indices for pixel i, denoted $RI_{i,1}, \ldots,$ $RI_{i,8}$, are the LRI-A indices, $I_{i,1}, \ldots, I_{i,8}$, circularly shifted by S_i. That is,

$$RI_{i,j} = I_{i,(j+s_i-2 \bmod 8)+1} \, .$$

When all eight $I_{i,d}$'s are zero, *i.e.*, when the pixel is not adjacent to an edge, we take the eight $RI_{i,d}$'s to be zero, as well. With this design, RI-LRI indices are rotation invariant.

Figure 4.1 shows several sample pixels next to an edge, together with their dominant directions and the RI-LRI indices. Parts (a), (b) and (c) show the ability of RI-LRI to achieve rotation invariance. As the edge rotates, the RI-LRI indices (the numbers below each figure) remain unchanged. By comparing (a) and (d), one can see that on both sides of an edge, pixels have the same dominant direction, namely, pointing to the larger intensity. However, their RI-LRI indices differ because they have different sets of neighbors. Part (e) shows a pixel in a transition region. Again, the dominant direction points to the larger intensity. Observe that if the definition of D_i used $|I_{i,d}|$ instead of $I_{i,d}$, then pixels in such transition regions would not have dominant direction pointing to the larger intensity. Parts (f), (g) and (h) present three sample pixels near corners. One can easily see the distinctive power of RI-LRI indices by comparing (f) and (g).

As with LRI, for each pixel, the *RI-LRI operator* outputs eight integer indices, one for each direction. Therefore, one could follow the procedure of LRI-A and compute eight histograms for an image as its feature vector. However, as we found during experiments, most of the distinctive information is contained in the dominant direction, the opposite direction, and the two orthogonal directions. In addition, the two orthogonal directions

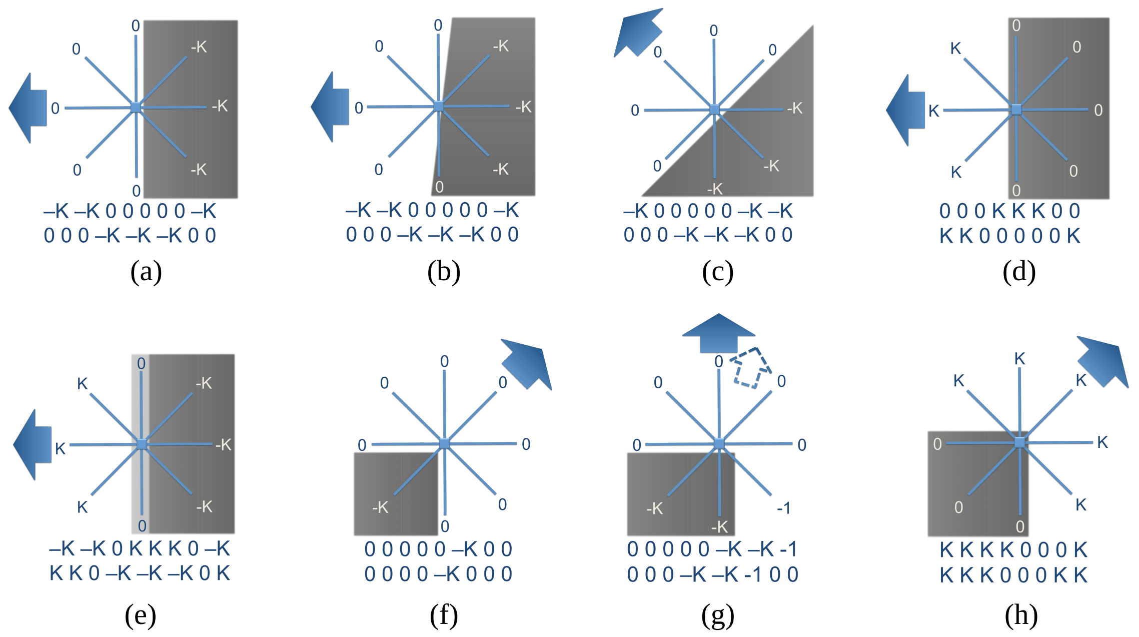


Figure 4.1: Eight image patches containing an edge. The original (above) and circularly shifted (below) sequences of LRI-A indices are shown, along with quanitzed dominant directions. The dashed arrow shows an unquantized dominant direction.

contain correlated information, which can be beneficially exploited by a two-dimensional histogram. Specifically, with the eight directions labeled counterclockwise, starting from the dominant direction, as $1, 2, \ldots, 8$, we claim that most of the information is included in directions 1, 3, 5 and 7. Orthogonal directions 1 and 3 contain correlated information. So, too, do orthogonal directions 5 and 7.

With this in mind, the *RI-LRI feature vector* consists of one two-dimensional histogram. Specifically, for each pair (m, n) of integers between $-K$ and K, we count the number of pixels i such that $(RI_{i,1}, RI_{i,3}) = (m, n)$ and add to it the number of pixels i such that $(RI_{i,5}, RI_{i,7}) = (m, n)$. The collection of such sums, $C(m, n), -K \leq m, n \leq K$, comprises the two-dimensional histogram that is the RI-LRI feature vector of length $(2K+1)^2$. A texture similarity metric is then defined by Kullback-Leibler divergence [82] between the normalized histograms for two images.

Table 4.1: Computational complexity comparison measured in operations per pixel.

Feature	Complexity	Feature	Complexity
$\text{LBP}^{riu}_{8,1}$ (No interp.)	30	$\text{LBP}^{riu}_{8,1}$	90
LRI ($K = 4$)	36	$\text{LBP}^{riu}_{16,2}$	242
RI-LRI ($K = 4$)	47	$\text{LBP}^{riu}_{24,3}$	394

4.1.1 Computational Complexity

As described earlier, the number of computations needed to compute LRI indices is around $4K$ operations per pixel. After considering the computation to determine the dominant direction, the overall complexity for the RI-LRI feature vector is $8K + 15$ ($= 47$ when $K = 4$) operations per pixel. Table 4.1 compares the complexity of RI-LRI and LBP with various parameters. One can see that RI-LRI is simpler than the simplest version of LBP with interpolation and a bit more complex than the simplest LBP without interpolation.

4.2 Rotation-Invariant Subband Contrast Distribution (RI-SCD)

As discussed earlier, the metric LRI$^+$ was proposed as a combination of LRI, LBP, SCD and IP, which have complementary properties. Here, we adopt a similar principle to design a new rotation-invariant metric, RI-LRI$^+$. However, we need to modify SCD (subband contrast distribution), since it is rotation sensitive.

Given the variances $\{\sigma^2_{s,r}\}$ of the bands in a subband decomposition of an image into S scales and R equally spaced orientations, one can form an approximately rotation invariant feature vector by ordering the variances for each scale from largest to smallest, and then concatenating. The resulting feature vector will be invariant to image rotations by $2\pi/R$, and if R is large enough, approximately invariant to any rotation.

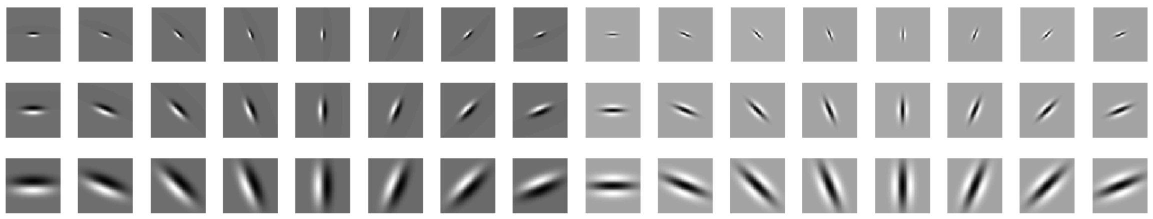


Figure 4.2: The 24 edge and 24 bar filters from RFS filter bank.

We found that the number of orientations was not large enough in either the usual 4-orientation steerable pyramid decomposition or the 4-direction pixel difference filters. Hence, motivated by LEBC [31], we use the edge (first derivative) and bar (second derivative) filters from the Root Filter Set (RFS) [38, 39]. In this work, we set the number of filter orientations to be eight and use three scales $\{(1,3),(2,6),(4,12)\}$, for a total of 48 filters, as shown in Fig. 4.2. We normalize the texture image to have unit variance before filtering, which creates invariance to global affine gray-scale changes. We end up with a 48-dimensional feature vector $F_x = (s_{x,1}, s_{x,2}, \ldots, s_{x,48})$ that is piecewise monotonic.

For two images x and y, with feature vectors F_x and F_y, the similarity score is

$$\text{RI-SCD}(x,y) = \prod_{i=1}^{48} \frac{2\sqrt{s_{x,i}s_{y,i}} + C}{s_{x,i} + s_{y,i} + C}\,,$$

where C is a small constant to increase robustness and avoid singularity. The RI-SCD(x,y) values lie between 0 and 1, with 1 meaning identical. By multiplicatively combining all 48 terms, we use the hypothesis that similar texture patterns should have similar variances in all 48 subbands.

4.3 Rotation-Invariant Texture Similarity Metric: RI-LRI$^+$

As in LRI$^+$, we propose the following way to combine RI-LRI with LBP and RI-SCD to obtain a very effective rotation-invariant metric:

$$\text{RI-LRI}^+ = \text{RI-LRI}^{p_1} \times \text{LBP}^{p_2} \times (1 - \text{RI-SCD})^{p_3} \, .$$

We found that $p_1 = 0.7$, $p_2 = 0.4$ and $p_3 = 1.3$ to be good values for weighting the three metric components. Different from LRI$^+$, the Intensity Penalty (IP) metric is not included in RI-LRI$^+$. This is because in texture classification tasks, people usually do not care about background luminance difference between two texture patterns. RI-LRI$^+$ values are non-negative, with 0 meaning identical and small values meaning similar. Since all three components are rotation and gray-scale invariant, so is RI-LRI$^+$. As can be seen from the experiments in Section 4.4.1, RI-LRI$^+$ is also insensitive to changes in the spectrum of the illuminance. With $K = 4$, the RI-LRI$^+$ feature vector has dimension 139 (= 81 + 10 + 48). This feature dimension is almost ten times less than the state-of-the-art metric, LEBC [31], which has a 1280-dimensional feature vector.

4.4 Metric Performance Evaluation

This section reports the results of testing RI-LRI and RI-LRI$^+$ based on its ability to do texture classification using the Outex [36] and CUReT databases [37] in the same ways that many previous metrics have been tested. The proposed metrics are compared to the state-of-the-art metric, LEBC [31], as well as other recently proposed metrics, including DLBP [28], CLBP [29], DNS [30], VZ-MR8 [38] and VZ-Joint [39].

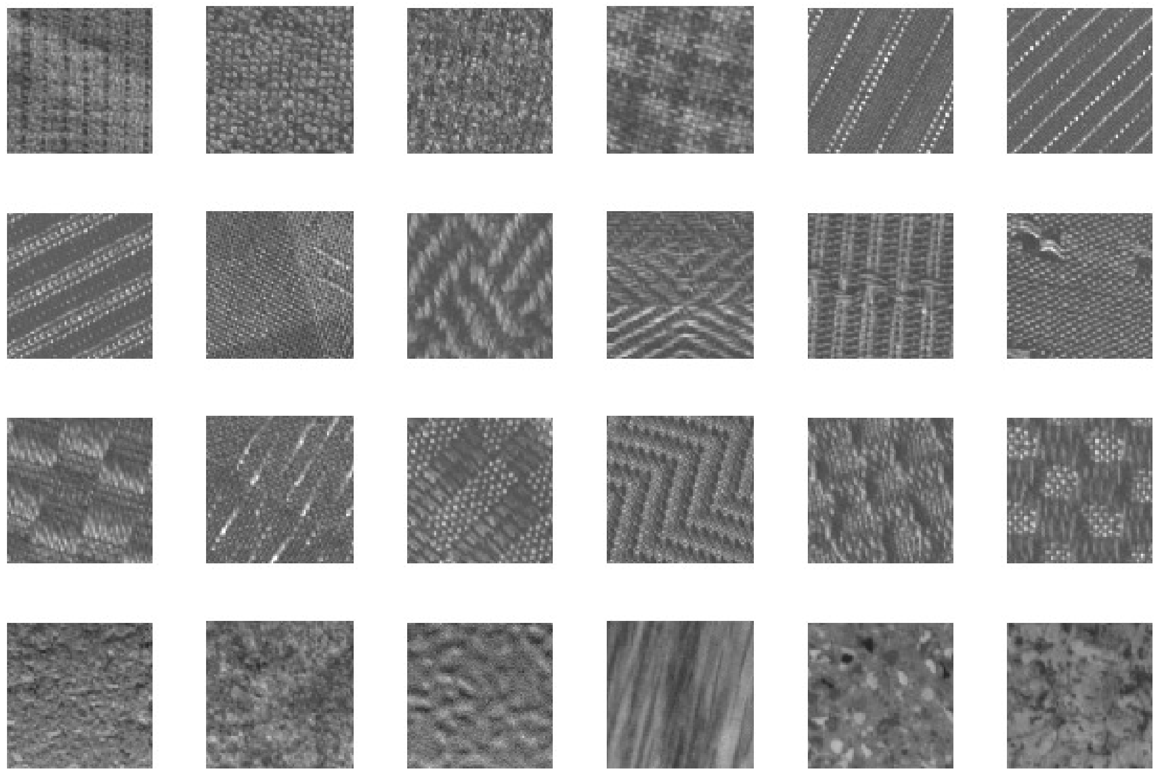


Figure 4.3: 24 textures from the Outex database.

4.4.1 Experiment 1: Classification on the Outex Database

To make a fair comparison with previous methods, experiments are performed in exactly the same way as in [27–31], using two test suites in the Outex database [36]: TC10 and TC12. Each test suite contains 24 texture classes obtained under three illumination conditions (“horizon”,“inca” and “t184”) and nine orientations ($0\,^\circ$, $5\,^\circ$, $10\,^\circ$, $15\,^\circ$, $30\,^\circ$, $45\,^\circ$, $60\,^\circ$, $75\,^\circ$ and $90\,^\circ$). For each texture class, under each illumination and orientation condition, there are 20 non-overlapped 128×128 texture samples. Figure 4.3 shows 24 sample images, each from one texture class used in the experiment. One can see that many texture patterns in the Outex database are highly directional. The texture classification experiment is based on a nearest neighbor classifier with a “dictionary” composed of the 480 TC10 “inca” images with orientation $0°$. Classification results are reported for three different sets of query images: (a) the remaining 3840 “inca” images in TC10, (b) all “t184”, and (c) all “horizon”

Table 4.2: Metric performance in the Outex database (%).

Metric	TC10	TC12/“t184”	TC12/“horizon”	Avg.
LBP$^{riu}_{8,1}$ [27]	85.1	67.5	62.7	71.8
RI-LRI ($K = 4$)	91.0	82.0	81.6	84.9
LBP/VAR [27]	97.7	87.3	86.4	90.5
VZ-Joint [29]	92.0	91.4	92.1	91.8
VZ-MR8 [29]	93.6	92.6	92.8	93.0
DLBP+NGF [28]	99.1	93.2	90.4	94.2
DNS+LBP [30]	99.3	94.4	92.9	95.5
CLBP [31]	99.2	95.2	95.6	96.7
LEBC [31]	99.5	98.4	98.2	98.7
RI-LRI$^+$ ($K = 4$)	99.4	97.7	98.2	98.4

images in TC12, of which there are 4320 of each. The first experiment is designed to test rotation invariance property of a metric, since the dictionary contains textures in only one orientation, yet is tested on textures in the other eight orientations. As a more challenging setup, in the second and third experiments, the dictionary contains textures in only one orientation and illumination, and is tested on textures in the other eight orientations using two different illuminants. Hence, the second and third experiments test not only rotation invariance, but also illuminance sensitivity.

In each experiment, the metric performance is measured by classification accuracy. Table 4.2 shows experimental results for our methods based on our experiments, as well as for other methods based on performance reported in [27–31]. For each metric, only the best performance in the cited reference is shown. The first three data columns show the three experiments described above, respectively. And the last data column gives the average performance of the three experiments. The first two lines show the performance of LBP$^{riu}_{8,1}$ and RI-LRI with $K = 4$, which are the two simplest and the most basic metrics. The results show that although RI-LRI is computationally more efficient, its classification accuracy significantly outperforms LBP. Because of its simplicity, RI-LRI could be added to existing or future texture similarity metrics without increasing complexity very much. Starting

Table 4.3: Metric performance in CUReT database (%). Numbers after "±" are standard deviations.

N	46	23	12	6
$\text{LBP}_{8,1}^{riu}$	80.8±1.3	75.2±1.7	67.9±2.5	59.0±4.1
RI-LRI ($K = 4$)	91.9±1.2	86.3±1.6	78.4±2.7	66.9±4.2
VZ-Joint [29]	97.7	94.6	89.4	81.1
VZ-MR8 [29]	97.8	95.0	90.5	82.9
CLBP [31]	97.0	93.6	87.7	78.3
DNS+LBP [30]	95.0	-	-	-
LEBC [31]	98.5	96.0	91.0	82.3
RI-LRI$^+$($K = 4$)	98.0±0.5	95.3±1.1	91.0±1.8	82.8±3.8

from the third line, the results of more sophisticated metrics are shown, with LEBC and RI-LRI$^+$ being the best. While LEBC has slightly better performance than RI-LRI$^+$, its 1280-dimensional feature vector is much larger than the 139-dimensional RI-LRI$^+$ feature vector, resulting in much higher storage demand and computational complexity.

4.4.2 Experiment 2: Classification on the CUReT Database

As mentioned previously, the CUReT database [37] contains 61 texture classes, each with 92 images obtained from different viewpoints and illumination conditions (5612 images in total). Following the experiment setups in [29, 31], N images per class are randomly selected as the dictionary for a nearest neighbor classifier, and the remaining images are used for testing. Four experiments are conducted with $N = 46, 23, 12$ and 6, respectively. Each experiment is repeated 100 times and the average classification accuracy is reported in Table 4.3. Compared to $\text{LBP}_{8,1}^{riu}$, RI-LRI gives significantly better classification accuracy. This is consistent with the result from the Outex database. LEBC has the best performance on average, with RI-LRI$^+$ and VZ-MR8 not far behind. However, if LEBC and VZ-MR8 methods have the same standard deviation as RI-LRI$^+$, then the differences in performance are not that significant. When feature dimension is considered, the 139-dimensional RI-LRI$^+$ feature vector is significantly more compact than the 960-dimensional VZ-MR8 feature vector and the 1280-dimensional LEBC feature vector. In short, RI-LRI$^+$ metric has comparable performance with the state-of-the-art metric, LEBC, with a much lower dimensional feature vector.

CHAPTER 5

The Distribution of LRI Indices for Tessellations

5.1 Introduction

In Chapter 3, two versions of LRI, namely LRI-A and LRI-D, were proposed. LRI-A captures *inter-edge distance* distributions in a texture pattern in eight directions, where an *inter-edge distance* is the distance between two adjacent edges in a certain direction. Due to the stochasticity of textures, such distances have a distribution. Theoretically speaking, inter-edge distance distributions reflect the sizes of texture elements in each direction and provide good distinguishing power as evidenced in a number of experiments. Different from LRI-A, LRI-D measures distribution of distance-to-nearest-edge in eight directions. At each pixel location, for each direction, there is a distance to its nearest edge. As one can imagine, such distances are highly correlated for adjacent pixels. Both LRI-A and LRI-D are good features to measure texture similarity.

This chapter focuses on a theoretical analysis of LRI. In particular, we use continuous space periodic tessellations to analyze LRI. It is intuitive to model textures with regular and repetitive patterns, such as Voronoi tessellations [19, 20] and structural displacement rules [21–23]. These methods assume textures are composed of texture elements with certain geometrical displacement rules. In order to gain insight into LRI features, we derive formulas to predict the distributions of the LRI-A and LRI-D induced by periodic tessellations. Numerical results demonstrate the accuracy of the formulas. In addition, a detailed model to pixelate a continuous image will be introduced, which may be useful for other image processing related topics.

5.2 Tessellation Model

In this section, we introduce a tessellation model to be used to analyze LRI. We seek a formula for the probability mass function of LRI indices in direction d for a discrete tessellation. To define a discrete tessellation, we start with a continuous image $X(\underline{t})$, $\underline{t} = (t_1, t_2)$, described by a tessellation $\{V_i\}$. In particular, X contains identical tiles V_i, all being translations of some fundamental tile V, such as a square, diamond or hexagon, that are each assigned a constant gray-scale intensity in such a way that adjacent tiles have significantly different intensities, as illustrated in Fig. 5.1. We assume that transitions in any direction d relevant to LRI are equally like to be positive or negative.

Now consider a pixelation induced by a square grid of $\Delta \times \Delta$ pixels overlaid on the image X. This square grid can be thought as a pixel tessellation $\{W_{\underline{m}}\}$, where $\underline{m} = (m_1, m_2)$, and m_1, m_2 are both integers. Each tile in $\{W_{\underline{m}}\}$ is a translation of a fundamental square tile: $[0, \Delta] \times [0, \Delta]$, which is defined to be $W_{(0,0)}$. Specifically, $W_{\underline{m}} = W_{(0,0)} + \Delta\underline{m}$. This grid induces a discrete-space, tessellated image $Y = \{Y[\underline{m}]\}$, with pixels indexed by integer coordinates $\underline{m} = (m_1, m_2)$, and with $Y[\underline{m}]$ inheriting the intensity of the tile V_i that has the most coverage of pixel $\underline{m}$.

By definition, both X and Y have infinite support. To analyze LRI, we limit our attention to some finite support. For X, we focus on the support region $[0, S] \times [0, S]$, for some $S > 0$. In particular, we choose S to be a multiple of Δ. And for Y, we limit the support to those $\underline{m}$ such that the corresponding square tile W_m overlaps $[0, S] \times [0, S]$. Accordingly, pixel index $\underline{m}$ takes integer value pairs within $\{0, 1, \ldots, S_\Delta - 1\} \times \{0, 1, \ldots, S_\Delta - 1\}$, where $S_\Delta = \frac{S}{\Delta}$. To avoid any possible dependence between the pixel grid $\{W_{\underline{m}}\}$ and the tessellation $\{V_i\}$, we assume that the pixel grid has a random shift described by a random

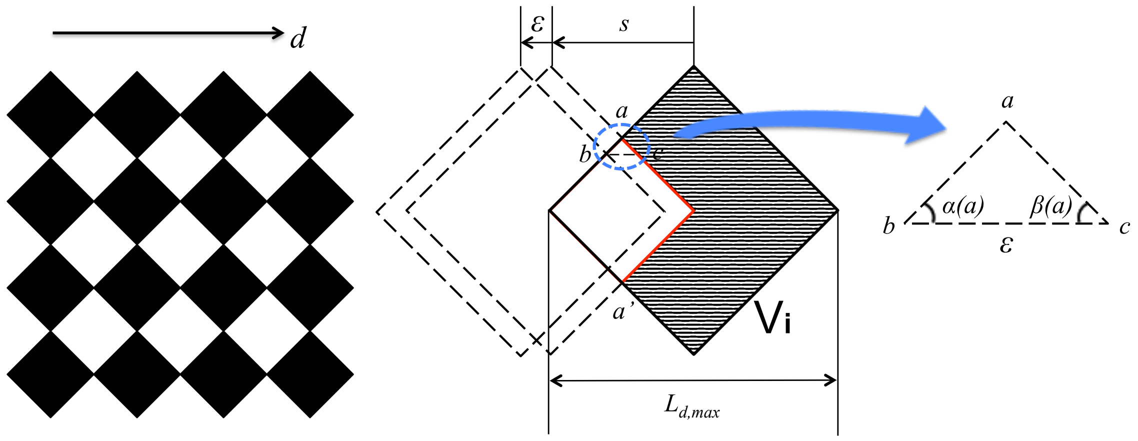


Figure 5.1: A tessellation example.

vector $\underline{U} = (U_1, U_2)$ uniformly distributed on $[0, \Delta) \times [0, \Delta)$, where $\underline{U}$ is independent of all other random variables. After this random shift, the new pixel grid is denoted as $\{\tilde{W}_{\underline{m}}\} = \{W_{\underline{m}}\} + \underline{U}$. As can be seen later, this random shift $\underline{U}$ helps analyze the distribution of LRI-A. In particular, in terms of X and $\underline{U}$, a discrete tessellated image $\tilde{Y}_{\underline{U}} = M(X, \underline{U})$ is now defined as follows:

$\tilde{Y}_{\underline{U}}[\underline{m}] = X(\underline{t})$ where $\underline{t}$ is any point in the V_i with largest overlap of pixel $\tilde{W}_{\underline{m}}$.

For a discrete-space image $\tilde{Y}_{\underline{U}}$, the LRI index at pixel $\underline{m}$ in direction d is denoted $I^d[\tilde{Y}_{\underline{U}}, \underline{m}]$. For an integer z and a random variable $\underline{U}$, the frequency with which an LRI index equals z is

$$\sum_{\underline{m}} \mathbb{1}\big(I^d[\tilde{Y}_{\underline{U}}, \underline{m}] = z\big) \frac{1}{S_\Delta^2} ,$$

where S_Δ^2 is the total number of pixels in $\tilde{Y}_{\underline{U}}$. Then we average this frequency over $\underline{U}$ to obtain a more accurate estimate of the frequency with which an LRI index equals z:

$$\int_0^\Delta \int_0^\Delta \sum_{\underline{m}} \mathbb{1}\big(I^d[\tilde{Y}_{\underline{u}}, \underline{m}] = z\big) \frac{1}{S_\Delta^2} \frac{1}{\Delta^2} \mathrm{d}u_1 \mathrm{d}u_2 .$$

This formula actually describes $\Pr(I^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z)$, given $\underline{M}$ and $\underline{U}$ are independent and uniformly distributed in $\{0, 1, \ldots, S_\Delta - 1\} \times \{0, 1, \ldots, S_\Delta - 1\}$ and $[0, \Delta) \times [0, \Delta)$, respectively. From now on, we focus on deriving an estimate of $\Pr(I^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z)$. In the next two sections, we derive estimates of $\Pr(I_D^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z)$ for LRI-D and $\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z)$

for LRI-A, respectively.

Now, for a point $\underline{t}$ in a continuous image X and a given grid location $\underline{U}$, define the LRI index at point $\underline{t}$ as $\tilde{I}^d(X, \underline{U}, \underline{t}) = I^d[M(X, \underline{U}), \pi(\underline{t}, \underline{U})]$, where $\pi(\underline{t}, \underline{U}) = \underline{m}$ if $\underline{t}$ falls within pixel $\tilde{W}_{\underline{m}}$. As mentioned above, the goal of the following two sections is to analyze LRI by finding an approximate formula for $\Pr(I^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z)$. We do this by first finding an approximate formula for $\Pr(\tilde{I}^d(X, \underline{U}, \underline{T}) = z)$, where $\underline{T} = (T_1, T_2)$ is uniformly distributed over the support of X ($[0, S] \times [0, S]$). For simplicity, we omit the effect of the size limit K in LRI, although the derived formulas can be easily truncated to show its effect. We assume throughout that Δ is small relative to the size of the fundamental cell V and that the image is large relative to the size of the tiles (so that boundary-of-image effects can be ignored).

5.3 Analysis of LRI-D

In this section, we analyze LRI-D using the tessellation model described in Section 5.2. For an integer z, we have

$$\begin{aligned}\Pr(I^d_D[\tilde{Y}_{\underline{U}}, \underline{M}] = z) &= \Pr(\tilde{I}^d_D(X, \underline{U}, \underline{T}) = z) \\ &\cong \Pr\left(|L^d(\underline{T}) - z\Delta| \le \frac{\Delta}{2}\right),\end{aligned}$$

where $L^d(\underline{t})$ equals the distance in direction d from $\underline{t}$ in the image domain to the nearest boundary of a tile if the boundary crossing is a positive change of intensity, and minus the distance if the boundary crossing is a negative change of intensity. We begin by considering the simplest case that all transitions in direction d have a positive change in intensity, and later consider the more general situation. Notice that the key approximation here is to ignore the small dependence on $\underline{U}$. Specifically,

$$\Pr\left(|L^d(\underline{T}) - z\Delta| \le \frac{\Delta}{2}\right) = \int_{z\Delta - \frac{\Delta}{2}}^{z\Delta + \frac{\Delta}{2}} f_{D,d}(\tau) \mathrm{d}\tau\,,$$

where $f_{D,d}(s)$ is the probability density function (PDF) of $L^d(\underline{T})$. To compute $f_{D,d}(s)$, first consider the cumulative distribution function (CDF)

$$F_{D,d}(s) = \Pr(L^d(\underline{T}) \leq s)\,.$$

From the periodicity of the tessellation, it suffices to assume $\underline{T}$ is uniform over one particular tile V_i. Referring to the illustration in Fig. 5.1 for a tessellation of diamonds, the probability of $L^d(\underline{T}) \leq s$ is the probability of the point $\underline{T}$ landing at a point in V_i such that $\underline{T} + s \cdot \underline{d}$ is not in V_i, where $\underline{d}$ is the unit vector in direction d. This is actually the shaded region in Fig. 5.1, with area denoted $S(s)$, which is what remains of the tile under consideration after removing all points contained in a translation of the tile by distance s in the direction opposite to d. Hence, $F_{D,d}(s)$ is the ratio of the of area of the shaded region to that of the entire tile. We will now show that $S(s)$ is closely related to $b(s)$, which is the portion of the boundary of the translated tile that remains in the original tile (shown in red in Fig. 5.1). Specifically,

$$S(s) = \int_0^s \mathrm{d}u \int_{b(u)} \sin\beta(\tau)\,\mathrm{d}\tau\,,$$

where $\beta(\tau)$ is the angle at point τ between $b(u)$ and direction d. One example is shown in Fig. 5.1. This leads to

$$F_{D,d}(s) = \frac{S(s)}{|V|}\,,$$

where $|V|$ denotes the area of a tile. Taking the derivative leads to the main result of this section:

$$f_{D,d}(s) = \frac{\int_{b(s)} \sin\beta(\tau)\mathrm{d}\tau}{|V|}\,,\ 0 \leq s \leq L_{d,\max}\,,$$

where $L_{d,\max}$ denotes the maximum width of a tile in direction d. Notice that $f_{D,d}(s)$ will be proportional to the length of $b(s)$ if $\beta(\tau)$ is constant, which is true if $b(s)$ is a straight line.

To account for the positivity and negativity of LRI-D indices, as mentioned earlier, we assume transitions in direction d are equally like to be positive or negative. In such a tessellation, the actual PDF of $L^d(\underline{t})$ is $\frac{f_{D,d}(-s)}{2} + \frac{f_{D,d}(s)}{2}$.

5.4 Analysis of LRI-A

In this section, we analyze LRI-A using the tessellation model proposed in Section 5.2. Different from LRI-D, in a discrete image, a pixel has a non-zero LRI-A index if and only if it is adjacent to an edge.

Again, we assume all transitions are positive, and that $\underline{T}$ is uniformly distributed on some tile V_i. Let $\tilde{I}_A^d \triangleq \tilde{I}_A^d(X, \underline{U}, \underline{T})$, and let B_{V_i} denote the boundary of V_i. For an integer $z \geq 0$, we can compute $\Pr\left(\tilde{I}_A^d = z\right)$ conditioned on whether $\tilde{I}_A^d = 0$:

$$\begin{aligned}\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z) \;&=\; \Pr\left(\tilde{I}_A^d = z\right) \\ &= \Pr\left(\tilde{I}_A^d = z | \tilde{I}_A^d = 0\right) \cdot \Pr\left(\tilde{I}_A^d = 0\right) \\ &\quad + \Pr\left(\tilde{I}_A^d = z | \tilde{I}_A^d \neq 0\right) \cdot \Pr\left(\tilde{I}_A^d \neq 0\right).\end{aligned}$$

In this formula, calculating $\Pr\left(\tilde{I}_A^d \neq 0\right)$ is the big challenge. We will derive this probability later in this section. Actually the event that $\{\tilde{I}_A^d \neq 0\}$ corresponds to the event that $\{\underline{T} \in B_{V_i}^d\}$, where $B_{V_i}^d = \{\underline{t} \in B_{V_i} : \exists s \geq 0 \; s.t. \; \underline{t} + \underline{d} \cdot s \in B_{V_i}\}$, $\underline{d}$ is the unit vector in direction d. To compute $\Pr\left(\tilde{I}_A^d = z | \tilde{I}_A^d \neq 0\right)$, the key assumption we make now is similar to what we did in the last section:

$$\begin{aligned}\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z) \;&=\; \Pr\left(\tilde{I}_A^d = z\right) \\ &= \Pr\left(\tilde{I}_A^d = z | \tilde{I}_A^d = 0\right) \cdot \Pr\left(\tilde{I}_A^d = 0\right) \\ &\quad + \Pr\left(\tilde{I}_A^d = z | \tilde{I}_A^d \neq 0\right) \cdot \Pr\left(\tilde{I}_A^d \neq 0\right) \\ &\cong \delta[z] \cdot \Pr\left(\tilde{I}_A^d = 0\right) \\ &\quad + \Pr\left(|L^d(\underline{T}) - z\Delta| \leq \frac{\Delta}{2} \middle| \underline{T} \in B_{V_i}^d\right) \cdot \Pr\left(\tilde{I}_A^d \neq 0\right) \\ &\cong \delta[z] \cdot \Pr\left(\tilde{I}_A^d = 0\right) + \int_{z\Delta - \frac{\Delta}{2}}^{z\Delta + \frac{\Delta}{2}} f_{A,d}(\tau)\,\mathrm{d}\tau \cdot \Pr(\tilde{I}_A^d \neq 0),\end{aligned}$$

where $f_{A,d}(s) \triangleq f_{L^d(\underline{T})|\underline{T} \in B_{V_i}^d}(s)$ for $s > 0$.

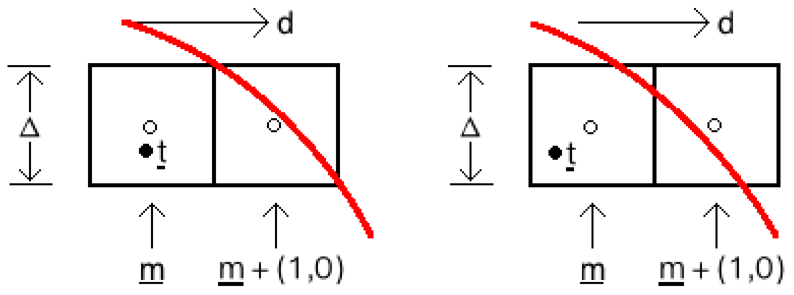


Figure 5.2: The location of pixel grid U influences the LRI-A index at pixel m that contains the interesting point t. The red curve is portion of the boundary between two tiles. (Suppose V_i is to the left of this boundary.) Left image: the pixel m has zero LRI-A index in direction d. Right image: the pixel m has non-zero LRI-A index in direction d.

Now let us compute $\Pr(\tilde{I}_A^d \neq 0)$. Specifically,

$$\begin{aligned}\Pr\left(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] \neq 0\right) &= \Pr\left(\tilde{I}_A^d(X, \underline{U}, \underline{T}) \neq 0\right) \\ &= \mathbb{E}_{\underline{T}}\mathbb{E}_{\underline{U}}\left[\mathbb{1}_{\neq 0}(\tilde{I}_A^d(X, \underline{U}, \underline{T}))\right],\end{aligned}$$

where $\mathbb{1}_{\neq 0}(\tilde{I}_A^d(X, \underline{U}, \underline{T})) = 1$, or equivalently $\tilde{I}_A^d(X, \underline{U}, \underline{T}) \neq 0$, if there is an intensity transition at pixel $\pi(\underline{T}, \underline{U})$ in direction d.

To compute $\mathbb{E}_{\underline{T}}\mathbb{E}_{\underline{U}}\left[\mathbb{1}_{\neq 0}(\tilde{I}_A^d(X, \underline{U}, \underline{T}))\right]$, we first fix $\underline{T} = \underline{t}$ and average over $\underline{U}$; subsequently we average over $\underline{t}$. We first compute $\mathbb{E}_{\underline{U}}\left[\mathbb{1}_{\neq 0}(\tilde{I}_A^d(X, \underline{U}, \underline{t})\right]$ given some $\underline{t}$. Consider some $\underline{t}$ close to the boundary of V_i in direction d. In this case, whether or not $\tilde{I}_A^d(X, \underline{U}, \underline{t}) = 0$ depends on $\underline{U}$. For some choices of $\underline{U}$, both the pixel containing $\underline{t}$ and the next pixel in direction d will have intensity of V_i and $\tilde{I}_A^d(X, \underline{U}, \underline{t}) = 0$, as illustrated in the left image in Fig. 5.2. For other values of $\underline{U}$, the pixel containing $\underline{t}$ will have the intensity of V_i but the next pixel in direction d will have the intensity that lies just beyond V_i, , as illustrated in the right image in Fig. 5.2. So $\Pr(\tilde{I}_A^d(X, \underline{U}, \underline{t}) \neq 0)$ is the probability of all $\underline{U}$'s that cause the second behavior.

From this analysis, for a given $\underline{t}$, we claim that $\mathbb{E}_{\underline{U}}\left[\mathbb{1}_{\neq 0}(\tilde{I}_A^d(X, \underline{U}, \underline{t})\right]$ is a function $f(l, \theta)$, where l is the perpendicular distance from $\underline{t}$ to the nearest boundary of V_i (l is

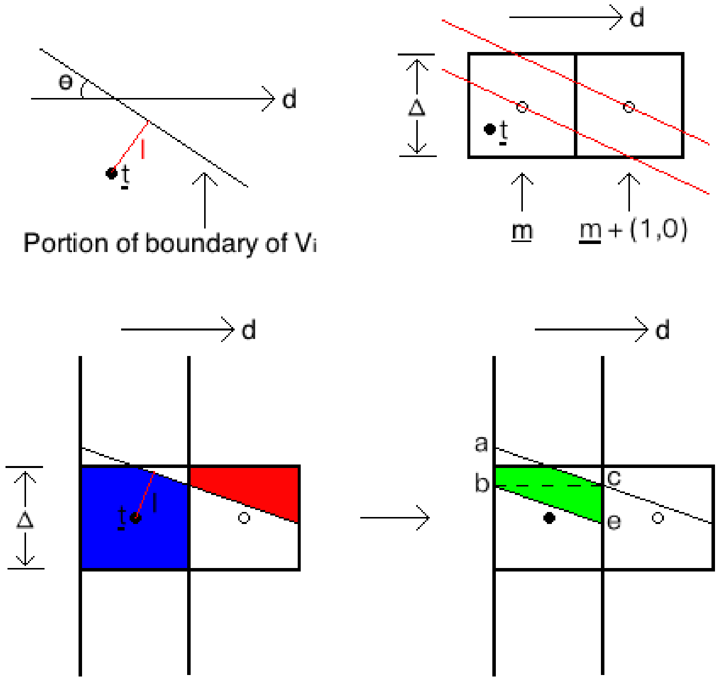


Figure 5.3: Upper left: Illustration of l and θ. Upper right: Range of tile boundary to trigger a non-zero LRI-A index at t. Second row: Demonstrate the computation of $f(l, \theta)$.

positive if the coordinate of the foot of the perpendicular line in direction d is larger than the coordinate of $\underline{t}$ in direction d, and negative otherwise), and θ is the angle between direction d and the slope of the boundary of V_i at the foot of the perpendicular line. Specifically, if direction d is rotated clockwise by θ, it is parallel to the boundary of V_i at the foot of the perpendicular line. The definitions of l and θ are illustrated in the upper left sub-image in Fig. 5.3. Due to symmetry, it is obvious that $f(l, \theta) = f(l, \pi - \theta)$. Hence, we just need to derive $f(l, \theta)$ for $0 \leq \theta \leq \frac{\pi}{2}$.

In the following, we consider horizontal direction to the right as shown in Fig. 5.3. The analysis for vertical, diagonal and anti-diagonal directions are similar. For some particular $\underline{U}$, the left square in the upper right sub-image in Fig. 5.3 is the pixel $\underline{m}$ that includes the interesting point $\underline{t}$. $\tilde{I}^d_A(X, \underline{U}, \underline{t}) \neq 0$ if $\tilde{Y}_{\underline{U}}[\underline{m}] \neq \tilde{Y}_{\underline{U}}[\underline{m} + (1, 0)]$. Equivalently, if the nearest boundary locates between the two red lines, this condition holds. Notice that the two red lines pass through the centers of pixel $\underline{m}$ and $\underline{m} + (1, 0)$, respectively. With this observation, we conclude that the centers of $\underline{m}$ and $\underline{m}+(1,0)$ should locate on different sides of the boundary in order to trigger a non-zero LRI-A index at $\underline{t}$ in direction d.

Now we can derive $f(l,\theta)$ as a function of l for any fixed θ, as illustrated in the second row of Fig. 5.3. Given a point $\underline{t}$, the center of $\underline{m}$ (such that $\tilde{W}_{\underline{m}}$ includes $\underline{t}$) can only locate at the square region centered at $\underline{t}$ with width Δ. Accordingly, the center of $\underline{m}+(1,0)$ can only locate at the next square region horizontally to the right. As discussed, the centers of $\underline{m}$ and $\underline{m}+(1,0)$ can only locate at the blue and red regions, respectively. Since there exists a 1-to-1 mapping from the center of $\underline{m}$ and the center of $\underline{m}+(1,0)$, the intersection of the blue and red regions is actually the green region. This green region can also be viewed as the intersection of the parallelogram $abce$ and the square centered at $\underline{t}$. As l decreases, the parallelogram $abce$ goes down. The ratio of the area of the green region over Δ^2 gives $\mathbb{E}_U(\mathbb{1}_{\mathbb{R}_{\neq 0}}(\tilde{I}_A^d(X,\underline{U},\underline{t}))$, which is actually $f(l,\theta)$.

First, for $0<\theta\leq\arctan\frac{1}{2}$, Figure 5.4 shows the movement of parallelogram $abce$ in the first row. And

$$f(l,\theta)=\begin{cases} \frac{\tan\theta}{2}(1-\frac{l-l_0}{\Delta\sin\theta})^2 & l\in(l_0,l_0+\Delta\sin\theta] \\ \frac{(\sin\theta+\frac{l_0-l}{\Delta})^2-2(\frac{l_0-l}{\Delta})^2}{\sin 2\theta} & l\in(l_0-\Delta\sin\theta,l_0] \\ \tan\theta & l\in(-l_0+2\Delta\sin\theta,l_0-\Delta\sin\theta] \\ f(-l+\Delta\sin\theta,\theta) & l\in(-l_0+\Delta\sin\theta,-l_0+2\Delta\sin\theta] \\ f(-l+\Delta\sin\theta,\theta) & l\in[-l_0,-l_0+\Delta\sin\theta] \end{cases},$$

where

$$l_0=\frac{\Delta\sin\theta+\Delta\cos\theta}{2}.$$

Second, for $\arctan\frac{1}{2}<\theta\leq\frac{\pi}{4}$, the movement of parallelogram $abce$ is a little different

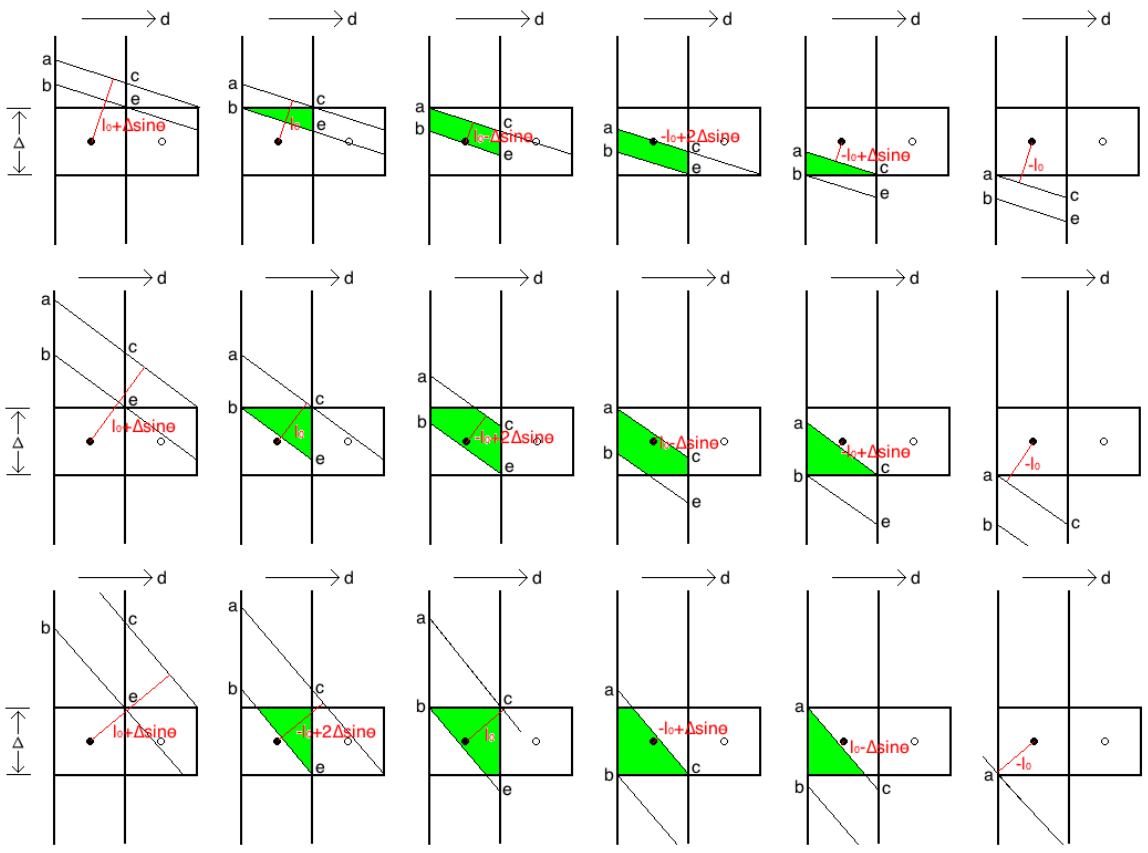


Figure 5.4: Computing $f(l, \theta)$ with different θ. First row: $0 < \theta \le \arctan \frac{1}{2}$. Second row: $\arctan \frac{1}{2} < \theta \le \frac{\pi}{4}$. Third row: $\frac{\pi}{4} \le \theta < \frac{\pi}{2}$. The solid dot in each sub-image represents t, and the left square is all possible positions of the center of pixel m.

than the first case, as shown in the second row of Fig. 5.4. And accordingly,

$$f(l,\theta) = \begin{cases} \frac{\tan\theta}{2}\left(1 - \frac{l-l_0}{\Delta\sin\theta}\right)^2 & l \in (l_0, l_0 + \Delta\sin\theta] \\ \frac{(\sin\theta + \frac{l_0-l}{\Delta})^2 - 2(\frac{l_0-l}{\Delta})^2}{\sin 2\theta} & l \in (-l_0 + 2\Delta\sin\theta, l_0] \\ 1 - \frac{(\frac{l_0-l}{\Delta})^2 + (\frac{l_0-l}{\Delta} - \cos\theta)^2}{\sin 2\theta} & l \in (l_0 - \Delta\sin\theta, -l_0 + 2\Delta\sin\theta] \\ f(-l + \Delta\sin\theta, \theta) & l \in (-l_0 + \Delta\sin\theta, l_0 - \Delta\sin\theta] \\ f(-l + \Delta\sin\theta, \theta) & l \in [-l_0, -l_0 + \Delta\sin\theta] \end{cases} .$$

At last, for $\frac{\pi}{4} \le \theta < \frac{\pi}{2}$, Figure 5.4 shows the movement of parallelogram $abce$ in row three.

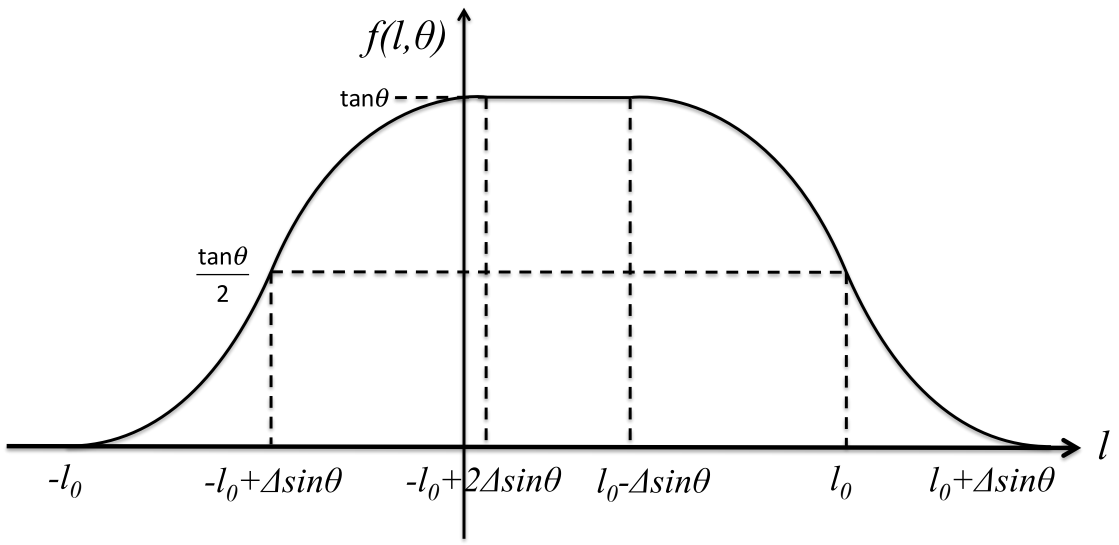


Figure 5.5: $f(l,\theta)$ for $0 < \theta \leq \arctan\frac{1}{2}$.

And

$$f(l,\theta) = \begin{cases} \frac{(\frac{l_0-l}{\Delta}+\sin\theta)^2}{\sin 2\theta} & l \in (-l_0+2\Delta\sin\theta, l_0+\Delta\sin\theta] \\ \frac{3}{2}-\frac{l}{\Delta\sin\theta} & l\in(l_0, -l_0+2\Delta\sin\theta] \\ 1-\frac{(\frac{l_0-l}{\Delta})^2+(\frac{l_0-l}{\Delta}-\cos\theta)^2}{\sin 2\theta} & l\in(-l_0+\Delta\sin\theta, l_0] \\ f(-l+\Delta\sin\theta,\theta) & l\in(l_0-\Delta\sin\theta, -l_0+\Delta\sin\theta] \\ f(-l+\Delta\sin\theta,\theta) & l\in[-l_0, l_0-\Delta\sin\theta] \end{cases}.$$

Figure 5.5, 5.6 and 5.7 present these three formulas. They are all first order continuous at each point. If $l > l_0 + \Delta\sin\theta$ or $l < -l_0$, $f(l,\theta) = 0$. Hence, we just need to consider $\underline{t}$ along the vicinity of the tile boundary.

Up to now, we have the formulas of $f(l,\theta)$ for $0 < \theta < \frac{\pi}{2}$. If $\theta = 0$, $f(l,0) = 0, \forall l$. If $\theta = \frac{\pi}{2}$, we have

$$f(l,\frac{\pi}{2}) = \begin{cases} \frac{3}{2}-\frac{l}{\Delta} & l\in(\frac{\Delta}{2},\frac{3\Delta}{2}] \\ \frac{1}{2}+\frac{l}{\Delta} & l\in[-\frac{\Delta}{2},\frac{\Delta}{2}] \end{cases}.$$

Now we can compute $\Pr(\tilde{I}^d_A(X,\underline{U},\underline{T}) \neq 0)$ by averaging $\mathbb{E}_{\underline{U}}\big[\mathbb{1}_{\neq 0}(\tilde{I}^d_A(X,\underline{U},\underline{t}))\big]$ over $\underline{t}$. For a periodic tessellation, instead of considering all $\underline{t}$, we can focus on one tile V_i and integrate along its boundary B_{V_i}. The result is organized in the following lemma.

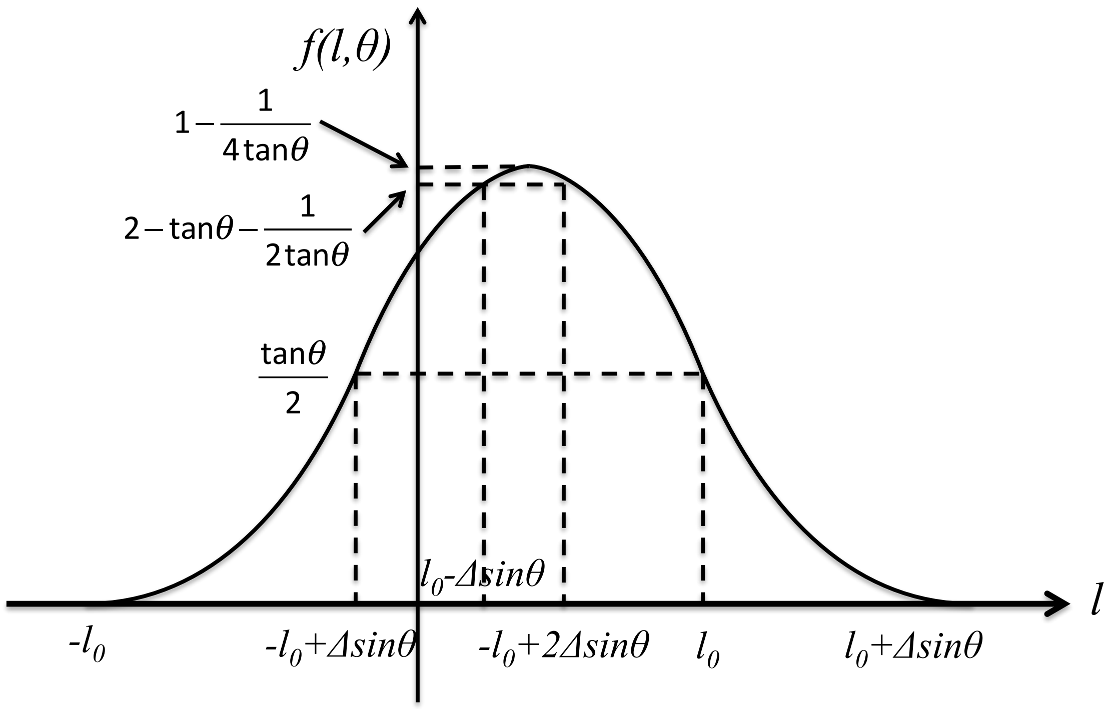


Figure 5.6: $f(l,\theta)$ for $\arctan\frac{1}{2} < \theta \leq \frac{\pi}{4}$.

Lemma 1. *After pixelation of a continuous tessellation with small pixel size Δ, the fraction of pixels that have an intensity transition in horizontal direction is*

$$\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) \approx \frac{\int_{B_{V_i}} \int_{-l_0(u)}^{l_0(u)+\Delta\sin\theta(u)} f(l,\theta(u))\mathrm{d}l\mathrm{d}u}{2|V_i|},$$

where $|V_i|$ denotes the area of V_i. Moreover, $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0)$ is proportional to Δ.

Note that the factor 2 in the above denominator is due to the fact that each boundary is double counted. The fact that $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0)$ is proportional to Δ can be proved by observing that $f(l,\theta)$ is a function of $\frac{l}{\Delta}$ for all l. After integrating $f(l,\theta)$ over a range of which both ends are proportional to Δ, the resulting formula is proportional to Δ.

As an aside, we note that this analysis can be generalized to non-periodic textures described by a partition of cells, with each cell having a constant intensity and adjacent cells having different intensities. In this case, instead of integrating over boundaries of a tile, we integrate $f(l,\theta)$ over all boundaries in the texture and divide it by the area of the entire

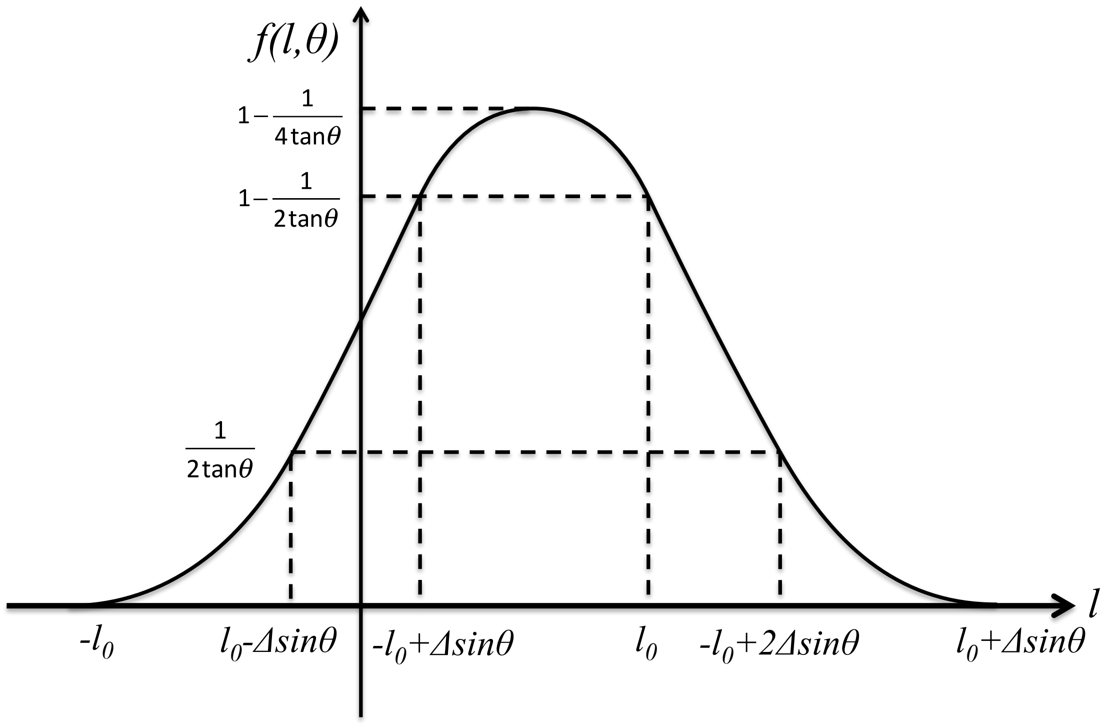


Figure 5.7: $f(l,\theta)$ for $\frac{\pi}{4} \le \theta < \frac{\pi}{2}$.

texture. Specifically,

$$\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) \approx \frac{\int_{\text{boundary}} \mathrm{d}u \int_{-l_0(u)}^{l_0(u)+\Delta\sin\theta(u)} f(l,\theta(u))\mathrm{d}l}{\text{area of texture}} .$$

For a general texture, we still have $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) \propto \Delta$.

Now two examples will be discussed. For tessellation of squares in Fig. 5.8, if d is horizontal direction,

$$\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) = \frac{2w\int_{-0.5\Delta}^{1.5\Delta}(f(l,\frac{\pi}{2})+f(l,0))\mathrm{d}l}{2w^2} = \frac{1}{w}\Delta .$$

If d is diagonal direction, similarly, $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) = \frac{2}{w}\Delta$. For tessellation of diamonds (with width w) in Fig. 5.8, if d is horizontal direction,

$$\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) = \frac{4w\int_{-\frac{\sqrt{2}}{2}\Delta}^{\sqrt{2}\Delta} f(l,\frac{\pi}{4})\mathrm{d}l}{2w^2} = \frac{\sqrt{2}}{w}\Delta .$$

If d is diagonal direction, similarly, $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) \neq 0) = \frac{\sqrt{2}}{w}\Delta$.

We now seek $f_{A,d}(s)$ for $s > 0$. Let $B_{V_i,s}^d = \{\underline{t} \in B_{V_i} : \underline{t} + \underline{d}\cdot s \in B_{V_i}\}$, so that

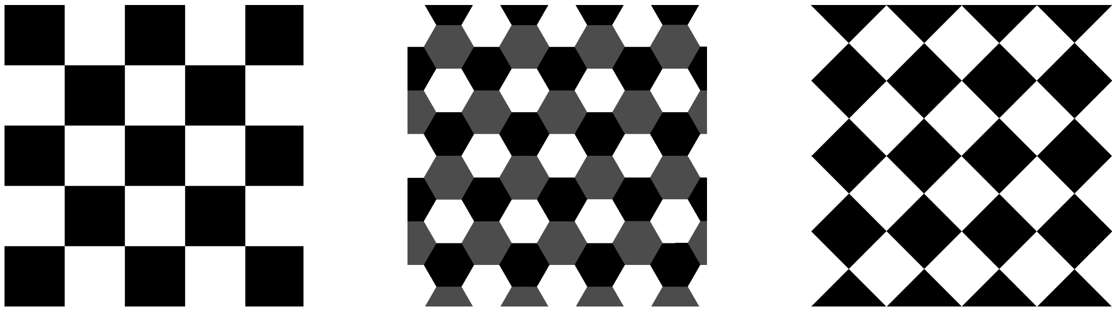


Figure 5.8: Checkerboard, hexagon and diamond tessellations.

$L^d(\underline{t}) = s$ for all $\underline{t} \in B^d_{V_i,s}$. As an example, points a and a' in Fig. 5.1 belong to $B^d_{V_i,s}$. Let us analyze point a, and a' is very similar. $\forall \underline{t} \in \overline{ab} \overset{\Delta}{=} r(a)$ satisfies that $s \leq L^d(\underline{t}) \leq s + \epsilon$. And

$$|r(a)| = |\overline{ab}| = \frac{\epsilon \times \sin\beta(a)}{\sin(\pi - \alpha(a) - \beta(a))},$$

where $\alpha(a)$ and $\beta(a)$ are defined in Fig. 5.1. For $s > 0$,

$$\begin{aligned} f_{A,d}(s) &= f_{L^d(\underline{T})|\underline{T}\in B^d_{V_i}}(s) \\ &= \lim_{\epsilon\to 0} \frac{\Pr(s \leq L(\underline{\tau}) \leq s+\epsilon | \underline{\tau} \in B^d_{V_i})}{\epsilon} \\ &= \frac{1}{|B^d_{V_i}|} \sum_{\underline{t}\in B^d_{V_i,s}} |r(\underline{t})| \\ &= \frac{1}{|B^d_{V_i}|} \sum_{\underline{t}\in B^d_{V_i,s}} \frac{\sin\beta(\underline{t})}{\sin(\pi - \alpha(\underline{t}) - \beta(\underline{t}))}. \end{aligned}$$

For convex shapes, $|B^d_{V_i,s}| \in \{0, 1, 2, \infty\}$. For nonconvex shapes, $|B^d_{V_i,s}|$ could take other positive integer values. $|B^d_{V_i,s}| = \infty$ if and only if $\alpha(\underline{t}) + \beta(\underline{t}) = \pi$, or there exist two lines separated by s on the boundary of a tile such that one is a translated version of the other in direction d. In this case, $f_{A,d}(s)$ has a delta function $\delta(\cdot)$ at s.

In summary,

$$\begin{aligned} \Pr(I^d_A[\tilde{Y}_{\underline{U}}, \underline{M}] = z) \;&=\; \Pr\left(\tilde{I}^d_A = z\right) \\ &\cong \delta[z] \cdot \Pr\left(\tilde{I}^d_A = 0\right) + \int_{z\Delta-\frac{\Delta}{2}}^{z\Delta+\frac{\Delta}{2}} f_{A,d}(\tau)\,\mathrm{d}\tau \cdot \Pr(\tilde{I}^d_A \neq 0), \end{aligned}$$

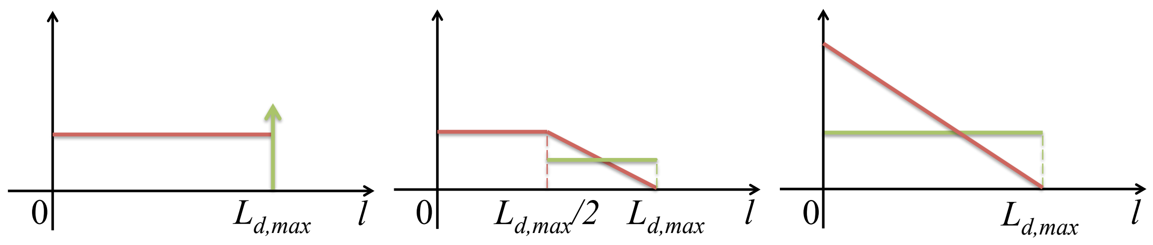


Figure 5.9: $f_{D,d}(s)$ (red) and $f_{A,d}(s)$ (green) for tessellations of square, hexagon and diamond, respectively. $L_{d,max}$ denotes the largest possible LRI index in each tessellation.

where $\Pr(\tilde{I}_A^d \neq 0)$ is given in Lemma 1 and $f_{A,d}(\tau)$ is given above.

Now several examples of $f_{D,d}(s)$ and $f_{A,d}(s)$ are given. For the checkerboard pattern in Fig. 5.8 and horizontal direction d,

$$f_{D,d}(s) = \frac{1}{L_{d,\max}}\ ,\ \ f_{A,d}(s) = \mathrm{P}_1\delta(s) + (1-\mathrm{P}_1)\delta(s - L_{d,\max})\ ,$$

where $\mathrm{P}_1 = \Pr(\tilde{I}_A^d(X, \underline{U}, \underline{T}) = 0)$ and $0 \leq s \leq L_{d,\max}(= w)$. For the diamond tessellation in Fig. 5.8 and direction d,

$$f_{D,d}(s) = \frac{2(L_{d,\max} - l)}{L_{d,\max}^2}\ ,\ \ f_{A,d}(s) = \mathrm{P}_1\delta(l) + (1-\mathrm{P}_1)\frac{1}{L_{d,\max}}\ .$$

For the hexagon tessellation in Fig. 5.8 and direction d,

$$f_{D,d}(s) = \begin{cases} \frac{4}{3L_{d,\max}} & \text{for} \quad 0 \leq s \leq \frac{L_{d,\max}}{2} \\ \frac{8(L_{d,\max}-l)}{3L_{d,\max}^2} & \text{for} \quad \frac{L_{d,\max}}{2} < s \leq L_{d,\max} \end{cases}$$

and

$$f_{A,d}(s) = \mathrm{P}_1\delta(l) + (1-\mathrm{P}_1)\frac{2}{L_{d,\max}}(H(\frac{L_{d,\max}}{2}) - H(L_{d,\max}))\ ,$$

where $H(\cdot)$ represents the Heaviside step function. The corresponding $f_{D,d}(s)$ and $f_{A,d}(s)$ of these three tessellations are shown in Fig. 5.9.

5.5 Relationship of LRI-A and LRI-D

One interesting observation from Fig. 5.9 is that the $f_{A,d}(s)$ behaves like the negative derivative of the $f_{D,d}(s)$. This is because on the one hand, $f_{D,d}(s_0)$ represents the den-

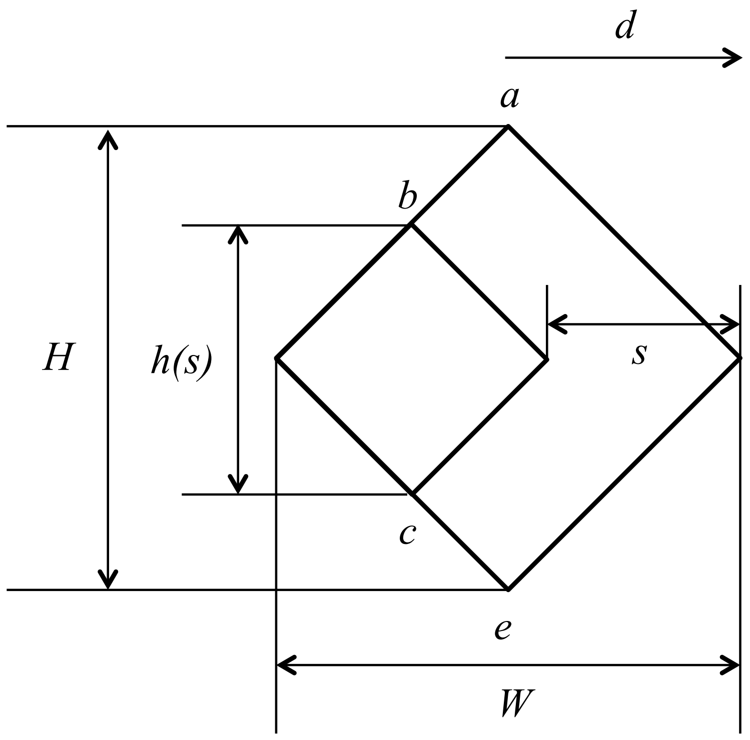


Figure 5.10: One tile V in a tessellation.

sity of points whose distances to the nearest edges in direction d equal to s_0. In other words, the inter-edge distances passing through these points must be greater than or equal to s_0. Hence, $f_{D,d}(s_0)$ is closely related to 1 minus the cumulative distribution function of the inter-edge distances in direction d. On the other hand, $f_{A,d}(s_0)$ represents the fraction of inter-edge distances in direction d that equal to s_0, which is essentially the probability density function of the inter-edge distances. We will prove this explanation in this section and the analysis reveals the fundamental relationship of LRI-A and LRI-D.

Due to the periodicity of tessellation, we can focus our analysis on one tile V, with area $|V|$, as shown in Fig. 5.10. Given horizontal direction d, we say that V has *size* W in direction d and *size* H in direction perpendicular to d. We first derive formulas for $f_{A,d}(s)$, and then link $f_{A,d}(s)$ to $f_{D,d}(s)$. In this section, we consider V to be pixelated with small pixel size Δ as before. In a discrete image, we define a *chord* as a straight line (a sequence of pixels) in direction d whose both ending points reside on the boundary of V. There only exists finite number of chords that begin and end in the center of pixels in direction d in V, namely $\frac{H}{\Delta}$. The leftmost pixel on each chord (or its neighboring pixel in direction d) has a non-zero LRI-A index equal to the length of the chord. Actually, the union of these leftmost pixels (or their neighboring pixels in direction d) is the set B_V^d. Recall that in Section 5.4, we have

$$\begin{aligned}\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z) &= \Pr\left(\tilde{I}_A^d = z\right)\\ &\cong \mathbb{1}_{=0}(z) \cdot \Pr\left(\tilde{I}_A^d{=}0\right) + \int_{z\Delta-\frac{\Delta}{2}}^{z\Delta+\frac{\Delta}{2}} f_{A,d}(\tau)\,\mathrm{d}\tau \cdot \Pr(\tilde{I}_A^d \neq 0)\,.\end{aligned}$$

From above we know for an integer $z > 0$,

$$\frac{\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = z)}{1 - \Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = 0)} = \int_{z\Delta-\frac{\Delta}{2}}^{z\Delta+\frac{\Delta}{2}} f_{A,d}(\tau)\,\mathrm{d}\tau \cong \Delta f_{D,d}(z\Delta)\,.$$

Summing the above over all the integers between z and $\frac{W}{\Delta}$ gives

$$\frac{\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] \geq z)}{1 - \Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = 0)} \cong \int_{z\Delta}^{W} f_{A,d}(\tau)\,\mathrm{d}\tau\,.$$

Only for $\underline{M} \in B_V^d$ do we have $\Pr(I_A^d[Y, \underline{M}] \neq 0)$. The left hand side in the above equation measures the fraction of pixels in B_V^d having LRI-A index greater than or equal to z, which is $\frac{h(z\Delta)}{H}$, where $h(\cdot)$ is illustrated in Fig.5.10. So we have

$$\frac{\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] \geq z)}{1 - \Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = 0)} = \frac{h(z\Delta)}{H} \cong \int_{z\Delta}^{W} f_{A,d}(\tau)\,\mathrm{d}\tau\,.$$

When Δ is very small, we can substitute $z\Delta$ for any positive real number $s < W$:

$$\int_{s}^{W} f_{A,d}(\tau)\,\mathrm{d}\tau = \frac{h(s)}{H}\,.$$

Before we derive the formula for $f_{D,d}(s)$, we first show a lemma that will be useful.

Lemma 2.

$$|V| \cong H \cdot \int_{0^+}^{W} \tau \cdot f_{A,d}(\tau)\mathrm{d}\tau\,.$$

Proof.

$$
\begin{aligned}
|V| &= \sum_{i=1}^{W/\Delta} i\Delta \cdot \frac{\Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = i)}{1 - \Pr(I_A^d[\tilde{Y}_{\underline{U}}, \underline{M}] = 0)} H \\
&\cong \sum_{i=1}^{W/\Delta} i\Delta \cdot \Delta f_{A,d}(i\Delta) H \\
&= H \sum_{j=\Delta}^{W} j \cdot f_{A,d}(j)\Delta \\
&\cong H \int_{0^+}^{W} \tau f_{A,d}(\tau)\mathrm{d}\tau \, .
\end{aligned}
$$

□

In the proof, the first equality holds since $|V|$ can be computed as the summation of areas of chords with different lengths, ranging from 1 to W/Δ. Note that $\int_{0^+}^{W} \tau f_{A,d}(\tau)\mathrm{d}\tau$ is the average length of chords in direction d.

Now let us link $f_{D,d}(s)$ to $f_{A,d}(s)$. Recall that

$$
f_{D,d}(s) = \lim_{\epsilon \to 0} \frac{\Pr(s \le L^d(\underline{T}) \le s + \epsilon)}{\epsilon} \, ,
$$

where $L^d(\underline{t})$ equals the distance in direction d from $\underline{t}$ in the image domain to the nearest boundary of a tile if the boundary crossing is a positive change of intensity, and minus the distance if the boundary crossing is a negative change of intensity. As before, we assume all transitions in direction d have a positive change in intensity. Under this setting, $f_{D,d}(s)$ measures the probability density of uniformly choosing a point $\underline{t} \in V$ such that $L^d(\underline{t}) = s$. We can characterize the random occurrence of $\underline{t}$ into two steps. First, choose a chord based on some probability distribution. And then uniformly choose a point on the chord. Since the random occurrence of $\underline{t}$ is uniform in V, longer chords have higher probability to be chosen than shorter ones. Based on Lemma 2, a chord with length τ is chosen with probability

$$
\frac{\tau f_{A,d}(\tau)}{\int_{0^+}^{W} r f_{A,d}(r)\mathrm{d}r} \, .
$$

In order to choose $\underline{t}$ on a chord C such that $L^d(\underline{t}) = s$, the length of C must be at least s.

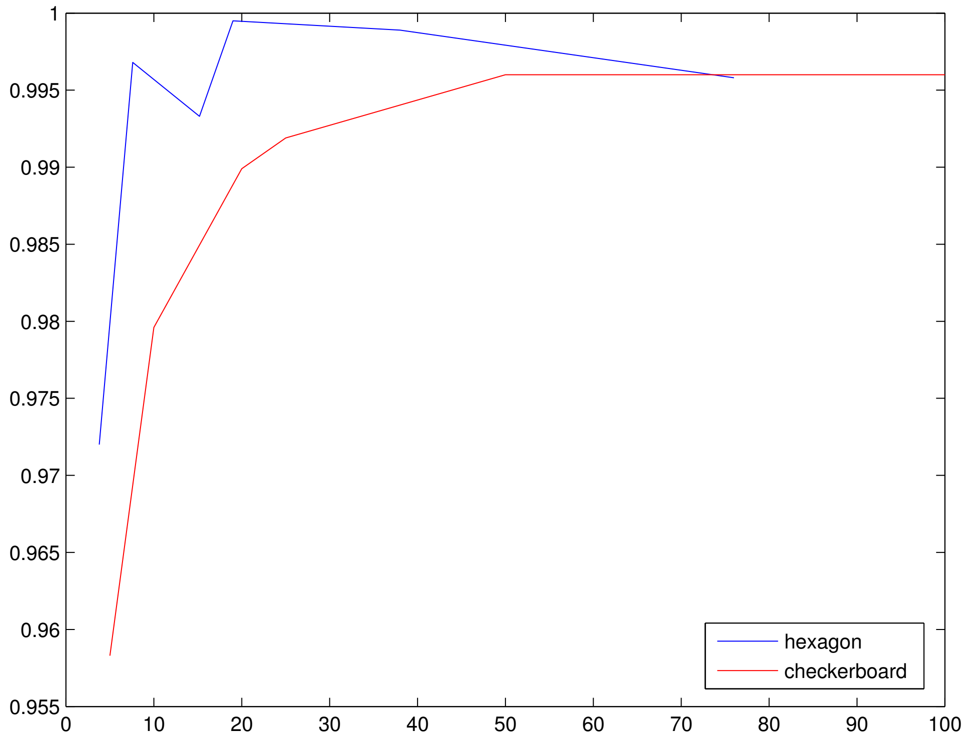


Figure 5.11: Accuracy of theoretical value of $\Pr(\text{LRI-A} \neq 0)$.

Hence, we have

$$f_{D,d}(s) = \int_s^W \frac{\tau f_{A,d}(\tau)}{\int_{0+}^W r f_{A,d}(r)\mathrm{d}r} \cdot \frac{1}{\tau}\mathrm{d}\tau = \frac{\int_s^W f_{A,d}(\tau)\mathrm{d}\tau}{\int_{0+}^W r f_{A,d}(r)\mathrm{d}r} = \frac{h(s)/H}{|V|/H} = \frac{h(s)}{|V|}\,.$$

This equation enables us to link $f_{D,d}(s)$ to $f_{A,d}(s)$:

$$f_{D,d}(s) = \frac{\int_s^W f_{A,d}(\tau)\mathrm{d}\tau}{|V|/H}\,,$$

or

$$f_{A,d}(s) = -\frac{|V|}{H} \cdot \frac{\mathrm{d}f_{D,d}(s)}{\mathrm{d}s}\,.$$

This explains why $f_{A,d}(s)$ behaves like the negative derivative of $f_{D,d}(s)$. This analysis reveals the fundamental relationship of LRI-A and LRI-D.

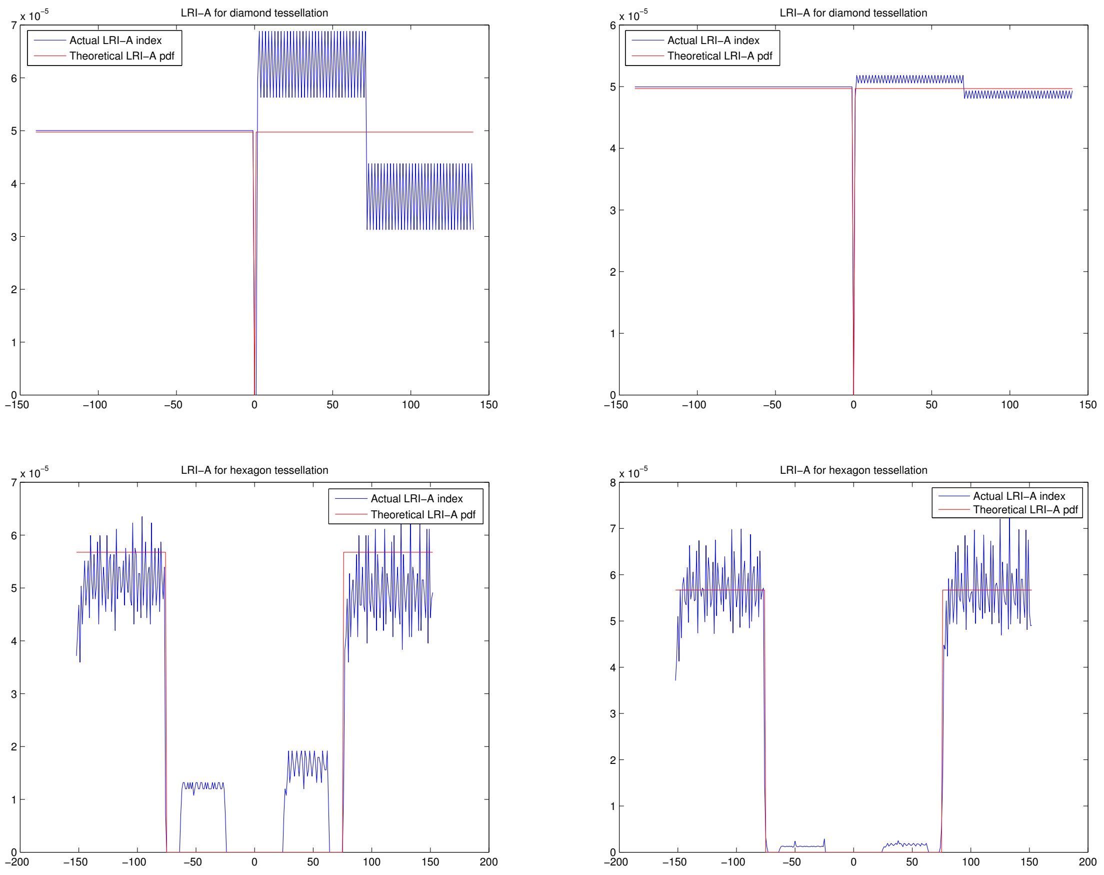


Figure 5.12: Theoretical vs. actual LRI-A distributions for diamond (first row) and hexagon (second row) tessellations.

5.6 Numerical Results

Figure 5.11 shows the *accuracy* of the Lemma 1 formula for $\Pr(\text{LRI-A} \neq 0)$ for the checkerboard and hexagonal tessellations of Fig. 5.8, as a function of the ratio of fundamental cell side length w to pixel size Δ. By *accuracy* we mean one minus the ratio of the absolute difference between the actual $\Pr(\text{LRI-A} \neq 0)$ for a discrete image and the formula, and the formula itself. Note that in this experiment we use the exact checkerboard and hexagonal tessellation images in Fig. 5.8, which have five and eight horizontal repetitions of the fundamental cells, respectively. One can see that the accuracy increases rapidly, and is greater than 0.99 for $\frac{w}{\Delta} > 20$.

Given that the accuracy of $\Pr(\text{LRI-A} \neq 0)$ is high, to match the remainder of the LRI-

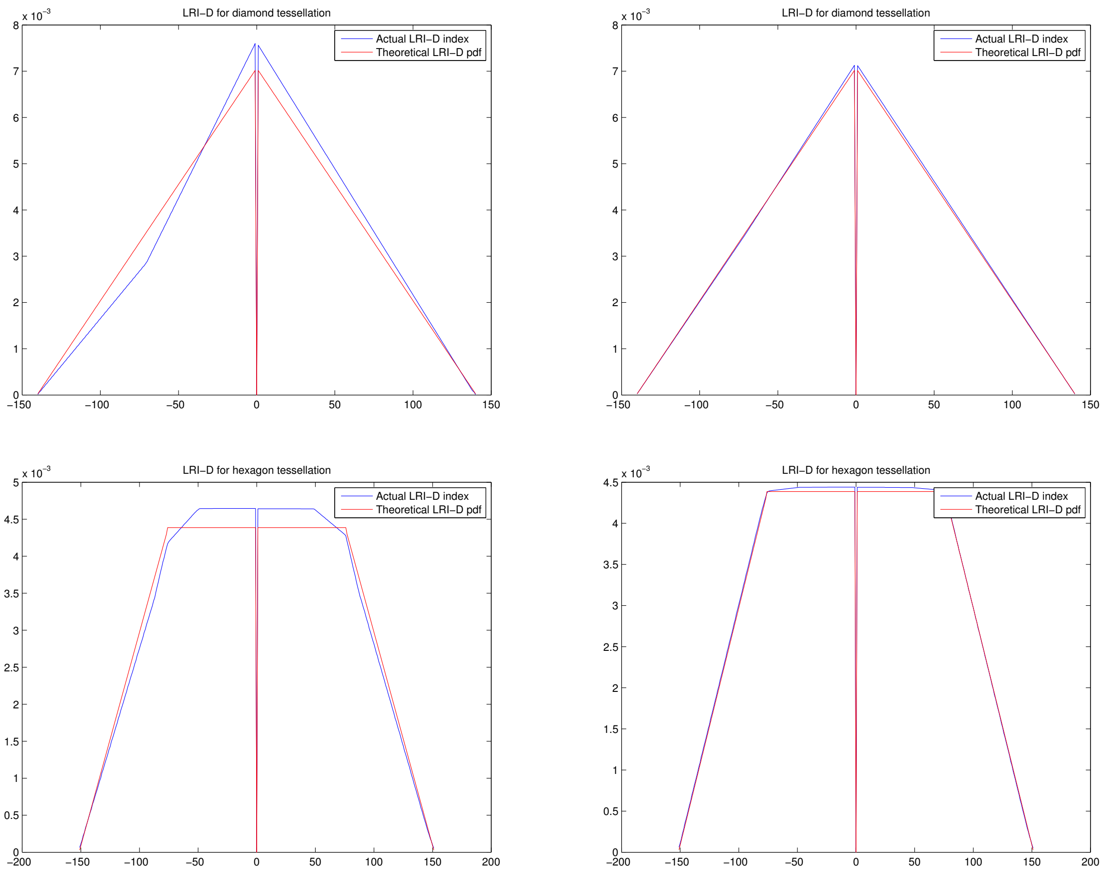


Figure 5.13: Theoretical vs. actual LRI-D distributions for diamond (first row) and hexagon (second row) tessellations.

A distribution, the predictions for $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) = z)$ simply need to be accurate for nonzero integer values of z. When the positivity and negativity of transitions are considered, one needs to compare $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) = z)/2$ to the actual LRI-A distributions for $\pm z$. This approximation is valid if our assumption that transitions in direction d in X are equally like to be positive or negative is satisfied. For the diamond and hexagon tessellations in Fig. 5.8 and horizontal direction, this condition is satisfied. Figure 5.12 shows numerical results, with the first row comparing the theoretical $\Pr(\tilde{I}_A^d(X,\underline{U},\underline{T}) = z)$ to the actual LRI-A distributions for a tessellations by diamonds and the second row for tessellations by hexagons. The two figures in the first column are based on the images in Fig. 5.8. Since these have only a few horizontal tiles, boundary effects cause mismatches of theoretical and actual LRI-A distributions. This effect can be expected to decrease as the images include more tiles. To test this hypothesis, we horizontally repeated the images in Fig. 5.8 ten times, recomputed the distributions, and show the results in the second column in Fig. 5.12. As expected, the theoretical and the actual LRI-A distributions become more similar. Note that in the second row, the actual LRI-A distribution has some fluctuations that cannot be explained by the theoretical distribution. This is partially because the pixelation of a hexagon will introduce round-up errors due to the existence of 120° angles. With the increase of $\frac{w}{\Delta}$, the fluctuations will become smaller.

The same accuracy tests were conducted for LRI-D using the tessellations of diamonds and hexagons in Fig. 5.8, and results are shown in Fig. 5.13. Again, the accuracies of the formulas increase as the image contains more horizontal tiles.

CHAPTER 6

Application to Photographic Paper Classification

6.1 Introduction

Since the early 20th century, photographic paper manufacturers have textured their products in different ways in order to provide photographers with a variety of esthetic effects and to distinguish their papers from others [99]. As photographers make conscious choices of paper, if textures can be distinguished, then analyses of the texture on which a photograph is printed can be useful in attributing it to a particular photographer or period of time. Similarly, it can be useful to distinguish contemporary papers developed for inkjet printers.

To facilitate such photographic paper analysis, Messier *et al.* [99, 100] constructed two datasets by taking raking light images of various papers. One dataset, called *b&w*, contains images of black and white silver gelatin photographic papers and the other, called *inkjet*, contains images of inkjet papers. Each dataset contains 120 high-resolution images from different manufacturers and brands. The datasets are challenging due to the high similarity of the different types of papers.

With this data set in mind, a number of texture similarity metrics have been proposed, including those based on eigentextures [100, 101], random-feature textons [100], anisotrpoic wavelet multiscale analysis [100, 102], pseudo-area-scale analysis [100, 103], POEM+VLAT [104], and HOG+VLAT [104]. The first four were stimulated by a collaborative competition organized by Messier, Johnson and Klein *et al* [100]. These methods were tested on the b&w and inkjet datasets described above and achieved good performance.

In this chapter, we apply Local Radius Index (LRI), more specially, LRI-A, and the similarity metrics based on it, to classify photographic papers in the aforementioned two datasets. LRI extracts texture features in spatial domain using simple pixel level operations. Hence, it is computationally very efficient. When combined with complementary features such as Local Binary Patterns LBP [27], and newly proposed Subband Contract Distribution (SCD) and Intensity Penalization (IP), LRI-based metrics have performed as well as state-of-the-art metrics, and sometimes significantly better, in problems such as identical texture retrieval, image compression, and texture classification. Moreover, the LRI based metrics require substantially less computation than the other state-of-the-art metrics. In the process of applying LRI to paper texture classification, it is found in the present work that for this application the metric can benefit from an improved method of adapting its threshold T to specific images. The results of this chapter show that classification based on the improved LRI metric outperforms previous methods, both on the groupings of images considered in previous analyses, as well as in some new groupings. The latter also suggest the ability to distinguish photographic papers by brand/style and reflectance.

6.2 Improved Threshold Selection for LRI Feature

As mentioned in Chapter 3, two parameters K and T govern the operation of LRI. The size limit K restricts the maximum size of texture elements that can be detected by LRI-A. As suggested in [34], $K = 4$ is usually large enough. The threshold T determines what is considered to be an edge in a texture pattern and also controls the noise sensitivity of the operator. Large T makes the LRI-A operator noise insensitive and only sharp edges are detected. Small T has the opposite effect. To better adapt to a texture pattern, T should be image dependent. In [34], T was chosen to be $\sigma/2$, where σ is the standard deviation of the image. This value was intended to be small enough to enable most of the interesting edges in texture patterns to be recognized, and at the same time, large enough so that it will not include small intensity transitions that are not important to the human eye. Experimental results in Section 3.5 showed that this choice is suitable for many texture patterns.

To provide an idea of how T influences texture classification, Figure 6.1 plots classification accuracy vs. D, when (a) $T = \sigma/D$, (b) the classification experiment is the one described in detail in Section 6.3.1 on the inkjet dataset grouped into the nine classes described there, each containing 10 images, and (c) classification accuracy is measured by three performance measures used there: P@1, MRR and MAP. For each performance measure, a high value indicates good accuracy. As can be seen, as T decreases, classification performance increases substantially and then plateaus when D reaches four. The fact that $D = 4$ works better in this experiment than $D = 2$, which worked well for the datasets in Section 3.5 suggests that D should be adapted to the images being classified.

Whereas the idea in Section 3.5 was to choose threshold $T = \sigma/2$ to be proportional to the variance of the image, the new idea is that the threshold T should depend on how much pixel-to-pixel variation the image contains. For example, although the image from the inkjet dataset shown in Fig. 6.2 has considerable variance, it has very small pixel-to-pixel changes, i.e., it is very smooth, and the texture is barely visible. Thus when $T = \sigma/2$, very few edges are detected and the LRI feature vector contains little distinctive information.

The new approach bases the threshold T on the distribution of interpixel differences. In particular, given an image, let $F(x)$ denote the empirical cumulative distribution function (CDF) for the magnitude difference between adjacent pixels horizontally, vertically and diagonally. For example, Figure 6.2 shows the CDF of the inkjet image shown there. We now propose

$$T = F^{-1}(1 - P)\,,$$

where P is a parameter such that when T is chosen as above, the fraction of pixel differences whose magnitudes exceed T is P. For example, if $P = 1$, then $T = 0$, and an

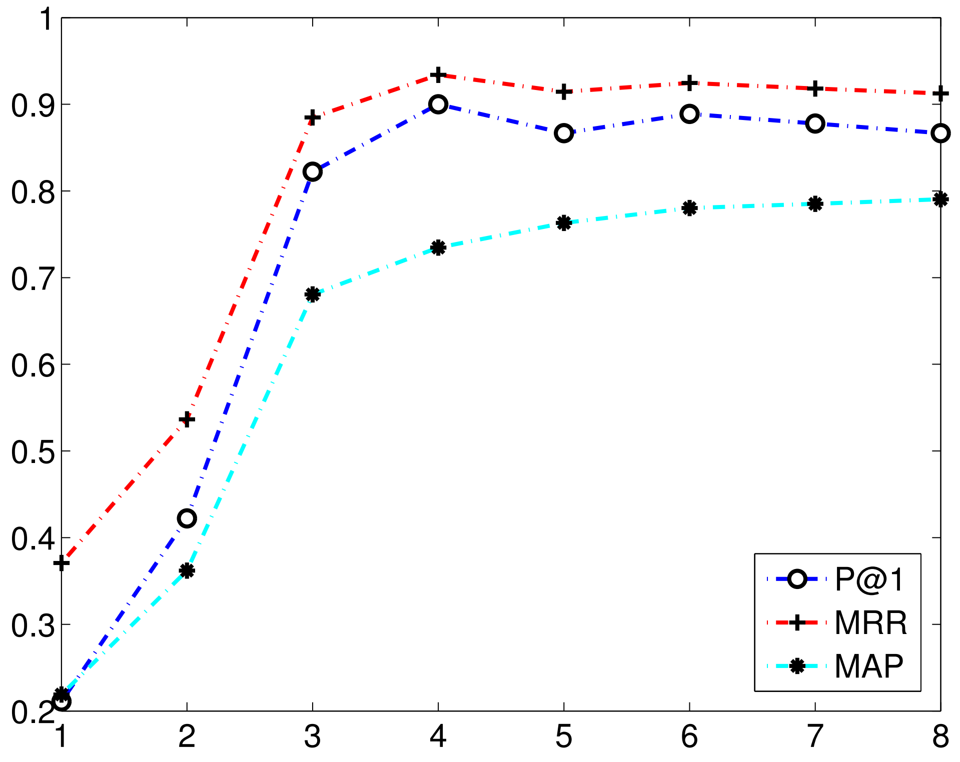


Figure 6.1: Classification accuracy vs. D for the inkjet dataset using LRI with threshold $T = \sigma/D$.

edge is *detected* between every pair of adjacent pixels. For several values of P, Table 6.1 shows classification accuracy on the inkjet dataset used in Fig. 8.13 when T is chosen as above. One can see that performance is good for P ranging from 0.1 to 0.4, i.e., it is not overly sensitive to the choice of T. Based on experiments such as shown in this table, we adopt $P = 0.3$ in this paper. For example, for the image in Fig. 6.2, this results in $T = 0.083 \times \sigma$. In contrast, choosing $T = \sigma/2$ corresponds to $F^{-1}(1 - P) = 0.5$, or equivalently, $F(0.5) = 1 - P$, which implies $P \approx 0$, i.e., essentially no edges are detected. As one may notice, actually the proposed threshold adaptation procedure is independent of σ, and purely depends on the empirical CDF of the interpixel differences.

Table 6.1: Classification accuracy for the inkjet dataset using LRI with several choices of P determining the threshold T.

P	P@1	MRR	MAP
0.1	86.7%	90.5%	77.8%
0.2	88.9%	90.7%	78.7%
0.3	88.9%	91.0%	78.8%
0.4	87.8%	90.7%	79.0%

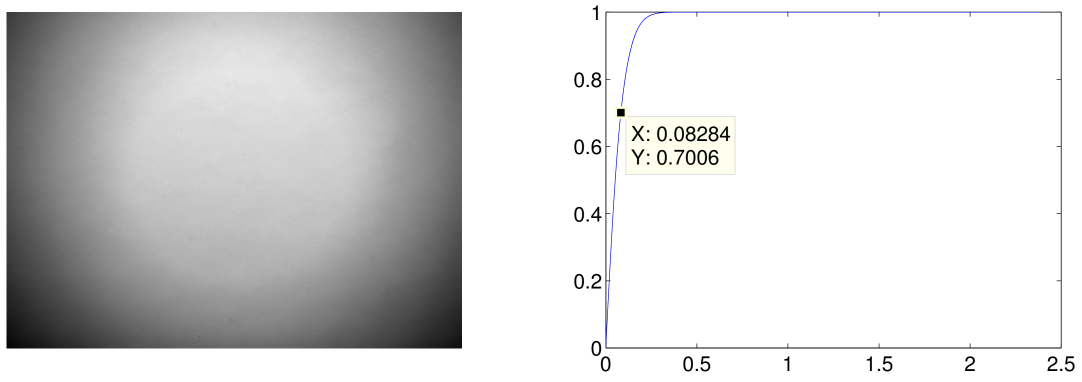


Figure 6.2: Left: a sample image from the inkjet dataset. Right: empirical CDF of interpixel differences.

6.3 Experiments and Results

In this section, we describe the results of applying LRI and LRI$^+$ with the new threshold adaptation procedure to the b&w and inkjet datasets [99, 100]. Each dataset contains 120 high-resolution photographic papers. As described in [100], the b&w dataset has the following structure:

a) Images 1-30 are taken from 3 papers, 10 images from each paper, with each paper having a different manufacturer and/or style, e.g., intended reflectance.

b) Images 31-60 are taken from 30 different papers, 10 papers coming from each of 3 different packages. Each package has a different manufacturer and/or style.

c) Images 61-70 are taken from 10 different papers, all manufactured by Kodak, all listed as glossy, dated from 1928 to 1939, with several different brand names.

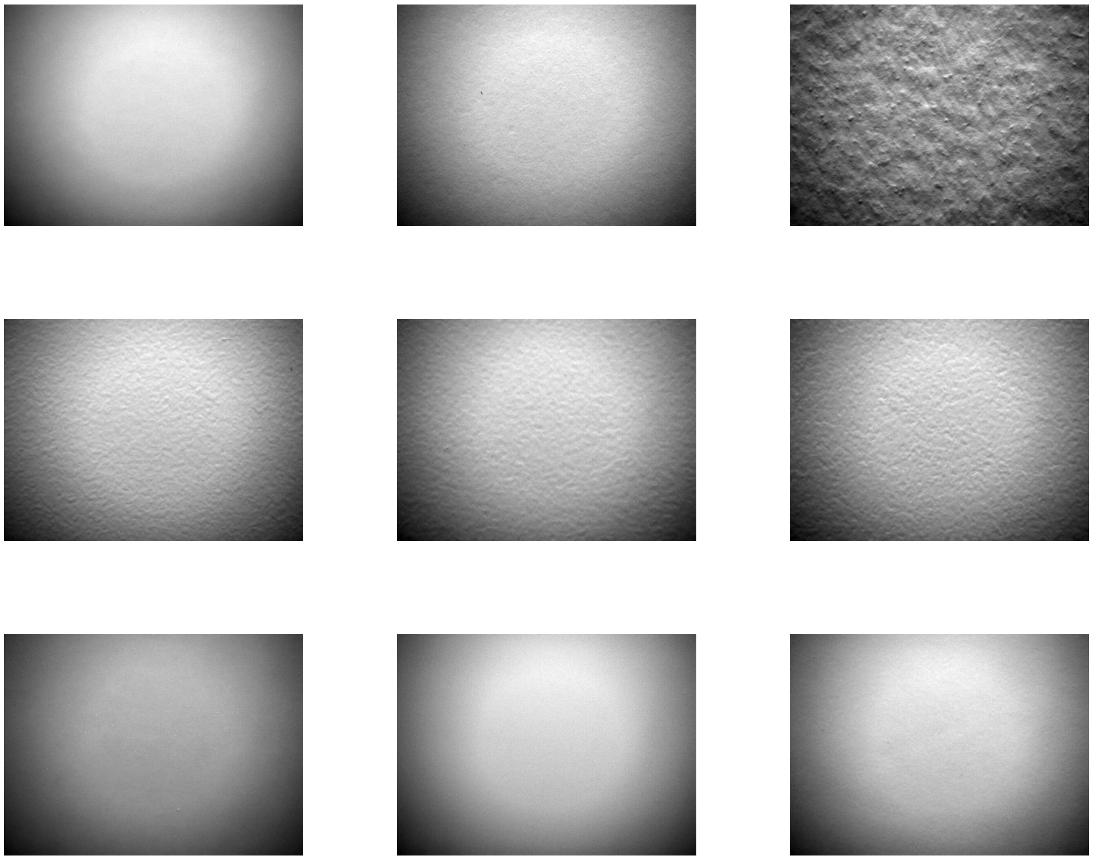


Figure 6.3: Images $9i + 1$, for $i = 0, 2, \ldots, 8$, from the inkjet dataset.

d) Images 71-80 are taken from 10 different papers, all manufactured by Dupont, all listed as semi-matte, dated from 1951 to 1958, with two different brand names.

e) Images 81-90 are taken from 10 different papers, all manufactured by Agfa, listed as lustre or glossy, dated from 1955 to 1976, with two different brand names.

f) Images 91-120 are randomly selected papers, including some from the previously groups. It is intended to increase the difficulty of classification.

The inkjet dataset has a similar structure.

Figures 6.3 shows images $9i+1$, for $i = 0, 2, \ldots, 8$, from the inkjet dataset. One can see that except for image 21, all the other eight photographic papers are very similar, with very

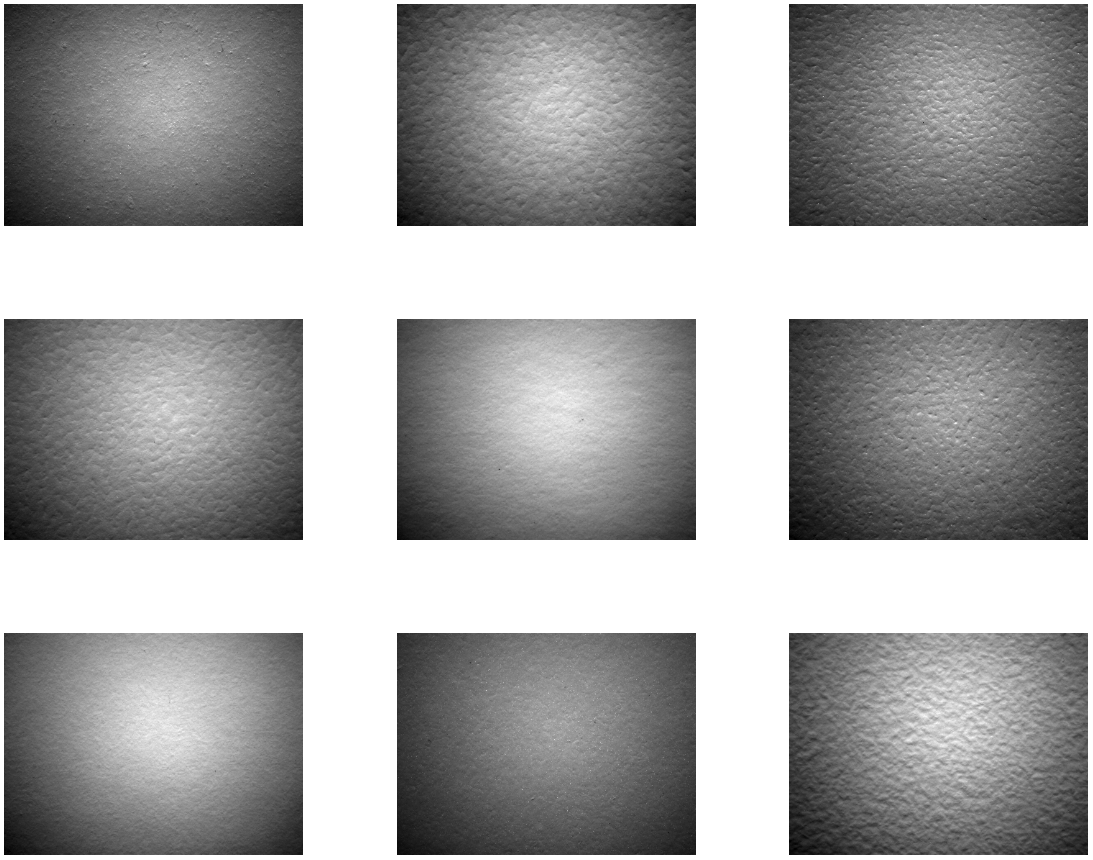


Figure 6.4: Images $9i + 1$, for $i = 0, 2, \ldots, 8$, from the b&w dataset.

detailed texture patterns. Similarly, Figure 6.4 shows images $9i + 1$, for $i = 0, 2, \ldots, 8$, from the b&w dataset. Images from the b&w dataset are even more similar to each other than images from the inkjet dataset. From images shown in these two figures, we can see how challenging it is to classify photographic papers. The resolution of these images is very high. Each image in the inkjet and b&w datasets has 1536×2080 pixels and measures only 1.00×1.35 cm^2 of the photographic paper. Also note that the imaging system causes the low frequency shading at the four corners of these images that is not intrinsic to the photographic papers.

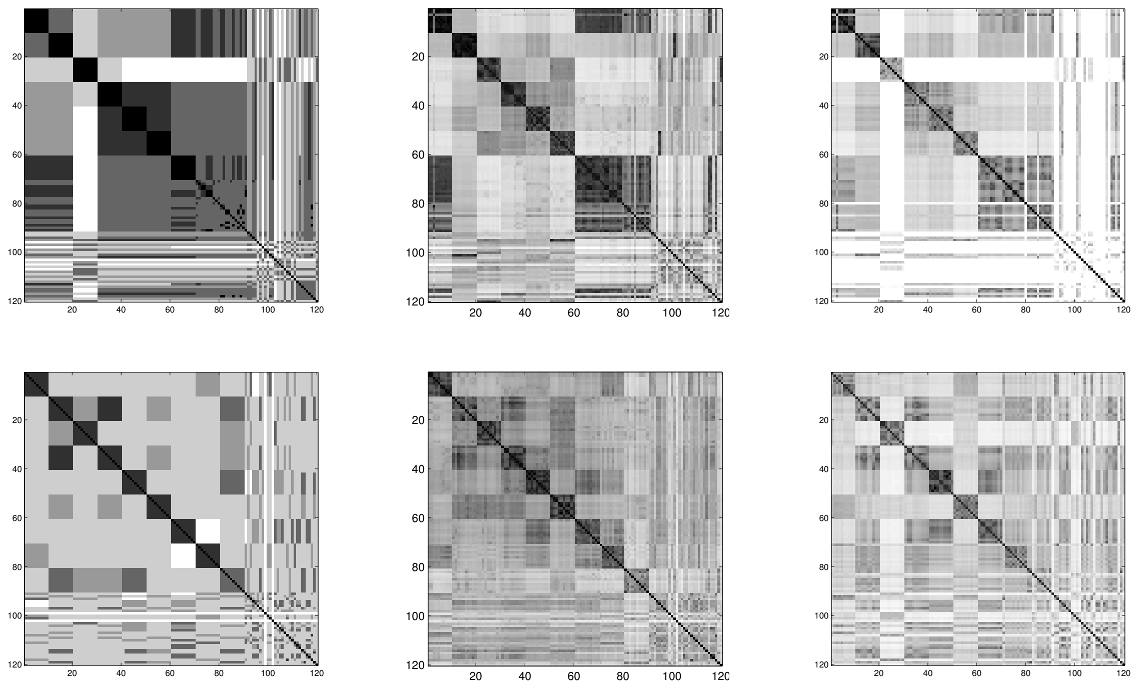


Figure 6.5: Score matrices of ground truth (Column 1), LRI$^+$ (Column 2) and WPI (Column 3) on inkjet (Row 1) and b&w (Row 2) datasets. Darker indicates more similar.

Since the dataset images are high-resolution (1536×2080), computational complexity can be large. However, as illustrated in Table 6.2, LRI-based classification works well when applied to down-sampled images. Hence, in this section, all results of LRI-based classification are computed from images down-sampled by five, both horizontally and vertically, which reduces complexity by a factor of 25. The results for other methods are not based on these down-sampled images, but are instead derived from the similarity matrices reported by the originators of the methods.

For each dataset, Figure 6.5 shows the resulting pairwise similarity matrices for LRI$^+$ and for the previous method with best overall performance, namely, WPI [100, 103]. Also shown is a manually-labelled similarity matrix from [100], serving as a kind of ground truth. As originally suggested in [100], such plots indicate roughly the similarities within and between classes, as well as a rough sense of the performance of a metric. As can be seen, in the inkjet database, classes 1, 7, 8 and 9 in inkjet datasets are very similar as

Table 6.2: Accuracy of LRI-based classification of the inkjet dataset, with and without down-sampling.

Down-sample factor	P@1	MRR	MAP
1 (original images)	88.9%	91.0%	78.8%
5	85.6%	89.6%	77.6%

Table 6.3: Classification results on inkjet and b&w datasets.

	inkjet dataset			b&w dataset		
Metric	P@1	MRR	MAP	P@1	MRR	MAP
LRI	85.6%	89.6%	77.6%	96.7%	97.1%	89.5%
LRI$^+$	**90.0%**	**92.9%**	**79.8%**	**98.9%**	**99.1%**	**91.5%**
Lyon [100, 102]	87.8%	90.4%	77.6%	62.2%	76.9%	65.0%
Tilburg [100]	82.2%	87.6%	76.9%	47.8%	62.6%	42.6%
WPI [100, 103]	87.8%	90.8%	77.7%	85.6%	89.9%	73.1%
HOG+VLAT [104]	-	-	69.7%	-	-	79.3%
POEM+VLAT [104]	-	-	73.6%	-	-	77.0%

one would expect from metadeta for these classes. To the authors of the present paper, it appears that the LRI$^+$ similarity matrix is closer to the ground truth than the matrix for WPI.

Quantitative comparisons of metrics can be based on the accuracy that results in specific classification experiments that use the metrics, as described in the next subsections.

6.3.1 Experiment 1

We first consider the experiment setup in [104]. For each dataset, the first 90 images are grouped into 9 classes, with 10 images in each. (Class i consists of images numbered from $10i - 9$ to $10i$.) Each of 90 will serve as a query into a database consisting of the 119 other images. Thus, the last 30 images in each dataset serve as a kind of noise, although in some

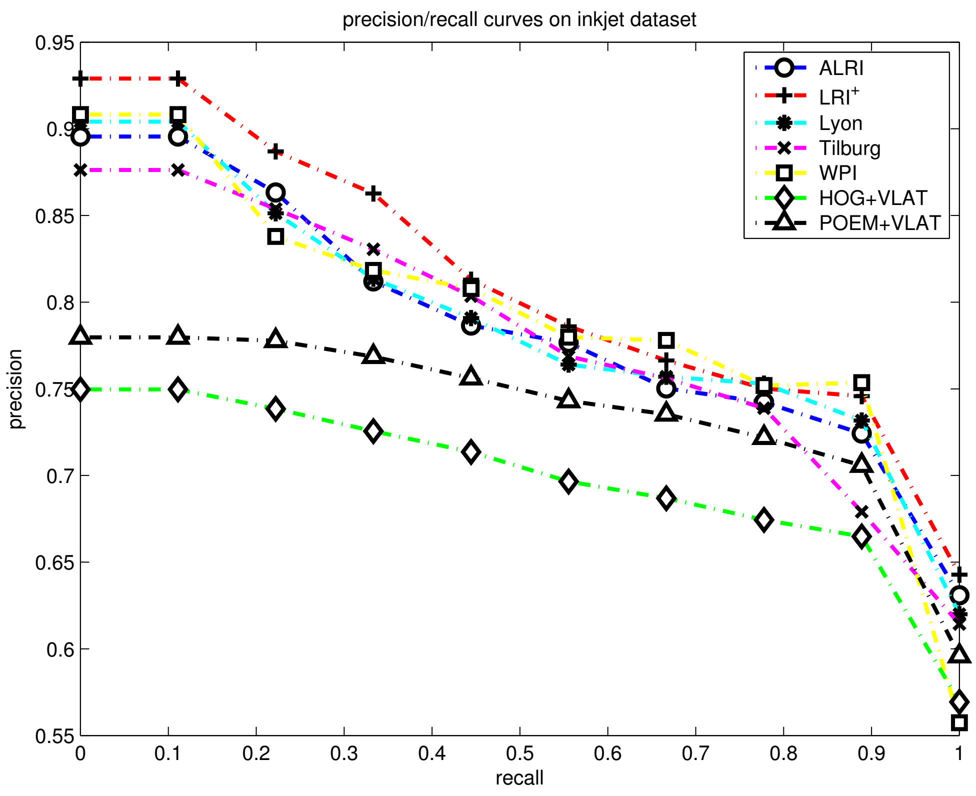


Figure 6.6: Precision/Recall plots on the inkjet dataset.

cases, they have the same manufacturer and style as a query.

Given a metric to be tested, for each query image, the remaining 119 images in the same dataset are ordered according to their similarity to the query, and accuracy is assessed in terms of Precision @ 1 (P@1), Mean Reciprocal Rank (MRR) [96], and Mean Average Precision (MAP) [97]. (For example, P@1 is the accuracy percentage of a nearest neighbor classifier.) The results are shown in Table 6.3. On both datasets, LRI$^+$ is the metric that is best able to distinguish these 10 classes. On the inkjet dataset, Lyon [100, 102] and WPI [100, 103] have the next best performance, followed by LRI. On the b&w dataset, LRI is nearly as good as LRI$^+$ and significantly better than any of the other metrics, some of which have great difficulty. To better visualize the results, Figure 6.6 and 6.7 present the precision and recall plots of all metrics being evaluated on both the inkjet and b&w datasets. As can be seen, the proposed LRI and LRI$^+$ outperform existing metrics, especially in the

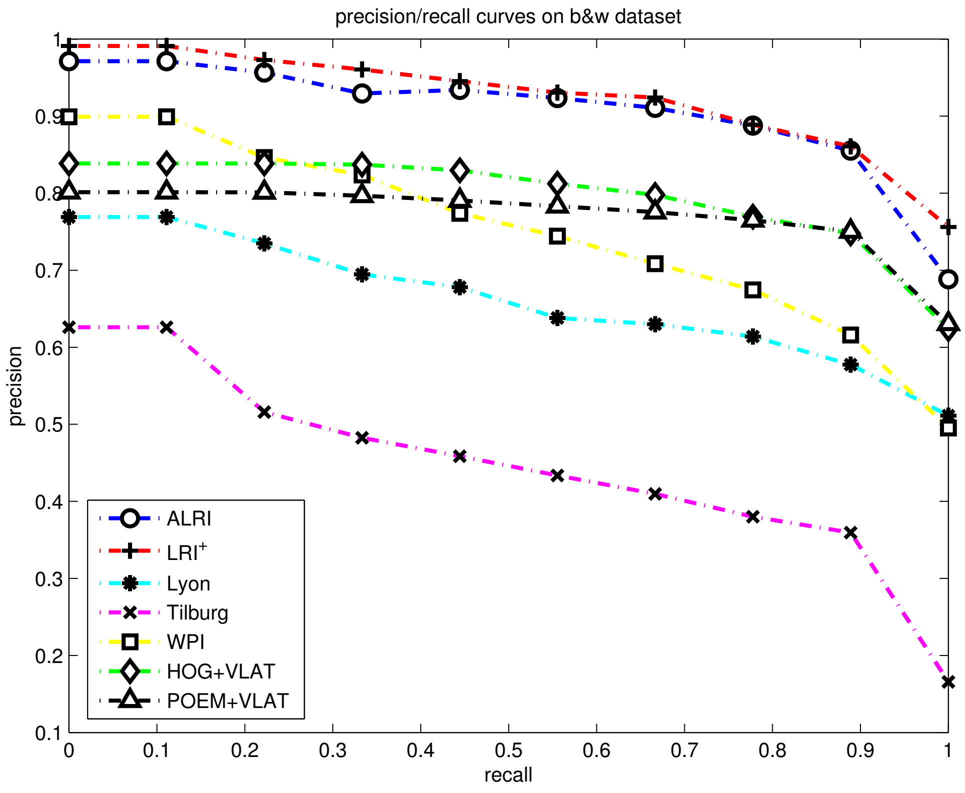


Figure 6.7: Precision/Recall plots on the b&w dataset.

b&w dataset.

6.3.2 Experiment 2

From the description of the datasets one expects that classes 1,2,3 should have very small intra-class distances, since each class consists of images from one piece of paper. The intra-class distance in classes 4,5,6 should be not quite as small, since the 10 images in each class come from different papers in the same package. Finally, classes 7,8,9 should have the largest intra-class distances, since images in each class come from different packages manufactured in different years. With this in mind, we did two additional experiments on the b&w dataset, one with only classes 1,2,3 and the other with only classes 4,5,6. The first experiment tests a metric's ability to distinguish one type of paper from another, when the dictionary of representative images contains samples in the same class that are very close

Table 6.4: Classification results on subsets of b&w dataset.

	classes 1,2,3 only			classes 4,5,6 only		
Metric	P@1	MRR	MAP	P@1	MRR	MAP
LRI	100%	100%	100%	100%	100%	98.9%
LRI$^+$	100%	100%	99.9%	100%	100%	99.8%
Lyon [100, 102]	100%	100%	100%	93.3%	95.4%	90.2%
Tilburg [100]	83.3%	90.0%	79.4%	46.7%	63.4%	44.2%
WPI [100, 103]	100%	100%	99.8%	100%	100%	99.4%

Table 6.5: Reflectance classes for b&w dataset.

Reflectance	Matte	Lustre	Chamois	Glossy	Half Matt	Semi Matt
# images	10	27	10	23	10	10

to the query. The second experiment is a bit more challenging in that it tests a metric's ability to distinguish different types of paper when the dictionary of representative papers does not contain samples quite as close to the query. (The closest samples come from the same package, but not the same paper.) Results in Table 6.4 shows that most metrics do very well in both experiments, but the second can give difficulty to some methods.

6.3.3 Experiment 3

In the appendix of [100], the detailed structure of b&w dataset is provided, with *reflectance* as an attribute. In this experiment, we classify papers according to their reflectance and test the various metrics on their ability to correctly classify. The first 90 images in b&w dataset are classified into six groups as shown in Table 6.5, while the remaining 30 images again serve as noise. This experiment setup is very challenging due to the large intra-class distances. The results, shown in Table 6.6, indicate that LRI and LRI$^+$ both perform very well, with WPI having the next best performance. This experiment suggests that the

Table 6.6: Classification results based on the "reflectance" attribute on the b&w dataset.

Metric	P@1	MRR	MAP
LRI	**95.6%**	97.1%	82.6%
LRI$^+$	**95.6%**	**97.4%**	**83.1%**
Lyon [100, 102]	62.2%	76.9%	55.7%
Tilburg [100]	48.9%	65.2%	41.0%
WPI [100, 103]	86.7%	89.7%	77.2%

proposed metrics are able to distinguish papers based on their reflectance.

CHAPTER 7

Application to Perceptual Image Compression

Another application of texture similarity metrics is perceptual image compression, which aims at outperforming existing compression algorithms in tradeoff between coding rate and reconstruction quality by considering perceptual similarity instead of pixel accuracy. In this chapter, a new perceptual image compression algorithm, called Matched Texture Coding (MTC), is proposed. MTC makes use of structural texture similarity metrics, *e.g.*, STSIM2 and LRI$^+$, to reduce coding rate while maintaining acceptable reconstruction quality.

The work described in this chapter has been published in ICIP 2012 [18] and ICASSP 2013 [34]. And a journal paper is under preparation to submit to IEEE Transaction on Image Processing.

During the past few decades, image compression techniques have seen impressive progress. Although existing approaches can dramatically reduce the coding rate compared to raw image data, there is still a long way to go before we can efficiently store and represent image data as our brains do. One possible way to go one step further is to represent texture more efficiently. Texture contains plenty of high frequency components, which makes it hard to compress. But meanwhile texture regions can appear visually indistinguishable even with substantial pixel-by-pixel differences, *e.g.*, Figure 1.3. The combination of low sensitivity of the human visual system (HVS) to texture and the high cost of its encoding makes texture an ideal breaking point to improve the tradeoff between coding efficiency and reconstruction quality. Since directly coding texture region is expensive, exploiting texture redundancy within an image and trying to encode similar texture patterns as few times as possible should be a better way. Such idea suggests that we can use texture similarity metrics to help quantify texture similarity and efficiently represent texture regions. The same idea can be extended to piecewise smooth regions and transition regions with sharp edges, as long as texture similarity metrics work well in such regions.

In perceptual image compression, the ultimate goal is to achieve *perceptually lossless* compression, which means when viewed side by side, human observers cannot distinguish the reconstructed image from its original. We can achieve this by using a high coding rate. However, since we want to compress images at low rates, our goal is to achieve *structurally lossless* compression. The reconstructed image is said to be structurally lossless relative to its original if it is difficult to distinguish the two images when viewed separately. However, when viewed side by side, visible differences are allowed. When this occurs, we say the reconstructed image and its original are *structurally similar* to each other. Hence structurally lossless compression allows pixel value differences as long as such differences do not affect human perception in this way. As an example, the two images in Fig. 1.3 are said to be structurally similar to each other. The MTC algorithm aims to achieve structurally lossless compression with low coding rate.

7.1 Matched Texture Coding (MTC)

As mentioned earlier, MTC is an image compression algorithm that relies on texture similarity metrics. MTC works in block fashion and tries to exploit texture redundancy among blocks in order to reduce coding rate, while maintaining acceptable reconstruction quality. In particular, the method tries to encode blocks of texture by pointing to already encoded blocks that are sufficiently similar. This idea of encoding by pointing is not new. However, MTC is the first image compression algorithm that combines this idea with effective texture similarity metrics. As one can see later, making this combination work is not trivial.

Since MTC is based on texture prediction from the already coded region, at the beginning of coding, we need to code the image with another coder, which we refer to as the baseline coder. The baseline coder can be any sort of commonly used coder, *e.g.*, JPEG and JPEG-2000. Our goal is to compress more than the baseline coder with the same reconstruction quality, or code at the same rate as the baseline coder but with higher quality. Currently, JPEG is the baseline coder. The reason to choose JPEG over JPEG-2000 will be discussed in Section 7.1.3.

In MTC, one divides the whole image into 32×32 blocks and then codes the blocks in raster order. In MTC, define the location of each block to be the location of the pixel in the upper left corner of the block. For each block the coding strategy is:

i. Tentatively code the 32×32 block with the baseline coder, which is JPEG. If JPEG gives sufficiently low coding rate, *e.g.*, 0.085 bits per pixel (bpp), then texture matching algorithm in Step 2 is not needed. Usually, very low JPEG rates indicate smooth blocks. Under this circumstance, output the JPEG encoded bits and go to Step 4. Otherwise, go to Step 2.

ii. Search the already coded region of the image and try to find a structurally similar 32×32 candidate block to the target block that is currently being coded. If successfully find a candidate block that is similar enough to the target, then encode the position of the chosen candidate block (will be described later) and go to Step 4. Otherwise, go to Step 3.

iii. Subdivide the target block into four 16×16 blocks, and try the texture matching algorithm on each of them in raster order. For each 16×16 block, if one successfully finds a structurally similar candidate block, encode the position of the chosen candidate block. Otherwise, JPEG will be used to encode this 16×16 block and the output is the JPEG encoded bits. Go to Step 4 after all four 16×16 blocks are encoded.

iv. Apply an image blending algorithm to avoid blocking artifacts. (Blocking artifacts denote the noticeable discontinuity around the border region of a block if this block is replaced by a candidate block. Image blending is a technique that modifies the pixel values in the vicinity of the block boundary to mitigate blocking artifacts.)

In Steps 2 and 3 above, we may need to encode the position of the chosen candidate block. Suppose the image has size $N \times M$ and the target block locates in row i. Since any candidate block must be located in the first i rows, we can encode the position of a candidate block using $\lceil \log_2(i) \rceil + \lceil \log_2(M) \rceil$ bits. In addition, for each 32×32 block, we need to encode the encoding *mode*, indicating how this block is coded. Basically we use one bit for each of the four 16×16 sub-blocks within a 32×32 block. For each 16×16 sub-block, we use "0" to indicate that this sub-block is coded using texture matching and "1" to indicate that this sub-block is coded using JPEG. The only confusion occurs when "0000" is generated, meaning that all the four sub-blocks are coded using texture matching. In this case, we need to further distinguish encoding as one 32×32 texture matching or four 16×16 texture matchings by adding another bit, with "0" meaning one 32×32 texture matching and "1" meaning four 16×16 texture matchings. Hence, either four or five bits are needed to encode the encoding mode of a 32×32 block, which translates to 0.0039 bpp or 0.0049 bpp.

A flow chart of the encoding strategy described above is shown in Fig. 7.1. With this coding strategy as the basic idea, a number of key issues and refinements are also needed, including candidate search strategy, image blending, JPEG quality parameter selection and a special strategy for smooth blocks. Without conquering these challenges, either the algorithm would be too complex and computationally unacceptable, or we could not guarantee good overall quality of the compressed images, even with a perfect texture similarity metric. In addition, we found that the compression algorithm can be improved by first doing a simple coding of the low frequency components of the image and then applying the MTC algorithm to the high frequency residual. In the following, we will consider these issues one by one.

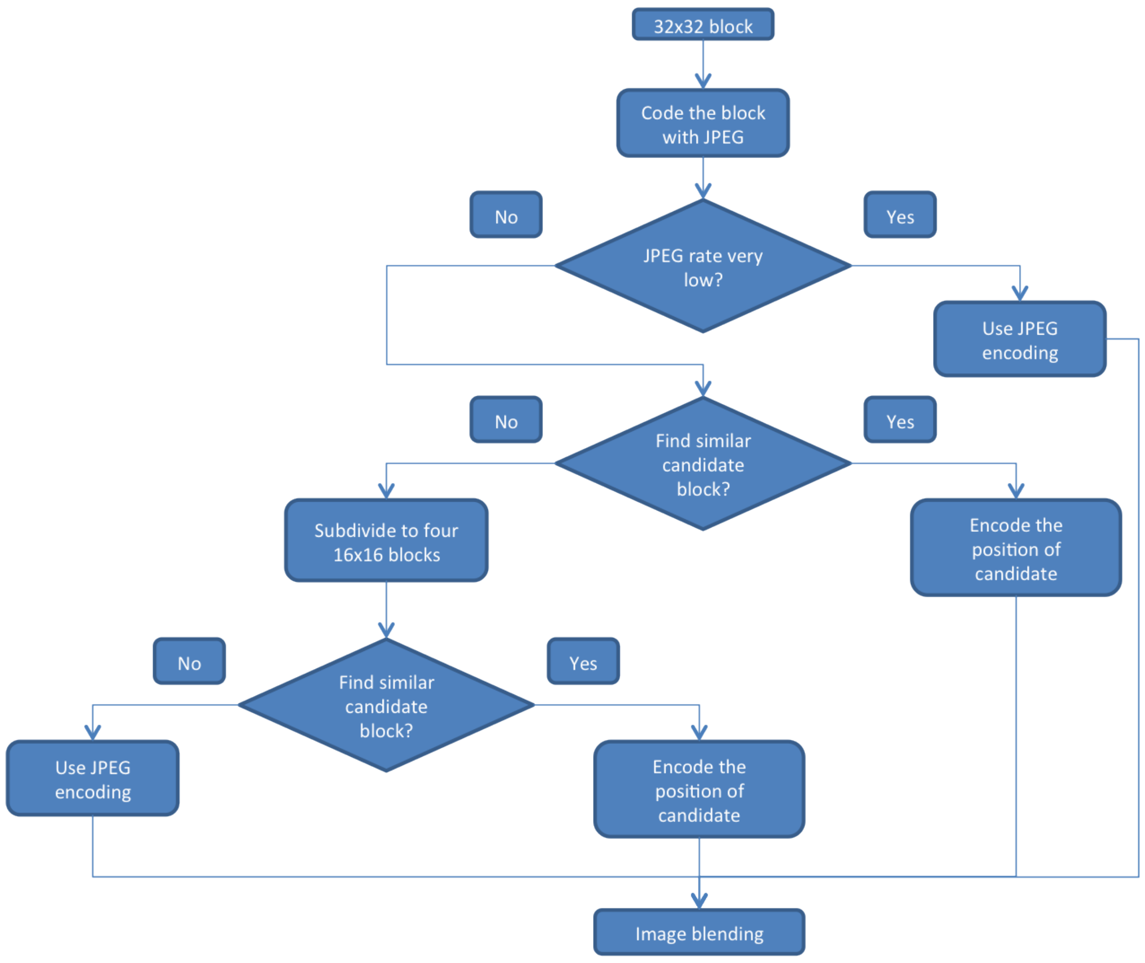

32x32 block
Code the block with JPEG
No
Yes
JPEG rate very low?
Use JPEG encoding
No
Yes
Find similar candidate block?
Subdivide to four 16x16 blocks
Encode the position of candidate
No
Yes
Find similar candidate block?
Use JPEG encoding
Encode the position of candidate
Image blending

Figure 7.1: Flow chart of the MTC coding strategy.

7.1.1 Candidate Search Strategy

As described above, MTC aims at finding previously coded blocks (candidate blocks) containing texture similar to the current coding block (target block). To quantify similarity, a texture similarity metric is applied, *e.g.*, STSIM2 or LRI$^+$. Due to computational constraints, we cannot afford to consider too many candidates for each target. For example, for the last block (in the right lower corner) in a 1024×1024 image, there are almost 1 million candidate blocks to consider. If we apply STSIM2 to every candidate block, it will take more than one week for just this one target block, which is obviously unacceptable. Hence a progressive search strategy that can pre-screen candidates is needed. In MTC, the candidate search strategy we propose for texture blocks is a 3-stage search strategy. In the first two stages, two simple metrics, namely the *variance metric* and the *inside side-matching metric*, are applied to rule out most candidates that are unlikely to be the best. And then in stage three, the structural texture similarity metric is applied to the few remaining candidates to make the final decision. With the 3-stage search strategy, the texture similarity metric is applied at most 8 to 16 times for each target block. In this way, we can avoid meaningless computation and speed up the coding algorithm dramatically. This strategy is crucial to MTC algorithm, since without it, MTC will be extremely complex and impossible to implement in practice. Moreover, as can be seen later, this 3-stage hierarchical search strategy can also improve search results and the reconstruction quality by rejecting some low-quality candidates that would be accepted by texture similarity metrics. Next we discuss the details of the 3-stage search strategy.

7.1.1.1 Search Stage 1

In this stage, we only consider candidates in the lattice Z8 (or Z4), which means we only consider candidates at positions $(8\times i+1, 8\times j+1)$ (or positions $(4\times i+1, 4\times j+1)$), where i and j are nonnegative integers. Then we apply the *variance metric* to each candidate block with the target block. The *variance metric* is a very simple metric that can help rule out many dissimilar candidates without using a complete texture similarity metric. Based on the *variance metric*, the best N_1 candidates are chosen.

The *variance metric* is described as follows. Given two 32×32 blocks, X and Y, subdivide each into 16 8×8 non-overlapped sub-blocks and calculate the local variances $\text{var}_X(i,j)$ and $\text{var}_Y(i,j)$, $i,j = 1,2,3,4$, for each sub-block. The *variance metric* is defined to be the average of the absolute differences between these local variances:

$$\text{Var}(X,Y) = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}\left|\text{var}_X(i,j) - \text{var}_Y(i,j)\right|.$$

When applied to 16×16 blocks, a similar subdivision is used, and in this case there are only 4 local variances.

Small Var(X, Y) implies the candidate and the target have roughly the same local variances. On the contrary, large Var(X, Y) means on average, local variances of the candidate and the target are very different, which is strong evidence against structural similarity. Figure 7.2 shows the relationship between the *variance metric* and STSIM2. In the experiment to obtain the figure, we randomly chose a 32×32 target block from the sweater area in the "woman" image (shown in Fig. 7.14), and computed both the *variance metric* and STSIM2 for every candidate in the lattice Z8. The target block is shown in the left image of Fig. 7.3. From Fig. 7.2, we have the following observations:

i. Where the *variance metric* has a small score, high STSIM2 score can be expected. This indicates the validity of Search Stage 1.

ii. Where STSIM2 has the smallest score (around 0.65), the *variance metric* approaches the average local variance of the target block (in this case around 1100). This happens when the candidate block is very smooth with near-to-zero local variances. In this case, the *variance metric* just gives the average local variance of the target block.

iii. Where both STSIM2 and the *variance metric* scores are large, candidate blocks have much higher local variances than the target block. Figure 7.3 shows such an example. The left image is the target block and the right one has both high STSIM2 and *variance metric* scores. In this case, the candidate has similar texture to the target for the most part, however, the upper right corner of the candidate is very different from the target. The high STSIM2 score shows the incapability of STSIM2 to distinguish these two image patches. Surely, we do not want to accept such a candidate and the *variance metric* can help rule it out. This example shows that the 3-stage search strategy actually improves the search strategy (in addition to speed up the algorithm).

Since in Search Stage 1, only candidates in the lattice Z8 (or Z4) are considered, we can pre-compute all 8×8 local variances needed and store them in a table. In this way, the

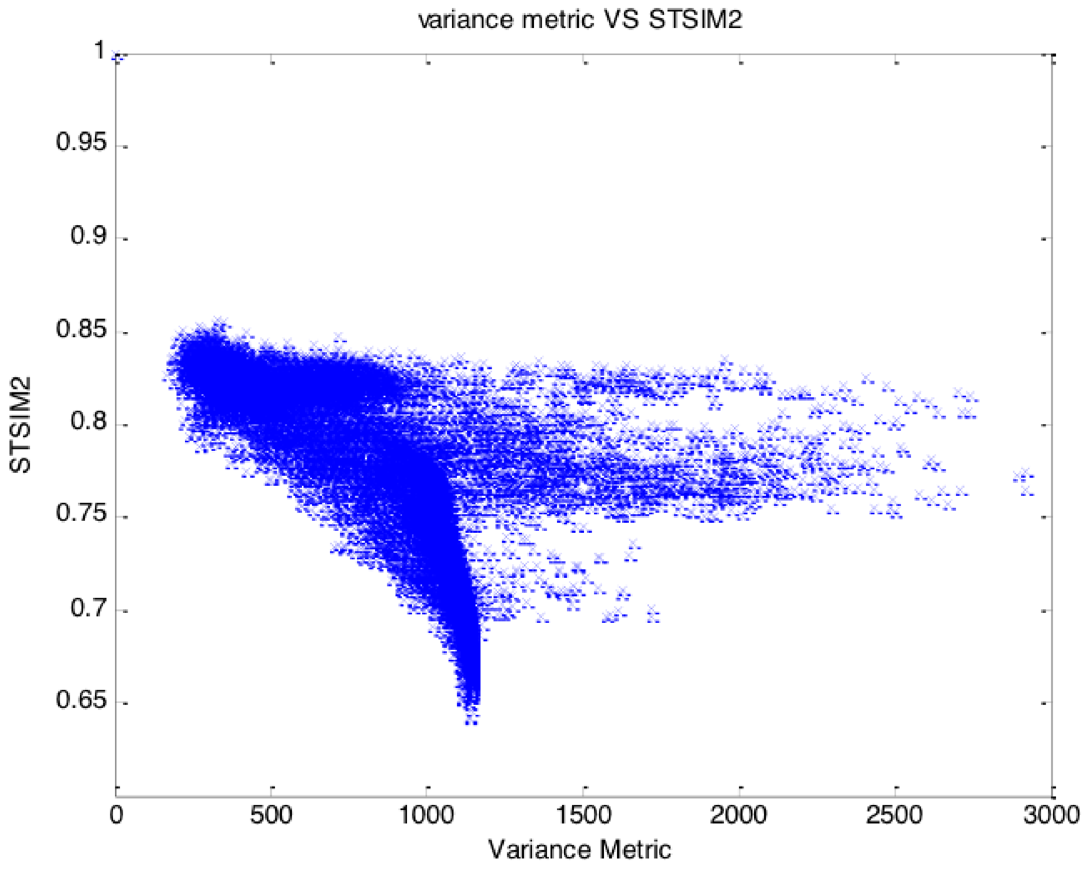


Figure 7.2: the *variance metric* vs. STSIM2.

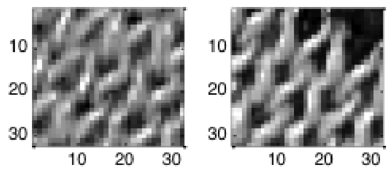


Figure 7.3: Left image: one 32×32 target block. Right image: one candidate block having high scores with the target block using both STSIM2 and the *variance metric*. Note that a high score in STSIM2 means similar and in the *variance metric* means dissimilar.

speed of Search Stage 1 becomes very fast.

7.1.1.2 Search Stage 2

In this stage, for each of the N_1 candidates chosen in Stage 1, we search its 8×8 neighborhood for the best candidate according to an *inside side-match metric*. So we will still have N_1 candidates left, but maybe at different locations than the original N_1 candidates. Next, sort the new N_1 candidates in order of these *inside side-matching metric* scores and discard those whose scores are worse than a threshold T_1. If there are any left, go to Search Stage 3. Otherwise, we fail to find a candidate that is structurally similar enough as the target block.

The *inside side-matching metric* is described as follows. For two 32×32 blocks (one is a candidate block and the other is the target block), compare the inner border region of each 32×32 candidate with that of the target block using Mean-Squared Error (MSE). The inner border of a 32×32 block is illustrated in Fig. 7.4 (similar for 16x16 blocks). Generally, for an $N \times N$ block, the width of the border region is $\frac{N}{4}$ pixels.

The motivation for the *inside side-matching metric* is to favor those candidate blocks whose borders are more similar to the target, which will facilitate image blending. We do not want to accept candidates having bad side-matching scores even if they are structurally similar to the target block, since a substitution with such a candidate will result in noticeable discontinuities in the border region that are unlikely to be fixed by an image blending algorithm. This issue will be discussed in detail in Section 7.1.2.

In summary, we can think of the relationship between Stage 1 and Stage 2 as follows. In Stage 1, the *variance metric* leads us to candidates that are roughly similar to the target block. And then in Stage 2, we apply the *inside side-matching metric* to further refine the candidates (match the border region) by searching in the geographic neighborhood of each candidate found in Stage 1.

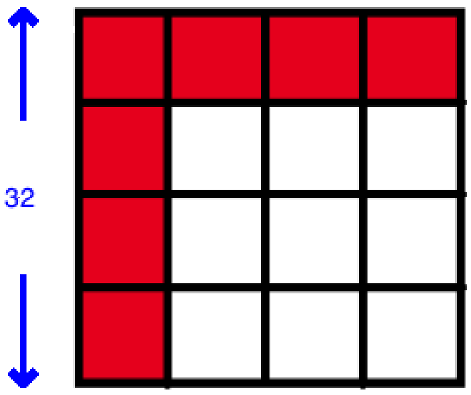


Figure 7.4: Red region is the inner border of a 32×32 block.

7.1.1.3 Search Stage 3

Suppose after Stage 2, there are N_2 ($\leq N_1$) candidates left. Apply a texture similarity metric, *e.g.*, STSIM2 or LRI$^+$, to these N_2 candidates and the target, one-by-one, until we find the first one with metric score better than a threshold T_2. If there exists such a candidate, then we have successfully found a candidate that is sufficiently structurally similar to the target, and we encode its position. Otherwise, we fail.

The texture similarity metrics applied to MTC were STSIM2 and the newly proposed LRI$^+$. Results in Section 7.3 show that both metrics can provide satisfactory tradeoff between coding rate and reconstruction quality, while the version with LRI$^+$ is significantly faster than the one with STSIM2.

To better fit this image coding application, a slightly modified version of STSIM2 is applied to the MTC algorithm. To compare the structural similarity of two 32×32 blocks with STSIM2, we apply the steerable pyramid subband decomposition [1, 66], with three scales and four orientations, to the 64×64 super-blocks around the 32×32 blocks and only retain the 32×32 subband coefficients in the center. The reason for applying the steerable pyramid subband decomposition to a larger region is to avoid boundary effects. Similarly, for 16×16 blocks, we apply the steerable pyramid subband decomposition to the 32×32 blocks and only retain the 16×16 subband coefficients in the center.

7.1.2 Image Blending

In MTC, we try to use the previously encoded candidate blocks to replace the target blocks, which may cause blocking artifacts at the border region. To avoid blocking artifacts, an image blending algorithm is applied after block replacement. Typically, blending is needed between a replaced block and a block coded with JPEG, and between two replaced blocks. (Blocking artifacts are unlikely to occur between two adjacent blocks both coded with JPEG.) The blending algorithm is applied systematically in MTC and costs no bits. In particular, blending is applied right after a block is encoded at the encoder. The decoder replicates the blending process in the encoder right after a block is decoded. The details of the image blending algorithm are as follows.

The image blending algorithm used in MTC is developed from the image quilting technique described in [105]. There are two kinds of blending used in the MTC algorithm.

i. Blending for MTC-encoded Blocks: (shown in Fig. 7.5):

   In Fig. 7.5, suppose we choose to replace the yellow target block with the green candidate block. Instead of only replacing the target block, we also replace part of the red L-shaped border region to the left and top of the target block. We choose a curve inside the red L-shaped region such that to the right and bottom of the curve will also be replaced by the candidate block. The curve is chosen to ensure that the blocking artifacts are "unnoticeable" after the replacement. To achieve this goal, we use the image quilting algorithm proposed in [105] to determine the curve inside the L-shaped region such that after replacement, the MSE of pixels to the left and right side of this curve is minimized. Finally, we use the green region in Fig. 7.5 to replace the yellow region.

ii. Blending for JPEG-encoded Blocks: (shown in Fig. 7.6):

   When JPEG is chosen as the final coding technique for a block instead of MTC, we have to check whether blending is needed.

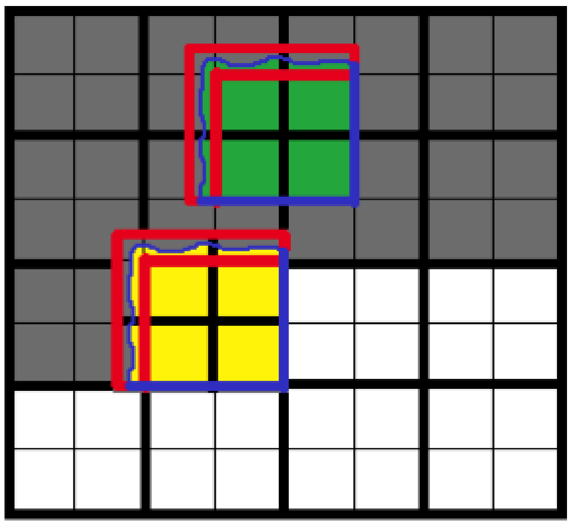


Figure 7.5: Blending for MTC-encoded blocks.

If to the left and top of a JPEG-encoded block are also JPEG-encoded blocks, no blending is needed, since it is unlikely to have blocking artifacts between two JPEG-encoded blocks.

However, if to the left or top of a JPEG-encoded block is an MTC-encoded block, then there may be blocking artifacts on the boundary. In this case we need to do blending. As shown in Fig. 7.6, the yellow part is coded with MTC, and the red part is coded with JPEG. We find a curve inside the red block such that to the left and top of the curve (blue region) we use texture matching. The curve is chosen using the same method as for blending for MTC-encoded blocks. The blue region comes from the surroundings of the candidate blocks used to substitute the yellow blocks.

7.1.3 JPEG Quality Parameter Selection

JPEG is currently used as the baseline coder in MTC. One may argue that JPEG-2000 is a better choice. However, we think JPEG is more suitable to this application due to the following reasons:

i. Although JPEG-2000 has a better performance than JPEG when applied to natural

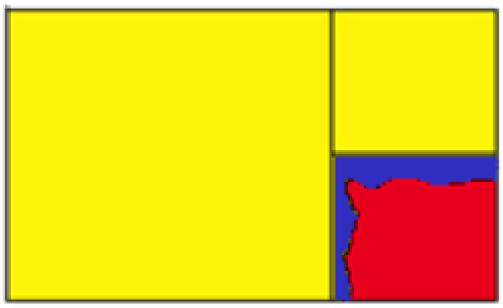


Figure 7.6: Blending for JPEG-encoded blocks.

images, its ability to compress texture is generally worse than JPEG. In addition, when the coding rate is low, *e.g.*, 0.2 or 0.3 bpp, JPEG usually gives better reconstruction quality than JPEG-2000.

ii. JPEG uses 8×8 DCT transform, which is easy to be embedded into the MTC coding system. However, JPEG-2000 uses a full frame DWT transform, which is very hard to handle when some target blocks are replaced by candidate blocks.

iii. Consider the compatibility to video coders, JPEG is a better baseline coder than JPEG-2000. The state-of-the-art video coders do not use the DWT transform. For example, HEVC [106], as MTC, uses the DCT transform. From this perspective, by using JPEG as the baseline coder, the MTC algorithm can be easily plugged into HEVC in the future.

As we know, JPEG can encode images with different qualities (or with different rates) by choosing a quality parameter, which is called JPEG quality factor. In this subsection we will discuss how to choose this quality factor in MTC.

JPEG encodes 8×8 blocks of an image using the Discrete Cosine Transform (DCT) and then uniformly quantizing each of the 64 DCT coefficients. We get to choose the quantizer step sizes, one for each coefficient. Normally, this is done by scaling the *de facto* standard set of thresholds, as shown in Table 7.1. The question is what scaling factor (quality factor)

Table 7.1: JPEG *de facto* quantization table.

Quantization step size	1	2	3	4	5	6	7	8
1	16	11	10	16	24	40	51	61
2	12	12	14	19	26	58	60	55
3	14	13	16	24	40	57	69	56
4	14	17	22	29	51	87	80	62
5	18	22	37	56	68	109	103	77
6	24	35	55	64	81	104	113	92
7	49	64	78	87	103	121	120	101
8	72	92	95	98	112	100	103	99

to use. (The actual JPEG quantization step sizes are numbers in Table 7.1 multiplied by the quality factor.) If the quality factor is large, coding rate will be small, and vice versa. Moreover, in a smooth 8×8 block, most of the quantized DCT coefficients are zero and JPEG can efficiently encode this block with very few bits. On the other hand, in a textured 8×8 block, JPEG tends to generate more bits.

The choice of JPEG quality factor is a tradeoff. If the quality factor is too large, meaning low quality JPEG is applied, we suffer from two problems. The first is that we can observe contour artifacts in smooth regions. This can be seen in the right image in Fig. 7.14. There is clear contour artifacts in the neck and finger areas of the woman. The second problem occurs when encoding the rest of the image. The JPEG coded region will be used for texture matching when encoding the rest of the image. So we can view the JPEG encoded region as part of a "codebook". Later when we try texture matching, we will search the current "codebook" to find the most similar candidate block. Low JPEG quality will lower the quality of the "codebook", which may influence the overall quality of the compressed image. On the contrary, if the quality factor is too small, meaning high quality JPEG is applied, we pay too much and the coding rate is high.

As described previously, under high quality factors, a JPEG-encoded image tends to have contour artifacts in smooth regions. However, such artifacts are not very obvious in texture regions. This means texture regions can tolerate worse JPEG quality than smooth regions without being noticed by human observers. Hence, when applying JPEG, we can use low quality factors in smooth regions and high quality factors in texture regions. By experimentation we have chosen the quality factor to be 2 in smooth regions and 4.5 in texture regions. In this way, we can avoid contour artifacts in smooth regions and save as many bits as possible at the same time. Variance is a good estimator of the smoothness of a block. If the variance of a block is above some threshold T_0, typically $T_0 = 50$ or 100, we apply the high quality factor. Otherwise we apply the low quality factor.

7.1.4 Special Strategy for Smooth Blocks

Up to now, we have focused only on texture matching, for texture blocks. However, in natural images, there are also many smooth blocks. For such blocks, texture similarity metrics may fail to provide robust similarity scores. So in our algorithm, we treat smooth blocks differently from texture blocks. Again, we use variance to indicate whether a block is smooth or textured. The overall encoding strategy for a smooth block is the same as what we described in Section 7.1. The difference is the strategy to search for similar candidate blocks. Instead of using the 3-stage search strategy, we consider all candidates one by one. For each candidate, we first calculate the block variance. If the variance of the candidate is above some threshold, meaning this block is not smooth and unlikely to be similar to a smooth target block, then we discard this candidate and go to the next one. Otherwise, we use a simple "max-error type" test to compare similarity of this candidate with the target block. (The "max-error type" test will be introduced in the next paragraph.) If we successfully find a candidate that passes the test, we accept this candidate without continuing to consider the rest of the candidates. So "max-error type" test is our only criterion to decide whether a smooth candidate is accepted or not.

The "max-error type" test used in MTC is motivated by an image patch similarity test developed for use in Quadtree Predictive Coding (QPC) [107, 108]. Specifically, we propose the following test for use in smooth regions: $M(X, Y)$ is the smallest value m such that every pixel in Y either differs from the corresponding pixel in X by m or less, or has 3 or 4 of its horizontally or vertically adjacent neighbors that have error m or less. Thus if $M(X, Y) = m$, pixel errors larger than m are either isolated or occur in horizontal or vertical clusters of size 2. A mathematical formula is:

$$M(X, Y) = c \times \max\{\min\{|X_i - Y_i|, E_i(2)\}\},$$

where $E_i(2)$ is the second largest value of $|X_j - Y_j|$ among the four horizontal and vertical neighbors of i, and where c is a constant to be discussed later. We compare $M(X, Y)$ with some threshold T. If $M(X, Y)$ is less than T, then the candidate passes the test. Otherwise, the candidate fails. This test is intended primarily for smooth X, and c should reflect the smoothness. In particular, if X is typically smooth over regions of size $n \times n$, then according to [109], the perceptual sensitivity to isolated errors increases with n, so c should increase. For example, in [107], it was applied to $n \times n$ patches with n ranging from 2 to 16, and it was found that c should increase by approximately 30% when n doubles. In summary, when X and Y are both smooth, or mostly smooth, this metric monotonically reflects their similarity. If X is smooth but Y is not, this metric will readily show their considerable dissimilarity. The details are shown in Fig. 7.13.

7.1.5 Low Frequency Coding

For each target block, we want to maximize the possibility of finding a good candidate block. The first approach to achieve this goal is to carefully design the search strategy and texture similarity metrics. Besides that, low frequency coding can also help. Suppose two blocks have similar texture patterns but different background luminance (low frequency component). For example, a target block consists of a pile of dark beans and a candidate block of light ones. On the one hand, STSIM2 would reject this candidate because of the difference in background luminance. However, if one separately encodes the background luminance of the target and attempts only to compare the residual high frequency component of the target to the high frequency component of the candidate (obtained by subtracting its low frequency component), one might find a good match, in which case one could encode by pointing to this candidate with the idea that the decoder will reproduce the target as the sum of the encoded low frequency component of the target plus the high frequency component of the candidate. On the other hand, if LRI$^+$ is used, due to its insensitivity to the background luminance, it would accept this candidate. As a result, the reconstructed target block would have noticeable discontinuous background luminance with its neighborhoods. However, if only the high frequency component of the candidate is used at the decoder along with an encoded low frequency component of the target block, this discontinuity would disappear. In both cases, a low frequency coding method can improve the MTC algorithm. Hence, the MTC encoder now works as follows. First, it estimates the low frequency component of the entire image. Then it applies the MTC algorithm described earlier to the high frequency residual. Bits produced in both steps are transmitted to the decoder.

A good low frequency coding method should remove texture patterns in texture regions while preserving edges in transition regions. The former property ensures that texture patterns in high frequency component are the same as in original images, which means we are not creating new kind of textures. The latter property avoids the blurring of sharp edges.

There are many ways to encode the low frequency component of an image. We need to find a way that satisfies the conditions described in the last paragraph and enables low rate coding. We considered several ways. One way is DCT-based low frequency coding. This method works in block fashion. First, apply the DCT transform to each 8×8 block. Then for those blocks that end up being MTC coded, retain and quantize only the 3 (or 4) coefficients representing the low frequency component. This method enables low coding rate for the low frequency component (around 0.05 bits per pixel (bpp)). However, we found that with this method, textures are not fully removed from textured regions of the image. Another possibility is to use the Total Variation as described in [110], which produces an

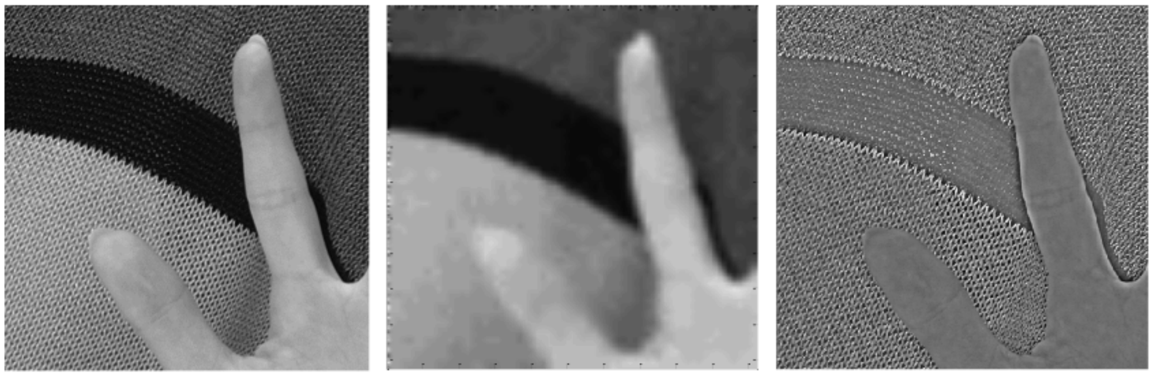


Figure 7.7: Left image: Original Image. Middle image: Low frequency component (coded at 0.0374 bpp). Right image: High frequency component.

excellent estimate of the low frequency component. However, we could not find an efficient way to encode it.

Finally, we found a simple and fast low frequency coding algorithm which can encode the low frequency part of an image at very low rate, while satisfying the conditions discussed above. The idea is developed from the Quadtree Predictive Image Coder as in [107] and will be introduced later. Figure 7.7 shows an example of the result of this algorithm, produced when coding at rate 0.0374 bpp. The image in the middle is the encoded low frequency component of the left image. And the image on the right is the high frequency component produced by subtracting the low frequency component from the original image. We can see that at such a low coding rate, the low frequency coding algorithm performs very well.

The proposed low frequency coding algorithm works as follows. For an image, it estimates the low frequency component of each 16×16 block in raster order. For each 16×16 block, the estimation of the low frequency component is a hierarchical linear interpolation of the values on the top border of the block, on the left border of the block, and four pixel values located strategically as shown in Fig. 7.8. These four values, which are called "feet", must be encoded, as they are not apriori known to the decoder. The following describes how their values are selected and encoded.

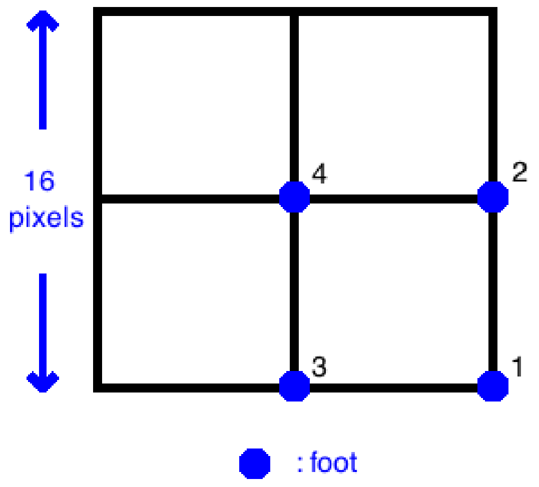


Figure 7.8: Positions of the four feet in a 16×16 block.

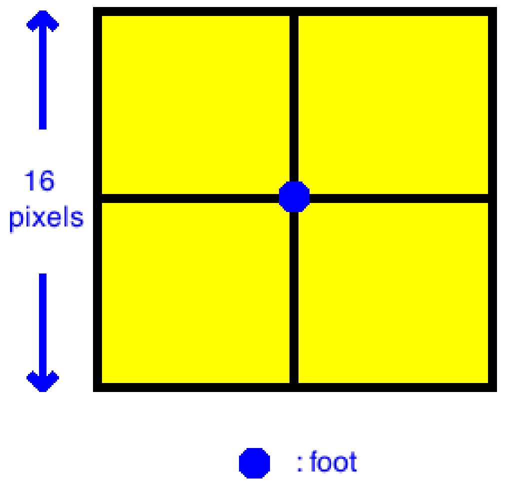


Figure 7.9: Foot takes the mean value of its 16×16 neighborhood (yellow region).

As mentioned previously, the locations of the four "feet" are shown in Fig. 7.8. Since texture has plenty of high frequency components, it is not robust to just take any single pixel value as a foot value. Hence, every foot takes the mean value of its 16×16 neighborhood, as shown in Fig. 7.9. These feet values then need to be encoded. Since feet are means of their respective neighborhoods, the difference between two adjacent feet values should have a smaller dynamic range than feet values themselves. Hence, instead of directly encoding each foot value, we predict the four feet in each 16×16 block from feet in the current and previous blocks. The prediction strategy is shown in Fig. 7.10.

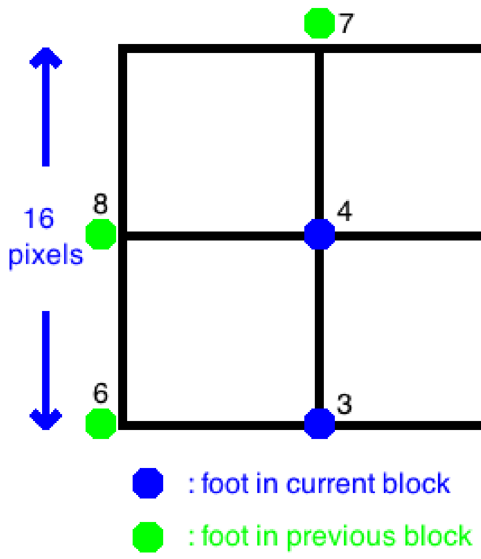


Figure 7.10: Foot prediction strategy: Predict foot 1 as the average of the previous feet 5 and 6; Predict foot 2 as the average of foot 1 and the previous foot 5; Predict foot 3 as the average of foot 1 and the previous foot 6; Predict foot 4 as the average of foot 2, foot 3 and the previous feet 7 and 8.

After obtaining the prediction errors, we quantize them with a uniform scalar quantizer. (A typical quantization step size is 10.) Then we encode the quantized prediction errors using the unary code. The first ten codewords in the codebook for the unary code is shown in Table 7.2.

After encoding the feet of a 16×16 block, a hierarchical linear interpolation is applied to reconstruct the low frequency component of the block. Specifically, we reconstruct its four 8×8 sub-blocks in raster order. The reconstruction of an 8×8 block is based on the pixels on the left and top borders of this block, together with the foot at its lower right corner. The interpolation method to reconstruct each pixel is the same as in [107]. Let us consider an 8×8 block $x[i,j]$, where $i,j = 1,2,\dots,8$. The problem becomes to interpolate $x[i,j]$ using $x[0,j]$, $j = 1,2,\dots,8$, $x[i,0]$, $i = 1,2,\dots,8$, and $x[8,8]$. Note that $x[0,j]$ is the top border of x, $x[i,0]$ is the left border of x, and $x[8,8]$ is the reconstructed foot. We first reconstruct the rightmost column by taking the average of already-known values, starting

Table 7.2: Unary codebook. (Only the first ten codewords are shown.)

	codeword
1	0
2	10
3	110
4	1110
5	11110
6	111110
7	1111110
8	11111110
9	111111110
10	1111111110

from $x[4,8]$:

$$x[4,8] = \frac{x[0,8] + x[8,8]}{2}\ ,\ x[2,8] = \frac{x[0,8] + x[4,8]}{2}\ ,\ x[1,8] = \frac{x[0,8] + x[2,8]}{2}\ \ldots$$

After that, reconstruct the bottom row in a similar way, starting from $x[8,4]$:

$$x[8,4] = \frac{x[8,0] + x[8,8]}{2}\ ,\ x[8,2] = \frac{x[8,0] + x[8,4]}{2}\ ,\ x[8,1] = \frac{x[8,0] + x[8,2]}{2}\ \ldots$$

Now, with all the boundary pixels, we can start to reconstruct the interior of block x, starting from the center pixel $x[4,4]$:

$$\begin{aligned} x[4,4] &= \frac{x[0,4] + x[8,4] + x[4,0] + x[4,8]}{4}\ , \\ x[2,2] &= \frac{x[0,2] + x[4,2] + x[2,0] + x[2,4]}{4}\ , \\ x[1,1] &= \frac{x[0,1] + x[2,1] + x[1,0] + x[1,2]}{4}\ \ldots \end{aligned}$$

Using this hierarchical linear interpolation algorithm, we can reconstruct block x step by step. To reconstruct 8×8 blocks at the left (or top) border of an image, we cannot use pixels to the left (or top) of this block. Instead, the reconstruction is based on pixels in the first column (or the first row) of the current blocks.

7.2 Matched Texture Coding Flow Charts

In summary, the most important parts of the MTC algorithm are:

i. Texture similarity metrics that can ensure frequent success in finding structurally similar candidate blocks for target blocks.

ii. Progressive candidate-search strategy that can guarantee both accuracy and efficiency.

iii. Pre-processing techniques, like low frequency coding of images (to get better compression results) and pre-calculation of certain image statistics (to speed up the algorithm).

iv. Post-processing techniques, like image blending (to avoid blocking artifacts).

Figures 7.11, 7.12 and 7.13 give detailed flow charts for the MTC algorithm. Figure 7.11 shows the overall MTC algorithm. Figure 7.12 shows the 3-stage search strategy for texture blocks and Fig. 7.13 shows the candidate search strategy for smooth blocks.

In all three flow charts, blue blocks are self explanatory; red blocks are explained in detailed flow charts; orange blocks have already been explained in Section 7.1.

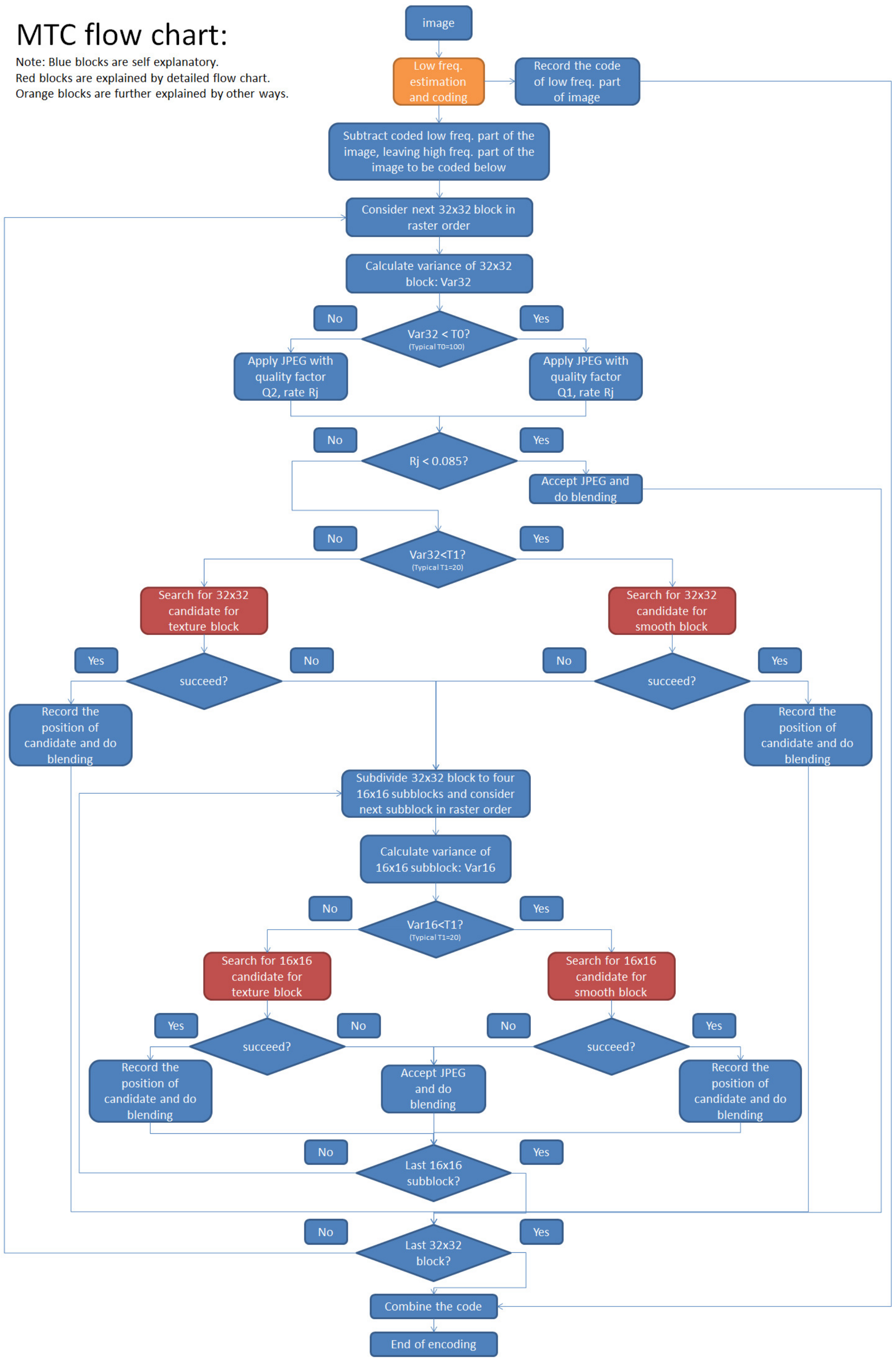

MTC flow chart:
Note: Blue blocks are self explanatory.
Red blocks are explained by detailed flow chart.
Orange blocks are further explained by other ways.
image
Low freq. estimation and coding
Record the code of low freq. part of image
Subtract coded low freq. part of the image, leaving high freq. part of the image to be coded below
Consider next 32x32 block in raster order
Calculate variance of 32x32 block: Var32
Var32 < T0? (Typical T0=100)
No
Yes
Apply JPEG with quality factor Q2, rate Rj
Apply JPEG with quality factor Q1, rate Rj
Rj < 0.085?
Accept JPEG and do blending
Var32<T1? (Typical T1=20)
Search for 32x32 candidate for texture block
Search for 32x32 candidate for smooth block
succeed?
Record the position of candidate and do blending
Subdivide 32x32 block to four 16x16 subblocks and consider next subblock in raster order
Calculate variance of 16x16 subblock: Var16
Var16<T1? (Typical T1=20)
Search for 16x16 candidate for texture block
Search for 16x16 candidate for smooth block
Accept JPEG and do blending
Last 16x16 subblock?
Last 32x32 block?
Combine the code
End of encoding

Figure 7.11: Flow chart of the MTC algorithm.

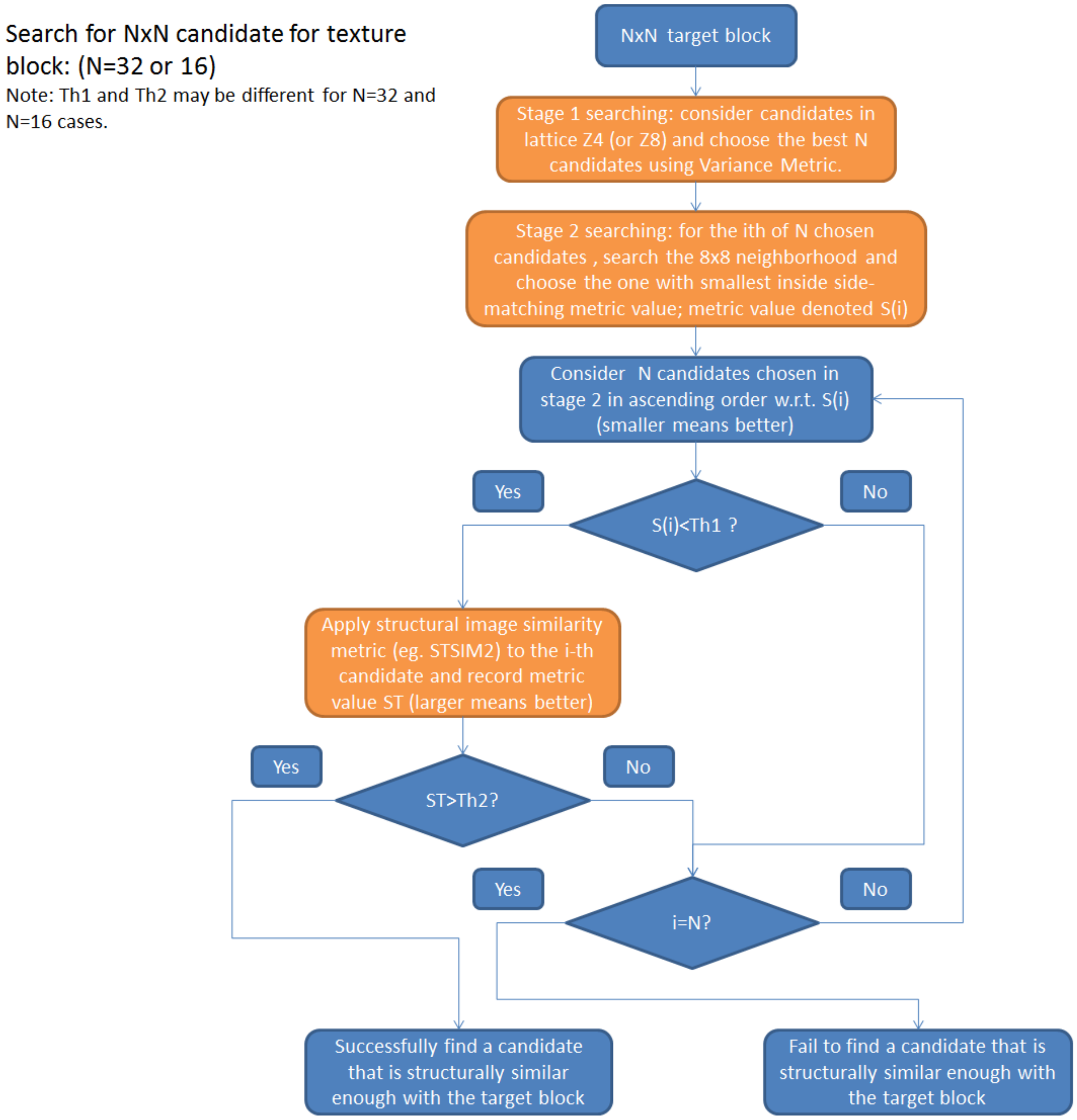

Search for NxN candidate for texture block: (N=32 or 16)
Note: Th1 and Th2 may be different for N=32 and N=16 cases.
NxN target block
Stage 1 searching: consider candidates in lattice Z4 (or Z8) and choose the best N candidates using Variance Metric.
Stage 2 searching: for the ith of N chosen candidates , search the 8x8 neighborhood and choose the one with smallest inside side-matching metric value; metric value denoted S(i)
Consider N candidates chosen in stage 2 in ascending order w.r.t. S(i) (smaller means better)
Yes
No
S(i)<Th1 ?
Apply structural image similarity metric (eg. STSIM2) to the i-th candidate and record metric value ST (larger means better)
Yes
No
ST>Th2?
Yes
No
i=N?
Successfully find a candidate that is structurally similar enough with the target block
Fail to find a candidate that is structurally similar enough with the target block

Figure 7.12: Flow chart of the candidate-search strategy for texture blocks.

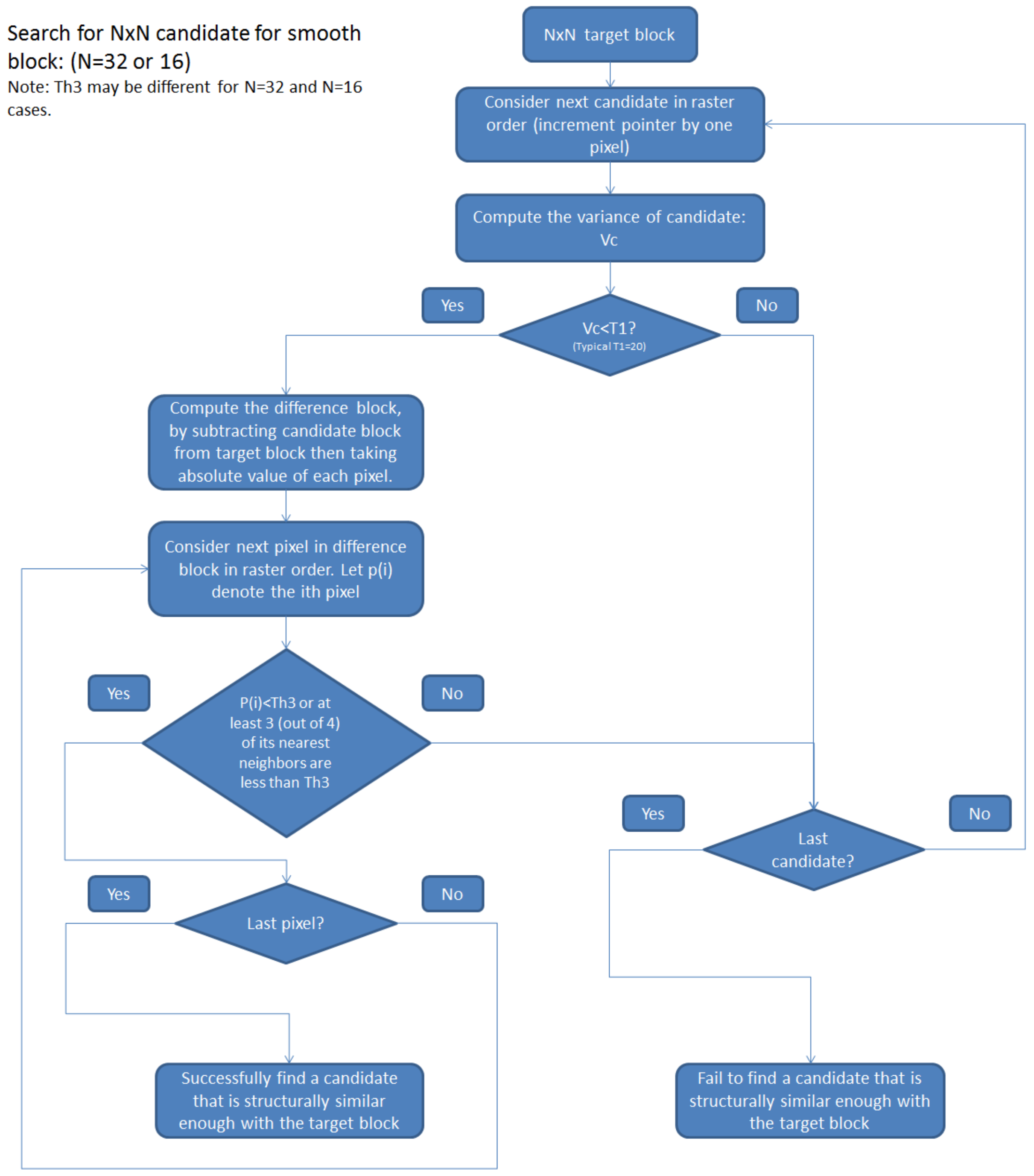

Search for NxN candidate for smooth block: (N=32 or 16)
Note: Th3 may be different for N=32 and N=16 cases.
NxN target block
Consider next candidate in raster order (increment pointer by one pixel)
Compute the variance of candidate: Vc
Yes
No
Vc<T1?
(Typical T1=20)
Compute the difference block, by subtracting candidate block from target block then taking absolute value of each pixel.
Consider next pixel in difference block in raster order. Let p(i) denote the ith pixel
Yes
No
P(i)<Th3 or at least 3 (out of 4) of its nearest neighbors are less than Th3
Yes
No
Last candidate?
Yes
No
Last pixel?
Successfully find a candidate that is structurally similar enough with the target block
Fail to find a candidate that is structurally similar enough with the target block

Figure 7.13: Flow chart of candidate-search strategy for smooth blocks.

7.3 Typical Results of the MTC Algorithm

7.3.1 MTC Using STSIM2

Figure 7.14 shows a result of an image coded with MTC, compared to the same image coded with JPEG. Both images are coded at 0.34 bpp. In this example, STSIM2 is the texture similarity metric in MTC. The lower right image shows the coding details, in which intensities 0 and 1 indicate MTC-encoded texture blocks, intensities from 2 to 4 indicate JPEG-encoded blocks and intensities 5 and 6 indicate MTC-encoded smooth blocks.[1] Figure 7.15 also shows the detailed coding information of the image coded with MTC.

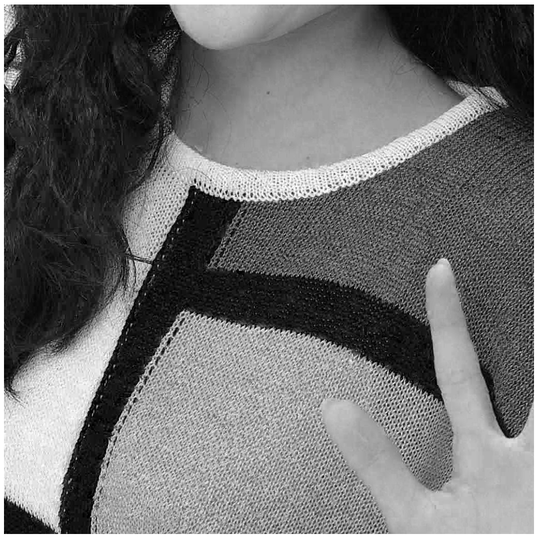
In the JPEG-encoded image in Fig. 7.14, there are clear artifacts in the hair, neck, hand and white sweater regions of the woman. These artifacts are due to the large quantization step sizes for DCT coefficients used in low quality JPEG. The MTC-encoded image in Fig. 7.14 has much better quality than the the JPEG-encoded one, although at the same rate. Hence, from Fig. 7.14, we claim that at low coding rates, MTC gives much better reconstruction quality than JPEG. Figure 7.15 shows the reason why MTC outperforms JPEG. Pie charts show that in image coded with MTC, although texture matching encodes 44% of the image, it contributes only 8% of the bits. As we know, JPEG tends to spend many bits to encode texture regions. When using MTC, most of the texture matching encoded regions are highly textured. Therefore, many bits can be saved in this way. Then the bits saved can be transferred to the JPEG-encoded regions. So in an image coded with MTC, better-quality JPEG can be applied without increasing the overall coding rate.

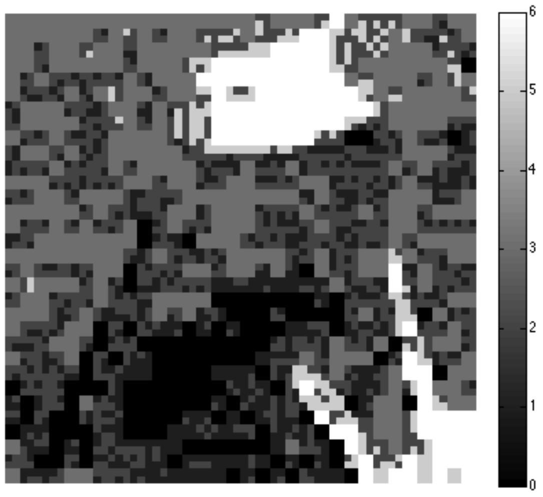
In the next example, the "house" image is coded with MTC and JPEG, respectively, at 0.34 bpp. Again, STSIM2 is used as the texture similarity metric in MTC. As shown in Fig. 7.16, the one encoded with JPEG has strong blocking artifacts in the roof and grass areas. Although a careful examination of the windows reveals some artifacts for the MTC-encoded image, its overall quality is still better than the JPEG-encoded one. The lower right

[1]In particular, intensities 0, 3, 4 and 6 indicate 32×32 blocks, and the remaining indicates 16×16 blocks.

Original “woman” image:1024×1024

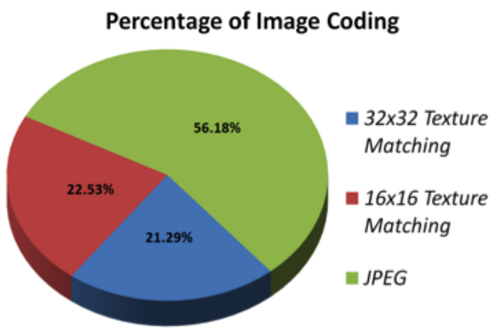
JPEG-coded image


MTC with STSIM2



MTC encoding details

Figure 7.14: Example of the “woman” image coded at 0.34 bpp.



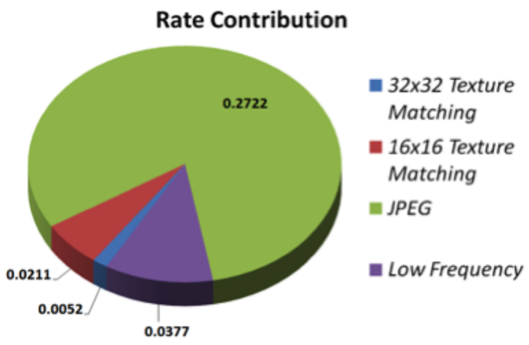
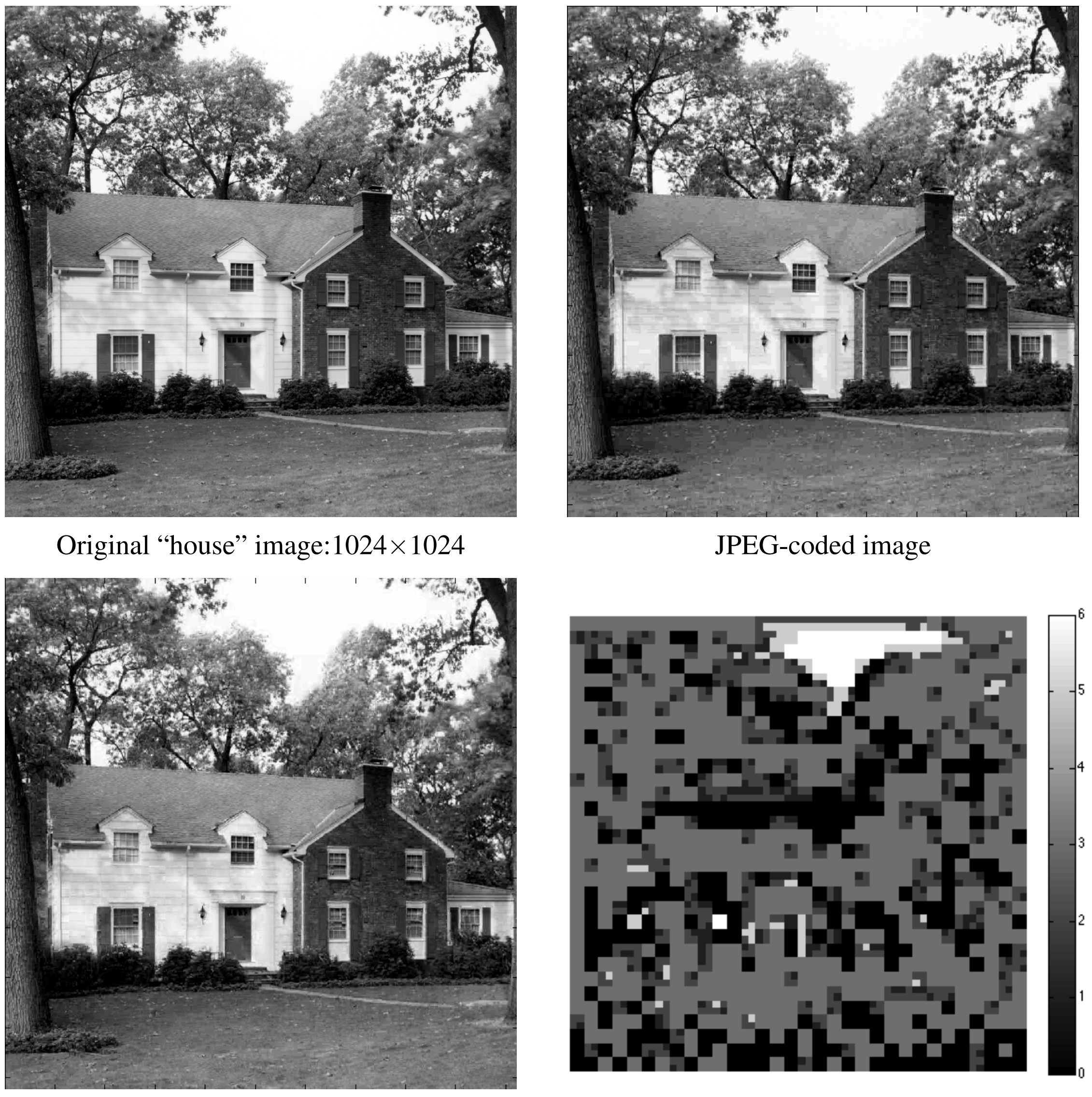


Figure 7.15: Detailed coding information of the “woman” image coded with MTC.



Original "house" image:1024×1024

JPEG-coded image

MTC with STSIM2

MTC encoding details

Figure 7.16: Example of the "house" image coded at 0.34 bpp.

Table 7.3: Statistics for the MTC-encoded “house” image.

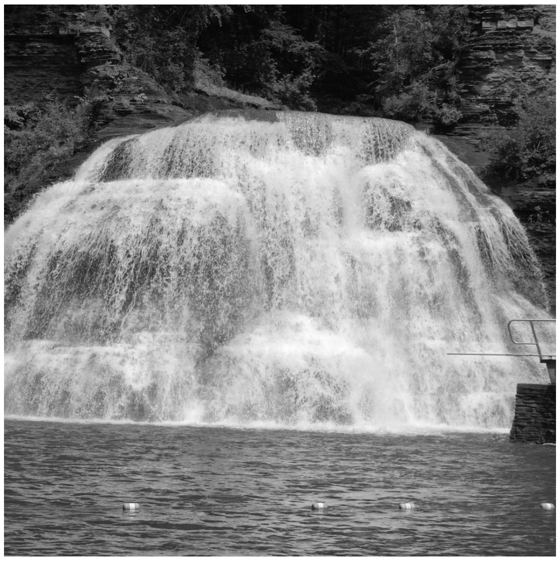
	32×32 MTC	16×16 MTC	8×8 JPEG	Low Freq. Coding	Overall
Image coded	25.7%	8.26%	66.1%		100%
Rate (bpp)	0.006	0.008	0.28	0.04	0.34

image in Fig 7.16 and Table 7.3 show the encoding details of the image coded with MTC. Again, many bits are saved since one third of the image is coded by the almost “bit-free” texture matching algorithm. The saved bits can enable better JPEG quality in the remaining two thirds of the image without increasing the overall coding rate.

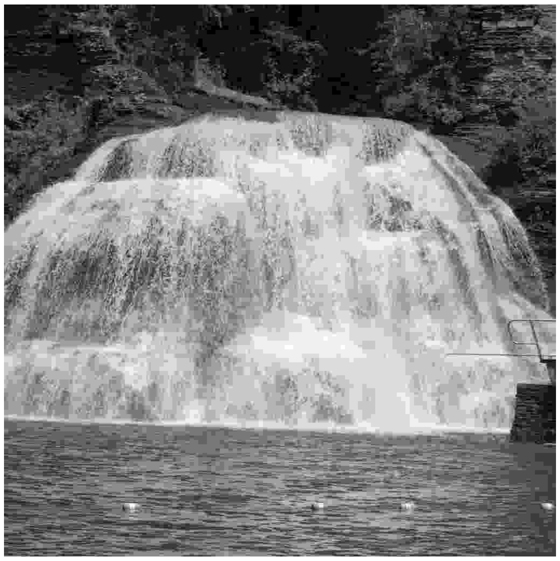
7.3.2 MTC Using LRI$^+$

One of the drawbacks of using STSIM2 is its computational complexity. In the next example, we use the newly proposed LRI$^+$ to substitute STSIM2 as the texture similarity metric in the MTC algorithm to encode an “waterfall” image. Specifically, we use the simplest version of LRI$^+$ proposed in Section 3.5.1, LRI$_b^+$ (LRI-A, LBP, SCD$_{\text{EST}}$ and IP). Figure 7.17 shows the original “waterfall” image together with three encoded versions at a very low coding rate, 0.18 bpp. It is clear that, at such a low coding rate, the JPEG-encoded image has the worst quality, with a lot of blocking artifacts in the waterfall region. And the two MTC-encoded images have roughly the same reconstruction quality, while the one using LRI$^+$ runs 6.3 time faster than the one using STSIM2. Hence, our newly developed LRI$^+$ is at least at good as STSIM2 when applied to the MTC algorithm, while substantially accelerates the encoding process.

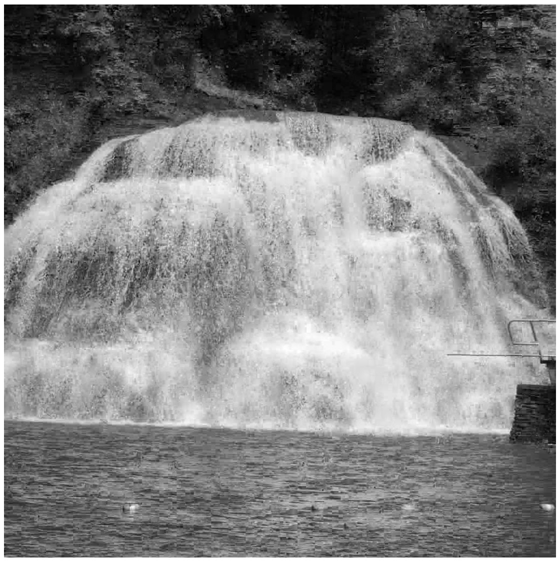
From Fig. 7.18 to Fig. 7.21, we show four more results of images encoded with MTC using LRI$_b^+$ as the texture similarity metric. In each figure, the upper left image is the original, the upper right image is encoded with JPEG, the lower left image is encoded with MTC at the same rate as the one encoded with JPEG, and the lower right image is the MTC coding details. Table 7.4 explains the lower right images in each figure. The MTC coding details can also be found in Table 7.5 - Table 7.8.



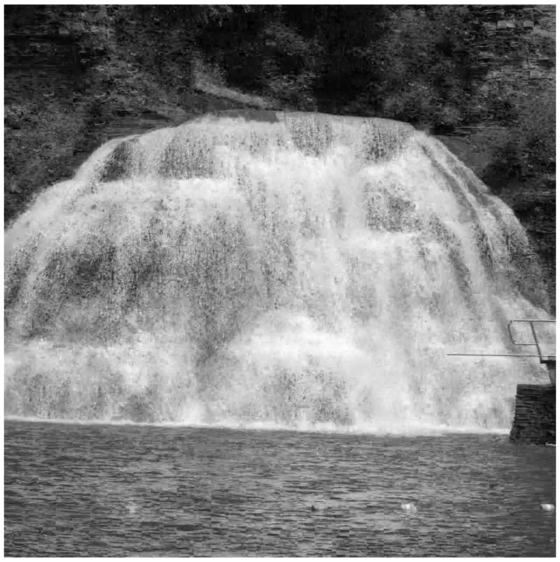
Original image:1024×1024



JPEG-coded image



MTC with STSIM2



MTC with LRI$^+$

Figure 7.17: Example of the “waterfall” image coded at 0.18 bpp.

Table 7.4: Interpretations of different pixel intensities in the lower right images in Fig. 7.18 - Fig. 7.21.

Pixel intensity	0 (Black)	1	2	3 (White)
Interpretation	32×32 MTC	16×16 MTC	16×16 JPEG	32×32 JPEG

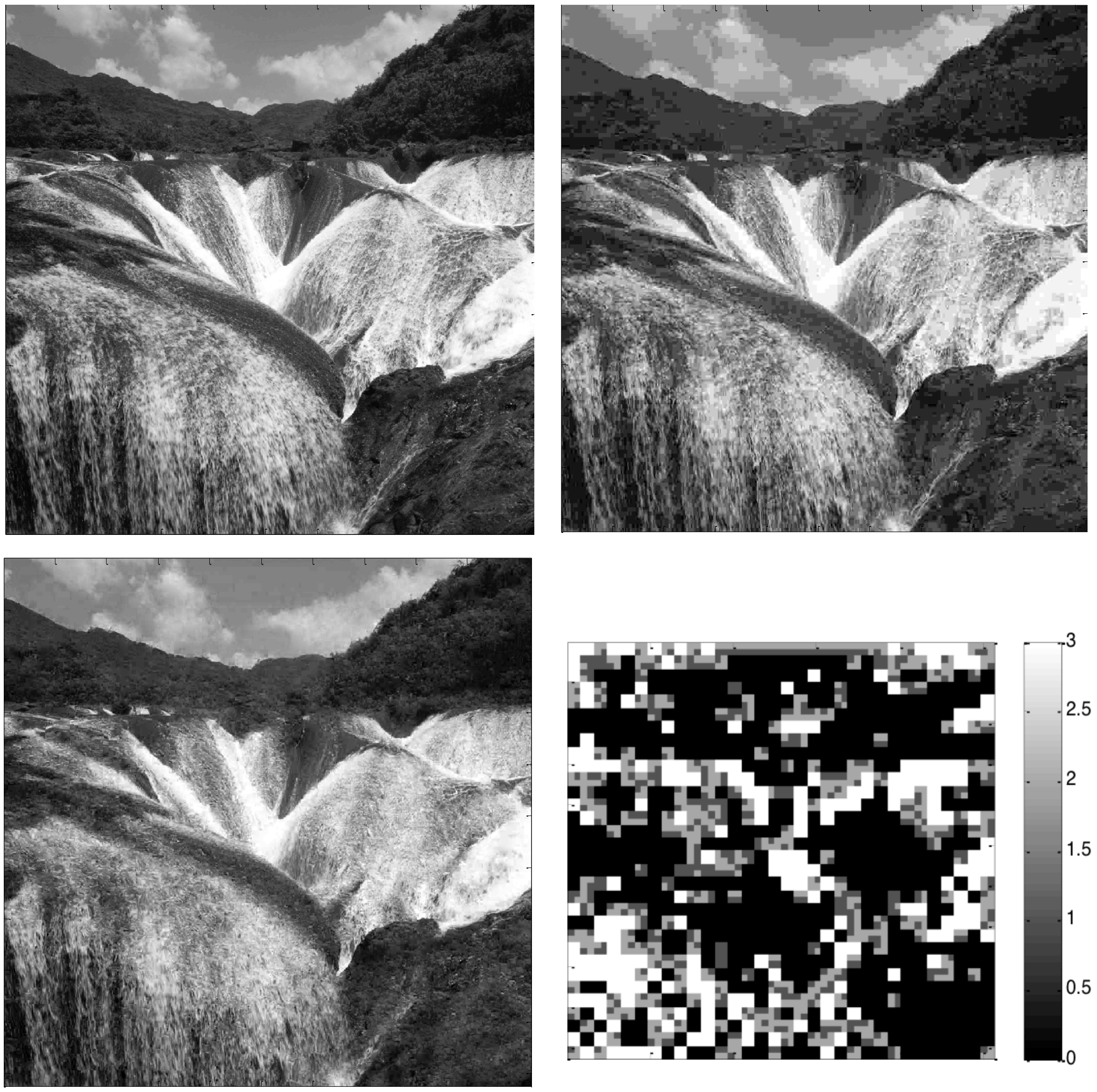


Figure 7.18: MTC coding example 1. Upper left: original image. Upper right: JPEG-encoded image at 0.20 bpp. Lower left: MTC-encoded image at 0.20 bpp (using LRI$^+$ as the texture similarity metric). Lower right: MTC coding details.

Table 7.5: Statistics for the MTC-encoded image in Fig. 7.18.

	32×32 MTC	16×16 MTC	JPEG	Low Freq. Coding	Overall
Image coded	53.1%	11.7%	35.2%		100%
Rate (bpp)	0.013	0.011	0.130	0.042	0.196

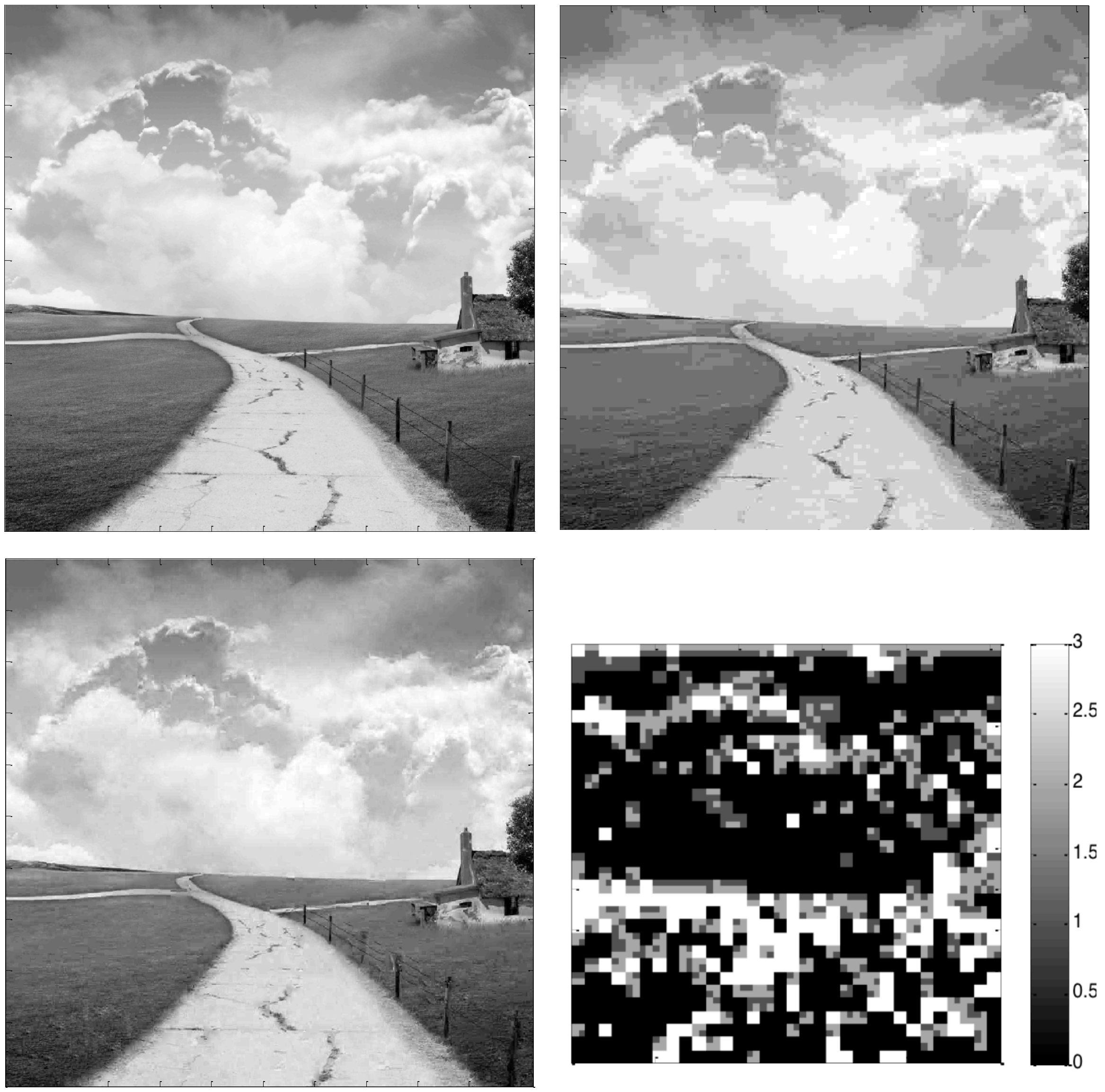


Figure 7.19: MTC coding example 2. Upper left: original image. Upper right: JPEG-encoded image at 0.13 bpp. Lower left: MTC-encoded image at 0.13 bpp (using LRI$^+$ as the texture similarity metric). Lower right: MTC coding details.

Table 7.6: Statistics for the MTC-encoded image in Fig. 7.19.

	32×32 MTC	16×16 MTC	JPEG	Low Freq. Coding	Overall
Image coded	57.5%	11.4%	31.1%		100%
Rate (bpp)	0.014	0.011	0.073	0.036	0.134

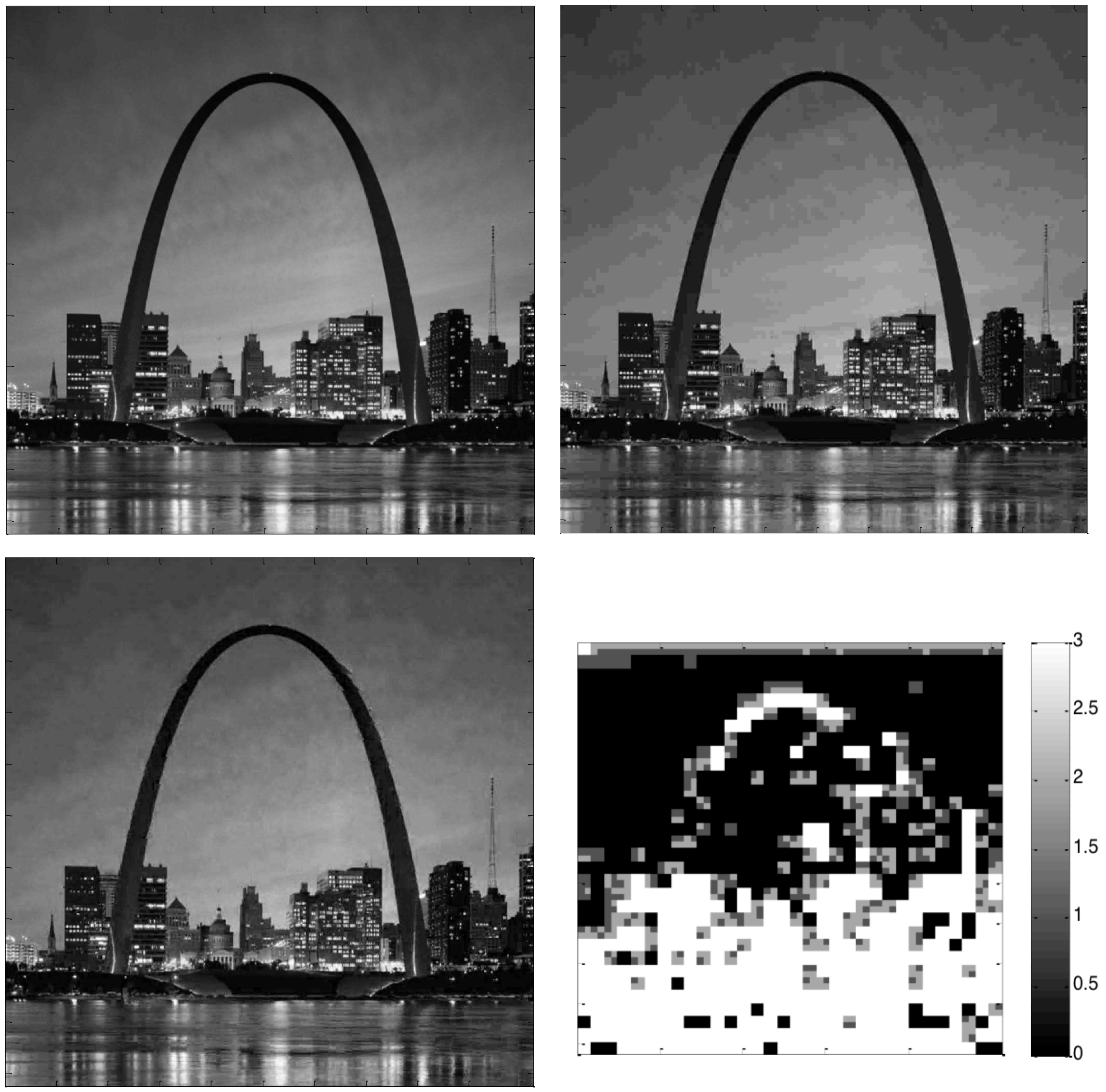


Figure 7.20: MTC coding example 3. Upper left: original image. Upper right: JPEG-encoded image at 0.18 bpp. Lower left: MTC-encoded image at 0.18 bpp (using LRI$^+$ as the texture similarity metric). Lower right: MTC coding details.

Table 7.7: Statistics for the MTC-encoded image in Fig. 7.20.

	32×32 MTC	16×16 MTC	JPEG	Low Freq. Coding	Overall
Image coded	47.1%	7.7%	45.2%		100%
Rate (bpp)	0.012	0.007	0.124	0.039	0.182

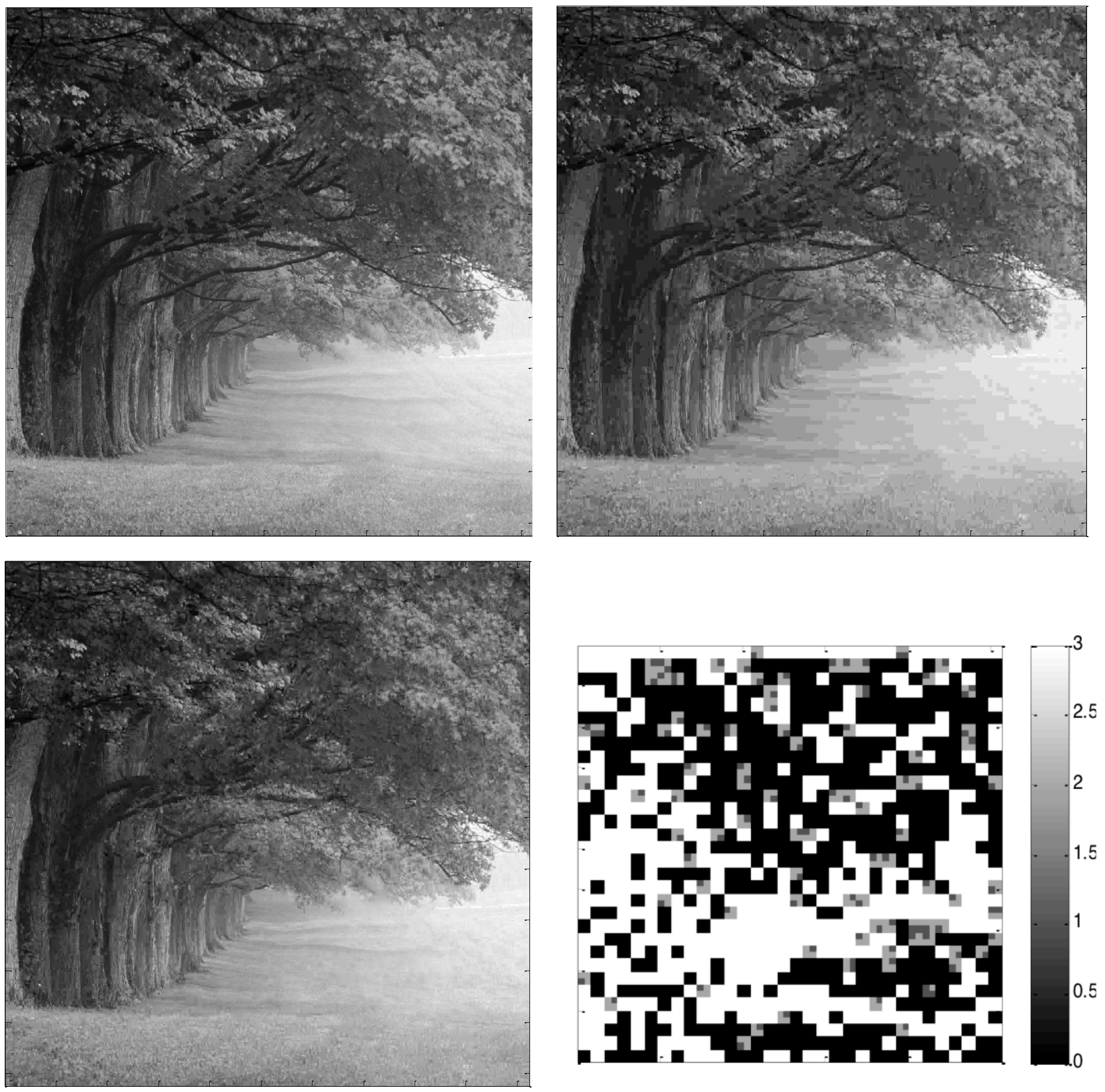


Figure 7.21: MTC coding example 4. Upper left: original image. Upper right: JPEG-encoded image at 0.20 bpp. Lower left: MTC-encoded image at 0.20 bpp (using LRI$^+$ as the texture similarity metric). Lower right: MTC coding details.

Table 7.8: Statistics for the MTC-encoded image in Fig. 7.21.

	32×32 MTC	16×16 MTC	JPEG	Low Freq. Coding	Overall
Image coded	45.9%	2.5%	51.6%		100%
Rate (bpp)	0.011	0.002	0.144	0.038	0.195

CHAPTER 8

Bilevel Image Similarity Metrics

This chapter presents a study of bilevel image similarity, including new objective metrics intended to quantify similarity consistent with human perception, and a subjective experiment to obtain ground truth for judging the performance of the objective similarity metrics. The focus is on scenic bilevel images, which are complex, natural or hand-drawn images, such as landscapes or portraits.

The ground truth was obtained from ratings by 77 subjects of 44 distorted versions of seven scenic images, using a modified version of the SDSCE testing methodology. The original and distorted images used in the subjective experiments, along with the subjective rating data obtained can be found in the "Bilevel Image Similarity Ground Truth Archive" at University of Michigan Deep Blue[1].

Based on hypotheses about human perception of bilevel images, several new metrics are proposed that outperform existing ones in the sense of attaining significantly higher Pearson and Spearman-rank correlation coefficients with respect to the ground truth from the subjective experiment. The new metrics include Adjusted Percentage Error (APE), Bilevel Gradient Histogram (GH) and Connected Components Comparison (CC). Combinations of these metrics are also proposed, which exploit their complementarity to attain even better performance.

These metrics and the ground truth are then used to assess the relative severity of various

[1]http://deepblue.lib.umich.edu/handle/2027.42/111059.

kinds of distortion and the performance of several lossy bilevel compression methods.

8.1 New Bilevel Image Similarity Metrics

This section proposes several new bilevel image similarity metrics, all calculated within $n \times n$ windows sliding across the image, for example, $n = 32$. This sliding-window structure is motivated, to a large degree, by the hypothesis that if the window size is of the order of foveal vision, which is the approximately two-degree-wide region[2] of clearest vision [111, p. 7], then what happens outside the window cannot mask errors within the window, whereas masking of errors can be caused by the contents of the window itself. If the window were chosen to be larger than fovial vision, then it could happen that the metric predicts masking that does not actually occur. On the other hand, if the window were chosen smaller than fovial vision, then the metric will be unable to take into account masking effects that occur outside the window but within foveal vision. We find that the hypothesis that the window's size should be of the order of fovial vision is supported by the experimental results in Section 8.4. Once the window size is specified, one must also specify the horizontal and vertical steps with which the window will slide across each image, which determine the *window overlapping rate*. In Section 8.4, we choose the overlapping rate based on experiments.

After computing the metric values $M(X_i, Y_i)$ at all window locations i in images X and Y, the final metric value $M(X, Y)$ is the average of all $M(X_i, Y_i)$:

$$M(X,Y) = \frac{1}{N_{\text{win}}} \sum_i M(X_i, Y_i) \, ,$$

where N_{win} is the total number of window locations.

We consider percentage error (PE) to be the *baseline metric*. Though PE treats all errors in all windows equally, in fact, error visibility depends significantly on the surrounding

[2]When viewing a computer monitor at 20 inches, two degrees is approximately 0.7 inches, or 70 pixels at 100 dpi.

content. For example, an error can be masked if the surrounding content is “busy” in the sense that there are many nearby black-white transitions, *i.e.*, adjacent pairs of pixels with one being black and the other white. Hence, PE can be improved by taking these effects into account. Figure 8.1 illustrates some shortcomings of PE. It shows an original scenic bilevel image together with four distorted versions. For each, both PE and subjective rating score are presented. Subjective rating scores, which range from 0 to 1, with 1 meaning identical, are obtained from the subjective experiment described in Section 8.2. Based on PE, the first distorted image is the most similar to the original, and the others are approximately equally dissimilar. However, the subjective rating scores indicate that human observers found the first two distorted images to be significantly less similar to the original than the last two.

Each of the metrics proposed in this section is motivated by some particular hypothesis about human perception, and attempts to outperform PE in measuring bilevel image similarity.

8.1.1 Adjusted Percentage Error

The first new metric is motivated by the hypothesis that when more pixels within a window, or adjacent to it, have one color than the other, then errors in (*i.e.*, changes to) the pixels with the minority color are more visible than errors in the pixels with the majority color. Moreover, the visibility of errors in minority pixels increases as their proportion decreases. From now on we refer to the pixels having the minority color as the *foreground* F and the remaining pixels as the *background* B.

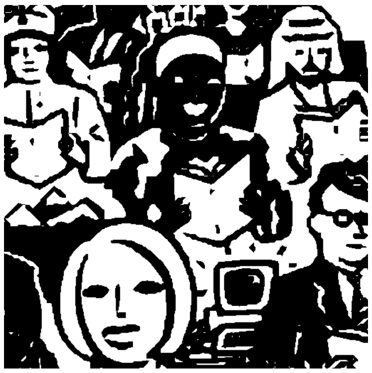
Based on this hypothesis, we define the Adjusted Percentage Error (APE) as follows. Suppose the window is $n \times n$, the size of foreground is $|F|$, the size of background is $|B| = n^2 - |F|$, the number of foreground errors is e_F, and the number of background errors is e_B. Then APE is the average of *foreground error rate* $\frac{e_F}{|F|}$ and *background error rate* $\frac{e_B}{|B|}$:

$$\text{APE} \triangleq \frac{1}{2} \times \frac{e_F}{|F|} + \frac{1}{2} \times \frac{e_B}{|B|},$$

Original image: 512×512

PE: 0.039 Subjective: 0.32

PE: 0.047 Subjective: 0.33


PE: 0.047 Subjective: 0.40

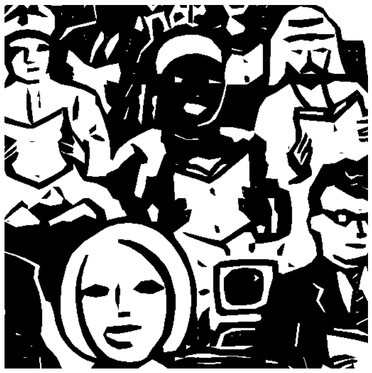

PE: 0.049 Subjective: 0.43

Figure 8.1: Several scenic bilevel images with different percentage errors (PE) and subjective rating scores.

which takes value in $[0, 1]$. Since $|F| \leq |B|$, individual foreground errors are given more weight than background errors. When $|F| = |B|$, APE = PE. For a given e_F and e_B, as $|F|$ shrinks, APE increases, consistent with the hypothesis that foreground errors become more significant as the size of foreground becomes smaller. One may also view PE as an average of background and foreground error rates:

$$\text{PE} \triangleq \frac{e_F + e_B}{|F| + |B|} = \frac{|F|}{|F| + |B|} \times \frac{e_F}{|F|} + \frac{|B|}{|F| + |B|} \times \frac{e_B}{|B|},$$

from which we see how PE emphasizes background error rate more than foreground error rate.

To link APE with intensity-based overlap metrics described in Section 2.2.1, let us formulate APE using the overlap counts a, b, c and d. One may observe that in Section 2.2.1, all intensity-based overlap metrics are symmetric with respect to image 1 and 2. In other words, b and c always play the same role in metrics. In contrast, APE is an asymmetric metric that focuses more on the original image, say image 1, than the distorted one (image 2). Since the foreground size $|F| = \min(a+b, c+d)$, APE can be rewritten as

$$\text{APE} = \frac{1}{2} \times \frac{b}{a+b} + \frac{1}{2} \times \frac{c}{c+d}.$$

Note that if image 2 were considered the original, then

$$\text{APE} = \frac{1}{2} \times \frac{c}{a+c} + \frac{1}{2} \times \frac{b}{b+d}.$$

from which it becomes clear that the metric is asymmetric.

We also consider two slight variations of APE:

$$\text{APE}' \triangleq \frac{1}{2} \times \frac{e_{F'}}{|F'|} + \frac{1}{2} \times \frac{e_{B'}}{|B'|}, \qquad \text{APE}'' \triangleq \frac{e_F + e_B}{|F|},$$

where F' is the one-step dilation of F using a 3×3 all ones structure element matrix, e'_F is the number of errors within F', $B' = W - F'$ denotes the remainder of the window W, and e'_B is the number of errors in B'. The hypothesis behind APE$'$ is that errors adjacent to F are as significant as foreground errors and therefore should be counted in the first term, which has the smaller denominator, rather than the second term, which has the larger denominator. APE$''$ is the ratio of total number of errors to the size of foreground. In this event, foreground and background errors are treated equally in APE$''$, just as in PE. However, the weight of errors within a window is inversely proportional to the size of foreground, so that errors within a window with small foreground count more than those within a window with large foreground.

8.1.2 Bilevel Gradient Histogram

For bilevel images, the contours between black and white regions contain most of the information. Hence, as considered in SmSIM [50], similar bilevel images should have similar contour smoothness, roughness and directionality. For grayscale images, a gradient histogram is a feature that captures such information. This is also true for bilevel images.

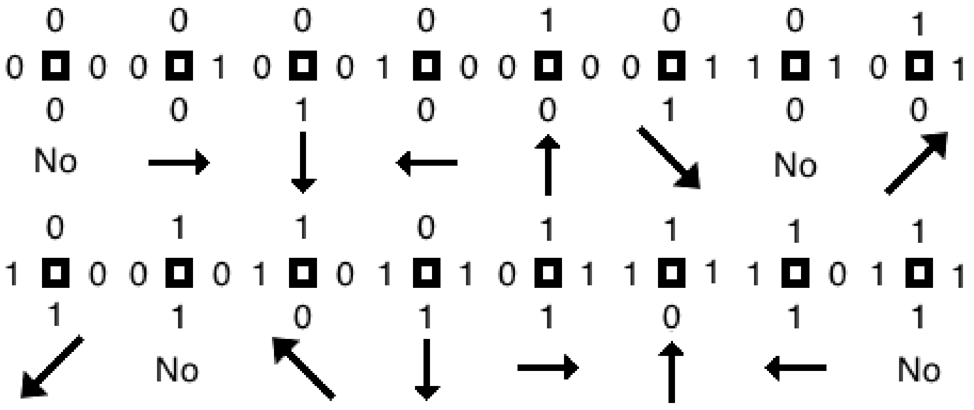


Figure 8.2: Bilevel Gradient.

However, a new definition of gradient is needed. With such, the similarity of bilevel gradient histograms of the original and distorted images becomes a good candidate for measuring similarity.

As the *bilevel gradient* at pixel $X(u,v)$, we propose $BG_{u,v} \triangleq \text{angle}(\underline{V}_{u,v})$, where $\underline{V}_{u,v}$ is the complex number

$$\underline{V}_{u,v} \triangleq X(u,v+1) - X(u,v-1) + j(X(u-1,v) - X(u+1,v)),$$

provided this number is not zero. When $\underline{V}_{u,v}$ is zero, for example when $X(u,v)$ lies in a monotone region, there is no direction at pixel $X(u,v)$, and $BG_{u,v}$ is not defined. It follows that $BG_{u,v}$ has the eight possible values illustrated in Fig. 8.2, and consequently, the gradient histogram for a given window position consists of eight values $C = \{C(1), \ldots, C(8)\}$.

Clearly, the proposed bilevel gradient histogram can distinguish different directional contours. Its ability to measure contour smoothness and roughness can be seen from the example shown in Fig. 8.3. The left image has a smooth contour, so that all pixels along the edge have the same gradient direction, while the rough contour in the right image causes a distinctly different gradient distribution.

To measure the similarity $S(C,D)$ of the histograms C and D corresponding to the original and distorted images, respectively, at a given window location, we propose three methods. In each, a small value indicates high similarity, and to avoid singularities, we

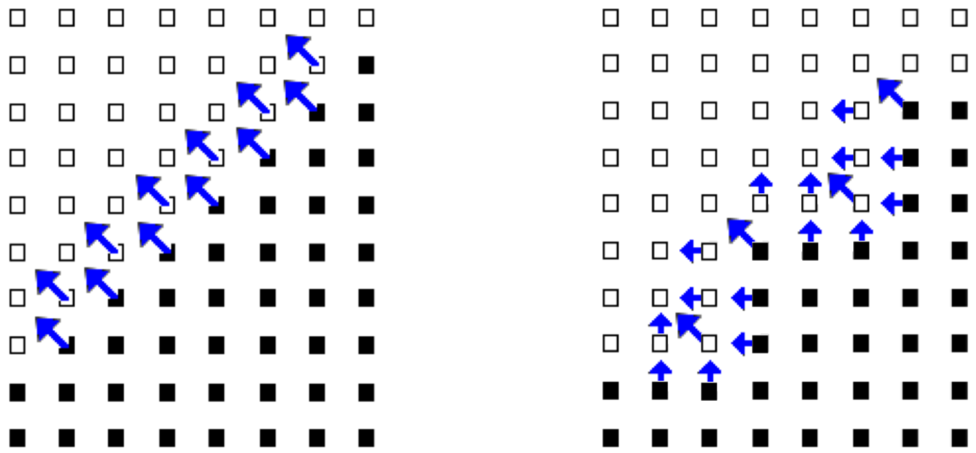


Figure 8.3: Exmples of smooth and rough contours.

increase any zero histogram value to one.

1. $$S^1(C,D) \triangleq 1-\prod_{k=1}^{8} \frac{2C(k)D(k)}{C^2(k)+D^2(k)} \,.$$

As each term in the product is the ratio of a geometric average to an arithmetic average (as commonly used for example in [3, 12, 14, 16, 17, 34, 112, 113]), it is less than or equal to one, making $S^1(C,D)$ non-negative. By multiplicatively combining eight terms, we tacitly assume that a distorted image has high similarity only when all eight values are similar to the original. Hence, this is a strict measure of histogram similarity, which may over penalize some histogram differences.

2. $$S^2(C,D) \triangleq \sum_{k=1}^{8} c(k) \log \frac{c(k)}{d(k)} \,,$$

where c and d denote C and D normalized so as to sum to one. This is the Kullback-Leibler divergence [82] of probability mass function d with respect to c.

3. $$S^3(C,D) \triangleq \left(\sum_{k=1}^{8} c(k) \log \frac{c(k)}{d(k)} \right) \times \frac{\max(\|C\|_1, \|D\|_1)}{\min(\|C\|_1, \|D\|_1)} \,.$$

In addition to the divergence of d with respect to c, this method also considers the similarities between the L_1 norms of C and D, which approximates the total number of pixels along edges within an image window.

We denote the Gradient Histogram metric with these three similarity methods as GH^1, GH^2 and GH^3, respectively.

We also experimented with a definition of bilevel gradient that depended on the eight nearest neighbors, rather than four, yielding 8 gradient directions, and another definition yielding 16. Since the improvements in the experiments of Section 8.4 resulting from these enhanced gradients were small, from now on we assume the gradient definition given previously.

8.1.3 Connected Components Comparison

The concept of connected components is useful in bilevel image analysis. Here, we hypothesize that distorted images should preserve the connected components of the foreground of the original. The simplest way to use this hypothesis is to compare the number of connected components in the original and distorted image windows. However, to avoid a small isolated dot adjacent to a large component from being counted as a new connected component, we do a one-step dilation with a 3×3 all ones structuring element before counting. Dilation helps connect isolated dots and islands that are close to some big connected components. We propose two methods to assess similarity using connected components.

The first compares the *effective number* of foreground connected components in a window W of the original and distorted images, where the effective number of connected components in a window W with N connected components is

$$N_W \triangleq \sum_{k=1}^{N} \min\Big(1, \frac{|cc_k|}{T_V}\Big) ,$$

where $|cc_k|$ is the size of the k^{th} connected component and T_V is a threshold greater than 1, which increases robustness by reducing the effect of small connected components, *e.g.*, isolated dots. In this paper, $T_V = 10$. All connected components with size less than T_V contribute less than one to N_W. Now, if X is the original and Y is the distorted image at

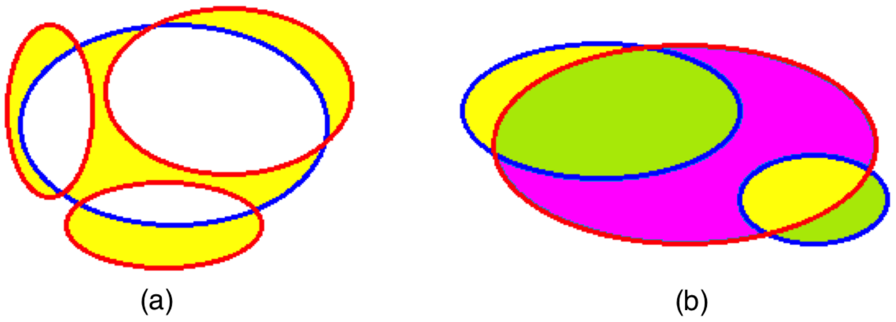


Figure 8.4: Examples of CC^2 calculation.

window W, the metric value is

$$\text{CC}^1 \triangleq 1 - \frac{\min(N_{W,X}, N_{W,Y})}{\max(N_{W,X}, N_{W,Y})} .$$

CC^1 takes values between 0 and 1, with 0 meaning identical.

The second method considers not only the number of connected components, but also errors inside or adjacent to each connected component in the original image. The hypothesis here is that a good reconstruction should not only preserve the number of connected components, but also their shapes. Suppose for some window W, the foreground connected components for the original and distorted images are $[cc_1, cc_2, \dots, cc_{N_1}]$ and $[cc_1^d, cc_2^d, \dots, cc_{N_2}^d]$, respectively. As explained below, the CC^2 metric value is, basically, the summation of individual metrics, CC$_i^2$, one for each connected component cc_i in the original.

If $N_1 > 0$, let $[cc_{i,1}^d, cc_{i,2}^d, \dots, cc_{i,k_i}^d]$ denote all connected components in the distorted image that overlap cc_i, and define

$$CC_i^2 \triangleq \left| cc_i \triangle \bigcup_{t=1}^{k_i} cc_{i,t}^d \right| \times (|k_i - 1| + 1)^p ,$$

where $A \triangle B$ denotes the symmetric difference between sets A and B. The term above within *size brackets* measures the total number of errors between cc_i and the union of $cc_{i,t}^d,\ t \in \{1, 2, \dots, k_i\}$. The second term penalizes the lack of any overlapping connected components ($k_i = 0$) or multiple connected components overlapping cc_i ($k_i > 1$). Parameter p, which we choose to equal to 1, controls the severity of the penalty. Figure 8.4(a) gives an example. The region enclosed by the blue curve is cc_i, and the distorted image has three connected components, enclosed by red curves, overlapping cc_i. Hence $k_i = 3$, and the size of the yellow region represents the first term in the formula above. Finally, we have

$$\mathrm{CC}^2 \triangleq \sum_{i=1}^{N_1} \mathrm{CC}_i^2 + \sum_{t=1}^{N_2} \delta\big[|cc_t^d \cap (\bigcup_{r=1}^{N_1} cc_r)|\big] \times |cc_t^d|\,,$$

where $\delta[0] = 1$ and $\delta[n] = 0, \forall n \neq 0$. The second summation above represents the penalty for having connected components in the distorted image that are disjoint with all connected components in the original. This term is important if the distorted image has many new connected components.

Note that if the original image window is monotone, *i.e.*, it contains only background, then $N_1 = 0$, and CC^2 reduces to

$$\mathrm{CC}^2 \triangleq \sum_{t=1}^{N_2} |cc_t^d|\,.$$

Note also that CC^2 is closely related to PE. If for all cc_i, $k_i = 1$, and each cc_i^d overlaps only one cc_j for some j, then CC^2 $=$ PE. However, when there are missing or split connected components, *e.g.*, Figure 8.4(a), CC^2 will penalize appropriately.

The false connection of two or more connected components is another interesting case. As illustrated in Fig. 8.4(b), two connected components, cc_i and cc_j, enclosed by blue curves, become one connected component in the distorted image, enclosed by the red curve. The yellow region is penalized in CC$_i^2$ and the green region is penalized in CC$_j^2$. The purple region, however, is penalized in both CC$_i^2$ and CC$_j^2$. Thus, we see that a false connection is penalized multiple times.

8.2 Subjective Evaluation Experiment

In Section 1.2 and Section 8.1, we reviewed existing bilevel image similarity metrics and proposed several new metrics, respectively. In order to compare their performance, ground truth is needed. This ground truth should consist of a collection of distorted scenic bilevel images with subjective similarity ratings to their original. This section describes a subjective experiment designed to obtain such ground truth using a modified version of simultaneous double stimulus for continuous evaluation (SDSCE) suggested in ITU-R BT.500-11 [51], as described next.

8.2.1 Experiment Design

In our experiments, each distorted image, called a *test image*, is shown simultaneously side by side with its original. Figure 8.5 shows an example of the screen that each subject saw during the experiment. Subjects are told which is the original and asked to rate the similarity of the distorted image to its original by dragging a slider on a continuous scale as in [53]. As benchmarks to help subjects make good ratings, the scale is divided into five equal portions, labeled “Bad”, “Poor”, “Fair”, “Good” and “Excellent”. Each rating is then rounded to the nearest integer between 0 and 100. In addition, unlike previous work, the rating time for each image by each subject was recorded for screening purposes. However, subjects were not informed of this. To make sure the recorded time information is as accurate as possible, a “Pause” button is added so that subjects could take rests during the experiment without influencing the rating times. Finally, since the number of test images is large, to prevent subjects from becoming impatient, we divide the 315 test images into 15 groups and display to subjects the number of remaining groups, instead of the number of remaining images, during the whole experiment. The grouping does not influence the data processing.

During the experiment, the ordering of test images is independently randomized for each

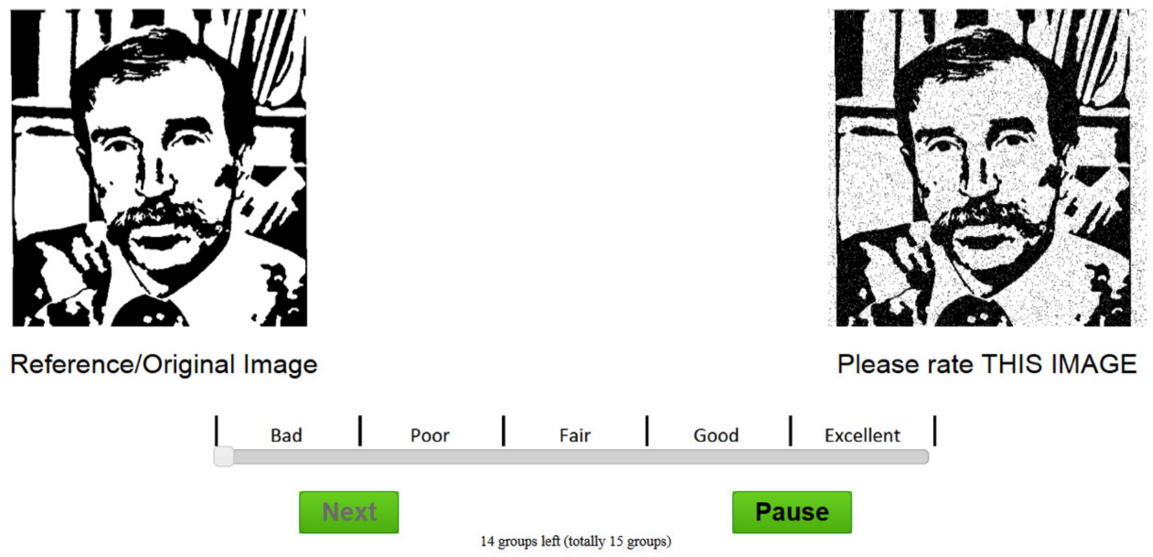


Figure 8.5: A sample screen of the subjective experiment.

subject to avoid systematic bias that might be caused by some fixed ordering. Moreover, to avoid contextual effects (discussed later), no two successive test images come from the same original.

The database of test images is developed from the seven scenic images shown in Fig. 8.6, each with size 512×512. The first six images are recognizable scenes and the last one, 'MRF', is typical of an Ising Markov random field model, which has been proposed as a model for scenic images [114, 115]. Seven kinds of distortions are created, resulting in 44 distorted images for each original:

i. Finite State Automata Coding (FSA) [116] with nine error rate factors: $[1, 100, 150, 200, 300, 400, 500, 700, 1000]$.
ii. Lossy Cutset Coding (LCC) [114,115] with eight grid sizes: $[2, 4, 6, 8, 10, 12, 14, 16]$.
iii. Lossy Cutset Coding with Connection Bits (LCC-CB) [114,115] with the same eight grid sizes as LCC.
iv. Hierarchical LCC (HC) [117] with eight MSE thresholds for block splitting: $[0, 0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 1]$.
v. Random bit flipping with five different probabilities: $[0.01, 0.03, 0.05, 0.10, 0.15]$.

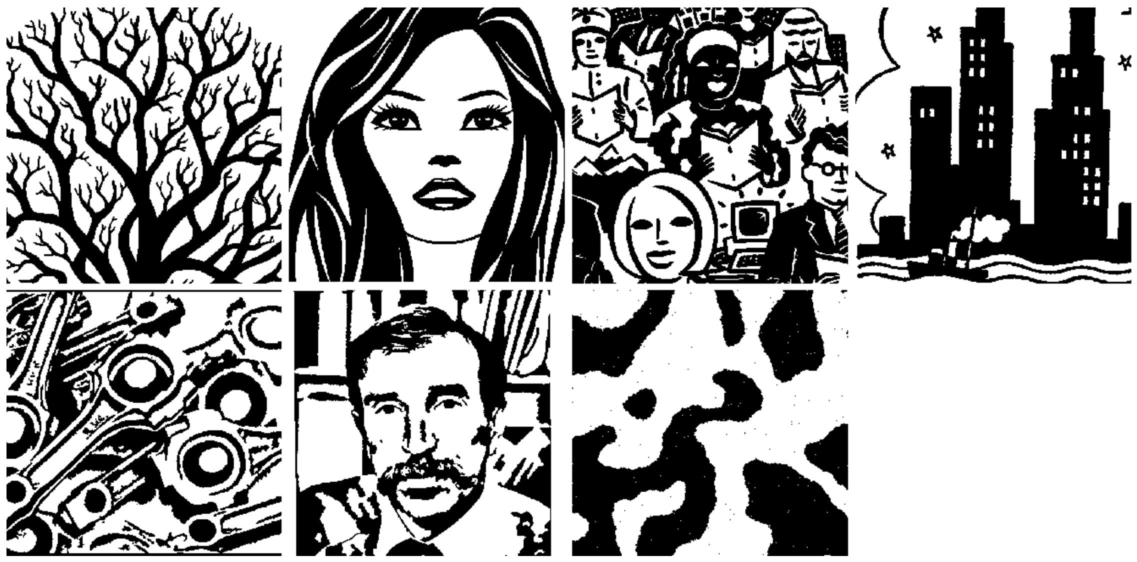


Figure 8.6: Seven scenic images: ‘tree’, ‘woman’, ‘people’, ‘boat’, ‘tools’, ‘Alc’, ‘MRF’.

vi. Dilation with 1, 2 and 3 iterations using a 3×3 all ones structuring element.

vii. Erosion with 1, 2 and 3 iterations using a 3×3 all ones structuring element.

Figure 8.7 shows the seven test images, each with a randomly selected distortion. Besides these distorted images, every original image itself is also included as a “distorted image” in order to verify that, as described later, subjects are making good faith judgments. Thus, since there are seven original images, each subject is asked to rate $45 \times 7 = 315$ images, each displayed side by side with the original at size $4'' \times 4''$. Subjects were asked to view the images from approximately 20 inches.

Before participating, each subject was given an explanation of the purpose of the experiment and a description of the procedure. In addition, several training images, similar to actual test images, are shown to subjects. These training images roughly cover the whole similarity range in the database.

8.2.2 Data Processing 1: Scaling the Ratings

In all, 77 subjects, all non-experts, completed a session in which they rated all 315 distorted images. For each subject, raw rating data, test image order and rating times were

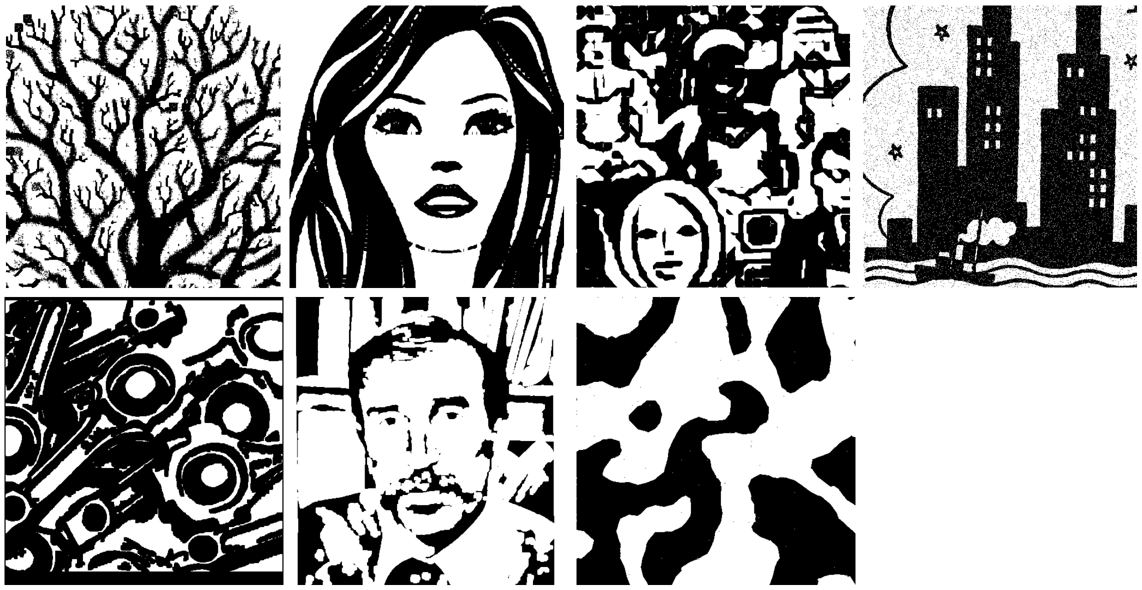


Figure 8.7: Seven randomly selected distorted images in the database, one for each original image.

recorded. As in [52], the raw rating data, Raw(i, j), for the j^{th} image by the i^{th} subject was then scaled to reduce systematic differences in ratings among subjects and to obtain values between 0 and 1, with 1 representing highest similarity:

$$\text{Scaled}(i,j) = \frac{\text{Raw}(i,j) - \min(\text{Raw}(i,k), \forall k)}{\max(\text{Raw}(i,k), \forall k) - \min(\text{Raw}(i,k), \forall k)} .$$

From now on, we will work with scaled rating data.

8.2.3 Data Processing 2: Subject Screening

Subject screening, such as in [51, 53], which is designed to rule out abnormal subjects and those who are just randomly rating, helps improve the quality of the ground truth. In this experiment, a subject is rejected if at least two of the following criteria are satisfied:

i. Total rating time is less than 10 minutes.

ii. More than 33 outlier ratings. (Described later.)

iii. At least two ratings of original images are outliers.

iv. Average of the scaled ratings for the seven original images is less than 0.5.

v. The “monotonicity test” is failed. (Described later.)

The motivation for criteria ii) and iii) is that the presence of many outlier ratings, especially for original images, indicate abnormal behavior or careless rating. Hence the corresponding subjects should be screened out. Similar to the approach taken in [53], a scaled rating Scaled(i, j) is considered an outlier if

$$|\text{Scaled}(i, j) - \text{avg}(j)| > \delta \times \text{std}(j)\,,$$

where avg(j) and std(j) are the expectation and standard deviation of scaled rating scores for image j by all subjects. δ is chosen to be 1.96 corresponding to a 95% confidence interval, assuming scaled rating scores are Gaussian.

The “monotonicity test” in criterion v) is a new idea, based on the property of our database that for each type of distortion, there is a clear monotonicity in the amount of distortion with respect to some parameter, such as bit flipping probability, number of dilation/erosion iterations, and coding rate for compression. Hence, if any subject's rating scores are too far from monotonic, the subject should be screened out. Specifically, for each subject i, a penalty counter $P(i)$ is initialized to zero. Now suppose

$$[\text{Scaled}(i, n_1), \text{Scaled}(i, n_2), \ldots, \text{Scaled}(i, n_k)]$$

are k ratings that should be monotonically non-increasing for reasons such as mentioned above. Then for each $t \in \{1, 2, \ldots, k-1\}$ such that

$$\text{Scaled}(i, n_{t+1}) > \text{Scaled}(i, n_t)\,,$$

$P(i)$ is increased by Scaled(i, n_{t+1}) $-$ Scaled(i, n_t). If, finally, $P(i) > 19$, subject i fails the monotonicity test and is screened out of the experiment.

Based on these five criteria, subject screening results are shown in Table 8.1. As can be seen, the three subjects that satisfy criterion 1 also satisfy almost all other four criteria. This fact is not surprising since short rating time usually indicates careless rating. And based on our total rating time threshold (10 minutes), the three subjects that satisfy criterion 1

Table 8.1: Subject screening result.

Criterion	Subjects that satisfy the corresponding screening criterion
1	4, 49, 54
2	4, 24, 49, 54, 59, 72
3	3, 4, 54, 55, 67
4	4, 49, 55, 67, 73
5	3, 4, 49, 54, 59

Table 8.2: Average rating time in seconds for each image.

Image	'tree'	'MRF'	'woman'	'Alc'	'tools'	'people'	'boat'
Time	4.00	4.10	4.21	4.56	4.64	4.72	4.93

spent less than 2 seconds to rate each test image, which is too short to make a responsible decision. After subject screening as described previously, seven subjects (3, 4, 49, 54, 55, 59, 67) were removed. From now on, all analyses are based only on the 70 remaining subjects.

8.3 Subjective Evaluation Results

8.3.1 Rating Time Analysis

For the 70 subjects retained, the average rating time was 23.4 minutes, with standard deviation 8.2. Table 8.2 shows the average rating times for each original image. Generally speaking, average rating time increases with image complexity, which makes sense because people need more time to evaluate a complex image than a simple one.

Figure 8.8 shows the relationship between subjective rating scores and average rating times. The red line is a linear regression fitting. It shows that average rating time increases

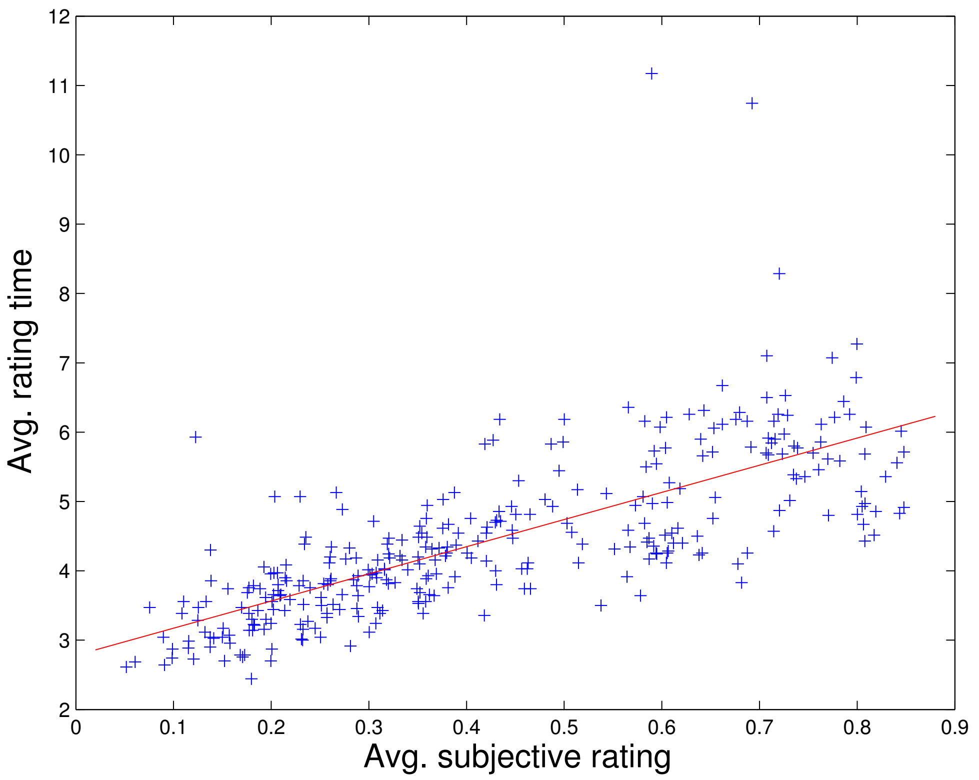


Figure 8.8: Average rating time in seconds vs. average subjective rating score for the 315 test images. Regression function: avg. rating time $= 3.92 \times$ avg. subjective rating $+ 2.78$.

with image similarity, which makes sense because it becomes harder to see and evaluate distortion as image similarity increases. This suggests that the subjects made serious efforts.

Another interesting result is the average rating time, over all sessions, for the nth displayed test image, as function of n. (Recall that the order of images is randomized for each test session.) As shown in Fig. 8.9, the average rating time decreases from almost 30 seconds for the first test image to 4 or 5 seconds after rating around 50 test images. The decline of average rating time indicates increasing familiarity with the experiment as the test session proceeds. On average, it takes about 50 images for a subject to be fully familiar with the experiment.

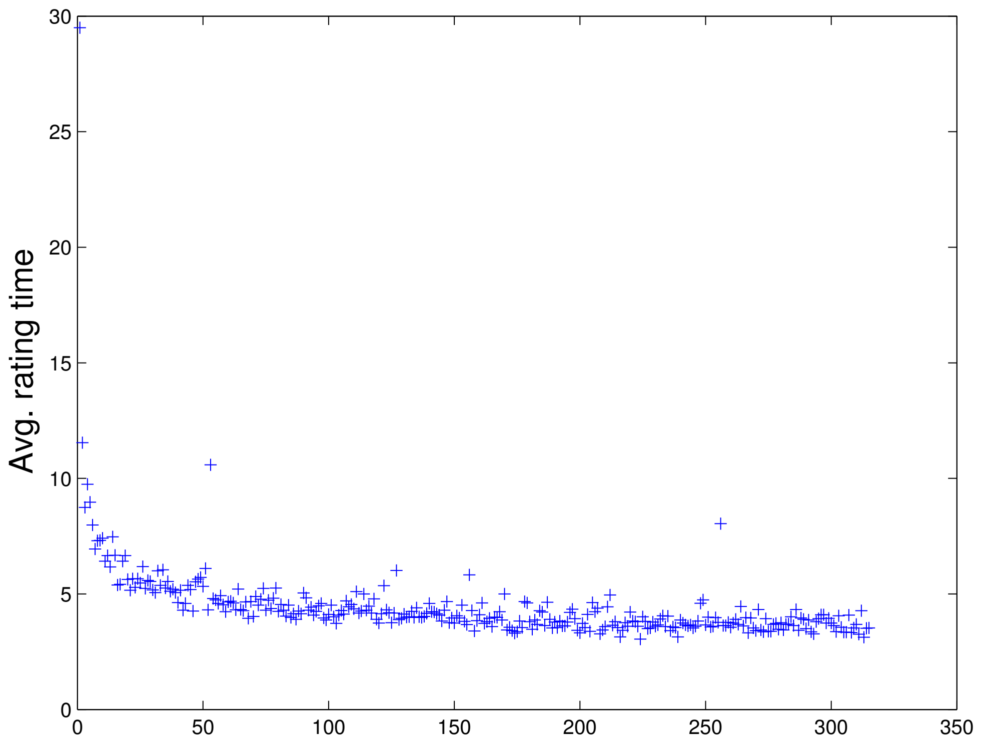


Figure 8.9: Average rating time as function of n.

8.3.2 Contextual Effects Analysis

As discussed in [51], contextual effects occur when the subjective rating of a test image is influenced by prior images presented to the subject, especially the previous test image. To check whether our testing procedure suffers from strong contextual effects, the following analysis is conducted. For each test image in each test session, we plot the relationship between:

i. The average rating score (over all sessions) of the previous test image in this session.
ii. The difference between the rating score of the current test image in the current test session and the average rating score for the current test image over all test sessions. This difference is called a "rating bias".

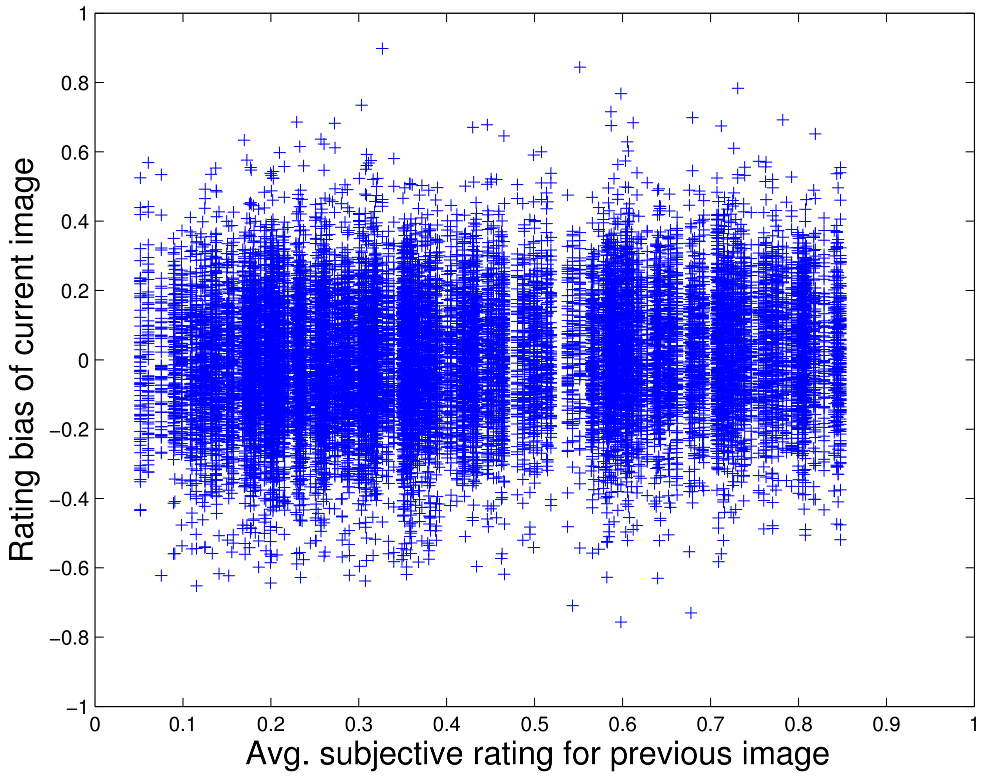


Figure 8.10: Results of contextual effects test.

If the testing procedure does not suffer from strong contextual effects, the rating bias of the current image should have symmetric distribution around zero, no matter the average rating score of the previous test image. The plot in Fig. 8.10 supports the hypothesis that the testing procedure is free from strong contextual effects.

8.3.3 Standard Deviation of Rating Scores

The ratings of different images have different standard deviations. Figure 8.11 presents a scatter plot showing the standard deviation of the scaled rating scores for each distorted image vs. its average rating score. The green solid line shows a quadratic regression fit, with two red dashed lines giving 2σ confidence bounds. As one would expect, for low and high similarity images, the standard deviations of rating scores are relatively small, meaning subjects are more consistent with their judgments. However, for images with

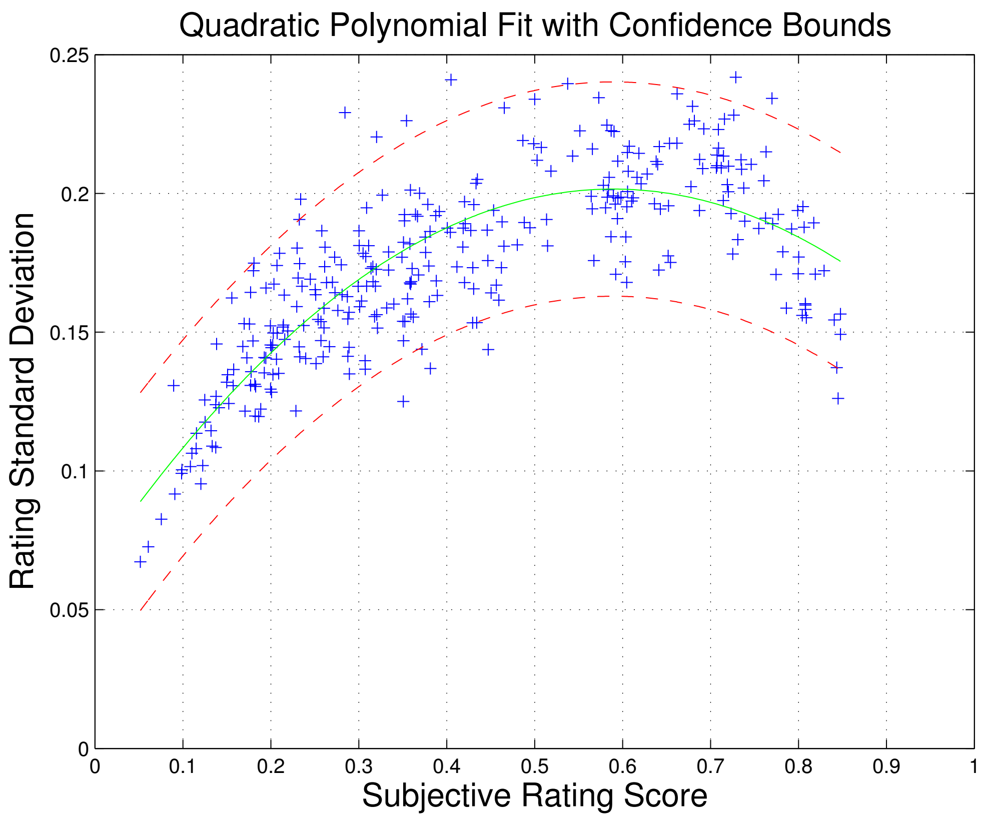


Figure 8.11: Standard deviation of ratings. Regression function:
std. dev. $= -0.39 \times$ avg.$^2 + 0.46 \times$ avg. $+ 0.07$.

moderate similarity, the standard deviations of rating scores are relatively large, showing less agreement among subjects.

Notice that since neither the lowest nor highest average rating scores are near zero or one, respectively, it does not appear that the standard deviation estimates are affected significantly by ceiling effects [118, p. 21].

8.3.4 Insensitivity of the 'MRF' Image to Distortion

As mentioned earlier, among the seven original test images, the first six contain recognizable scenes while the last, 'MRF', does not. From the experimental results, we found that human observers are fairly sensitive to the amounts of distortion added to the first six images, but are not so sensitive to the amounts of distortion added to the 'MRF' image. Figure

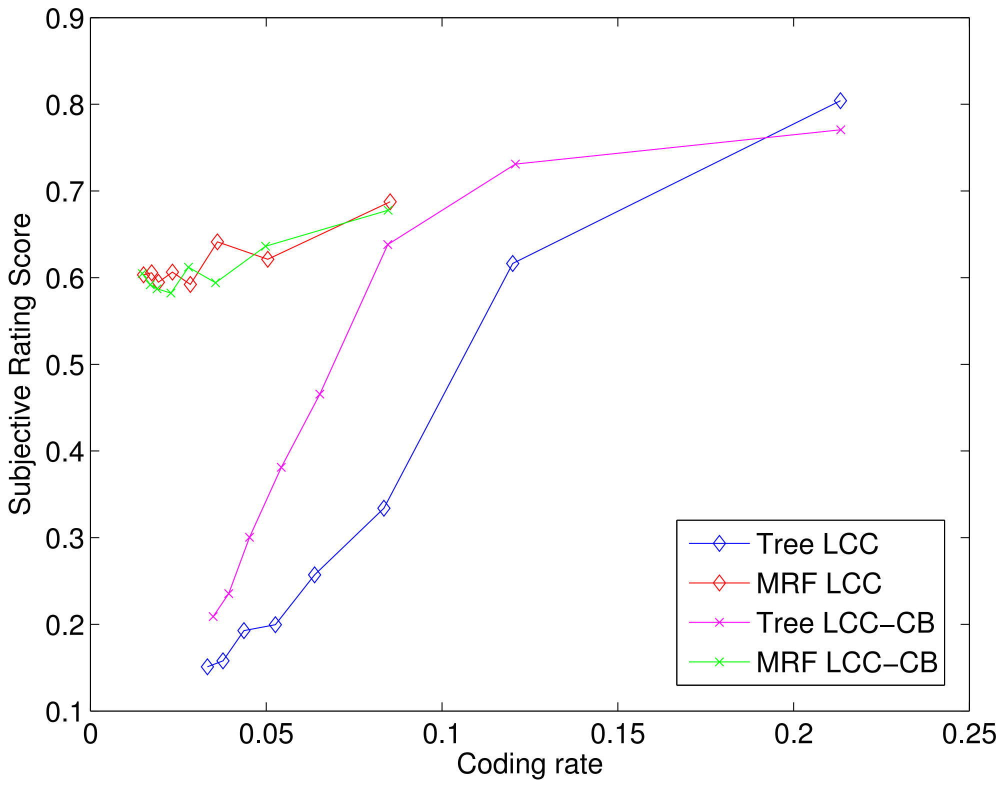


Figure 8.12: Subjective rating scores for ‘tree’ and ‘MRF’ coded with LCC and LCC-CB.

8.12 illustrates this finding by showing the subjective rating scores of both the ‘tree’ and ‘MRF’ images coded with LCC and LCC-CB, respectively. As can be seen, the subjective rating scores of the ‘tree’ images increase monotonically with coding rate. However, this is not the case for ‘MRF’ images. One possible reason is that when viewing an image with a recognizable scene, observers have a “ground truth” in mind with which to compare. Hence, it is relatively easy for them to observe the effects of increasing or decreasing distortion. However, if the image contains unfamiliar or abstract content, *e.g.*, the ‘MRF’ image, observers may have a hard time observing changes to the distortion. For this reason, the ‘MRF’ image is not used in the tests of the next two sections.

8.4 Tests of Bilevel Image Similarity Metrics

In this section, we analyze the performance of existing and new bilevel image similarity metrics using the ground truth obtained from the subjective experiments described above. In addition, we analyze several similarity metrics designed for grayscale images, namely, SSIM [12], LBP [27] and LRI [34]. LBP is computed using the eight surrounding pixels without interpolation. As suggested in [27], only uniform patterns with less than or equal to two 0/1 transitions are labeled. LRI-A is applied with $K = 4$ and $T < 1$, where $T < 1$ guarantees that all 0/1 transitions trigger non-zero LRI-A indices. While these other metrics were not specifically designed for bilevel images, they can obviously be applied. Generally speaking, they have considerably higher computational complexity.

The performance of each metric is evaluated using Pearson and Spearman-rank correlation coefficients to rate its consistency with the ground truth consisting of 44 distorted versions of the six images in Fig. 8.6 with recognizable scenes. (As mentioned earlier, we decided not to use 'MRF'.) The Pearson correlation is computed after nonlinear transformation of metric values by the 5-parameter logistic model proposed in [53] and shown below, with parameters chosen to maximize correlation for the metric being evaluated. In particular, the logistic model is:

$$Y = \beta_1 \text{logistic}(\beta_2, (X - \beta_3)) + \beta_4 X + \beta_5 \,,$$

where X is a metric value, Y is the transformed metric value and

$$\text{logistic}(\tau, X) = \frac{1}{2} - \frac{1}{1 + \exp(\tau X)} \,.$$

This is the usual strategy that avoids penalizing a metric simply for having a nonlinear relationship to the ground truth.

The next two subsections discuss the influence of window size and window overlapping rate, respectively.

8.4.1 Window Size Selection

In our experiments, each metric was evaluated with a variety of $n \times n$ window sizes: $n = 8, 16, 32, 64, 128, 256, 512$. Different metrics reacted differently to changes in n. We found that APE gives the best performance with $n = 64$ and 128. For small and large n, the performance decreases. We believe this result is closely related to the size of foveal vision (2 degrees) described in Section 8.1, which under the environment of our subjective experiment is approximately $0.7''$ or 90 pixels. We found that GH performs best for moderate window sizes ($n = 16$ and 32). On the one hand, when the window size is too small, the histogram is not robust. On the other hand, when the window size is greater than 32, the histograms naturally become more similar, even if the original and distorted images do not. The performance of CC decreases monotonically as n decreases, which is not surprising since small windows are not robust to the consideration of connected components. Finally, as a compromise, we choose window size 32×32 for all metrics evaluated in this section. However, this choice is influenced by viewing distance and image resolution, and might not be optimal if the experimental environment changes.

Note that while the intensity-based overlap metrics mentioned in Section 2.2.1 were originally applied globally to images, they can also be applied locally by computing and averaging metric values for windows sliding across both images, and in the results of this section, they are applied with the same 32×32 window as the new metrics.

8.4.2 Window Overlapping Rate Selection

On the one hand, if windows are not overlapped, then distortion in an image edge lying on the boundary between two windows could be missed by the metric. On the other hand, a high rate of window overlapping can significantly increase the computational load. In our experiments, we compared overlapping rates of 0%, 25%, 50% and 75% for several metrics; results are shown in Table 8.3. We see that for all chosen metrics, as window overlapping rate increases, the performance measured by Pearson and Spearman-rank cor-

Table 8.3: Window Overlapping Rate Selection.

Overlapping rate	0%	0%	25%	25%	50%	50%	75%	75%
Metric	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman
APE	0.87	0.86	0.87	0.86	0.87	0.87	0.87	0.87
GH^1	0.88	0.80	0.88	0.80	0.88	0.80	0.88	0.80
GH^2	0.92	0.88	0.92	0.88	0.92	0.89	0.93	0.89
GH^3	0.91	0.85	0.91	0.86	0.92	0.87	0.92	0.88
CC^1	0.87	0.84	0.85	0.83	0.86	0.85	0.88	0.86
LBP	0.90	0.84	0.90	0.84	0.91	0.85	0.91	0.85
LRI	0.89	0.84	0.89	0.84	0.90	0.84	0.90	0.85

relation coefficients either stays constant or increases by very small amounts. Since the improvements are not significant, we use non-overlapped windows from now on for both the new and existing metrics.

8.4.3 Evaluation of Metrics

This subsection compares metric performance by reporting the Pearson and Spearman-rank correlation coefficients with the ground truth using non-overlapped 32×32 windows. In Table 8.4, the newly proposed similarity metrics are compared to the existing metrics designed for bilevel images, *e.g.*, intensity-based overlap metrics described in Table 2.1 and SmSIM [50], as well as metrics designed for grayscale images, *i.e.*, SSIM [12], LBP [27] and LRI [34].

All but two of the existing intensity-based overlap metrics (the first column in Table 8.4), have very competitive performance. Compared to these intensity-based overlap metrics and the baseline metric PE, the proposed APE gives better performance, especially in Spearman-rank correlation coefficients. Surprisingly, SmSIM [50] performs a little worse than the baseline metric, PE, meaning it is not a very effective similarity metric. SSIM is designed to measure grayscale image quality. Results in Table 8.4 show that SSIM does not provide satisfactory performance for scenic bilevel images. Although LBP and LRI are designed to measure grayscale texture similarity, results suggest that they are also capable of measuring bilevel image similarity.

All three versions of APE outperform PE, proving that its hypothesis is good. Specifically, the fact that APE and APE$'$ work better than PE and APE$''$ indicates that foreground errors are more visible than background errors, and should be penalized harder. The fact that APE outperforms APE$'$ suggests that dilation of the foreground is not necessary. Among the three versions of bilevel gradient histogram metrics, GH^1 is the worst, suggesting that multiplicatively combining eight terms may cause over-penalization. Both GH^2 and GH^3 provide very good results, suggesting that divergence is suitable for comparing histogram similarity in this application. In addition, GH^2 is the overall best similarity metric. CC^1 and CC^2 give comparable performance to APE. We know CC^2 is closely related to PE. The fact that CC^2 outperforms PE suggests that the consideration of connected components helps predict human judgments on scenic bilevel image similarity.

8.4.4 Combining Different Metrics

Since the different metrics assess complementary aspects, one can expect to attain better performance by combining them. After testing many combinations, the best ones are shown in Table 8.5. The formula for combining metrics $X_i,\ i = 1, 2, \ldots, m$, is

$$Y = \prod_{i=1}^{m} X_i^{p_i} ,$$

where the X_i's are similarity metric values after nonlinear transformation. The best combination we found is APE and GH^2 (with $p_1 = 0.2$ and $p_2 = 0.4$), where APE measures the overall accuracy of the distorted image to the original, while GH^2 quantifies the contour similarity. The motivation behind this combination is similar to that for SmSIM [50]. Similarly, PE and CC^2 also provide accuracy information and are complementary to GH^2.

Table 8.4: Metric evaluation. (P = Pearson, S = Spearman.)

Metric	P	S	Metric	P	S
PE	0.84	0.81	SmSIM [50]	0.81	0.74
Jaccard [42]	0.87	0.81	SSIM [12]	0.77	0.78
Kulczynski [43]	0.75	0.76	LBP [27]	0.90	0.84
Kulczynski [43]	0.86	0.82	LRI [34]	0.89	0.84
Braun-Blanquet [44]	0.85	0.79	APE	0.87	0.86
Dice [45]	0.86	0.81	APE′	0.88	0.80
Ochiai [46]	0.86	0.81	APE″	0.86	0.84
Sokal & Michener [85]	0.86	0.82	GH^1	0.88	0.80
Simpson [86]	0.61	0.54	**$\mathbf{GH}^2$**	**0.92**	**0.88**
Rogers & Tanimoto [87]	0.85	0.82	GH^3	0.91	0.85
Sokal & Sneath [88]	0.86	0.82	CC^1	0.87	0.84
Sokal & Sneath [88]	0.87	0.81	CC^2	0.87	0.83

The combination of LBP, LRI and GH^2 also gives comparable performance. However, as the computational load of LBP and LRI is much higher, this combination is not suggested. The fact that all of the best combinations include GH^2 suggests that the bilevel gradient histogram contains information that is important to predicting human perception of scenic bilevel image similarity.

8.5 Assessing Bilevel Image Distortion

This section uses the ground truth and new similarity metrics to compare the performance of several lossy bilevel compression methods and to assess the relative severity of several types of distortion, including random bit flipping, dilation and erosion.

Table 8.5: Metric combination evaluation.

Overlapping rate	0%	0%	75%	75%
Combination	Pearson	Spearman	Pearson	Spearman
APE & GH^2	0.94	0.92	**0.95**	**0.94**
PE & GH^2	0.93	0.90	0.94	0.91
CC^2 & GH^2	0.93	0.90	0.94	0.91
LBP & LRI & GH^2	0.93	0.89	0.94	0.90

8.5.1 Comparing Lossy Compression Algorithms

As mentioned earlier, one important application of similarity metrics is to judge the performance of compression algorithms. In this subsection, we use both the metrics and the ground truth to compare the four lossy compression algorithms used in the subjective experiment, namely, Finite State Automata (FSA) [116], Lossy Cutset Coding (LCC) [114, 115], Lossy Cutset Coding with Connection Bits (LCC-CB) [114, 115] and Hierarchical LCC (HC) [117].

Figure 8.13 shows the subjective rating scores of the reconstructed images produced by the four lossy bilevel compression algorithms, averaged over the six images with recognizable scenes in Fig.1.1, and plotted vs. coding rate in bits per pixel (bpp). As can be seen, HC has the best performance at all coding rates. The runner-ups are two versions of Lossy Cutset Coding (LCC-CB and LCC). FSA has the lowest rating scores at each coding rate, although its difference to LCC at low coding rates is negligible. Moreover, the plot for HC suggests that coding at rates between 0.04 and 0.06 bpp is quite attractive, as higher coding rates do not substantially increase the subjective rating scores, while lower rates suffer a significant drop.

Next, as an application of objective similarity metrics, Figure 8.14 compares the same four lossy compression methods on the basis of the new metric with the best performance, namely, the combination of APE and GH^2 (computed using 32×32 windows and 75%

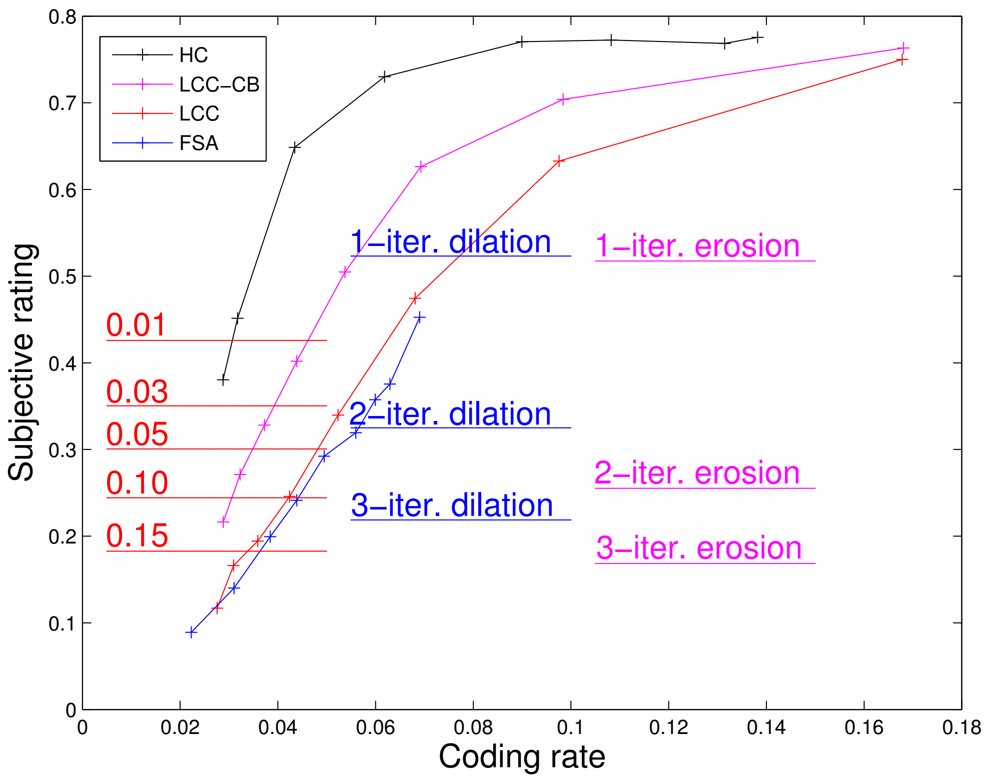


Figure 8.13: Experimental results. Red: random bit flipping with different probabilities. Blue: dilation. Purple: erosion.

window overlapping rate). (The metric values plotted are those obtained after the nonlinear transformation that maximizes the Pearson correlation.)

Compared to the subjective rating scores in Fig. 8.13, Figure 8.14 preserves the relative relationship of the four compression methods. However, one can see the relative sizes of the gains from one coding method to another are not always accurately reflected in the objective metric values. For example, according to the ground truth results in Fig. 8.13 at rates around 0.04 bpp, the subjective rating score for LCC is a little better than FSA, LCC-CB is considerably better than LCC, and HC is considerably better than LCC-CB. Figure 8.14 shows the same relationship. However, the advantage of HC with respect to LCC-CB is smaller and the advantage of LCC with respect to FSA is larger.

Figure 8.15 shows the four ‘tree’ images coded at rates around 0.04 bpp with differ-

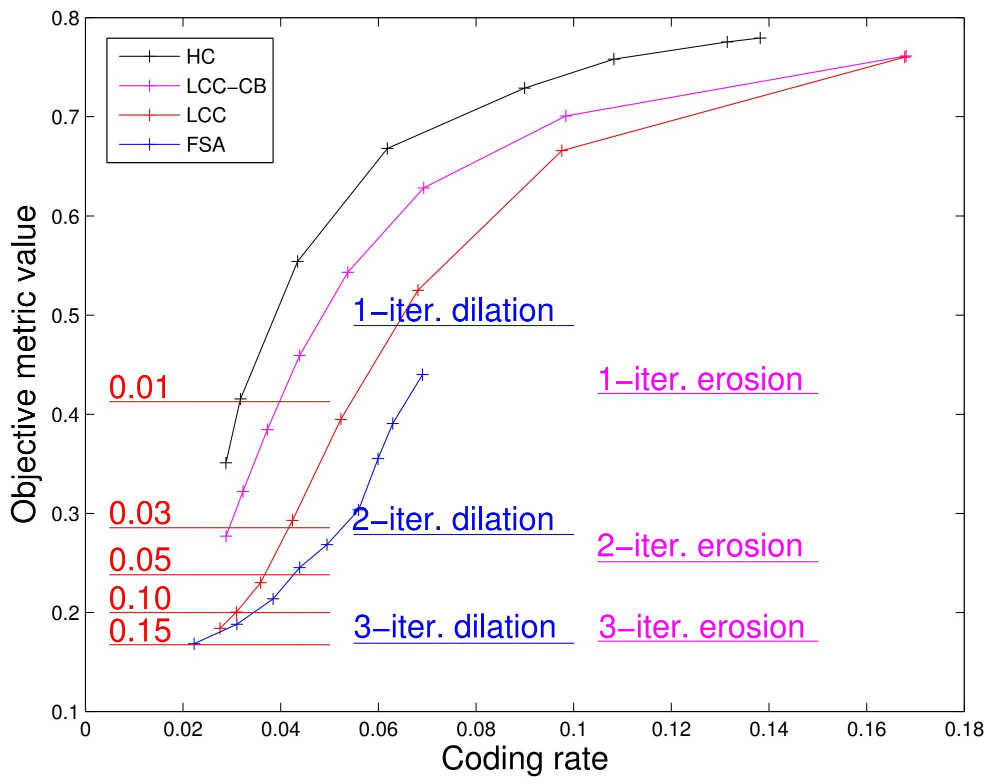
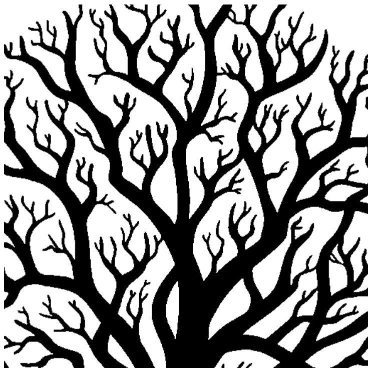
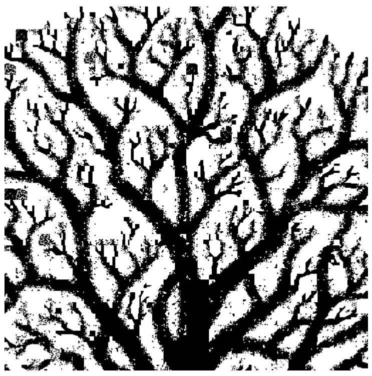
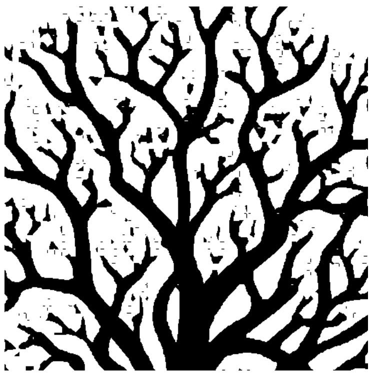


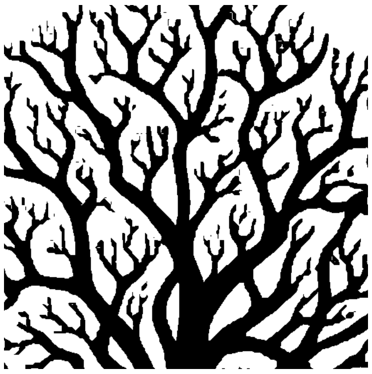
Figure 8.14: Objective metric values of distorted bilevel images coded with lossy compression methods.

ent compression algorithms. The corresponding subjective rating scores, objective rating scores and PE are shown below each image. Note that the object metric values and PE are after the non-linear transformation and are supposed to match the subjective rating scores. From this figure, one can observe several things. First, the objective metric values match the subjective rating scores significantly more than PE. Second, while according to the ground truth the FSA and LCC images have nearly the same similarity, the natures of their distortion are different. The FSA image appears noisy, while the LCC images appears to have incomplete structure. Finally, while the HC image is rated significantly higher than the LCC-CB image, the LCC-CB image actually looks quite good when viewed on its own. However, subjects did not rate it nearly as highly as the HC image because when viewed side-by-side with the original, differences can be easily seen in the LCC-CB image, but not


Original image: 512×512


0.24, 0.25, 0.31


0.25, 0.29, 0.34


0.40, 0.46, 0.42

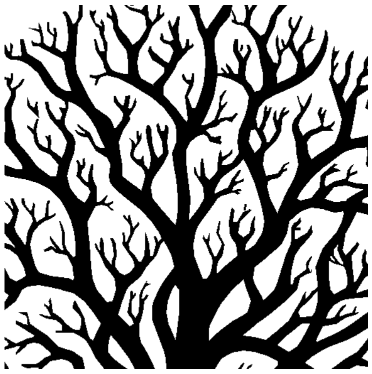

0.65, 0.55, 0.51

Figure 8.15: The 'tree' image coded at around 0.04 bpp with four different lossy compression algorithms. Starting from the second image: FSA, LCC, LCC-CB, HC. Numbers shown below each image are: subjective rating scores, transformed objective metric values and transformed percentage errors, from left to right.

in the HC image. We believe the differences mentioned above between the ground truth and objective metric values are partially caused by the specific form of the nonlinear transformation. They might be reduced by better transformations or by future improvements to objective bilevel image similarity metrics.

8.5.2 Comparing the Impact of Different Types of Distortion

One can also use the ground truth and objective similarity metrics to make judgments on the relative severity of the different types of man-made distortion that were introduced in the test images in the subjective experiment.

Figure 8.13 overlaps the subjective rating scores due to random bit flipping, dilation and

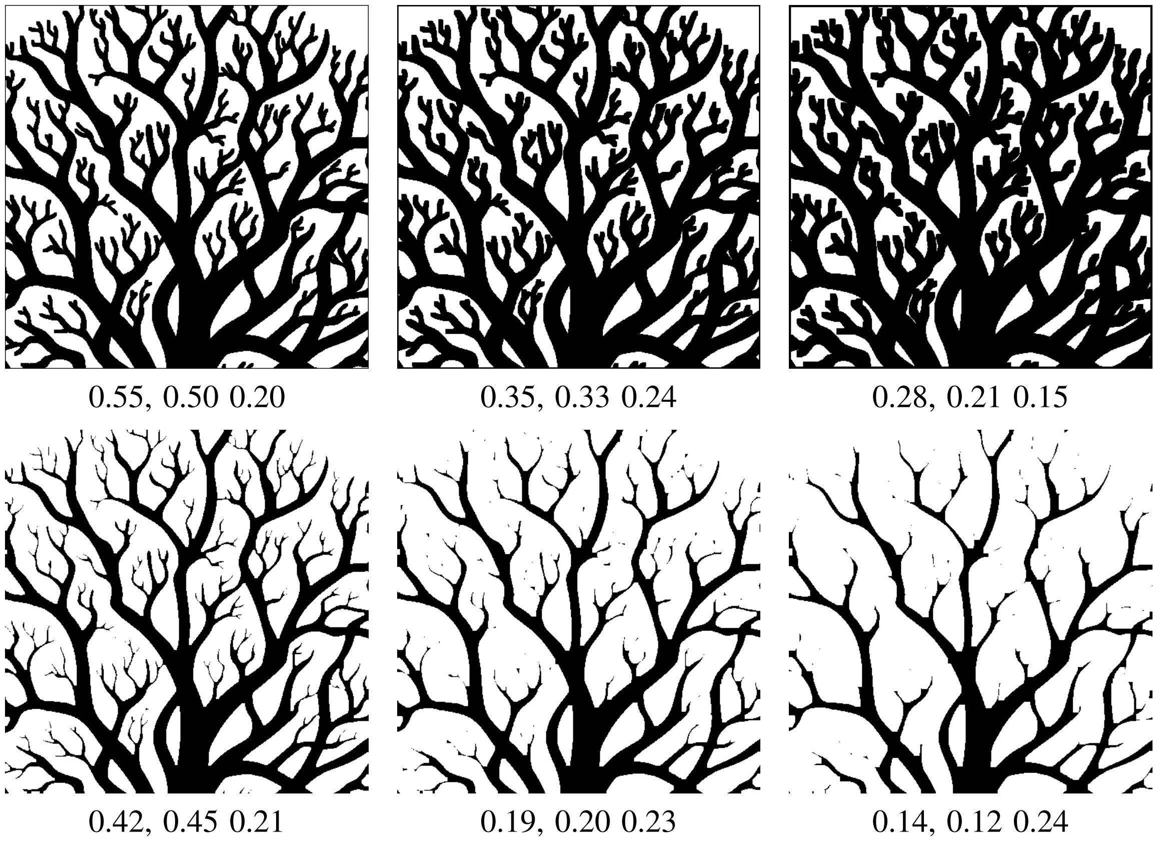


Figure 8.16: Dilation and erosion added to the ‘tree’ image. First row: dilation with 1, 2 and 3 iterations. Second row: erosion with 1, 2 and 3 iterations. Numbers shown below each image are: subjective rating scores, transformed objective metric values and transformed percentage errors, from left to right.

erosion with those due to the four compression algorithms. One can see that all three kinds of man-made distortions seriously impact image similarity, even at their lowest levels, *i.e.*, they give subjective rating scores of 0.55 or less. Random bit flipping with probability only 0.01 has a subjective rating score similar to HC with the lowest coding rate. Morphological transformations, *i.e.*, dilation and erosion, with two or more iterations have very low similarity based on human perception. Also note that there is a large gap between the scores for one and two iterations of the morphological transformations. The gap is illustrated visually in Fig. 8.16, where one sees that the second iteration of dilation or erosion has a larger visual effect than the first. Another interesting fact is that people are more tolerant of dilation than erosion, which suggests that people have lower tolerance to incomplete structures.

Similarly, in Fig. 8.14, the objective metric values due to the three types of distortion overlap with those due to the four compression algorithms. The objective metric values, subjective rating scores and PE's are also shown in Fig. 8.16. One can easily see that the objective metric values match the subjective rating scores much better than PE.

CHAPTER 9

Concluding Remarks

In this chapter, we first summarize the dissertation, and then suggest future work for LRI, MTC and bilevel image similarity metrics, respectively.

9.1 Summary

My PhD research focuses on perceptual image similarity metrics and their applications, *e.g.*, content-based image retrieval, perceptual image compression, image similarity assessment and texture analysis. The ultimate goal of this research area is to understand how the human visual system (HVS) processes visual signals and measures similarity. This is actually a multidisciplinary research topic involving signal and image processing, neurobiology, *etc*. As an image processor, my research focuses on designing objective similarity metrics that can be used to predict image similarity consistent with human perception. During the past few decades, image similarity metrics evolved from Mean-Squared Error (MSE) to Structural Similarity (SSIM) metrics [12], from pixel-by-pixel comparison to statistics extraction. This is a very interesting and challenging topic to work on, with many new findings and improvements going on every year.

Generally speaking, my past five years' research consists of two parts, one on grayscale images and the other on bilevel images. The first part is covered from Chapter 3 to Chapter 7; and the second part is presented in Chapter 8. As portion of my research on grayscale images, Chapter 7 describes a perceptual image compression algorithm, MTC, that I want to emphasize here.

The first part of my research aims at designing objective metrics to assess grayscale image similarity. In particular, texture similarity is given special attention. The goal is to develop objective texture similarity metrics that are consistent with human perception. A new family of statistical texture similarity features, called Local Radius Index (LRI), and the corresponding texture similarity metrics are proposed in Chapter 3. LRI extracts texture features using simple pixel value comparisons in the spatial domain and is designed for applications in which shifts can be tolerated but rotations should be penalized monotonically. Effective LRI-based metrics can be constructed by combining LRI with complementary texture features, *e.g.*, LBP, and two newly proposed features, Subband Contrast Distribution (SCD) and Intensity Penalty (IP). Compared to state-of-the-art metrics in the STSIM family, the LRI-based metrics achieve better texture retrieval performance with much less computation, as described in Chapter 3. When applied to the recently developed perceptual image coder, Matched Texture Coding (MTC), an LRI-based metric enables similar performance while significantly accelerating the encoding, as discussed in Chapter 7. In addition, in photographic paper classification, the LRI-based metrics also outperform existing metrics and achieve state-of-the-art performance, as presented in Chapter 6. We believe that LRI has a broad range of applications in practice and are still actively looking for new applications for LRI, such as climate prediction quality assessments.

To fulfill the needs of texture classification and other applications in which rotation invariance is desired, a rotation-invariant version of LRI, called RI-LRI, is proposed in Chapter 4. Although its name only emphasizes rotation invariance, the RI-LRI feature is also gray-scale and illuminance insensitive. From the experimental results in Chapter 4, we found that the corresponding texture similarity metric has comparable texture classification accuracy to state-of-the-art metrics, when tested on the Outex and CUReT databases. Moreover, it has a much lower dimensional feature vector and requires substantially less computation and storage space than other state-of-the-art texture features, such as those based on LBP.

The second part of my research focuses study of bilevel image similarity, including new objective metrics intended to quantify similarity consistent with human perception, and a subjective experiment to obtain ground truth for judging the performance of the objective similarity metrics. As mentioned previously, this part of work is presented in Chapter 8. The focus is on scenic bilevel images, which are complex, natural or hand-drawn images, such as landscapes or portraits, but are not halftoned.

In order to provide ground truth to assess the performance of objective similarity metrics, as well as for subjectively comparing various image processing systems, such as compression, we conduct a subjective experiment to obtain similarity rating scores for a collection of distorted scenic bilevel images. In this subjective experiment, seven scenic images are each distorted in forty-four ways, including random bit flipping, dilation, erosion and lossy compression. To produce subjective rating scores, the distorted images are each viewed by 77 subjects. The ratings from each subject are normalized. Several screening tests are applied to rule out subjects whose ratings are not sufficiently good. The normalized ratings are then analyzed on the basis of rating time, contextual effects and standard deviation. The result is a set of 264 subjectively rated pairs of images to use as ground truth to compare the performance of four compression algorithms, to assess the severity of the various kinds of distortion, and to provide ground truth to assess how well various objective metrics measure bilevel image similarity. The original and distorted images used in the subjective experiments, along with the subjective rating data obtained can be found in the "Bilevel Image Similarity Ground Truth Archive" at University of Michigan Deep Blue[1].

Based on hypotheses about human perception of bilevel images, we propose several objective bilevel image similarity metrics that outperform existing ones in the sense of

[1]http://deepblue.lib.umich.edu/handle/2027.42/111059.

attaining significantly higher Pearson and Spearman-rank correlation coefficients with respect to the ground truth attained in the subjective experiment. The new metrics include Adjusted Percentage Error (APE), Bilevel Gradient Histogram (GH), Connected Components Comparison (CC) and combinations of such. The best performance is achieved by the combination of APE and GH, attaining Pearson and Spearman-rank correlation coefficients as high as 0.95 and 0.94, respectively. We anticipate the proposed similarity metrics will be useful in a number of applications, either to judge the performance of lossy bilevel image compression methods, or to be used as part of the system, as is needed in a retrieval application or image segmentation quality assessments.

Within the first part of my research, Chapter 7 describes the development of the perceptual image compression algorithm mentioned above, MTC, which achieves better tradeoff between coding rate and reconstruction quality than other compression methods, when the image being compressed contains a substantial amount of texture. MTC is a block-based image coder that uses texture similarity metrics, *e.g.*, LRI and STSIM2, to decide if blocks of the image can be encoded by pointing to structurally similar ones in the already coded region. The key to its success is an effective texture similarity metric, such as an LRI-based metric, and an effective search strategy. Compared to traditional image compression algorithms, *e.g.*, JPEG, MTC achieves similar coding rate with higher reconstruction quality. And the advantage of MTC becomes larger as coding rate decreases.

9.2 Suggested Future Work

9.2.1 Future Work to Improve LRI

Texture similarity metrics, like LRI, aim at measuring texture similarity consistent with human perception. Although LRI has shown to be successful in many image processing fields, *e.g.*, Chapter 6 and Chapter 7, there are still many aspects in which it can be improved. The following list shows the ultimate goals of an ideal texture similarity metric.

We believe ideal texture similarity metrics should

i. assess image similarity in a manner consistent with human perception.

ii. be appropriate for use in a variety of applications and should enable such applications to perform better.

iii. have robust metric performance over a wide range of image pairs, *i.e.*, its values should make sense for a wide range of image pairs.

iv. give values monotonically related to perceived similarity when perceived (Our view is that quantifying really bad distortion is not needed, nor even feasible, not even by humans.)

v. permit a scale factor to be input so users can indicate the maximum scale of interest. (Indeed the similarity of two images might be expressed as a similarity vs. scale curve.)

vi. be the simplest possible for a given pair of images. (To reduce computational complexity and to maximize robustness.)

vii. adapt to its characteristics, *e.g.*, smooth, texture, or edge, when applied to an (relatively small) image patch.

viii. adapt to different image regions, as needed (but the adaptation should be relatively slow), when applied to an entire image (or a very large image patch).

ix. asymmetrically focus more on the original image than the reproduction.

We suggest that future researchers could focus on this list to further improve LRI, or a new family of texture similarity metric.

9.2.2 Future Work to Improve MTC

Although experiments have proven the effectiveness of the MTC algorithm, there still exist some limitations. One of the main drawback is the computational complexity. Compared to JPEG, MTC is too complicated to be applied in practice. Although it may not be our main concern for now, this problem should be solved in the future. There are two possible solutions to reduce the computational complexity:

i. Design better search strategy to find good candidates more efficiently and effectively. Now the 3-stage search strategy is doing a much better job than a brute force search. However, we believe better search strategy can still be developed. The new search strategy should balance the efficiency, *i.e.*, computational complexity, and the effectiveness, *i.e.*, frequency to find a perceptually similar candidate block for a given target block, given the existence of such a candidate block.

ii. Reduce the complexity of texture similarity metrics without sacrificing compression performance. STSIM2, as mentioned many times, is computationally expensive. Compared to STSIM2, our newly developed LRI$^+$ is at least as good, while substantially accelerates the encoding process, as showed in Chapter 7. With future work to further improve LRI$^+$, better performance or lower computational complexity can be expected.

Besides computational complexity, sometimes MTC-encoded blocks tend to mess up the periodicity of some fine texture or pattern. This problem could be serious if certain noticeable edges or texture boundaries are badly predicted from similar, but not identical, candidate blocks. For example, under careful examination, such problems can be seen from the sweater area in the “woman” image in Fig. 7.14 and the broken window in the “house” image in Fig. 7.16. In some other cases, small objects inside a large homogeneous textured region could be missed by inaccurate texture matching. One example would be the missing floats in the “waterfall” image in Fig. 7.17.

These problems occur because texture similarity metrics are not perfect. There is a tradeoff between the frequency of successful texture matching and the overall matching quality. If we set the threshold of a texture similarity metric too loose, many blocks will be MTC-encoded, even when the candidate blocks are not sufficiently similar, and then become part of the "codebook" to encode the rest of the image. However, such low-quality MTC-encoded blocks will lower the overall quality of the "codebook". Hence the compressed image will also suffer from low quality. On the contrary, if the threshold is set to be too strict, most of the blocks will be coded by the baseline coder instead of texture matching, and accordingly, the coding rate will increase. In this case, what we have is actually a compression algorithm much slower than the baseline coder, while yielding very similar compression results. A better texture similarity metric is needed to improve the tradeoff mentioned above. The new metric should have acceptable complexity, and meanwhile be more consistent with human perception.

9.2.3 Future Work to Improve Objective Bilevel Similarity Metrics

The ultimate goal of this research is to find interpretations for each similarity metric that can fit the ground truth well. Such interpretations include the relationship between objective similarity metrics and the human visual system (HVS). Also, each developed metric should be tested under different subsets of the ground truth to find its best working domain. For example, metric M_1 may be suitable to measure bilevel image similarity under certain compression algorithm or certain kind of distortion. Meanwhile, metric M_2 may work well for compressed bilevel images with high (or low) coding rates. Such study can help provide intuition to explain the performance for different bilevel similarity metrics. And correspondingly it may also help us understand human perception of bilevel image similarity.

Another research topic is the study of valid criteria for evaluating image similarity metrics. Pearson correlation coefficient is used when a linear relationship between the objective metric values and the subjective rating scores is needed. And Spearman-rank correlation coefficient is applied when a monotonic relationship is important. Both of them are good criteria to assess the performance of image similarity metrics. However, as can be seen from Fig. 8.11, the standard deviation of the subjective rating scores varies at different locations on the similarity scale. At the high end and low end of the similarity scale, human observers are more consistent. However, in the middle of the similarity scale, people tend to disagree with each other, resulting in high standard deviations of the rating scores. Hence, we should allow objective similarity metrics to have a similar behavior as human beings. An ideal criterion should be stricter at the high end and low and of the similarity scale and looser in between. The design of such criterion could rely on the subjective rating scores obtained in the experiment described in Chapter 8.

BIBLIOGRAPHY

[1] J. Portilla and E. P. Simoncelli, “A parametric texture model based on joint statistics of complex wavelet coefficients,” *Int. J. Comput. Vis.* , 40: 49-70, Oct. 2000.

[2] T.N. Pappas, “An adaptive clustering algorithm for image segmentation,” *IEEE Tr. Sig. Proc.*, vol. 40, no. 4, pp. 901-914, Apr. 1992.

[3] J. Zujovic, T. N. Pappas and D. L. Neuhoff, “Structural texture similarity metrics for image analysis and retrieval,” *IEEE Transaction on Image Processing*, vol. 22, no. 7, pp. 2545-2558, 2013.

[4] J. Zujovic, T. N. Pappas and D. L. Neuhoff, “Perceptual similarity metrics for retrieval of natural textures,” *IEEE Int. Workshop Multimedia Sig. Proc. (MMSP)*, Rio De Janeiro, pp. 1-5, Oct. 2009.

[5] T. N. Pappas, D. L. Neuhoff, H. de Ridder and J. Zujovic, “Image analysis: focus on texture similarity,” invited, *Proc. IEEE*, vol. 101, pp. 2044-2057, Sep. 2013.

[6] R. M. Haralick, K. Shanmugam and I. Dinstein, “Textural features for image classification,” *IEEE Trans. Syst. , Man, Cybern.* , vol. 3, no. 6, pp. 610-621, 1973.

[7] S. K. Saha, A. K. Das and B. Chanda, “CBIR using perception based texture and colour measures,” *Proc. 17th Int. Conf. Pattern Recognition (ICPR)*, vol. 2, pp. 985-988, 2004.

[8] T. Mita, T. Kaneko, B. Stenger and O. Hori, “Discriminative feature co-occurrence selection for object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1257-1269, 2008.

[9] *The Handbook of Pattern Recognition and Computer Vision (2nd Edition),* (C. H. Chen, L. F. Pau, P. S. P. Wang, eds.), World Scientific Publishing Co. , 1998.

[10] R. L. Kashyap, R. Chellappa and A. Khotanzad, "Texture classification using features derived from random field models," *Pattern Recognition Letters*, vol. 1, no. 1, pp. 43-50, 1982.

[11] B. S. Manjunath and R. Chellappa, "Unsupervised texture segmentation using markov random field models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 478-482, 1991.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transaction on Image Processing*, vol. 13, pp. 600-612, Apr. 2004.

[13] Z. Wang and E. P. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. II, pp. 573-576, 2005.

[14] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik and M. K. Markey, "Complex wavelet structural similarity: a new image similarity index," *IEEE Transaction on Image Processing*, vol. 18, pp. 2385-2401, Nov. 2009.

[15] J. Zujovic, *Perceptual Texture Similarity Metrics*, Ph.D. Thesis, Northwestern University, 2011.

[16] X. Zhao, M. G. Reyes, T. N. Pappas and D. L. Neuhoff, "Structural texture similarity metrics for retrieval applications," *IEEE International Conference on Image Processing (ICIP)*, pp. 1196-1199, Oct. 2008.

[17] J. Zujovic, T. N. Pappas and D. L. Neuhoff, "Structural similarity metrics for texture analysis and retrieval," *IEEE International Conference on Image Processing (ICIP)*, pp. 2225-2228, Nov. 2009.

[18] G. Jin, Y. Zhai, T. N. Pappas and D. L. Neuhoff, "Matched-texture coding for structurally lossless compression," *IEEE International Conference on Image Processing (ICIP)*, pp. 1065-1068, Oct. 2012.

[19] N. Ahuja, "Dot Pattern Processing Using Voronoi Neighborhoods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4, pp. 336-343, 1982.

[20] M. Tuceryan and A. K. Jain, "Texture Segmentation Using Voronoi Polygons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12, pp. 211-216, 1990.

[21] S. W. Zucker, "Toward a model of Texture," *Computer Graphics and Image Processing*, vol. 5, pp. 190-202, 1976.

[22] H. Voorhees and T. Poggio, "Detecting textons and texture boundaries in natural images," *Proceedings of the First International Conference on Computer Vision*, pp. 250-258, 1987.

[23] D. Blostein and N. Ahuja, "Shape from Texture: Integrating Texture-Element Extraction and Surface Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11, pp. 1233-1251, 1989.

[24] S. Lazebnik, C. Schmid and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1265-1278, Aug. 2005.

[25] M. Mellor, B. -W. Hong and M. Brady, "Locally rotation, contrast, and scale invariant descriptors for texture analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 52-61, Jan. 2008.

[26] G. -S. Xia, J. Delon and Y. Gousseau, "Shape-based invariant texture indexing," *Int. J. Comput. Vis.* , DOI 10.1007/s11263-009-0312, Nov. 2009.

[27] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolu-tion gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 971-987, Jul. 2002.

[28] S. Liao, M. Law and A. Chung, "Dominant local binary patterns for texture classification," *IEEE Transaction on Image Processing*, vol. 18, no. 5, pp. 1107-1118, May 2009.

[29] Z. Guo, L. Zhang and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Transaction on Image Processing*, vol. 19, no. 6, pp. 1657-1663, Jun. 2010.

[30] F. M. Khellah, "Texture classification using dominant neighborhood structure," *IEEE Transaction on Image Processing*, vol. 20, no. 11, pp. 3270-3279, Nov. 2011.

[31] T. Song, F. Meng, B. Luo and C. Huang, "Robust texture representation by using binary code ensemble," *Visual Communications and Image Processing (VCIP)*, pp. 1-6, 2013.

[32] J. Zujovic, T. N. Pappas, D. L. Neuhoff, R. van Egmond, and H. de Ridder, "Subjective and objective texture similarity for image compression," *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1369-1372, Mar. 2012.

[33] J. Zujovic, T. N. Pappas, D. L. Neuhoff, R. van Egmond and H. de Ridder, "A new subjective procedure for evaluation and development of texture similarity metrics," *Proc. IEEE 10th IVMSP Workshop*, pp. 123-128, June 2011.

[34] Y. Zhai, D. L. Neuhoff and T. N. Pappas, "Local radius index - a new texture similarity feature," *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1434-1438, May 2013.

[35] Y. Zhai and D. L. Neuhoff, "Rotation-invariant local radius index - a compact texture similarity feature for classification," *IEEE International Conference on Image Processing (ICIP)*, pp. 5711-5715, Oct. 2014.

[36] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen and S. Huovinen, "Outex - new framework for empirical evaluation of texture analysis algorithms," *Proc. ICPR*, pp. 701-706, 2002.

[37] K. J. Dana, B. Van-Ginneken, S. K. Nayar, J. J. Koenderink, "Reflectance and texture of real world surfaces," *ACM Transactions on Graphics (TOG)*, vol. 18, no. 1, pp. 1-34, Jan. 1999.

[38] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *Int. J. Comp. Vision*, vol. 62, no. 1-2, pp. 61-81, 2005.

[39] M. Varma and A. Zisserman, "A statistical approach to material classification using image patch exemplars," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2032-2047, Nov. 2009.

[40] Z. Hubalek, "Coefficients of association and similarity, based on binary (presence-absence) data: An evaluation," *Biol. Rev.* , vol. 57, no. 4, pp. 669-689, 1982.

[41] G.R. Shi, "Multivariate data analysis in palaeoecology and palaeobiogeography - a review," *Palaeogeogr. , Palaeoclimatol. , Palaeoecol.* , vol. 105, no. 3-4, pp. 199-234, Nov. 1993.

[42] P. Jaccard, "The distribution of flora in the Alpine zone," *New Phytol.* , vol. 11, pp. 37-50, 1912.

[43] S. Kulczynski, “Zespoly rslin w pieninach,” *Bull. Int. Acad. Pol. Sci. Let-ters*, vol. 2, pp. 57-203, 1928.

[44] J. Braun-Blanquet, “Plant Sociology: The Study of Plant Communities,” New York: McGraw Hill, 1932.

[45] L.R. Dice, “Measures of the amount of ecologic association between species,” *Ecology*, vol. 26, no. 3, pp. 297-302, 1945.

[46] A. Ochiai, “Zoogeographic studies on the soleoid fishes found in Japan and its neighbouring regions,” *Bull. Jpn. Soc. Sci. Fish*, vol. 22, pp. 526-530, 1957.

[47] S.A. Forbes, “On the local distribution of certain Illinois fishes: an essay in statistical ecology,” *Bulletin of the Illinois State Laboratory for Natural History*, vol. 7, pp. 273-303, 1907.

[48] A. Zijdenbos, B. Dawant, R. Margolin and A. Palmer, “Morphometric analysis of white matter lesions in MR images: Method and validation,” *IEEE Trans. Med. Imag.* , vol. 13, no. 4, pp. 716-724, Apr. 1994.

[49] K. Zou *et al.* , “Statistical validation of image segmentation quality based on a spatial overlap index,” *Acad. Radiol.* , vol. 11, no. 2, pp. 178-189, Feb. 2004.

[50] M. Reyes, X. Zhao, D. L. Neuhoff and T. N. Pappas, “Structure-preserving properties of bilevel Image compression,” *Human Vision Electr. Im. XIII*, *Proc. SPIE*, vol. 6806, pp. 680617-1-12, Jan. 2008.

[51] ITU-R BT. 500-11, “Methodology for the subjective assessment of the quality of television pictures,” *International Telecommunication Union*, 2002.

[52] VQEG, “Final report from the video quality experts group on the validation of objective models of video quality assessment, phase II,” [Online]. Available: *http://www.vqeg.org*, Aug. 2003.

[53] H. Sheikh, M. Sabir and A. Bovik, “A statistical evaluaIon of recent full reference image quality assessment algorithms,” *IEEE Transaction on Image Processing*, vol. 15, pp. 3440-3451, Nov. 2006.

[54] Y. Zhai, D. L. Neuhoff and T. N. Pappas, “Measuring texture similarity for compression, retrieval and classification with Local Radius Index,” submitted to *IEEE Transaction on Image Processing*, 2015.

[55] Y. Zhai and D.L. Neuhoff, “Distributions of Local Radius Indices on periodic tessellations,” accepted by *IEEE International Conference on Image Processing (ICIP)*, Sep. 2015.

[56] Y. Zhai and D.L. Neuhoff, “Photographic paper classification via Local Radius Index metric,” accepted by *IEEE International Conference on Image Processing (ICIP)*, Sep. 2015.

[57] Y. Zhai, D. L. Neuhoff and T. N. Pappas, “Subjective similarity evaluation for scenic bilevel images,” *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 156-160, May 2014.

[58] Y. Zhai and D. L. Neuhoff, “Objective similarity metrics for scenic bilevel images,” *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2793-2797, May 2014.

[59] Y. Zhai and D. L. Neuhoff, “Similarity of Scenic Bilevel Images,” submitted to *IEEE Transaction on Image Processing*, 2015.

[60] J. L. Mannos and D. J. Sakrison, “The effects of a visual fidelity criterion on the encoding of images,” *IEEE Trans. Information Theory*, vol. 4, pp. 525-536, 1974.

[61] A. M. Eskicioglu and P. S. Fisher, “Image Quality Measures and Their Performance,” *IEEE Transactions on Communications*, 43(12): 2959-2965, Dec. 1995.

[62] T. N. Pappas and R. J. Safranek, “Perceptual criteria for image quality evaluation,” *Handbook of Image and Video Proc.* , (A. Bovik, ed.), Academic Press, 2000.

[63] A. B. Watson, “The cortex transform: Rapid computation of simulated neural images,” *Computer Vision, Graphics, and Image Processing*, vol. 39, pp. 311-327, 1987.

[64] P. J. Burt and E. H. Adelson, “The Laplacian pyramid as a compact image code,” *IEEE Trans. Comm.* , vol. 31, pp. 532-540, 1983.

[65] W. T. Freeman and E. H. Adelson, “The design and use of steerable filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 891-906, Sept. 1991.

[66] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, “Shiftable multiscale transforms,” *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 587-607, Mar. 1992.

[67] S. Daly, “The visible differences predictor: an algorithm for the assessment of image fidelity,” *Digital Images and Human Vision*, (A. B. Watson, ed.), pp. 179-206, Cambridge, MA: the MIT Press, 1993.

[68] J. Lubin, “The use of psychophysical data and models in the analysis of display system performance,” *Digital Images and Human Vision*, (A. B. Watson, ed.), pp. 163-178, Cambridge, MA: the MIT Press, 1993.

[69] P. C. Teo and D. J. Heeger, “Perceptual image distortion,” *IEEE International Conference on Image Processing (ICIP)*, vol. II, pp. 982-986, Nov. 1994.

[70] A. B. Watson, “DCT quantization matrices visually optimized for individual images,” *Proc. SPIE*, vol. 1913, pp. 202-216, 1993.

[71] A. B. Watson, J. Hu and J. F. III. McGowan, “DVQ: A digital video quality metric based on human vision,” *Journal of Electronic Imaging*, vol. 10, no. 1, pp. 20-29, 2001.

[72] A. B. Watson, G. Y. Yang, J. A. Solomon and J. Villasenor, “Visibility of wavelet quantization noise,” *IEEE Transaction on Image Processing*, vol. 6, pp. 1164-1175, 1997.

[73] A. P. Bradley, “A wavelet visible difference predictor,” *IEEE Transaction on Image Processing*, vol. 5, pp. 717-730, May 1999.

[74] Y. K. Lai and C. -C. J. Kuo, “A Haar wavelet approach to compressed image quality measurement,” *Journal of Visual Communication and Image Representation*, vol. 11, pp. 17-40, Mar. 2000.

[75] M. P. Eckert and A. P. Bradley, “Perceptual quality metrics applied to still image compression,” *Signal Processing*, vol. 70, pp. 177-200, 1998.

[76] P. Chen and T. Pavlidis, “Segmentation by texture using correlation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 64-69, 1983.

[77] B. Julesz, E. Gilbert and J. Victor, “Visual discrimination of textures with identical third-order statistics,” *Biological Cybernetics*, vol. 31, no. 3, pp. 137-140, 1978.

[78] B. Julesz, “A theory of preattentive texture discrimination based on first-order statistics of textons,” *Biological Cybernetics*, vol. 41, no. 2, pp. 131-138, 1981.

[79] M. Varma and A. Zisserman, “Texture classification: are filter banks necessary?,” *Proc. IEEE Conf. Comp. Vision Pattern Recogn. (CVPR)*, vol. 2, pp. 691-698, 2003.

[80] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *J. Optical Soc. America A*, vol. 2, pp. 1160-1169, 1985.

[81] J. P. Jones and L. A. Palmer, "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex," *J. Neurophysiology*, vol. 58, no. 6, pp. 1233-1258, Dec. 1987.

[82] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.* , vol. 22, pp. 79-86, Mar. 1951.

[83] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for Distributions with applications to image databases," *IEEE International Conference on Computer Vision*, pp. 59-66, Jan. 1998.

[84] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *Int. J. Comput. Vision*, vol. 62, no. 1-2, pp. 61-81, Apr. 2005.

[85] R. Sokal and C. Michener, "A statistical method for evaluating systematic relationships," *Univ. Kansas Sci. Bull.* , vol. 38, pp. 1409-1438, 1958.

[86] G. Simpson, "Notes on the measurement of faunal resemblance," *Amer. J. Sci.* , vol. 258, pp. 300-311, 1960.

[87] D. Rogers and T. Tanimoto, "A computer program for classifying plants," *Science*, vol. 132, pp. 1115-1118, 1960.

[88] R. Sokal and P. Sneath, "Principles of Numerical Taxonomy," San Francisco, CA: W. H. Freeman, 1963.

[89] J. Zujovic, T. N. Pappas, D. L. Neuhoff, R. van Egmond and H. de Ridder, "Effective and efficient subjective testing of texture similarity metrics," *Journal of the Optical Society of America A* , vol. 32, no. 2, pp. 329-342, Feb. 2015.

[90] T. N. Pappas, J. Zujovic and D. L. Neuhoff, "Image analysis and compression: renewed focus on texture," *Vis. Inf. Proc. Comm.* , Proc. SPIE vol. 7543, pp. 75430N-1-12, Jan. 2010.

[91] J.F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, pp. 679-698, 1986.

[92] S.J. Daly, "Visible differences predictor: an algorithm for the assessment of image fidelity," *Proc. SPIE 1666, Human Vision, Visual Processing, and Digital Display III*, pp. 2-15, Aug. 1992.

[93] R. J. Safranek and J. D. Johnston, "A perceptually tuned sub-band image coder with image dependent quantization and post-quantization data compression," *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1945-1948, vol. 3, 1989.

[94] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comput.* , 19: 297-301, 1965.

[95] "Corbis stock photography," http://www.corbis.com.

[96] E. M. Voorhees, "The trec-8 question answering track report," *Proc. of TREC-8*, pp. 77-82, 1999.

[97] E. M. Voorhees, "Variations in relevance judgments and the measurement of retrieval effectiveness," *Inform. Process. Manag.* , vol. 36, pp. 697-716, Sept. 2000.

[98] M. Maggioni, G. Jin, A. Foi and T. N. Pappas, "Structural texture similarity metric based on intra-class variances," *IEEE International Conference on Image Processing (ICIP)*, pp. 1992-1996, Oct. 2014.

[99] P. Messier, C. R. Johnson, H. Wilhelm, W. A. Sethares, A. G. Klein, P. Abry, S. Jaffard, H. Wendt, S. G. Roux, N. Pustelnik, N. van Noord, L. van der Maaten and E. Postma, "Automated surface texture classification of inkjet and photographic media," *the 29th International Conference on Digital Printing Technologies*, pp. 85-91, IS&T: The Society for Imaging Science and Technology, 2013.

[100] C. R. Johnson, P. Messier, W. A. Sethares, A. G. Klein, C. Brown, P. Klausmeyer, P. Abry, S. Jaffard, H. Wendt, S. G. Roux, N. Pustelnik, N. van Noord, L. van der Maaten, E. Postma, J. Coddington, L. A. Daffner, H. Murata, H. Wilhelm, S. Wod and M. Messier, "Pursuing automated classification of historic photographic papers from raking light photomicrographs," *Journal of the American Institute for Conservation*, vol. 53, no. 3, pp. 159-170, Aug. 2014.

[101] W. A. Sethares, A. Ingle, T. Krc and S. Wood, "Eigentextures: an SVD approach to automated paper classification," *Proc. Asilomar Conf. on Signals, Systems, and Computers*, pp. 1109-1113, Nov. 2014.

[102] S. Roux, P. Abry, H. Wendt and S. Jaffard, “Hyperbolic wavelet transform for historic photographic paper classification challenge,” *Proc. Asilomar Conf. on Signals, Systems, and Computers*, pp. 1119-1123, Nov. 2014.

[103] A. G. Klein, A. Do, C. A. Brown and P. Klausmeyer, “Texture classification via area-scale analysis of raking light images,” *Proc. Asilomar Conf. on Signals, Systems, and Computers*, pp. 1114-1118, Nov. 2014.

[104] D. Picard, N. Vu and I. Fijalkow, “Photographic paper texture classification using model deviation of local visual descriptors,” *IEEE International Conference on Image Processing (ICIP)*, pp. 5701-5705, Oct. 2014.

[105] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” *SIGGRAPH-01*, pp. 341346, 2001.

[106] G.J̇. Sullivan, J. -R. Ohm, W. -J. Han and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) standard”, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 22, No. 12, pp. 1649-1668, Dec. 2012.

[107] C. - Y. Teng and D. L. Neuhoff, “A new quadtree predictive image coder,” *IEEE International Conference on Image Processing (ICIP)*, pp. II73-II76, Oct. 1995.

[108] K. M. Holt and D. L. Neuhoff, “Strategies for quadtree predictive image coding,” *IEEE International Conference on Image Processing (ICIP)*, vol. II, pp. 247-250, Oct. 2003.

[109] A. Rose, “The sensitivity performance of the human eye on an absolute scale,” *Journal of the Optical Society of America*, vol. 38, pp. 196-208, Feb. 1948.

[110] L. I, Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: Nonlinear Phenomena*, vol. 60, issues 1-4, pp. 259 - 268, Nov. 1992.

[111] Fairchild, Mark, *Color Appearance Models*. Reading, Mass.: Addison, Wesley, & Longman, ISBN 0-201-63464-3, 1998.

[112] L. Zhang, L. Zhang and X. Mou, “RFSIM: a feature based image quality assessment metric using Riesz transforms,” *IEEE International Conference on Image Processing (ICIP)*, pp. 321–324, 2010.

[113] L. Zhang, D. Zhang, X. Mou and D. Zhang, “FSIM: a feature similarity index for image quality assessment,” *IEEE Transaction on Image Processing*, vol. 20, pp. 2378–2386, 2011.

[114] M. G. Reyes, X. Zhao, D. L. Neuhoff and T. N. Pappas, “Lossy compression of bilevel images based on Markov random fields,” *IEEE International Conference on Image Processing (ICIP)*, pp. II-373-II-376, 2007.

[115] M. G. Reyes, D. L. Neuhoff and T. N. Pappas, “Lossy cutset coding of bilevel images based on Markov random fields,” *IEEE Transaction on Image Processing*, vol. 23, no. 4, pp. 1652-1665, 2014.

[116] K. Culik, V. Valenta and J. Kari, “Compression of silhouette-like images based on WFA,” *Journal of Universal Computer Science*, 3(10):1100-1113, 1997.

[117] S. Zha, T. N. Pappas and D. L. Neuhoff, “Hierarchical bilevel image compression based on cutset sampling,” *IEEE International Conference on Image Processing (ICIP)*, pp. 2517-2520, 2012.

[118] D. Cramer, D. L. Howitt, “The SAGE Dictionary of Statistics: A Practical Resource for Students in the Social Sciences (Third ed.)”, 2005.