Improved Subband-Based and Normal-Mesh-Based Image Coding

by

Di Xu
B.S., Beijing Normal University, China, 2003

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Improved Subband-Based and Normal-Mesh-Based Image Coding

by

Di Xu
B.S., Beijing Normal University, China, 2003

Supervisory Committee

Dr. Michael D. Adams, (Department of Electrical and Computer Engineering)

Supervisor

Dr. Wu-Sheng Lu, (Department of Electrical and Computer Engineering)

Departmental Member

Dr. Reinhard Illner, (Department of Mathematics and Statistics)

Outside Member

Supervisory Committee

Dr. Michael D. Adams, (Department of Electrical and Computer Engineering)

Supervisor

Dr. Wu-Sheng Lu, (Department of Electrical and Computer Engineering)

Departmental Member

Dr. Reinhard Illner, (Department of Mathematics and Statistics)

Outside Member

ABSTRACT

Image coding is studied, with the work consisting of two distinct parts. Each part focuses on different coding paradigm.

The first part of the research examines subband coding of images. An optimization-based method for the design of high-performance separable filter banks for image coding is proposed. This method yields linear-phase perfect-reconstruction systems with high coding gain, good frequency selectivity, and certain prescribed vanishing-moment properties. Several filter banks designed with the proposed method are presented and shown to work extremely well for image coding, outperforming the well-known 9/7 filter bank (from the JPEG-2000 standard) in most cases. Several families of perfect reconstruction filter banks exist, where the filter banks in each family have some common structural properties. New filter banks in each family are designed with the proposed method. Experimental results show that these new filter banks outperform previously known filter banks from the same family.

The second part of the research explores normal meshes as a tool for image coding, with a particular interest in the normal-mesh-based image coder of Jansen, Baraniuk, and Lavu. Three modifications to this coder are proposed, namely, the use of a data-dependent base mesh, an alternative representation for normal/vertical offsets, and a different scan-conversion scheme based on bicubic interpolation. Experimental results show that our proposed changes lead to improved coding performance in terms of both objective and subjective image quality measures.

Contents

List of Tables

List of Figures

List of Acronyms

1-D one-dimensional

2-D two-dimensional

3-D three-dimensional

CDT constrained Delaunay triangulation

CR compression ratio

DCT discrete cosine transform

DT Delaunay triangulation

EZW embedded zerotree wavelet

FIR finite-length impulse response

HVS human visual system

IIR infinite-length impulse response

ISO International Organization for Standardization

JPEG Joint Photographic Experts Group

MRA multiresolution approximation

MSE mean squared error

PR perfect reconstruction

PSLG planar straight line graph

PSNR peak-signal-to-noise ratio

SOCP second-order cone programming

SPIHT set partitioning in hierarchical trees

UMD uniformly maximally decimated

Acknowledgments

This thesis would never have been done without the help and support from numerous people. Needless to say, I thank all of them from the bottom of my heart. I would like to take this opportunity to express my thankfulness to the following individuals.

First and foremost, I would like to express my sincerest gratitude towards my supervisor Dr. Michael Adams for his support and guidance throughout my studies. Michael led me to the wonderful world of digital signal processing, and allowed me wide freedom in choosing my thesis topic. Those have yielded agreeable study experience for me. He has always been tireless and meticulous when helping me to improve my technical writing skills. I also thank him for being not only a good supervisor but also a great friend.

Many thanks also go to my marvellous professor, Dr. Wu-Sheng Lu. I was very lucky to have the opportunities to take his digital signal processing (III) and optimization (I) courses. Being amazed by his knowledge and enthusiasm in teaching, I then had a chance to audit his optimization (II) course. Through my master studies, Dr. Lu has always provided me many valuable suggestions and insights, and expanded my scope of thought even during his sick leave. I sincerely wish him a full and speedy recovery.

Next, I would like to thank my friends Yi Chen and Ana-Maria Sevcenco for their friendship and help in my research. I would also like to express my gratitude to the following individuals, who have kindly helped me during my master studies: Dr. Reinhard Illner, Dr. Pan Agathoklis, Dr. Aaron Gulliver, Dr. Dale Olesky, Catherine Chang, Lynne Barrett, Vicky Smith, Mary-Anne Teo, Moneca Bracken, Erik Laxdal, Shirley Shi, Behzad Bahr-Hosseini, Akshay Rathore, Dhaval Shah, Ping Wan, and Ze Dong.

Lastly, I would like to thank the Natural Sciences and Engineering Research Council of Canada and the University of Victoria for providing financial support in the form of a research grant and a university fellowship, respectively. The support definitely helped me resolve the financial burden pertaining to my studies.

To my family, whom I love and who loves me

Chapter 1

Introduction

1.1 Image Coding

Image coding seeks to find more efficient representations of images so that the data size in bits is minimized while an acceptable visual quality (i.e., minimal/no distortion) is maintained. Since there is often a correlation between adjacent pixels of an image, image coding allows us to compress the image data by reducing this spatial redundancy in order to store or transmit the information in an efficient manner. Image coding is very useful, since uncompressed images require considerable storage capacity and transmission bandwidth. Despite rapid progress in mass storage and digital communications techniques, the available storage and bandwidth resources are still far from enough to meet the increasing heavy demand of applications, which makes the development of efficient image coding systems essential.

1.2 Historical Perspective

Numerous image-coding techniques have been developed over the past sixty years. One of the most popular transforms used for image coding is the discrete cosine transform (DCT) [31]. A commonly used image coding standard created by the Joint Photographic Experts Group, namely the JPEG standard, is a DCT-based transform coding system. Although JPEG is simple and can be easily implemented in hardware, it suffers, at low bit rates, from annoying blocking artifacts, due to its use of a block transform. To overcome this problem, several improved coding systems have been proposed, such as ones using the lapped orthogonal transform [27], the down-scaling scheme [7], the variable projection approach [42], and the combined adap-

tive and averaging strategies [35]. Although the blocking artifacts are reduced, the increased computational complexity and the relative performance of the improved coding systems make such coders less competitive than some other coding systems.

Subband coding has been widely adopted in many of today's best image coders, such as JPEG 2000 [20]. In numerous applications, subband coding outperforms other image-coding systems including DCT-based image coders. In addition, subband image coding offers multiresolution representations [26], which readily facilitate resolution scalability. Despite JPEG-2000's many advantages, the 9/7-J filter bank used in JPEG 2000 is designed by a spectral factorization scheme [11], and is not necessarily optimal in terms of energy compaction (e.g., high coding gain). Although the 9/7-J filter bank is known to have high coding gain, there is still room for the filter bank to be improved. Therefore, an optimal filter bank having high coding gain, linear phase, perfect reconstruction (PR), good frequency selectivity, and a prescribed number of vanishing moments is highly desirable, as such a filter bank would likely offer better performance than the 9/7-J filter bank used in JPEG 2000.

Many subband coding systems are based on wavelet transforms. The wavelet transform is good at representing point singularities as well as horizontal/vertical/diagonal line singularities in images. Unfortunately, intensity-discontinuity contours (i.e., edges) of images can have arbitrary orientations, and wavelet transforms cannot efficiently represent such edges. This has led to an interest in schemes that employ geometric representations of images, such as ridgelets [16], curvelets [8], edgelets [14], contourlets [13], wedgelets [15], bandelets [30], and normal meshes [22]. Recently, an image coder based on normal meshes was proposed by Jansen, Baraniuk, and Lavu [22], which we henceforth refer to as the JBL coder. A normal-mesh-based representation is adaptive and can efficiently capture the geometric information in images. The JBL coder, however, depends solely on the adaptivity of the normal-mesh scheme in locating points on image edges. A performance improvement may be achieved by explicitly providing some points on image edges as opposed to depending solely on the adaptivity of the normal-mesh scheme to find such points.

1.3 Overview and Contribution of the Thesis

Two types of image coding systems, namely subband-based and normal-mesh-based systems, are studied in this thesis. The first part of the research relates to subband image coding. An optimization-based approach for the design of high-performance filter banks for subband coding is proposed. The designed filter banks make a good tradeoff between several design criteria, and are shown to be highly effective for image coding, outperforming the 9/7-J filter bank in JPEG 2000. In the second part of the research, we propose methods

to enhance the coding performance of a normal-mesh-based image coder by more fully exploiting normal meshes. The improved coder is shown to be efficient, outperforming the JBL coder in both objective and subjective measures.

The remainder of this thesis is organized as follows. Chapter 2 introduces the preliminaries necessary to understand the work presented in this thesis. Some of the notation and terminology used herein are briefly presented, followed by background information pertaining to image coding. Finally, some fundamentals relating to subband-based and geometric-based image coding are introduced.

Subband coding is a widely used tool in many of today's best image coders. Subband coding systems are based on filter banks. Hence, high-performance filter banks are desirable for realizing good subband coding systems. The filter banks used in today's best coders, however, are not necessarily the very best that can be designed. In Chapter 3, we present a novel optimization-based filter bank design technique that yields improved filter banks for image coding. Appropriate design criteria are first discussed and gathered to form an abstract optimization problem. Then, three general approaches for solving the abstract optimization problem are proposed and analyzed. One approach is chosen for implementation, and the experimental results obtained from it are presented in detail. The choice of parameters and the objective function is discussed. The optimal filter banks obtained with the proposed method are shown to be highly effective for image coding. Some of our optimal filter banks outperform the well-known 9/7-J filter bank for both lossy and lossless compression, an impressive feat given that the 9/7-J filter bank is known for its exceptional lossy coding performance. Several families of perfect reconstruction filter banks are also discussed, where the filter banks in each family have some common structural properties. New filter banks in each family are designed with the proposed method. Experimental results show that these new filter banks outperform previously known filter banks from the same family.

In Chapter 4, some improvements to a multiscale normal-mesh-based image-coding system are discussed. The improved scheme is based on the JBL coder [22]. We first describe how the JBL coder works and our implementation of it. Then, we identify some shortcomings of the JBL coder. In order to overcome the shortcomings, three proposed modifications to the JBL coder are presented, namely modifications to the base mesh, normal/vertical offset format, and scan-conversion scheme. Experimental results show the effectiveness of the preceding three modifications. In particular, our proposed data-dependent base meshes help to locate horizons more efficiently and preserve image edges better. Using integers as opposed to real numbers to represent normal/vertical offsets improves coding efficiency. By exploiting bicubic interpolation with adaptive vertex selection, smoother reconstructed images with sharp edges are obtained.

Finally, Chapter 5 concludes the thesis by summarizing some of the more important results, as well as

suggesting some directions for future research.

Chapter 2

Preliminaries

2.1 Introduction

To facilitate a better understanding of the work presented in this thesis, some essential background information is introduced in this chapter. First, we introduce some of the notation and terminology used herein. Then, we discuss some topics related to image coding, including subband-based and normal-mesh-based image-coding systems.

2.2 Notation and Terminology

Before proceeding further, a brief digression is in order regarding the notation and terminology used herein. The sets of natural numbers, integers, odd integers, even integers, real numbers, and positive real numbers are denoted as $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}_{\text{odd}}$, $\mathbb{Z}_{\text{even}}$, $\mathbb{R}$, and $\mathbb{R}^+$, respectively. The symbol j denotes $\sqrt{-1}$. The subset of $\mathbb{R}$ $\{x \in \mathbb{R} \,:\, a \leq x \leq b\}$ is denoted as $[a,b]$. The Cartesian product of the sets X_1 and X_2 is denoted as $X_1 \times X_2$, and the n-fold Cartesian product of the set X is denoted as X^n. The magnitude of a real or complex number α is denoted as $|\alpha|$. The notation $a|b$ means that a divides b (i.e., $\frac{b}{a} \in \mathbb{Z}$). For $\alpha \in \mathbb{R}$, the notation $\lfloor \alpha \rfloor$ and $\lceil \alpha \rceil$ denote the largest integer not more than α (i.e., the floor function) and the smallest integer not less than α (i.e., the ceiling function), respectively. For $\alpha \in \mathbb{R}$, the signum function is defined as

$$\operatorname{sgn} \alpha \triangleq \begin{cases} 1 & \text{for } \alpha > 0 \\ 0 & \text{for } \alpha = 0 \\ -1 & \text{for } \alpha < 0. \end{cases} \tag{2.1}$$

Variable assignment in algorithms is denoted by the symbol ":=". The symbol O is used to denote an asymptotic upper bound for the magnitude of a function in terms of another, usually simpler, function. Suppose $f(x)$ and $g(x)$ are two functions defined on some subset of real numbers. We say that $f(x) \in O\big(g(x)\big)$ as $x \to \infty$, if there exists x_0 and $M > 0$ such that $|f(x)| \leq M|g(x)|$ for all $x > x_0$.

Curvature is the amount by which a geometric object deviates from being flat. For a planar curve, the curvature at a given point equals the reciprocal of the radius of an osculating circle (i.e., a circle that closely touches the curve at the given point). As the radius r of the osculating circle becomes smaller, the magnitude of the curvature ($\frac{1}{r}$) becomes larger. Therefore, where a curve is nearly straight, its curvature is close to zero, while when a curve undergoes a sharp turn, its curvature is large. For a planar curve given parametrically as $c(t) = \big(x(t),\, y(t)\big)$, the curvature κ of $c(t)$ can be computed as

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}}, \tag{2.2}$$

where x', x'', y', and y'' denote the first and second order derivatives of x, and the first and second order derivatives of y, respectively. Note that the above equation is only valid when both x and y are twice differentiable. For discrete curves, however, curvature is not well defined. Some methods such as [28] can be applied to estimate curvatures for discrete curves.

Matrices and vectors are denoted by uppercase and lowercase boldface letters, respectively. The transpose of the matrix/vector $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^T$. For matrix multiplication, we define the product notation as

$$\prod_{k=M}^{N} \boldsymbol{A}_k \triangleq \boldsymbol{A}_N \boldsymbol{A}_{N-1} \ldots \boldsymbol{A}_{M+1} \boldsymbol{A}_M, \tag{2.3}$$

where $N \geq M$. The symbols $\boldsymbol{I}$, $\boldsymbol{0}$, and $\boldsymbol{1}$ denote an identity matrix, a vector of all zeros, and a vector of all ones, respectively, the size of which should be clear from the context. For a positive semi-definite matrix $\boldsymbol{A}$, we denote its square root (e.g., as defined in [24]) as $\boldsymbol{A}^{1/2}$. Given a vector $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_n]^T$, a general l_p norm $\|\boldsymbol{x}\|_p$ is defined as

$$\|\boldsymbol{x}\|_p = \begin{cases} \Big(\sum_{i=1}^{n} |x_i|^p\Big)^{1/p} & \text{for } \ 1 \leq p < \infty \\ \max\big\{|x_1|, \cdots, |x_n|\big\} & \text{for } \ p = \infty, \end{cases} \tag{2.4}$$

where the subscript p is omitted when clear from the context. For a function $f(\boldsymbol{x})$, its gradient with respect to the vector variable $\boldsymbol{x}$ is defined as $\nabla_{\boldsymbol{x}} f = \big[\frac{\partial f}{\partial x_1}\ \frac{\partial f}{\partial x_2}\ \cdots\ \frac{\partial f}{\partial x_n}\big]^T$, and its transposed gradient is denoted as $\nabla_{\boldsymbol{x}}^T f$, where the subscript $\boldsymbol{x}$ may be omitted when clear from the context.

The **Fourier transform** of $x(t)$, denoted as $\hat{x}(\omega)$, is defined as

$$\hat{x}(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt. \tag{2.5}$$

The **inverse Fourier transform** of $\hat{x}(\omega)$ is given by

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{x}(\omega) e^{j\omega t} d\omega. \tag{2.6}$$

The above equation represents the signal $x(t)$ in terms of complex sinusoids at all frequencies. The **z-transform** of a discrete-time signal $x[n]$, denoted $X(z)$, is defined as

$$X(z) = \sum_{n \in \mathbb{Z}} x[n] z^{-n}. \tag{2.7}$$

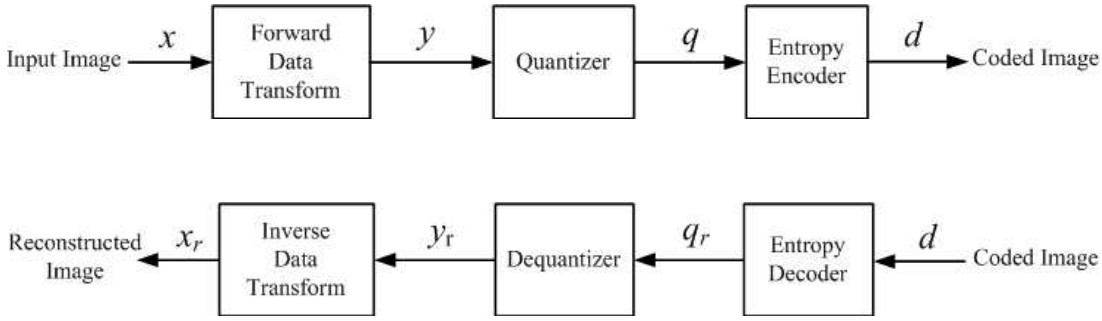
The transfer function, impulse response, and frequency response of a filter H are denoted as $H(z)$, h, and $\hat{h}(\omega)$, respectively. A filter H is said to have linear phase if the phase response of H is a linear function of frequency.

2.3 Image Coding

As mentioned earlier, the particular type of image coding in which we are interested is image compression. This type of image coding tries to reduce the number of bits used for representing images while still being able to reproduce faithful duplicates of the originals. In the sections that follow, we introduce the general structure of an image coding system as well as several compression-performance measures. We also discuss two important concepts, namely, quantization and entropy.

2.3.1 Image Coding Systems

The general structure of a typical image coding system is shown in Figure 2.1. The image coding system consists of an encoder and a decoder. The encoder has three components, namely, a forward data transform, a quantizer, and an entropy encoder. The first component, the forward data transform, usually applies a transformation or filtering process to the input data, aimed at achieving energy compaction. In other words, the forward data transform reduces the number of nonzero or large-magnitude parameters in the representation as much as possible. Therefore the signal y has more zero or small-magnitude parameters than the input image signal x, which makes the signal y more compressible than x in subsequent processing. The forward data transform also assumes the role of separating important and less important information. The separation is especially helpful for the design of a quantizer, where one needs to decide which information to discard in order to minimize distortion. To design an energy-compact forward data transform, we need to exploit the statistical properties of images. The second component of the encoder is the quantizer. It converts the signal y into a less precise signal q. The quantizer simply reduces the number of bits required to represent



(b)

Figure 2.1: General structure of a typical image coding system. (a) Encoder and (b) decoder.

the signal by discarding less important information, which does not significantly affect signal quality. The third component, an entropy encoder, is needed to encode the quantized signal q. The compressed signal d preserves all information in q.

The structures of the encoder and decoder are highly symmetric. The image decoder is comprised of an entropy decoder, a dequantizer, and an inverse data transform. The decoder takes the coded image data as the input, and applies the entropy decoding, dequantization, and inverse data transformation, yielding the reconstructed image x_r as the output.

If the original signal can be recovered from its compressed representation, the coding is said to be **lossless**; otherwise it is called **lossy**. In the lossy case, the difference between the original and reconstructed signals is referred to as distortion. Lossy coding is commonly used for image and video data, where a certain amount of information loss is acceptable for many applications.

To evaluate the performance of lossless and lossy compression systems, we now introduce several compression-performance measures. For both lossless and lossy compression, the **compression ratio** is often employed, which is defined as

$$\text{compression ratio} = \frac{\text{original signal size in bits}}{\text{compressed signal size in bits}}. \tag{2.8}$$

The reciprocal of compression ratio is referred to as the **normalized bit rate**. The **bit rate** denotes the number of bits per sample in the coded representation.

In the case of lossy compression, the distortion is usually measured in terms of **mean-squared error** (MSE) and **peak-signal-to-noise ratio** (PSNR). For the original image x and reconstructed image x_r of size $N_0 \times N_1$, the MSE and PSNR are defined as

$$\text{MSE} = \frac{1}{N_0 N_1} \sum_{n_0=0}^{N_0-1} \sum_{n_1=0}^{N_1-1} \big(x_r[n_0,n_1] - x[n_0,n_1]\big)^2 \quad \text{and} \tag{2.9}$$

$$\text{PSNR} = 20\log_{10}\left(\frac{2^P - 1}{\sqrt{\text{MSE}}}\right), \tag{2.10}$$

respectively, where P is the number of bits per sample in the original image x. Lower MSE does not always correspond to reconstructed images with better visual quality. Therefore, subjective measures of distortion (i.e., as judged by human observers) are considered to be quite important.

2.3.2 Quantization

As shown in Figure 2.1, the second component of a typical image encoder is a quantizer. The quantizer converts a signal with a continuous range of values or a very large set of possible discrete values to integer indices in order to obtain an efficient representation of the signal. Quantization is usually not invertible. A scalar quantizer quantizes each value individually, and is relatively simple to implement. A generally used quantizer is given by

$$q = (\text{sgn}\, x)\left\lfloor \frac{|x|}{\Delta} + \frac{1}{2} \right\rfloor, \tag{2.11}$$

where $x \in \mathbb{R}$ is the true value, $q \in \mathbb{Z}$ is the quantizer index, and $\Delta \in \mathbb{R}^+$ is the quantizer step size. The quantizer step size determines the tradeoff between signal distortion and the bit rate. In order to achieve an optimal bit rate allocation, the quantizer step size may be selected to take into account the properties of the human visual system (HVS) [33]. Information to which the HVS is less sensitive is less important and can therefore be heavily quantized.

A dequantizer in the decoder is the counterpart of a quantizer in the encoder. The dequantizer reconstructs the original signal by converting quantizer indices into a signal with the same format as the original with some potential loss in accuracy. The true value x is approximated with $\hat{x}$ by the dequantizer, and for the quantizer given by (2.11), the resulting dequantized value $\hat{x}$ is given by

$$\hat{x} = q\Delta. \tag{2.12}$$

2.3.3 Entropy Coding

As shown in Figure 2.1, the third component of a typical image encoder is the entropy encoder. It tries to achieve an optimal lossless compression rate, which is bounded by the entropy of the quantized signal data. The concept of entropy [36] plays a central role in information theory. Entropy is sometimes referred to as a measure of uncertainty. The entropy H of a discrete random variable X with possible outcomes $x_1, \cdots, x_n$ is defined in terms of its probability distribution as

$$H = -\sum_{k=1}^{n} Pr(x_k)\log_2 Pr(x_k), \tag{2.13}$$

where $Pr(x_k)$ is the probability of the kth outcome x_k of X.

A commonly used probability distribution is the Laplacian, which is also known as the double exponential distribution. A random variable is a Laplacian if its probability density function is of the form

$$p(x) = \frac{\lambda}{2} e^{-\lambda|x-\mu|}, \tag{2.14}$$

where $\mu \in \mathbb{R}$ and $\lambda \in \mathbb{R}^+$ are location and scale parameters, respectively. A random variable is said to have a uniform distribution on the interval $[a,b]$ if its probability density function is of the form

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

In the image coding system of Figure 2.1, the entropy encoder tries to achieve the lossless compression rate associated with the estimated entropy of the signal using a probability model. Two commonly used entropy coding schemes are Huffman and arithmetic coding. More information about entropy coding techniques can be found in [18, 29].

2.4 Subband-Based Image Coding Systems

Subband coding has been proven to be an extremely valuable tool for image coding applications. Subband coding has been successfully used in the **embedded zerotree wavelet** (EZW) [37], **set partitioning in hierarchical trees** (SPIHT) [34], and JPEG-2000 coding systems. As will become clear later, filter banks are used to realize subband coding. In subband-based image coding systems, the forward and inverse data transform components of Figure 2.1 are associated with filter banks. We introduce some concepts related to subband coding and multirate filter banks in the sections that follow.

2.4.1 Multirate Systems

A system is said to be multirate if the system employs more than one sampling rate. Since a multirate system has more than one sampling rate, a means for changing sampling rates is needed. The downsampling and upsampling operations are therefore essential parts of a multirate system. The operations are performed by processing elements called downsamplers and upsamplers, respectively. Here, we give the formal definitions of the two operations.

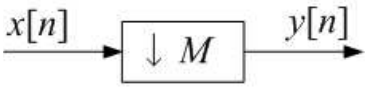


Figure 2.2: M-fold downsampler.

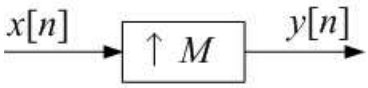


Figure 2.3: M-fold upsampler.

Definition 2.1 (Downsampling)**.** The M-fold downsampler, as shown in Figure 2.2, reduces the sampling rate of the input signal $x[n]$ by a **factor** of M. Mathematically, downsampling is defined as

$$y[n] = (\downarrow M)x[n] = x[Mn], \tag{2.16}$$

where $M \in \mathbb{N}$ and $M \geq 2$. The downsampling process retains only every Mth sample of the input signal $x[n]$.

Definition 2.2 (Upsampling)**.** The M-fold upsampler, as shown in Figure 2.3, increases the sampling rate of the input signal $x[n]$ by a **factor** of M. Mathematically, upsampling is defined as

$$y[n] = (\uparrow M)x[n] = \begin{cases} x\left[\frac{n}{M}\right] & \text{if } M|n \\ 0 & \text{otherwise,} \end{cases} \tag{2.17}$$

where $M \in \mathbb{N}$ and $M \geq 2$. The upsampling process inserts $M-1$ zeros between the original samples of the input signal $x[n]$.

2.4.2 Multirate Filter Banks

Multirate filter banks provide an efficient way to realize the wavelet transforms used in subband coding. One important type of multirate filter bank is the **uniformly maximally decimated** (UMD) filter bank. The general structure of a UMD filter bank is depicted in Figure 2.4, where M denotes the number of channels. The input signal $x[n]$ is processed by the analysis filters $\{\mathrm{H}_k(z)\}_{k=0}^{M-1}$, and the resulting signals are downsampled by M. Each of the resulting subband signals $\{y_k[n]\}_{k=0}^{M-1}$ has a sampling density that is $\frac{1}{M}$th of the sampling density of the input signal $x[n]$. On the synthesis side, the subband signals $\{y_k[n]\}_{k=0}^{M-1}$ are upsampled by M, and then processed by the synthesis filters $\{\mathrm{G}_k(z)\}_{k=0}^{M-1}$. Finally, all of the synthesis filter outputs are added together, producing the reconstructed signal $x_r[n]$. As a matter of terminology, a filter bank is called **maximally decimated** if the sum of the sampling densities in all subbands equals the sampling density of the input signal. A filter bank is called **uniformly decimated** if the filter bank is such that the signal in each channel is downsampled by the same sampling factor. In the case that $M = 2$, a UMD filter bank has the canonical form shown in Figure 2.5.

For a UMD filter bank, if the output signal $x_r[n]$ is identical to the input signal $x[n]$, namely, $x_r[n] = x[n]$,

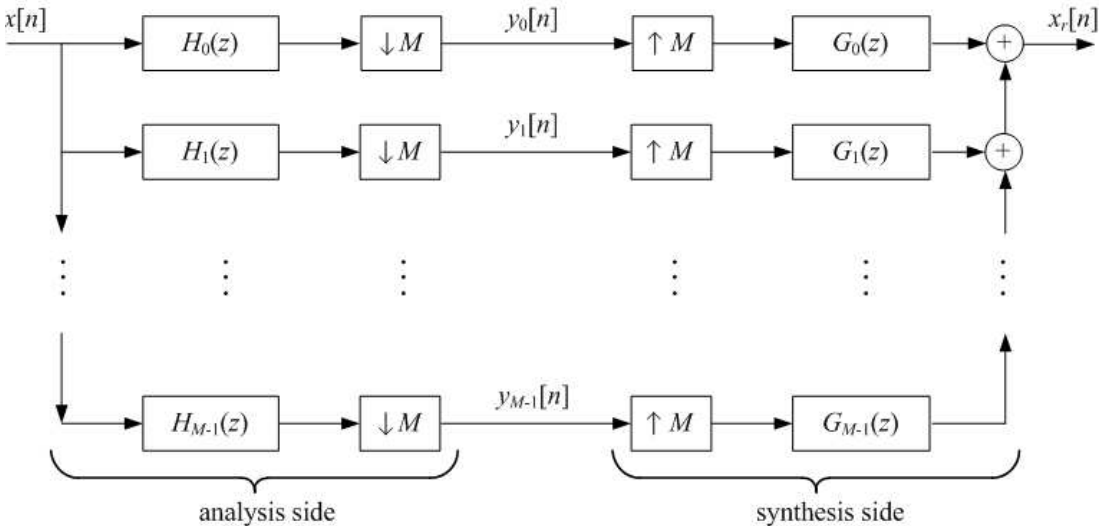


Figure 2.4: General structure of an M-channel UMD filter bank.

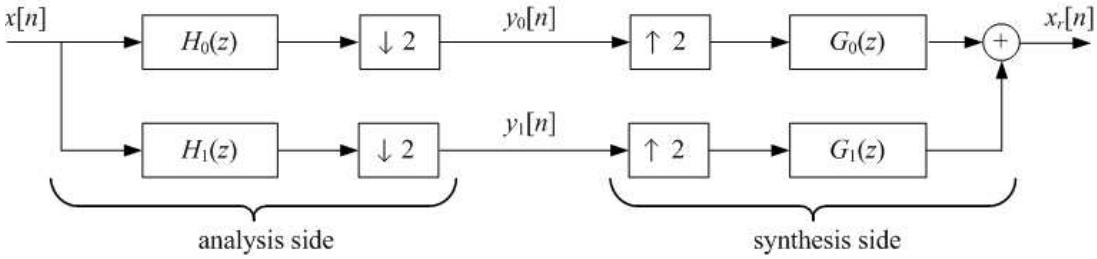


Figure 2.5: The canonical form of a 1-D two-channel UMD filter bank.

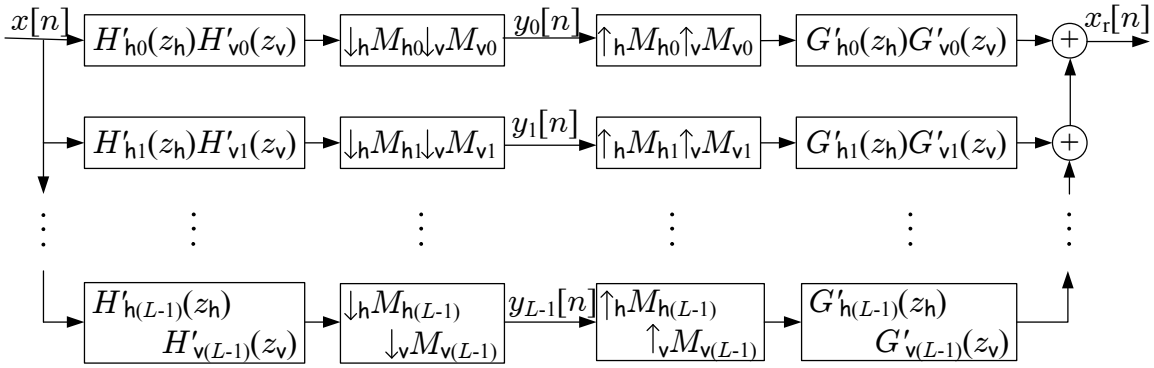


Figure 2.6: The equivalent L-channel nonuniform filter bank associated with the N-level tree-structured filter bank.

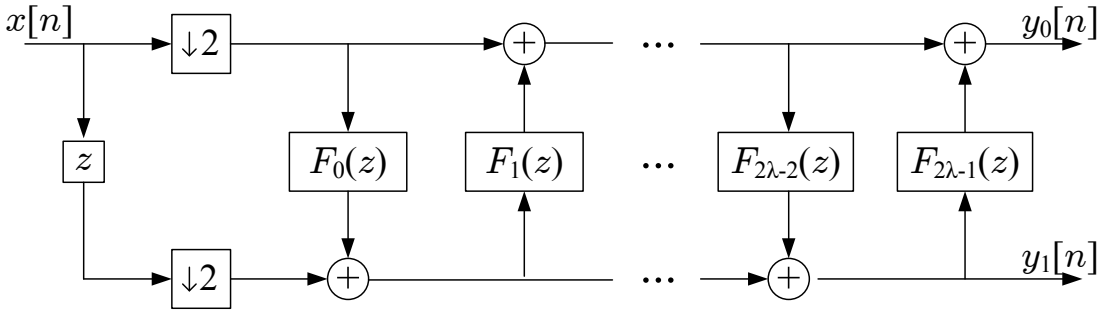
the filter bank is said to possess the (shift-free) **perfect reconstruction** (PR) property. The PR property is desirable in many applications, such as image coding.

2.4.3 2-D Octave-Band Filter Banks

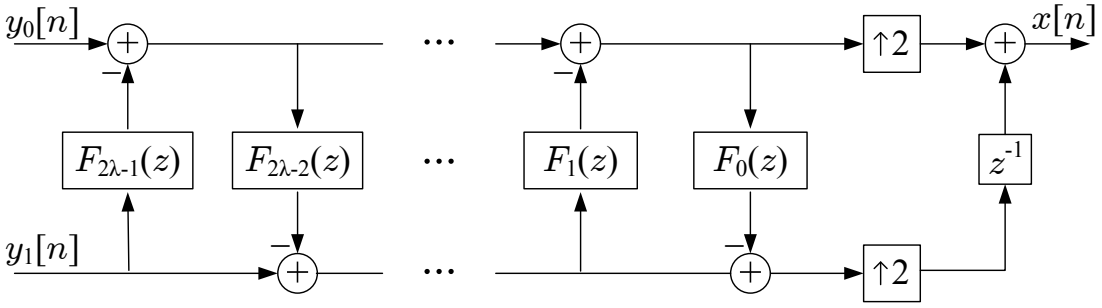
Since images are two-dimensional (2-D) signals, their processing requires multidimensional systems. This can be accomplished by constructing multidimensional systems based on 1-D building blocks. In particular, to construct a 2-D filter bank from a 1-D filter bank, we simply apply the 1-D two-channel filter bank in each of the two dimensions of the signal in succession. This results in a separable four-channel 2-D filter bank. Furthermore, in practice, we usually apply the 2-D filter bank in an N-level tree structure, decomposing the lowest-frequency subband signal at each level in the tree. The resulting N-level tree-structured filter bank can be equivalently expressed (via the noble identities [43]) in the form of an L-channel nonuniform filter bank as shown in Figure 2.6, where $L = 3N + 1$. In the diagram, the $\{H'_{hk}\}$, $\{H'_{vk}\}$, $\{G'_{hk}\}$, and $\{G'_{vk}\}$ denote the equivalent horizontal analysis, vertical analysis, horizontal synthesis, and vertical synthesis filters, respectively, and the $\{M_{hk}\}$ and $\{M_{vk}\}$ denote horizontal and vertical upsampling/downsampling factors, respectively. As a matter of notation, the subscripts "h" and "v" on $\downarrow$, $\uparrow$, and z indicate correspondences with the horizontal and vertical directions, respectively.

2.4.4 Lifting Realization of Filter Banks

UMD filter banks can also be implemented by the lifting realization [41]. The lifting realization of filter banks is superior to the canonical form in several respects. First, the lifting realization structurally imposes the PR property. Furthermore, the lifting structure also helps in the construction of reversible integer to



(a)



(b)

Figure 2.7: The lifting realization of a 1-D two-channel UMD filter bank. (a) Analysis and (b) synthesis sides.

integer transforms. Such transforms are of great interest in many applications, since they remain invertible even in the presence of finite-precision arithmetic.

The lifting realization of a 1-D two-channel filter bank is shown in Figure 2.7, where there are 2λ lifting filters $\{\mathrm{F}_k\}_{k=0}^{2\lambda-1}$. Without loss of generality, we assume that only $F_0(z)$ and $F_{2\lambda-1}(z)$ may be identically zero. The analysis filter transfer functions can be calculated from the lifting parametrization as

$$H_0(z) = H_{0,0}(z^2) + zH_{0,1}(z^2) \quad \text{and} \tag{2.18a}$$

$$H_1(z) = H_{1,0}(z^2) + zH_{1,1}(z^2), \tag{2.18b}$$

where

$$\begin{bmatrix} H_{0,0}(z) & H_{0,1}(z) \\ H_{1,0}(z) & H_{1,1}(z) \end{bmatrix} = \prod_{k=0}^{\lambda-1} \left(\begin{bmatrix} 1 & F_{2k+1}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ F_{2k}(z) & 1 \end{bmatrix} \right). \tag{2.19}$$

The transfer functions of the synthesis filters can be similarly derived as

$$G_0(z) = G_{0,0}(z^2) + z^{-1}G_{1,0}(z^2) \quad \text{and} \tag{2.20a}$$

$$G_1(z) = G_{0,1}(z^2) + z^{-1}G_{1,1}(z^2), \tag{2.20b}$$

where

$$\begin{bmatrix} G_{0,0}(z) & G_{0,1}(z) \\ G_{1,0}(z) & G_{1,1}(z) \end{bmatrix} = \prod_{k=1-\lambda}^{0} \left(\begin{bmatrix} 1 & 0 \\ -F_{-2k}(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -F_{-2k+1}(z) \\ 0 & 1 \end{bmatrix} \right). \tag{2.21}$$

Consequently, the synthesis filters are completely determined by the analysis filters as given by

$$G_0(z) = -z^{-1}H_1(-z) \quad \text{and} \tag{2.22a}$$

$$G_1(z) = z^{-1}H_0(-z). \tag{2.22b}$$

2.4.5 Relationship Between Filter Banks and Wavelet Systems

Since a 2-D separable filter bank or wavelet transform can be realized by successively applying a 1-D filter bank or wavelet transform in each of the two dimensions of the signal, we discuss only 1-D filter banks and wavelet transforms in this section. Filter banks and wavelets were initially shown to be closely connected in [26]. In particular, it was demonstrated that wavelet transforms [32] can be computed by a tree-structured filter bank based on UMD-filter-bank building blocks. Figure 2.8 gives the tree structure of an N-level octave-band filter bank with the building blocks being 1-D two-channel UMD filter banks from Figure 2.5. In the octave-band filter bank, the analysis side of the building block decomposes the lowest-frequency subband signal at each level.

In order to present the relation between filter banks and wavelet systems in a mathematical manner, we will introduce some equations related to wavelet representations shortly. A wavelet system can represent functions at different resolutions. We focus our attention on dyadic wavelet systems, where successive resolution differs in scale by a factor of two. In a dyadic wavelet system, the primal scaling function ϕ satisfies a scaling equation of the form

$$\phi(t) = \sqrt{2} \sum_{n \in \mathbb{Z}} c[n]\phi(2t - n). \tag{2.23}$$

The equation shows the relationship between the function ϕ and the translated and dilated versions of itself. The primal wavelet function ψ is related to the primal scaling function ϕ. The relationship can be expressed in terms of the wavelet equation

$$\psi(t) = \sqrt{2} \sum_{n \in \mathbb{Z}} d[n]\phi(2t - n). \tag{2.24}$$

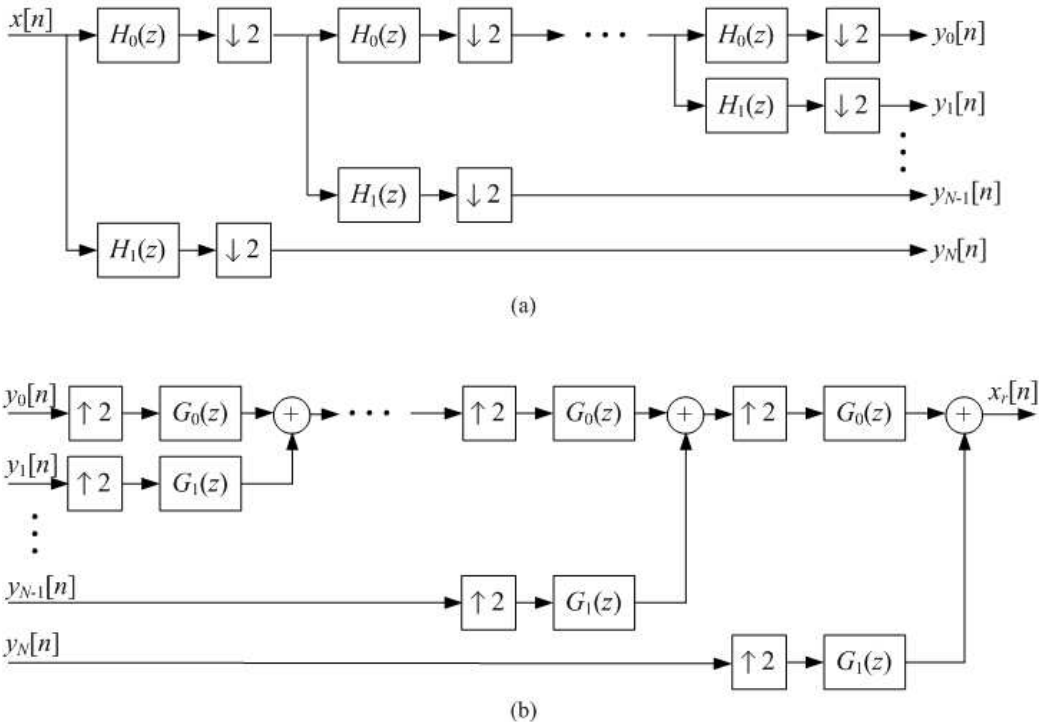


Figure 2.8: The tree structure of an N-level octave-band filter bank. (a) Analysis side and (b) synthesis side.

A **multiresolution approximation** (MRA) can be derived from the scaling and wavelet equations. MRAs occur in pairs. The complementary MRA of any MRA is called the dual MRA. In the **dual MRA**, the dual scaling and dual wavelet equations follow the same pattern as the primal scaling and primal wavelet functions, respectively. The dual scaling function $\tilde{\phi}$ and dual wavelet function $\tilde{\psi}$ are respectively given by

$$\tilde{\phi}(t) = \sqrt{2} \sum_{n \in \mathbb{Z}} \tilde{c}[n] \tilde{\phi}(2t - n) \tag{2.25}$$

and

$$\tilde{\psi}(t) = \sqrt{2} \sum_{n \in \mathbb{Z}} \tilde{d}[n] \tilde{\phi}(2t - n). \tag{2.26}$$

For an octave-band filter bank constructed by 1-D two-channel filter-bank (as shown in Figure 2.5) building blocks, the filter impulse responses relate to the above MRA in the following manner:

$$h_0[n] = \tilde{c}^*[-n], \quad h_1[n] = \tilde{d}^*[-n], \quad g_0[n] = c[n], \quad \text{and} \quad g_1[n] = d[n]. \tag{2.27}$$

In the above equation, h_0, h_1, g_0, and g_1 are respectively the impulse responses for the analysis lowpass, analysis highpass, synthesis lowpass, and synthesis highpass filters of the 1-D two-channel UMD filter bank, and $c[n]$, $d[n]$, $\tilde{c}[n]$, and $\tilde{d}[n]$ are the coefficient sequences for the primal scaling, primal wavelet, dual scaling, and dual wavelet equations, respectively.

Because of the aforementioned relationship, the structures of operations in an octave-band filter bank and a wavelet transform are identical. There exists a one-to-one mapping between a wavelet transform and an octave-band filter bank with the building blocks having certain properties. Consequently, the choice of filter coefficients determines the shape of the associated scaling and wavelet functions. When a signal is processed by an octave-band filter bank and the subband signals are quantized, the reconstructed signal tends to have the type of artifacts in the shape of the impulse-response functions associated with the synthesis filters. Furthermore, as the number of building blocks used in the cascade algorithm [39] in the synthesis bank increases, the discrete-time basis functions approach to sampled versions of primal scaling and primal wavelet functions. The cascade algorithm converges very quickly. Only after a small number of iterations, discrete-time basis functions reasonably approximate the scaling and wavelet functions. Therefore, most of the discrete-time basis functions are fairly good approximations of sampled versions of scaling and wavelet functions. Hence, the shapes of the primal scaling and primal wavelet functions are of great importance in image coding.

2.4.6 Vanishing Moments

One useful concept is that of the moment of a sequence as defined below.

Definition 2.3 (Moment)**.** The kth moment of a sequence h is defined as

$$m_k = \sum_{n\in\mathbb{Z}} n^k h[n]. \tag{2.28}$$

Let $d[n]$ and $\tilde{d}[n]$ be the coefficient sequences for the primal and dual wavelet equations, respectively. The kth moments of primal and dual wavelet coefficient sequences are given by

$$m_k = \sum_{n\in\mathbb{Z}} n^k d[n] \tag{2.29}$$

and

$$\tilde{m}_k = \sum_{n\in\mathbb{Z}} n^k \tilde{d}[n], \tag{2.30}$$

respectively. Let h_1 and g_1 be the impulse responses of the highpass analysis and synthesis filters, respectively. From (2.27), we know the relationships $h_1[n] = \tilde{d}^*[-n]$ and $g_1[n] = d[n]$. Therefore, the kth moments of primal and dual wavelet coefficient sequences can also be expressed as $m_k = \sum_{n\in\mathbb{Z}} n^k g_1[n]$ and $\tilde{m}_k = (-1)^k \sum_{n\in\mathbb{Z}} n^k h_1[n]$, which have the same magnitude as the kth moments of g_1 and h_1, respectively.

The numbers of vanishing (i.e., equal to zero) moments for primal and dual wavelet coefficient sequences are important quantities. In terms of wavelet systems, the number of vanishing moments for dual wavelet coefficient sequence determines the highest order of polynomials that the primal scaling functions can represent. In terms of filter banks, the number of vanishing moments associated with the impulse response sequence of highpass analysis filter determines the ability of the lowpass synthesis filter to represent polynomials. Since smooth functions can be well approximated by polynomials, it is important for an efficient filter bank to have a certain number of vanishing moments associated with the impulse-response sequence of the highpass analysis filter. More specifically, if an original signal can be well approximated by a polynomial of order η, a filter bank with η or more vanishing moments of the impulse-response sequence of highpass analysis filter leads to few/no nonzero coefficients in the highpass and bandpass subbands. The transformed coefficients consisting mostly of zeros can be efficiently represented, and are favorable in signal coding applications.

For a wavelet system associated with a two-channel PR UMD filter bank, we are interested in the number of primal and dual vanishing moments of wavelet functions. Let h_0, h_1, g_0, and g_1 be the impulse responses of the lowpass analysis, highpass analysis, lowpass synthesis, and highpass synthesis filters, respectively. The wavelet system associated with a two-channel PR UMD filter bank has η primal vanishing moments if $\hat{h}_0(\omega)$ has a ηth order zero at $\omega = \pi$ or $\hat{g}_1(\omega)$ has a ηth order zero at $\omega = 0$. The wavelet system has η dual vanishing moments if $\hat{h}_1(\omega)$ has a ηth order zero at $\omega = 0$ or $\hat{g}_0(\omega)$ has a ηth order zero at $\omega = \pi$. The following theorem offers alternative ways to determine the number of primal and dual vanishing moments of wavelet functions.

Theorem 2.1 (Sum rule)**.** *Let h_0 and h_1 be sequences with Fourier transforms $\hat{h}_0(\omega)$ and $\hat{h}_1(\omega)$, respectively. Then, $\hat{h}_0(\omega)$ has a ηth order zero at $\omega = \pi$ if and only if*

$$\sum_{n\in\mathbb{Z}} (-1)^n n^k h_0[n] = 0, \quad \text{for } k = 0,\ 1,\ \cdots\ \eta - 1, \tag{2.31}$$

and $\hat{h}_1(\omega)$ has a ηth order zero at $\omega = 0$ if and only if

$$\sum_{n\in\mathbb{Z}} n^k h_1[n] = 0, \quad \text{for } k = 0,\ 1,\ \cdots\ \eta - 1. \tag{2.32}$$

Therefore, for a wavelet system associated with an iterative tree-structured filter bank to have η primal and dual vanishing moments, the impulse responses of the corresponding lowpass and highpass filter are required to respectively satisfy the linear equation (2.31) and (2.32), where quantities in terms of n are equation coefficients.

2.4.7 Coding Gain

Coding gain [23, 12] is a measure of the energy compaction abilities of a filter bank. This measure is defined as the ratio between the reconstruction error variance obtained by quantizing a signal directly to that obtained by quantizing the corresponding subband coefficients using an optimal bit allocation strategy. Coding gain is a useful quantity as it provides an indication of how well a filter bank is likely to perform for image coding purposes.

An L-channel filter bank, as shown in Figure 2.6, has the coding gain G_{SBC} given by

$$G_{\mathsf{SBC}} = \prod_{k=0}^{L-1} \left(\frac{\alpha_k}{A_k B_k}\right)^{\alpha_k}, \tag{2.33}$$

where

$$A_k = \sum_{m\in\mathbb{Z}} h'_k(m) \sum_{n\in\mathbb{Z}} h'_k(n) r(m-n), \quad B_k = \alpha_k \sum_{n\in\mathbb{Z}} g'^2_k(n), \quad \alpha_k = \frac{1}{M_k},$$

M_k is the downsampling factor of the kth subband, and r is the normalized autocorrelation function of the input image. In practice, the normalized autocorrelation function is chosen, depending on the most appropriate image model, as follows:

$$r(x,y) = \begin{cases} \rho^{|x|+|y|} & \text{for separable model} \\ \rho^{\sqrt{x^2+y^2}} & \text{for isotropic model,} \end{cases} \tag{2.34}$$

where ρ is a correlation coefficient (typically, $0.9 \le \rho \le 0.95$ for images).

2.5 Normal-Mesh-Based Image Coding Systems

In the sections that follow, we introduce some basic concepts pertaining to geometric-based image coding systems. We first introduce triangulations, followed by some subdivision concepts. Lastly, scan conversion is discussed.

2.5.1 Triangulations

The notion of a triangulation is an important concept in geometric representations. In what follows, we first introduce the concepts of convex and convex hull, followed by the formal definition of a triangulation.

Definition 2.4 (Convex)**.** A subset S of the plane is called convex if and only if for any pair of points $p, q \in S$ the line segment $\overline{pq}$ is completely contained in S.

Definition 2.5 (Convex hull)**.** The convex hull of a set S is the smallest convex set that contains S. To be more precise, it is the intersection of all convex sets that contain S.

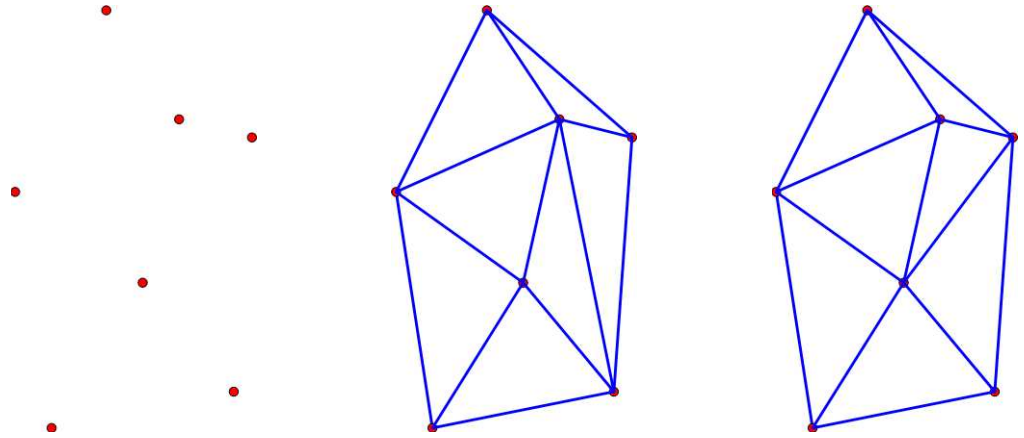
Definition 2.6 (Triangulation)**.** A triangulation of a set V of vertices in $\mathbb{R}^2$ is a set T of triangles such that: the union of the vertices of all triangles in T is V; the interiors of any two triangles in T are disjoint; and the union of the triangles in T is the convex hull of V.

The triangulation of a point set is usually not unique. Figure 2.9 gives an example of a triangulation. Figure 2.9(a) shows a given set V of vertices, and Figures 2.9(b) and (c) present two different triangulations of V. A triangulation helps to simplify the operations on a space. For instance, an approximation of a function can be processed in pieces when the function domain is partitioned into (triangular) regions.

One important type of triangulation is the Delaunay triangulation, which has a number of useful properties. The definition of a Delaunay triangulation involves the concept of a circumcircle. The circumcircle of a triangle is the unique circle that passes through all three vertices of the triangle. Now we are ready to introduce the definition of a Delaunay triangulation.

Definition 2.7 (Delaunay triangulation (DT))**.** A triangulation is said to be Delaunay if each triangle in the triangulation is such that the interior of its circumcircle contains no vertices.

An example of a DT, namely the DT of the vertex set in Figure 2.9(a), is shown in Figure 2.10. In the figure, the circumcircle of each triangle in the triangulation is also drawn. Each circumcircle contains no vertices strictly in its interior. Therefore, the triangulation is a DT.



(a) (b) (c)

Figure 2.9: Example of a triangulation. (a) A set V of vertices, (b) a triangulation of V, and (c) another triangulation of V.

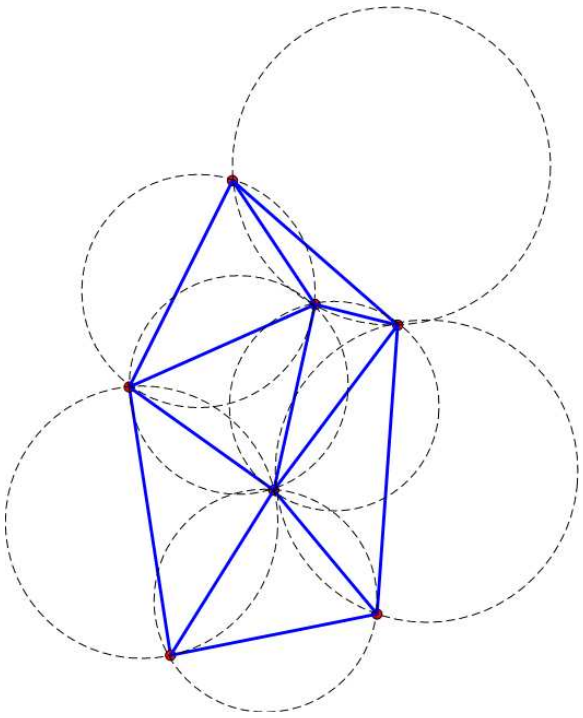


Figure 2.10: Example of a Delaunay triangulation.

A DT maximizes the minimum angle of all interior angles of the triangles in the triangulation. A DT also tends to avoid sliver triangles to whatever extent is possible. It is known that there exists a unique DT for V, if V is a set of vertices such that no four vertices are cocircular. For situations in which the DT is not unique, some methods exist for choosing a unique DT from amongst the numerous possibilities. One such scheme is presented in [17].

In many applications, some prescribed line segments are required as part of a triangulation. Therefore, segment constraints need to be imposed during the triangulation process. A planar straight line graph describes the collection of vertices and line segments used to construct a constrained triangulation.

Definition 2.8 (Planar straight line graph (PSLG))**.** A planar straight line graph is a set V of vertices in $\mathbb{R}^2$ and a set L of line segments, denoted (V, L), such that: each line segment in L must have its endpoints in V; and any two line segments in L must either be disjoint or intersect at most at a common endpoint.

To help define a constrained Delaunay triangulation, it is convenient to think of constrained segments as blocking the view of one point from another. Then, two points are **invisible** from each other if the segment between the two points intersects a constrained segment. Now, let us define a constrained DT as follows.

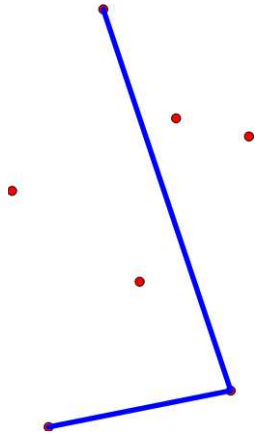
Definition 2.9 (Constrained Delaunay triangulation (Constrained DT))**.** A triangulation is said to be constrained Delaunay if each triangle in the triangulation is such that: the interior of the triangle does not intersect any constraining line segment; and any vertex inside the triangle's circumcircle is invisible from the interior of the triangle.

An example of a constrained DT is shown in Figure 2.11. Figures 2.11(a) and (b) show the given PSLG and its corresponding constrained DT, respectively. The thick segments $\overline{ab}$ and $\overline{bg}$ correspond to the constrained segments in the given PSLG. The circumcircle of each triangle is shown in Figure 2.11(b) as well. Vertices d and e are inside $\triangle abc$'s circumcircle, while they are invisible from the interior of the triangle, since the line segment from d or e to any interior point of $\triangle abc$ intersects the constrained segment $\overline{ab}$ of $\triangle abc$. Similarly, vertices c, f, and g are inside $\triangle abd$'s circumcircle, while they are invisible from the interior of $\triangle abd$. Therefore, the triangulation is a constrained DT.

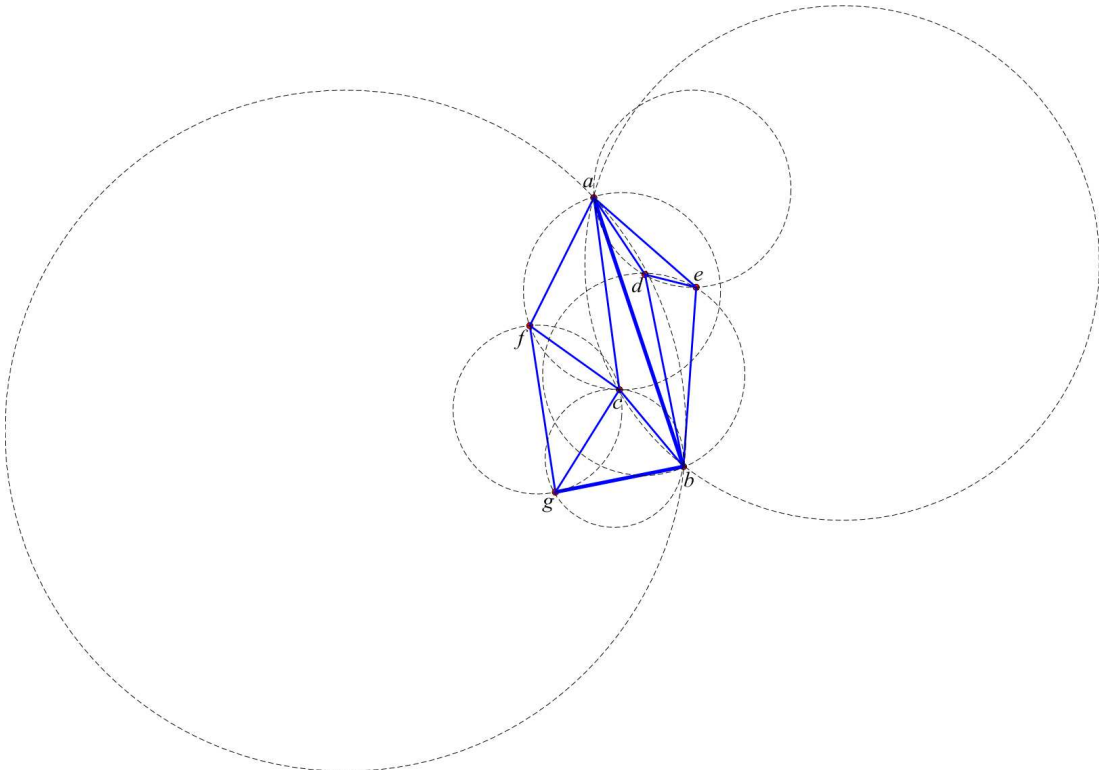
Similar to a DT, a constrained DT of a PSLG also tends to avoid sliver triangles, except the sliver triangles caused by satisfying the segment constraints. A constrained DT is not (except in rare cases) a DT.

2.5.2 Quaternary Subdivision

Quaternary subdivision is an important concept related to the triangulations of geometric representations. A **quaternary subdivision** is a means of splitting a triangle into four new non-overlapping triangles. This is



(a)



(b)

Figure 2.11: Example of a constrained Delaunay triangulation. (a) A PSLG and its corresponding (b) constrained Delaunay triangulation.

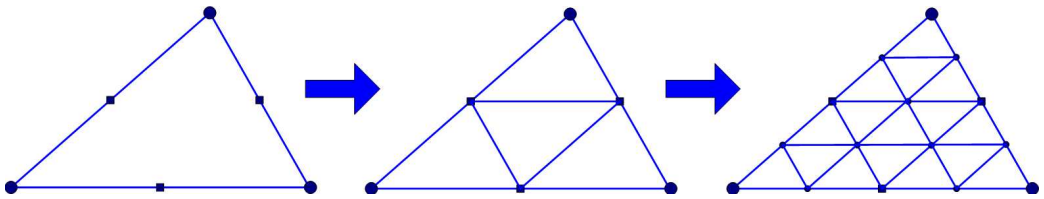


Figure 2.12: Quaternary subdivision.

accomplished by adding a new point associated with each triangle edge. As shown in Figure 2.12, a new vertex (denoted by a small square) is added for each edge in the triangulation, and each triangle in the coarse mesh is divided into four small triangles by connecting among the new added vertices and the three vertices of the triangle. The quaternary subdivision is done iteratively following the same simple topology structure.

2.5.3 Scan Conversion

Geometric objects defined on a continuous domain must be represented on discrete grids in order to use raster scan output devices, such as printers and computer screens. The process of converting geometric objects defined on a continuous domain to a raster representation is called **scan conversion**. Interpolation schemes can be used in the scan-conversion process. Interpolation is a method of constructing new data points from a set of known data points in some space. The values at new data points are usually determined by the values at some known data points nearby. We focus our attention on interpolation methods for functions defined on a plane. A point in 2-D together with its corresponding height value can be viewed as a point in 3-D. The interpolated function values together with their positions can be considered as a surface in 3-D. Some popular interpolation methods include nearest neighbor, planar, bilinear, and bicubic interpolation.

Chapter 3

Design of High-Performance Filter Banks for Image Coding

Overview

In this chapter, an optimization-based method for the design of high-performance separable filter banks for image coding is proposed. This method yields linear-phase perfect-reconstruction (PR) systems with high coding gain, good frequency selectivity, and certain prescribed vanishing-moment properties. Several two-channel PR UMD filter banks designed with the proposed method are presented and shown to work extremely well for image coding, outperforming the well-known 9/7 filter bank from the JPEG-2000 standard in most cases. Several families of two-channel PR UMD filter banks are also discussed, where the filter banks in each family have some common structural properties. Filter banks in each family are designed using the proposed method. Experimental results show that the designed filter banks in each family outperform the other well-known filter banks from the same family. Some of the work described in this chapter has also been presented in [46].

3.1 Introduction

Filter banks have proven to be an extremely valuable tool for image coding applications [20, 37, 34]. In order to be effective in such applications, however, a filter bank must typically have a number of desirable characteristics, namely, PR, linear phase, high coding gain [23], good frequency selectivity, and certain vanishing-

moment properties. To design filter banks having all of the preceding characteristics is a challenging task. In this chapter, we propose a new optimal design method based on [10] for designing high-performance separable filter banks with all of the aforementioned desirable characteristics. Techniques for making tradeoffs between various design criteria are also presented.

The remainder of this chapter is structured as follows. We begin, in Section 3.2, by introducing our proposed design method. The filter bank structure and design criteria are introduced. Then, an abstract optimization problem is presented. Next, three general approaches for solving the abstract optimization problem are proposed. After a comparison of the three approaches, one particular approach is selected for our filter-bank design method. In Section 3.3, various experimental results are presented. The selection of design parameters is first discussed. By choosing suitable parameters, a good tradeoff between different design criteria is achieved. Next, the best choice of objective function is determined based on experimentation. Several optimal filter banks are designed and shown to outperform the well-known 9/7 filter bank from the JPEG-2000 standard. Finally, filter banks are designed for various families of two-channel PR UMD filter banks. The designed filter banks in each family are shown to outperform previously-known filter banks from the same family.

3.2 Design Method

In the sections that follow, we present our proposed design method in detail. First, we introduce the appropriate design criteria. Since the lifting parametrization is employed in our design method, all criteria are first related to the lifting parametrization. Then, based on our design criteria, an abstract optimization problem with three different cases is proposed. Next, we present an alternative form for the most complex of these cases. Finally, we propose three general approaches to solve the abstract optimization problem. After a comparison of the three approaches, one approach is selected for use in our design method.

3.2.1 Design Criteria

As mentioned earlier, high-performance filter banks need to have several desirable characteristics, namely PR, linear phase, high coding gain, good frequency selectivity, and certain vanishing moment properties. The PR property ensures that the reconstructed signals have no distortion in the absence of quantization, which is especially important in image coding applications. The linear phase property is also often quite desirable. Visually annoying distortion, due to frequency selective delays, is avoided by the linear phase property. High coding gain aims at achieving good energy compaction. In other words, high coding gain results in many

transform coefficients either being zero or having small magnitude, allowing for more efficient compression. Good frequency selectivity results in better separation of high and low frequencies, and therefore reduces aliasing effects. Having a certain number of vanishing moments helps the highpass analysis filter to annihilate polynomials. Therefore, signals that are well approximated with a certain order polynomial can be efficiently represented, due to a reduced number of nonzero transform coefficients. In what follows, we present the filter-bank structure used in our design method, and relate the filter-bank design criteria to the structure.

3.2.2 Lifting Parametrization of Filter Banks

In our design method, rather than representing a filter bank in its canonical form as shown in Figure 2.5 (on page 12), we instead use the lifting framework as depicted in Figure 2.7 (on page 14). The use of the lifting framework has a number of advantages over the canonical form. The key benefit, however, is that the PR condition is automatically satisfied. Additionally, the linear phase requirement can be easily met by choosing the lifting filters $\{\mathrm{F}_k\}_{k=0}^{2\lambda-1}$ to have certain symmetry properties, as we shall see shortly. Since the PR and linear-phase conditions can be imposed via the lifting framework, there is no need for explicit optimization constraints to ensure that these conditions are satisfied. This greatly reduces the complexity of the subsequent optimization problem.

As suggested above, the linear-phase condition can be easily imposed through a clever choice of the lifting filters $\{\mathrm{F}_k\}_{k=0}^{2\lambda-1}$. It can be shown [3] that if the $\{F_k(z)\}_{k=0}^{2\lambda-1}$ are chosen to be of either of the following two forms, then the resulting filter bank will have linear phase:

$$F_k(z) = \begin{cases} \sum_{i=0}^{(L_k-2)/2} a_{k,i}(z^{-i} + z^{i+1}) & \text{for even } k \\ \sum_{i=0}^{(L_k-2)/2} a_{k,i}(z^{-i-1} + z^{i}) & \text{for odd } k \end{cases} \tag{3.1a}$$

or

$$F_k(z) = \begin{cases} -1 & \text{for } k = 0 \\ \frac{1}{2} + \tilde{F}_1(z) & \text{for } k = 1 \\ \tilde{F}_k(z) & \text{for } k \geq 2, \end{cases} \tag{3.1b}$$

where $\tilde{F}_k(z) = \sum_{i=1}^{(L_k-1)/2} \tilde{a}_{k,i}(z^{-i} + z^{i})$, L_k is the length of the lifting filter F_k, and the $\{L_k\}$ are all even and all odd in (3.1a) and (3.1b), respectively. Since the lifting-filter coefficients have certain symmetry properties, some coefficients can be derived from others. Let $\boldsymbol{x}$ denote the vector of independent lifting-filter coefficients. In $\boldsymbol{x}$, the $\{a_{k,i}\}$ and $\{\tilde{a}_{k,i}\}$ in (3.1) are presented in lexicographic order. In other words, coefficients are sorted first by increasing lifting-filter index k and then by increasing i. For example, the vector $\boldsymbol{x}$ of independent

lifting-filter coefficients for a filter bank with the lifting filters $F_0(z) = \sum_{i=0}^{2} a_{0,i}(z^{-i} + z^{i+1})$ and $F_1(z) = \sum_{i=0}^{1} a_{1,i}(z^{-i-1} + z^{i})$ is denoted as $\boldsymbol{x} = [a_{0,0}\ a_{0,1}\ a_{0,2}\ a_{1,0}\ a_{1,1}]^T$. Equation (3.1a) parameterizes all PR linear-phase **finite-length impulse response** (FIR) filter banks with odd-length analysis/synthesis filters (up to a normalization), while (3.1b) parameterizes only a subset of all PR linear-phase FIR filter banks with even-length analysis/synthesis filters (up to a normalization). Consequently, the parametrization in (3.1a) has more potential to lead to good filter banks than the parametrization in (3.1b). This suspicion will also be confirmed later in Section 3.3 by experimental evidence.

With the lifting framework, the synthesis filters are completely determined by the analysis filters as given in (2.22). Therefore, we focus primarily on the design of the analysis side of the filter bank in what follows. Since we have elected to use a lifting parametrization for our later optimization problem formulation, we need to relate the analysis-filter frequency responses, vanishing-moment properties, and coding gain to the lifting-filter coefficients. In the case of the frequency responses and the moment properties of primal and dual wavelet coefficient sequences, these relationships can be derived in a straightforward manner via (2.18), (2.22), (3.1), (2.29), and (2.30). In the case of the coding gain, the relationship can be derived as follows. First, we determine the filters of the equivalent nonuniform filter bank, shown in Figure 2.6 (on page 13), via (2.18), (3.1), and the noble identities. Then, one can show [23] that the coding gain G_{SBC} is given by

$$G_{\mathsf{SBC}} = \prod_{k=0}^{L-1} \left(\frac{\alpha_k}{A_k B_k}\right)^{\alpha_k}, \quad \text{where} \tag{3.2a}$$

$$A_k = \sum_{m\in\mathbb{Z}} h'_{\mathsf{h}k}(m) \sum_{n\in\mathbb{Z}} h'_{\mathsf{v}k}(n) \sum_{p\in\mathbb{Z}} h'_{\mathsf{h}k}(p) \sum_{q\in\mathbb{Z}} h'_{\mathsf{v}k}(q) r(m-p, n-q),$$

$$B_k = \alpha_k \sum_{m\in\mathbb{Z}} g'^2_{\mathsf{h}k}(m) \sum_{n\in\mathbb{Z}} g'^2_{\mathsf{v}k}(n),$$

$$\alpha_k = (M_{\mathsf{h}k} M_{\mathsf{v}k})^{-1},$$

and r is the normalized autocorrelation function. Depending on the most appropriate image model, r is chosen as follows:

$$r(x,y) = \begin{cases} \rho^{|x|+|y|} & \text{for separable model} \\ \rho^{\sqrt{x^2+y^2}} & \text{for isotropic model,} \end{cases} \tag{3.2b}$$

where ρ is a correlation coefficient (typically, $0.9 \le \rho \le 0.95$). In all of our subsequent work, we assume $\rho = 0.95$ for both separable and isotropic models.

3.2.3 Abstract Optimization Problem

As indicated earlier, we seek to design filter banks having numerous desirable characteristics, namely, PR, linear phase, high coding gain, good frequency selectivity, and certain prescribed vanishing-moment proper-

ties. Since the PR and linear-phase properties are structurally imposed via the lifting framework, we need not consider them further. Thus, the design problem at hand reduces to one explicitly involving only coding gain, frequency selectivity, and vanishing-moment properties.

Now, let us consider the formulation of the design problem as an optimization. Recall that we use $\boldsymbol{x}$ to denote the vector of independent lifting-filter coefficients. We choose G, a measure related to coding gain, as the function to maximize. Let G_{sep} and G_{iso} denote the coding gain (in dB) obtained from (3.2) using the separable and isotropic models, respectively. In our work, we consider three possible choices for G as given by

$$G(\boldsymbol{x}) = \begin{cases} G_{\mathsf{sep}}(\boldsymbol{x}) & \text{separable only} \quad \text{(case 1)} \quad \text{(3.3a)} \\ G_{\mathsf{iso}}(\boldsymbol{x}) & \text{isotropic only} \quad \text{(case 2)} \quad \text{(3.3b)} \\ \min\{G_{\mathsf{sep}}(\boldsymbol{x}), G_{\mathsf{iso}}(\boldsymbol{x})\} & \text{joint} \quad \text{(case 3).} \quad \text{(3.3c)} \end{cases}$$

That is, we consider the maximization of each of the separable and isotropic coding gains individually as well as the joint maximization of both coding gains. Case 3 in (3.3) is motivated by the observation that many images are nonstationary, exhibiting both separable and isotropic behavior in different regions. Thus, we might suspect that there is an advantage to having both coding gains high.

The remaining filter bank properties are handled as constraints. To quantify the frequency selectivity of the analysis filters, we employ a stopband-energy measure. In particular, we define the stopband energy of the analysis filter H_k as

$$b_k(\boldsymbol{x}) \triangleq \int_{S_k} |\hat{h}_k(\omega, \boldsymbol{x})|^2 d\omega, \quad k \in \{0,1\}, \tag{3.4}$$

where $S_0 = [\pi - \omega_b, \pi]$, $S_1 = [0, \omega_b]$, $\hat{h}_k$ denotes the frequency response of H_k, and ω_b denotes the stopband width of the analysis filters. In passing, we note that a choice of $\omega_b = \frac{3\pi}{8}$ is made in our work. The reason we choose to use only a stopband constraint is twofold. First, limiting the stopband energy leakage can be quite effective in avoiding aliasing. Second, since the filter banks in which we are interested have short analysis/synthesis filters, the number of degrees of freedom for the filter-bank design is small. If the passband response is also constrained, the room left for other criteria to be good may become too small, resulting in poorer designs.

To facilitate the introduction of moment constraints, we define the moment-constraint functions

$$c_k(\boldsymbol{x}) \triangleq \|\boldsymbol{m}_k(\boldsymbol{x})\|, \quad k \in \{1,2,\ldots,n\}, \tag{3.5}$$

where $\boldsymbol{m}_k$ is a ν_k-dimensional vector function with its elements corresponding to the moments of interest (i.e., moments that are to be constrained). Each $\boldsymbol{m}_k$ may contain only one moment (i.e., $\nu_k = 1$) or a group

of several moments (i.e., $\nu_k > 1$). In this way, moments can be controlled either individually or jointly. Each moment of primal and dual wavelet coefficient sequences can be computed by (2.29) and (2.30) (on page 18), respectively.

Above, we introduced the objective function (3.3) to be maximized. Conventionally, however, an optimization problem is usually stated in terms of minimizing (rather than maximizing) an objective function. Because the maximizer of a function is equivalent to the minimizer of the negative of the function, we now write our optimization problem as a minimization. Combining (3.3), (3.4), and (3.5), we obtain the following abstract optimization problem to be solved:

$$\text{minimize} \quad -G(\boldsymbol{x}) \tag{3.6a}$$

$$\text{subject to:} \quad b_k(\boldsymbol{x}) \leq \varepsilon_k, \;\; k \in \{0,1\} \;\; \text{and} \tag{3.6b}$$

$$c_k(\boldsymbol{x}) \leq \gamma_k, \;\; k \in \{1,2,\ldots,n\}, \tag{3.6c}$$

where the $\{\varepsilon_k\}$ and $\{\gamma_k\}$ are strictly positive tolerances for the stopband-energy and moment constraints. Since the $\{\gamma_k\}$ are strictly positive, we do not impose exact vanishing-moment conditions. Rather, we only ensure that the moments in question are very small. In a practical sense, there is no significant disadvantage to allowing moments to deviate slightly from zero, as true vanishing-moment properties are usually lost during implementation anyhow, due to finite-precision effects. This more relaxed form of moment constraint is, in fact, advantageous, resulting in more design flexibility and therefore potentially better designs.

3.2.4 Alternative Form of the Third Case of the Abstract Optimization Problem

In the abstract optimization problem (3.6), there are three cases for the objective function (3.3). Cases 1 and 2 (i.e., the separable only and isotropic only cases) are already expressed as standard types of minimization problems. Case 3 is indeed a min-max problem. For the purposes of implementation, however, case 3 needs to be converted to a standard type of minimization problem. Since for all $\alpha, \beta \in \mathbb{R}$, $-\min\{\alpha,\beta\} = \max\{-\alpha,-\beta\}$, the objective function in case 3 can be rewritten as minimize $\max\{-G_{\mathsf{sep}}(\boldsymbol{x}), -G_{\mathsf{iso}}(\boldsymbol{x})\}$. Then, we further transform this min-max problem into the following standard minimization problem:

$$\text{minimize} \quad t \tag{3.7a}$$

$$\text{subject to:} \quad -G_{\mathsf{sep}}(\boldsymbol{x}) \leq t, \tag{3.7b}$$

$$-G_{\mathsf{iso}}(\boldsymbol{x}) \leq t, \tag{3.7c}$$

$$b_k(\boldsymbol{x}) \leq \varepsilon_k, \;\; k \in \{0,1\}, \;\; \text{and} \tag{3.7d}$$

$$c_k(\boldsymbol{x}) \leq \gamma_k, \;\; k \in \{1,2,\ldots,n\}, \tag{3.7e}$$

where t is an auxiliary variable. By introducing the new augmented vector variable $\tilde{\boldsymbol{x}} \triangleq [t \quad \boldsymbol{x}]^T$, case 3 of problem (3.6) can be trivially cast as an optimization with respect to $\tilde{\boldsymbol{x}}$ as follows:

$$\text{minimize} \quad [1 \quad \boldsymbol{0}]\tilde{\boldsymbol{x}} \tag{3.8a}$$

$$\text{subject to:} \quad -G_{\mathsf{sep}}(\boldsymbol{x}) \leq [1 \quad \boldsymbol{0}]\tilde{\boldsymbol{x}}, \tag{3.8b}$$

$$-G_{\mathsf{iso}}(\boldsymbol{x}) \leq [1 \quad \boldsymbol{0}]\tilde{\boldsymbol{x}}, \tag{3.8c}$$

$$b_k(\boldsymbol{x}) \leq \varepsilon_k, \;\; k \in \{0,1\}, \;\; \text{and} \tag{3.8d}$$

$$c_k(\boldsymbol{x}) \leq \gamma_k, \;\; k \in \{1,2,\ldots,n\}. \tag{3.8e}$$

For notational convenience, we still use $\boldsymbol{x}$ in some places in (3.8) instead of the equivalent expression in terms of $\tilde{\boldsymbol{x}}$ (i.e., $\boldsymbol{x} = [\boldsymbol{0} \quad \boldsymbol{I}]\tilde{\boldsymbol{x}}$).

3.2.5 Approaches to Solve the Abstract Optimization Problem

For the optimal design, the formulation of the abstract optimization problem is given in (3.6), with a complex case (case 3) reformulated as (3.8). Unfortunately, the abstract optimization problem (3.6) is highly nonlinear and somewhat difficult to solve directly. For this reason, several methods based on iterative solutions of reduced-order problems are considered.

Before introducing various approaches for solving the abstract optimization problems, we first give the general iterative reduced-order optimization method of the abstract optimization problem in what follows.

Algorithm 3.1 (General iterative reduced-order optimization method)**.**

Step 1. Let $i := 0$. Choose an initial point $\boldsymbol{x}_i$. Initialize the tolerances τ_1 and τ_2.

Step 2(a). For the ith iteration, we use a Taylor series expansion to approximate each of the $\{b_k\}$, $\{c_k\}$, and $G_{\mathsf{sep}}(\boldsymbol{x})$ and/or $G_{\mathsf{iso}}(\boldsymbol{x})$. That is, each function of $\boldsymbol{x}$, say $f(\boldsymbol{x})$, is approximated by its linear or quadratic approximation as $f(\boldsymbol{x}_i) + \nabla^T f(\boldsymbol{x}_i)\boldsymbol{\delta}$ or $f(\boldsymbol{x}_i) + \nabla^T f(\boldsymbol{x}_i)\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T\{\nabla[\nabla^T f(\boldsymbol{x}_i)]\}\boldsymbol{\delta}$, respectively, where $\boldsymbol{x}_i$ is a constant vector, and the new variable $\boldsymbol{\delta}$ is a perturbation about $\boldsymbol{x}_i$. An additional constraint $\|\boldsymbol{\delta}\| \leq \beta$ needs to be imposed in the reduced-order optimization problem to ensure a solution in the vicinity of the operating point $\boldsymbol{x}_i$.

Step 2(b). Solve the reduced-order optimization problem in the variable $\boldsymbol{\delta}$. Let $\boldsymbol{\delta}_i$ be the corresponding optimal solution.

Step 3. Set $\boldsymbol{x}_{i+1} := \boldsymbol{x}_i + \boldsymbol{\delta}_i$.

Step 4. If either $|G_{\boldsymbol{x}_{i+1}} - G_{\boldsymbol{x}_i}| \leq \tau_1$ or $\|\boldsymbol{\delta}_i\|_2 \leq \tau_2$ is satisfied, then output the solution $\boldsymbol{x}^* = \boldsymbol{x}_{i+1}$ and stop. Otherwise, set $i := i+1$ and go to step 2.

In step 2(a) of the above algorithm, since closed-form expressions for the derivatives of $G_{\mathsf{sep}}(\boldsymbol{x})$ and/or $G_{\mathsf{iso}}(\boldsymbol{x})$ are extremely difficult to obtain, numerical approximation is used instead. In the case of the less nonlinear functions $\{b_k\}$ and $\{c_k\}$, closed-form expressions are used for derivative computation.

Since a Taylor series expansion $f(\boldsymbol{x}) = f(\boldsymbol{x}_i + \boldsymbol{\delta}) = f(\boldsymbol{x}_i) + \nabla^T f(\boldsymbol{x}_i)\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T\{\nabla[\nabla^T f(\boldsymbol{x}_i)]\}\boldsymbol{\delta} + R(\boldsymbol{\delta})$ is used in the approximation of the original function $f(\boldsymbol{x})$, high order terms $\frac{1}{2}\boldsymbol{\delta}^T\{\nabla[\nabla^T f(\boldsymbol{x}_i)]\}\boldsymbol{\delta} + R(\boldsymbol{\delta})$ or $R(\boldsymbol{\delta})$ are negligible only when $\|\boldsymbol{\delta}\|$ is sufficiently small. To ensure obtaining good approximations of the original functions, we must impose an additional constraint $\|\boldsymbol{\delta}\| \leq \beta$ to the optimization in order to limit the approximation of f in the neighborhood of the operating point $\boldsymbol{x}_i$ (i.e., for small $\|\boldsymbol{\delta}\|$). The norm function used in $\|\boldsymbol{\delta}\| \leq \beta$ is either an l_2 or l_∞ norm in our work.

In step 3 of the above algorithm, we update our operating point. As the solution in step 2(b) is limited to the neighborhood of the operating point, we repeat the process in steps 2 to 4 unless one of the termination conditions is satisfied. This permits solutions farther away from the current operating point to be found. Note that for case 3 (i.e., in (3.8)) of the abstract optimization problem, the vector variables $\boldsymbol{x}$ and $\boldsymbol{\delta}$ in the above iterative optimization method become the augmented vector variables $\tilde{\boldsymbol{x}} \triangleq [t \quad \boldsymbol{x}]^T$ and $\tilde{\boldsymbol{\delta}} \triangleq [t \quad \boldsymbol{\delta}]^T$, respectively.

We notice that the building block of step 2 of Algorithm 3.1 can be formulated differently by using various orders of approximation for different function $f(\boldsymbol{x})$. We now present two approaches for handling the optimization problem in step 2, namely using linear programming and second-order cone programming.

Formulation of Linear Programming and Convex Quadratic Programming Problems

We begin to discuss the formulation of step 2 for the simplest approach, i.e., a linear programming scheme. A linear programming problem has the general form:

$$\text{minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} \tag{3.9a}$$

$$\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \tag{3.9b}$$

where $\boldsymbol{c} \in R^{n\times 1}$ with $\boldsymbol{c} \neq \boldsymbol{0}$, $\boldsymbol{A} \in R^{p\times n}$, and $\boldsymbol{b} \in R^{p\times 1}$ are given, and $\boldsymbol{A}$ has full row rank. To formulate step 2 of the optimization design problem into a linear programming problem, the three cases of the abstract optimization problem need to be discussed. In what follows we give the formulations of step 2 of Algorithm 3.1 for these three cases.

We start with the first and second cases. We begin our formulation by deriving the functions $\{b_k\}$, $\{c_k\}$, and $G_{\mathsf{sep}}(\boldsymbol{x})$ and/or $G_{\mathsf{iso}}(\boldsymbol{x})$ via (3.4), (3.5), (3.2), (2.18), (2.22), (3.1), (2.29), and (2.30). Then we find Taylor series approximations to $\{b_k\}$, $\{c_k\}$, and $G_{\mathsf{sep}}(\boldsymbol{x})$ and/or $G_{\mathsf{iso}}(\boldsymbol{x})$. Next, we substitute these approximated

functions into (3.6), and add an additional constraint $\|\boldsymbol{\delta}\| \leq \beta$ to the optimization problem. Finally, the problem can be rewritten into a linear programming problem by applying simple transforms to the l_∞ norms as discussed shortly. The resulting linear programming problem from cases 1 and 2 of the abstract optimization problem (3.6) is given as follows:

$$\begin{aligned}
\text{minimize} \quad & -[\nabla^T G(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} && \text{(3.10a)}\\
\text{subject to:} \quad & -2\big[\int_{S_k} \hat{h}_k(\omega,\boldsymbol{x})\nabla_{\boldsymbol{x}}^T\hat{h}_k(\omega,\boldsymbol{x})d\omega\big]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq b_k(\boldsymbol{x}_i)-\varepsilon_k,\ \ k\in\{0,1\}, && \text{(3.10b)}\\
& [\nabla^T \boldsymbol{m}_k(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq -\boldsymbol{m}_k(\boldsymbol{x}_i)-\gamma_k,\ \ k\in\{1,2,\ldots,n\}, && \text{(3.10c)}\\
& -[\nabla^T \boldsymbol{m}_k(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq \boldsymbol{m}_k(\boldsymbol{x}_i)-\gamma_k,\ \ k\in\{1,2,\ldots,n\}, && \text{(3.10d)}\\
& \boldsymbol{\delta} \geq -\beta\mathbf{1},\ \text{ and} && \text{(3.10e)}\\
& -\boldsymbol{\delta} \geq -\beta\mathbf{1}, && \text{(3.10f)}
\end{aligned}$$

where $\boldsymbol{\delta}$ is a perturbation from the operating point $\boldsymbol{x}_i$, and β is a positive real number. If an l_∞-norm constraint $\|\boldsymbol{x}\| \leq c$ is used, the constraint is split into two constraints $\boldsymbol{x} \leq c\mathbf{1}$ and $-\boldsymbol{x} \leq c\mathbf{1}$ in our formulation. Equations (3.10c) and (3.10d) are equivalent to the linear approximations of the moment constraints $c_k(\boldsymbol{x}) \triangleq \|\boldsymbol{m}_k(\boldsymbol{x})\| \leq \gamma_k$, where $k \in \{1,2,\ldots,n\}$, with l_∞ norms being applied. Equations (3.10e) and (3.10f) are equivalent to $\|\boldsymbol{\delta}\| \leq \beta$ using an l_∞ norm. The preceding problem (3.10) is of the form of (3.9), where

$$\boldsymbol{A} = \begin{bmatrix}
-2\big[\int_{S_0} \hat{h}_0(\omega,\boldsymbol{x})\nabla_{\boldsymbol{x}}^T\hat{h}_0(\omega,\boldsymbol{x})d\omega\big]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\\
-2\big[\int_{S_1} \hat{h}_1(\omega,\boldsymbol{x})\nabla_{\boldsymbol{x}}^T\hat{h}_1(\omega,\boldsymbol{x})d\omega\big]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\\
[\nabla^T \boldsymbol{m}_1(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\\
\vdots\\
[\nabla^T \boldsymbol{m}_n(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\\
-[\nabla^T \boldsymbol{m}_1(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\\
\vdots\\
-[\nabla^T \boldsymbol{m}_n(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\\
\boldsymbol{I}\\
-\boldsymbol{I}
\end{bmatrix}, \quad
\boldsymbol{b} = \begin{bmatrix}
b_0(\boldsymbol{x}_i)-\varepsilon_0\\
b_1(\boldsymbol{x}_i)-\varepsilon_1\\
-\boldsymbol{m}_1(\boldsymbol{x}_i)-\gamma_1\\
\vdots\\
-\boldsymbol{m}_n(\boldsymbol{x}_i)-\gamma_n\\
\boldsymbol{m}_1(\boldsymbol{x}_i)-\gamma_1\\
\vdots\\
\boldsymbol{m}_n(\boldsymbol{x}_i)-\gamma_n\\
-\beta\mathbf{1}\\
-\beta\mathbf{1}
\end{bmatrix},$$

and the optimization is performed with respect to the variable $\boldsymbol{\delta}$.

The alternative form of the abstract optimization problem (3.8) associated with case 3 of (3.6) can be

transformed to yield the following:

$$\text{maximize} \quad [1 \quad \mathbf{0}]\tilde{\boldsymbol{\delta}} \tag{3.11a}$$

$$\text{subject to:} \quad [\nabla^T G_{\text{iso}}(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq -[1 \quad \mathbf{0}]\tilde{\boldsymbol{\delta}} - G_{\text{iso}}(\boldsymbol{x}_i), \tag{3.11b}$$

$$[\nabla^T G_{\text{sep}}(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq -[1 \quad \mathbf{0}]\tilde{\boldsymbol{\delta}} - G_{\text{sep}}(\boldsymbol{x}_i), \tag{3.11c}$$

$$-2\Big[\int_{S_k} \hat{h}_k(\omega,\boldsymbol{x})\nabla_{\boldsymbol{x}}^T \hat{h}_k(\omega,\boldsymbol{x})d\omega\Big]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq b_k(\boldsymbol{x}_i) - \varepsilon_k, \;\; k \in \{0,1\}, \tag{3.11d}$$

$$[\nabla^T \boldsymbol{m}_k(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq -\boldsymbol{m}_k(\boldsymbol{x}_i) - \gamma_k, \;\; k \in \{1,2,\ldots,n\}, \tag{3.11e}$$

$$-[\nabla^T \boldsymbol{m}_k(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} \geq \boldsymbol{m}_k(\boldsymbol{x}_i) - \gamma_k, \;\; k \in \{1,2,\ldots,n\}, \tag{3.11f}$$

$$\boldsymbol{\delta} \geq -\beta\mathbf{1}, \;\; \text{and} \tag{3.11g}$$

$$-\boldsymbol{\delta} \geq -\beta\mathbf{1}, \tag{3.11h}$$

where $\tilde{\boldsymbol{\delta}} \triangleq [t \quad \boldsymbol{\delta}]^T$ is the new augmented vector variable. For notational convenience, we still use $\boldsymbol{\delta}$ in some places in (3.11) instead of the equivalent expression in terms of $\tilde{\boldsymbol{\delta}}$ (i.e., $\boldsymbol{\delta} = [\mathbf{0} \quad \boldsymbol{I}]\tilde{\boldsymbol{\delta}}$).

Incidentally, we also considered formulating step 2 of Algorithm 3.1 as a convex quadratic programming (QP) problem. A convex QP problem has the general form:

$$\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{p} \tag{3.12a}$$

$$\text{subject to:} \;\; \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \tag{3.12b}$$

where $\boldsymbol{H} \in R^{n\times n}$ is positive semidefinite, and $\boldsymbol{A} \in R^{p\times n}$ has full row rank. The objective function of a convex QP problem needs to be a convex quadratic function. Therefore, the matrix $\boldsymbol{H}$ in (3.12a) is required to be positive semidefinite. Since the matrix $\boldsymbol{H}$ associated with the quadratic approximation of the coding gain function $G(\boldsymbol{x})$ may not be positive semidefinite, for practical reasons, we do not consider the convex QP formulation further.

Formulation of SOCP Problem

Another commonly used optimization approach is second-order cone programming (SOCP) [25]. The step 2 of Algorithm 3.1 can be cast as a SOCP problem. A SOCP problem has the general form:

$$\text{minimize} \;\; \boldsymbol{b}^T\boldsymbol{x} \tag{3.13a}$$

$$\text{subject to:} \;\; \|\boldsymbol{A}_k^T\boldsymbol{x} + \boldsymbol{c}_k\| \leq \boldsymbol{b}_k^T\boldsymbol{x} + d_k, \quad k \in \{1,2,\ldots,K\}, \tag{3.13b}$$

where $\boldsymbol{b} \in R^{n\times 1}$, $\boldsymbol{x} \in R^{n\times 1}$, $\boldsymbol{A}_k \in R^{n\times m_k}$, $\boldsymbol{c}_k \in R^{m_k\times 1}$, $\boldsymbol{b}_k \in R^{n\times 1}$, and $d_k \in R$ for $1 \leq k \leq K$. To express step 2 of Algorithm 3.1 as a SOCP problem, we first use a Taylor series expansion to approximate the $\{b_k\}$,

$\{c_k\}$, and $G_{\mathsf{sep}}(\boldsymbol{x})$ and/or $G_{\mathsf{iso}}(\boldsymbol{x})$, such that the approximation of $G_{\mathsf{sep}}(\boldsymbol{x})$ and/or $G_{\mathsf{iso}}(\boldsymbol{x})$ is linear, and the approximations of the $\{b_k\}$ and $\{c_k\}$ are at most quadratic with respect to $\boldsymbol{\delta}$. This reduced-order problem can then be expressed as a SOCP problem.

It can be shown that cases 1 and 2 of the abstract optimization problem (3.6) can be expressed in the following form:

$$\begin{aligned}
\text{minimize} \quad & -[\nabla^T G(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta} && \text{(3.14a)}\\
\text{subject to:} \quad & \|\boldsymbol{Q}_k^{1/2}(\boldsymbol{x}_i)\boldsymbol{\delta}+\boldsymbol{q}_k(\boldsymbol{x}_i)\| \le \varepsilon_k - b_k(\boldsymbol{x}_i)+\boldsymbol{q}_k^T(\boldsymbol{x}_i)\boldsymbol{q}_k(\boldsymbol{x}_i),\ \ k\in\{0,1\}, && \text{(3.14b)}\\
& \|[\nabla^T \boldsymbol{m}_k(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta}+\boldsymbol{m}_k(\boldsymbol{x}_i)\| \le \gamma_k,\ \ k\in\{1,2,\ldots,n\},\ \text{and} && \text{(3.14c)}\\
& \|\boldsymbol{\delta}\| \le \beta, && \text{(3.14d)}
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{Q}_k(\boldsymbol{x}) &= \int_{S_k} [\nabla_{\boldsymbol{x}}\hat{h}_k(\omega,\boldsymbol{x})]\nabla_{\boldsymbol{x}}^T\hat{h}_k(\omega,\boldsymbol{x})d\omega,\\
\boldsymbol{q}_k(\boldsymbol{x}) &= \boldsymbol{Q}_k^{-1/2}(\boldsymbol{x})\int_{S_k}\hat{h}_k(\omega,\boldsymbol{x})\nabla_{\boldsymbol{x}}^T\hat{h}_k(\omega,\boldsymbol{x})d\omega,
\end{aligned}$$

and $\boldsymbol{\delta}$ is a perturbation from the operating point $\boldsymbol{x}_i$. Clearly, the preceding problem is of the form of (3.13), where the optimization is performed with respect to the variable $\boldsymbol{\delta}$. Note that in (3.14d), an l_2 norm is applied for the constraint on $\boldsymbol{\delta}$.

To complete our discussion, we must now consider case 3 of the abstract optimization problem (3.6) as re-expressed in (3.8). In this case, by introducing the new augmented vector variable $\tilde{\boldsymbol{\delta}} \triangleq [t \quad \boldsymbol{\delta}]^T$, the reduced-order optimization problem at hand can be trivially cast into a SOCP problem with respect to $\tilde{\boldsymbol{\delta}}$, as given by

$$\begin{aligned}
\text{maximize} \quad & [1 \quad \mathbf{0}]\tilde{\boldsymbol{\delta}}\\
\text{subject to:} \quad & [\nabla^T G_{\mathsf{iso}}(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta}+G_{\mathsf{iso}}(\boldsymbol{x}_i)-[1 \quad \mathbf{0}]\tilde{\boldsymbol{\delta}} \ge 0,\\
& [\nabla^T G_{\mathsf{sep}}(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\boldsymbol{\delta}+G_{\mathsf{sep}}(\boldsymbol{x}_i)-[1 \quad \mathbf{0}]\tilde{\boldsymbol{\delta}} \ge 0,\\
& \|\tilde{\boldsymbol{Q}}_k(\boldsymbol{x}_i)\tilde{\boldsymbol{\delta}}+\tilde{\boldsymbol{q}}_k(\boldsymbol{x}_i)\| \le \varepsilon_k - b_k(\boldsymbol{x}_i)+\boldsymbol{q}_k^T(\boldsymbol{x}_i)\boldsymbol{q}_k(\boldsymbol{x}_i),\ \ k\in\{0,1\},\\
& \|[\nabla^T \tilde{\boldsymbol{m}}_k(\boldsymbol{x})]\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\tilde{\boldsymbol{\delta}}+\tilde{\boldsymbol{m}}_k(\boldsymbol{x}_i)\| \le \gamma_k,\ \ k\in\{1,2,\ldots,n\},\ \text{and}\\
& \|\boldsymbol{\delta}\| \le \beta,
\end{aligned}$$

where

$$\tilde{\boldsymbol{Q}}_k(\boldsymbol{x}) = [\mathbf{0} \quad \boldsymbol{Q}_k^{1/2}(\boldsymbol{x})],\quad \tilde{\boldsymbol{q}}_k(\boldsymbol{x}) = [0 \quad \boldsymbol{q}_k(\boldsymbol{x})]^T,\quad \tilde{\boldsymbol{m}}_k(\boldsymbol{x}) = [0 \quad \boldsymbol{m}_k(\boldsymbol{x})]^T,$$

$$\boldsymbol{Q}_k(\boldsymbol{x}) = \int_{S_k} [\nabla_{\boldsymbol{x}} \hat{h}_k(\omega,\boldsymbol{x})] \nabla_{\boldsymbol{x}}^T \hat{h}_k(\omega,\boldsymbol{x}) d\omega, \quad \text{and} \quad \boldsymbol{q}_k(\boldsymbol{x}) = \boldsymbol{Q}_k^{-1/2}(\boldsymbol{x}) \int_{S_k} \hat{h}_k(\omega,\boldsymbol{x}) \nabla_{\boldsymbol{x}}^T \hat{h}_k(\omega,\boldsymbol{x}) d\omega.$$

For notational convenience, in the above equation, we still use $\boldsymbol{\delta}$ in some places instead of the equivalent expression in terms of $\tilde{\boldsymbol{\delta}}$ (i.e., $\boldsymbol{\delta} = [\boldsymbol{0} \quad \boldsymbol{I}]\tilde{\boldsymbol{\delta}}$).

Formulation of SQP Problem

Instead of using Algorithm 3.1 for solving the abstract optimization problem, let us now consider applying a different nonlinear optimization method. In a general nonlinear optimization problem, both the objective function and the constraints are nonlinear and not even necessarily convex. The problem has the general form:

$$\text{minimize} \quad f(\boldsymbol{x}) \tag{3.15a}$$

$$\text{subject to:} \quad c_k(\boldsymbol{x}) \geq 0, \quad k \in \{1,2,\ldots,K\}, \tag{3.15b}$$

where $f(\boldsymbol{x})$ and $c_k(\boldsymbol{x})$ are continuous and have continuous second-order partial derivatives, and the feasible region derived from (3.15b) is nonempty. Since cases 1 and 2 of the abstract optimization problem (3.6) and the alternative form (3.8) of case 3 are already in the form of (3.15), in order to get filter bank designs satisfying all of our criteria, we need to solve the general nonlinear optimization problem (3.15). Such a problem can be solved by sequential quadratic programming (SQP), which is also an iterative reduced-order approach. In a SQP problem, the objective functions or the Lagrangian functions are quadratically approximated, and the constraints are linearly approximated. For detailed algorithms for solving a SQP problem, the reader is referred to [4].

Comparison of the Three General Approaches

All three methods discussed above (i.e., linear-programming, SOCP, and SQP) provide iterative solutions to solve nonlinear optimization problems. We now compare the three methods, and choose one as the solution method for our design problem.

A linear programming problem can be solved by using either simplex [44] or interior-point methods [45]. Some schemes for solving a linear programming problem allow nonfeasible initialization (i.e., the initial point need not satisfy all of the constraints of the optimization). This offers additional flexibility in choosing the initial points. In comparison with the SOCP and SQP approaches, the linear-programming method uses the lowest-order approximation of the original problem. Therefore, we would expect in many cases that the average iteration time of linear programming is shorter than the other two approaches. The linear programming problem, however, may need many more iterations to find the solution to the original problem.

A SOCP problem can be efficiently solved by interior-point methods. The SOCP approach uses a relatively high-order approximation of the original problem, and highly probably takes less iterations but longer average iteration time than linear programming. Luckily, some well developed software such as SeDuMi [40] is available for efficiently solving SOCP problems.

Last but not least, the general nonlinear optimization problem (3.15) can be solved by a SQP approach, which uses the highest order of approximation of the original problem among the methods we compare. The SQP approach using the Lagrange-Newton method performs well when the initial point is sufficiently close to a local minimizer, but may fail to give any solution if the initial point is poorly chosen. The SQP approach with a line search is always able to find solutions with arbitrary initial points. In comparison with the linear programming and SOCP approaches, this SQP approach often takes the least number of iterations but has the longest average iteration time due to the computational complexity.

Based on the above discussion, in our design method, we chose to use the SOCP approach, which makes a good tradeoff between the execution time per iteration and the number of iterations in total. Furthermore, the availability of software for solving SOCP problems is another reason for us to choose this approach. All of the results we present in the remainder of this chapter are produced by the SOCP approach.

3.3 Experimental Results

Having introduced our design method, we would now like to present some detailed experimental results related to our method. In the sections that follow, we first introduce the test environment in which all experiments are conducted. Then, the selection of the design parameters and objective function is discussed. Several filter banks are designed by the proposed method with the selected optimal design parameters and objective function. Some of the designed filter banks are shown to outperform the well-known 9/7 filter bank from the JPEG-2000 standard. Finally, filter banks are also designed for various families of two-channel PR UMD filter banks. Both subjective and objective coding results show that the designed filter banks in each family outperform previously known filter banks from the same family.

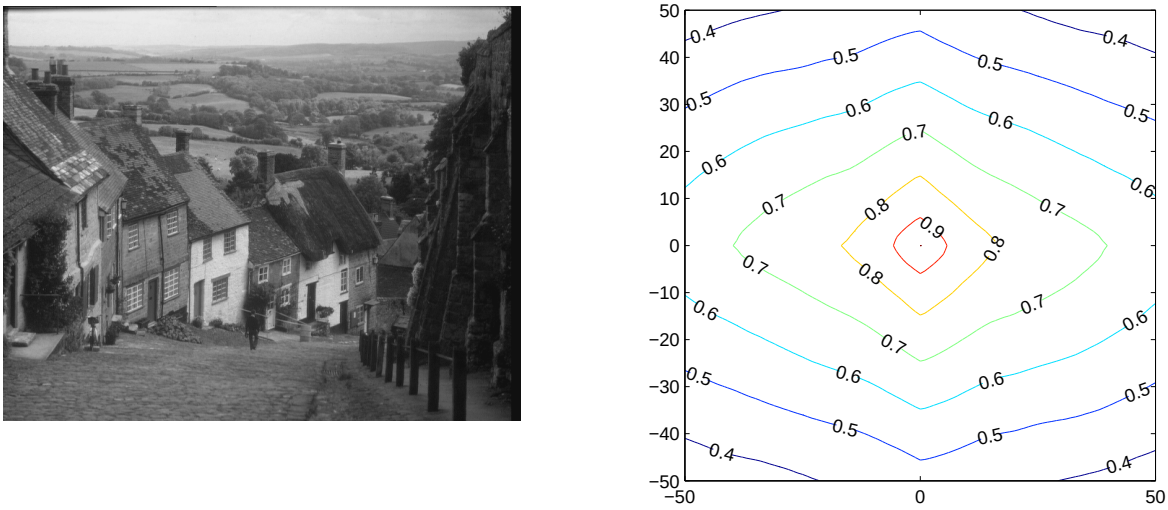
3.3.1 Test Environment

Before presenting some experimental results, we first introduce some details concerning our test environment. In all of our experiments, for test data, all of the twenty-six reasonably-sized continuous-tone grayscale images from the JPEG-2000 test set [21] were employed. Table 3.1 provides the sizes, numbers of bits per pixel, and a brief description of each image. Often, in what follows, we will focus our attention on

Table 3.1: Test images for filter bank design

Image	Size	Precision†	Description
aerial2	2048×2048	8	aerial photograph
bike	2048×2560	8	collection of objects
cafe	2048×2560	8	outside of cafe
cats	3072×2048	8	cats
chart	1688×2347	8	color chart
cmpnd1	512×768	8	mixed text and graphics
cmpnd2	5120×6624	8	mixed text and graphics
cr	1744×2048	10	computer radiology
ct	512×512	12	computer tomography
elev	1201×1201	12	fractal-like pattern
finger	512×512	8	fingerprint
gold	720×576	8	houses and countryside
hotel	720×576	8	hotel
mat	1528×1146	8	mountains
mri	256×256	11	magnetic resonance
sar1	1024×1024	16	synthetic aperture radar
sar2	800×800	12	synthetic aperture radar
seismic	512×512	8	seismic data
target	512×512	8	patterns and textures
tools	1524×1200	8	collection of tools
txtur1	1024×1024	8	aerial photograph
txtur2	1024×1024	8	aerial photograph
us	512×448	8	ultrasound
water	1465×1999	8	water and land
woman	2048×2560	8	lady
xray	2048×1680	12	x-ray

†precision in bits per sample

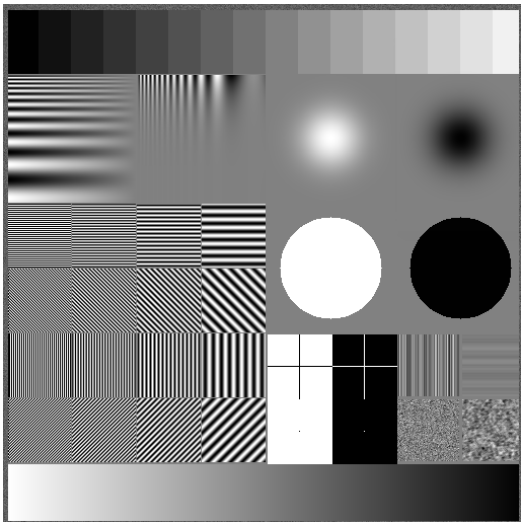


(a) (b)

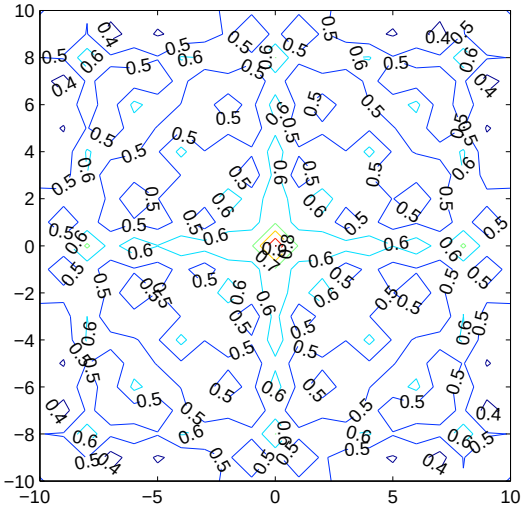
Figure 3.1: (a) The `gold` image and (b) a contour plot of its normalized autocorrelation function.

a representative subset of these images, namely, the `gold`, `sar2`, and `target` images. These three test images and the contour plots of their normalized autocorrelation functions are shown in Figures 3.1, 3.2, and 3.3. Now, we make several observations. First, the autocorrelation function of the `gold` image fits the separable model better than the isotropic model. Second, the autocorrelation function of the `sar2` image fits the isotropic model better than the separable model. Third, the autocorrelation function of the `target` image is similar to neither the separable model nor the isotropic model.

To evaluate the performance of filter banks for image coding, the EZW [37], SPIHT [34], and MIC [1] codecs were employed, all of which utilize reversible integer-to-integer transforms. Since similar results were obtained with all three codecs, we mainly present results for the MIC codec. In fact, unless explicitly indicated otherwise, all results should be assumed to be from this codec. In all of our coding experiments, a six-level wavelet decomposition was employed. In what follows, we use the notation l_0/l_1 to indicate that a filter bank has lowpass and highpass analysis filters of lengths l_0 and l_1, respectively. For a filter bank using the lifting framework, the **lifting configuration** of the filter bank refers to the number and lengths of the lifting filters. We denote the lifting configuration using the notation $\{L_0, L_1, \ldots\}$, where L_k is the length of the kth lifting filter. For example, the notation $\{1,3,5\}$ indicates a family of filter banks having three lifting filters where the first, second, and third lifting filters have the lengths of one, three, and five, respectively.



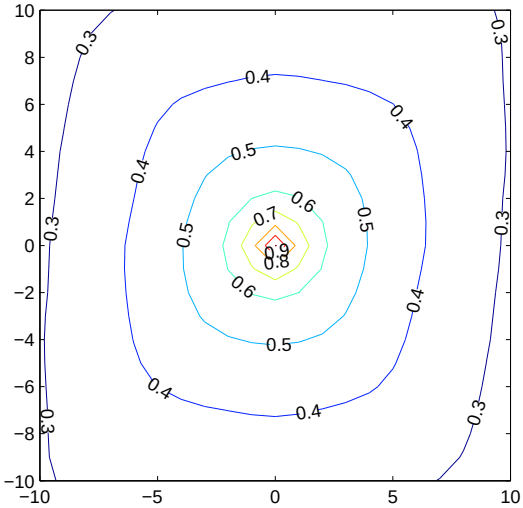
(a)



(b)

Figure 3.2: (a) The `target` image and (b) a contour plot of its normalized autocorrelation function.

(a)



(b)

Figure 3.3: (a) The `sar2` image and (b) a contour plot of its normalized autocorrelation function.

3.3.2 Design-Parameter Selection

Having introduced the test environment, we now briefly comment on the selection of several key design parameters. In the case of the frequency-selectivity constraints, an appropriate choice of tolerances $\{\varepsilon_k\}$ in (3.6b) is critical to achieving good designs. Based on our experiments, for a stopband width of $\omega_b = \frac{3\pi}{8}$, a choice of $\varepsilon_k \in [0.02, 0.14]$ is highly effective. The selection of the moment constraints (i.e., the $\{\boldsymbol{m}_k\}$ and $\{\gamma_k\}$) is also quite important. Let μ_k be the kth moment associated with the sequence of a highpass analysis or synthesis filter impulse response. Since filter banks satisfying (3.1a) or (3.1b), respectively, have the moment μ_k, $k \in \mathbb{Z}_{\text{even}}$ or $k \in \mathbb{Z}_{\text{odd}}$ automatically vanish (providing that the μ_{k-1} moment vanishes), only the remaining moments need to be considered. Again, based on our experiments, good designs satisfying (3.1a) are obtained when we constrain the zeroth moments of sequences for impulse responses of highpass analysis and synthesis filters. An effective choice is that the l_2 norm of the above moments is less than $2 \cdot 10^{-5}$.

Due to the highly nonlinear nature of the abstract optimization problem (3.6), the optimal solution found by our method will most likely not be a global minimizer. The particular solution obtained depends on the choice of the initial point $\boldsymbol{x}$. Therefore, the quality of the design can be improved by finding multiple solutions and then selecting the best one. As a practical matter, we found that an effective strategy in this regard is to consider many initial points with lifting-filter coefficients of magnitude 2 or less, as the best designs typically have coefficients in this range.

The frequency-selectivity constraints used in (3.6b) only consider the frequency responses of the lowpass and highpass analysis filters in their stopbands. The frequency responses in the passbands are not considered explicitly. Experimental results show, however, that all good filter banks designed by constraining only stopband energy also have relatively good passband frequency responses. Besides using $\frac{3\pi}{8}$ as the cutoff frequency ω_b, we also tried using values in the range of $[\frac{\pi}{5}, \frac{3\pi}{8}]$, and the performance of the optimal designs showed little variation with the change in ω_b and associated tolerances $\{\varepsilon_k\}$. In passing, we note that, in some cases, the frequency-response constraints are even inactive at the final solution.

Now let us discuss the number of decomposition levels used in the coding gain computation. Coding gain is an important quantity in our optimization-based filter bank design. The number of decomposition levels is a parameter in the coding gain formula. Although having more decomposition levels tends to increase coding gain, coding gain rapidly approaches a limiting value as the number of decomposition levels increases. Moreover, too many decomposition levels exacerbate boundary filtering problems [6] and increase computational complexity. In practice, five or six levels of decomposition is often quite effective for image coding. Therefore, five or six decomposition levels are a reasonable choice for the coding gain computation. We observe, from experimental results, that the optimizers of the coding-gain functions associated with two, three, and

four levels of decomposition do not have significant differences, although the coding-gain function values vary with different decomposition levels. To reduce the computational complexity of the filter-bank design method, we can use relatively fewer decomposition levels. Another practical technique is to refine the design from an optimal filter bank associated with few (e.g., 2 or 3) decomposition levels to that associated with many (e.g., 5) decomposition levels. The optimizers pertaining to few decomposition levels are considered as initial points pertaining to many decomposition levels. Since the optimizers for few decomposition levels are good indications of the optimizers for many decomposition levels, fewer evaluations of coding gain associated with many decomposition levels are needed with the above technique than directly optimizing coding gain with respect to many decomposition levels.

Now, we consider the adjustment of the upper bound β for the constraint $\|\boldsymbol{\delta}\| \leq \beta$ as specified in (3.14d), (3.10e), and (3.10f). When iteratively solving the optimization problem, if no solution is found at a certain iteration, the step size limit β needs to be increased to enlarge the search region (i.e., feasible region). If the reduction of the objective function is very small at a certain iteration, the step size limit β needs to be reduced, so that a more accurate approximation of the original optimization problem is achieved via the Taylor series expansion.

3.3.3 Choice of Objective Function

As indicated earlier, our design method allows for three possible objective functions as given by (3.3). We also suggested that, of these possibilities, (3.3c) might be the most desirable. Now, we experimentally show this to be the case.

To begin, for each of several different lifting configurations, we used our method to design three filter banks, one for each of the three choices of objective function in (3.3). In so doing, we were able to make an interesting observation. As it turns out, in all of our tests, optimizing with respect to either of the objective functions (3.3b) or (3.3c) always led to the same optimal design. This is due to the fact that, for filter banks with good frequency selectivity, G_{sep} always tends to be greater than G_{iso}. Therefore, maximizing (3.3c) is equivalent (in a practical sense) to maximizing (3.3b). With the above observation in mind, we combine the cases of (3.3b) and (3.3c) in the remainder of this discussion.

Herein, we consider two sets of optimal designs with analysis filter lengths of 9/7 and 6/14 (which correspond to the lifting configurations $\{2,2,2,2\}$ and $\{1,3,5\}$, respectively). The characteristics of these filter banks are shown in Table 3.2, where the suffixes "sep", "iso", and "jnt" are used to designate the optimal designs obtained using the objective functions (3.3a), (3.3b), and (3.3c), respectively. For comparison purposes, in the case of each of the two sets, we also consider a previously proposed filter bank that has an identical

Table 3.2: Characteristics of the filter banks designed using different objective functions for the 9/7 and 6/14 cases

Transform	$G^{\ddagger}_{\mathsf{sep}}$	$G^{\ddagger}_{\mathsf{iso}}$	$b_0^{\dagger}$	$b_1^{\dagger}$	Van. Mom.*
9/7-ref	14.9734	12.1781	0.0628	0.0347	4, 9.560
9/7-sep	14.9735	12.1781	0.0628	0.0347	2, 5.79E-5
9/7-iso/jnt	14.9326	12.1809	0.0570	0.0351	2, 0.0041
6/14-ref	14.8801	12.0457	0.0340	0.0241	3, 3.000
6/14-sep	15.0912	11.9285	0.0252	0.0131	1, 0.0483
6/14-iso/jnt	14.9766	12.0738	0.0212	0.0229	1, 0.0643

*index and magnitude of the first moment of dual wavelet coefficient sequence with moment magnitude greater than $2 \cdot 10^{-5}$

‡coding gain (in dB) for a six-level decomposition

†stopband energy as defined by (3.4)

lifting configuration and is known to be effective for image coding. In the table, the reference filter bank is designated by the suffix "ref".

Having produced several sets of optimal filter banks as described above, we then proceeded to compare the coding performance of the filter banks in each set. For each of the filter banks in each set, we compressed all of the twenty-six test images in a lossy manner at several bit rates. In each case, we measured the relative difference between the distortions (in PSNR) obtained with our design and the appropriate reference (i.e., "ref") filter bank. The results are summarized in statistical form in Table 3.3(a). In particular, we provide the mean and median relative differences of PSNR (with positive values corresponding to our designs outperforming the reference filter bank). As well, we indicate the percentage of cases in which our filter bank outperforms the reference filter bank. From Table 3.3(a), we can see that, in both the 9/7 and 6/14 cases, designs based on the (joint) objective function (3.3c) (designated by the suffix "jnt") have better performance relative to the reference filter bank than the designs based on the (separable-only) objective function (3.3a). In Table 3.3(b), we provide the actual distortions obtained for three representative images. Here, we can see that the filter banks with the jointly-highest coding gains perform better overall for all three images, in spite of the images having significantly different autocorrelation models. Clearly, the above experiment demonstrates that there is a potential benefit to jointly optimizing both of the separable and isotropic coding gains (i.e., using objective function (3.3c)).

Based on experimental results, for the same tolerance ε_k, the frequency-response constraints $b_k(\boldsymbol{x}) \leq \varepsilon_k$

Table 3.3: Lossy compression results for the filter banks designed using different objective functions. (a) Summary statistical results over all twenty-six test images and five bit rates. (b) Specific results for three images.

(a)

Transform	Mean (%)	Median (%)	Outperform (%)
9/7-sep	-0.0049	-0.0001	46.15
9/7-iso/jnt	0.1488	0.1070	87.69
6/14-sep	-0.5848	-0.5112	20.77
6/14-iso/jnt	0.0331	0.0279	61.54

(b)

Image	CR†	PSNR (dB)			
		9/7-sep	9/7-iso/jnt	6/14-sep	6/14-iso/jnt
gold	8	36.76	36.88	36.77	36.96
	16	33.77	33.84	33.50	33.78
	32	31.26	31.27	31.07	31.15
	64	29.15	29.15	28.82	28.83
	128	27.33	27.32	27.06	27.26
target	8	41.48	41.59	40.94	41.68
	16	33.55	33.55	32.79	32.91
	32	27.08	27.19	27.37	27.00
	64	22.70	22.84	22.39	22.43
	128	19.13	19.14	18.49	18.26
sar2	8	30.33	30.35	30.24	30.39
	16	26.61	26.62	26.49	26.49
	32	24.69	24.70	24.64	24.78
	64	23.55	23.55	23.48	23.46
	128	22.73	22.73	22.59	22.68

†compression ratio

Table 3.4: Characteristics of the various filter banks

Transform	$\{L_k\}$	$G^{\ddagger}_{\mathsf{sep}}$	$G^{\ddagger}_{\mathsf{iso}}$	$b_0^{\dagger}$	$b_1^{\dagger}$	VM*	$\hat{h}_0(0)$	$\hat{h}_0(\pi)$	$\hat{h}_1(0)$	$\hat{h}_1(\pi)$
9/7	{2,2,2,2}	14.933	12.181	0.057	0.035	2, 0.004	1.25	2.1E-6	-9.1E-6	1.60
9/11	{4,2,2}	14.928	12.112	0.111	0.043	2, 0.276	1.24	-5.6E-6	-9.4E-6	1.61
13/11	{4,2,2,2}	15.041	12.206	0.030	0.027	2, 0.068	1.20	2.2E-5	-1.9E-5	1.67
17/11	{2,2,4,4}	15.117	12.218	0.031	0.028	2, 0.337	1.19	9.9E-6	4.9E-6	1.69
13/15	{6,2,2}	14.641	12.074	0.094	0.035	2, 0.169	1.35	-5.4E-6	-9.4E-6	1.49
9/7-J	{2,2,2,2}	14.973	12.178	0.063	0.035	4, 9.560	1.23	2.3E-9	3.9E-9	1.63

*index and magnitude of the first moment of dual wavelet coefficient sequence with moment magnitude greater than $2 \cdot 10^{-5}$

‡coding gain (in dB) for a six-level decomposition

†stopband energy as defined by (3.4)

for $k \in \{0,\ 1\}$ are active at the solution points when considering the separable coding gain (i.e., (3.3a)) as the objective function, and inactive at the solution points when considering isotropic coding gain (i.e., (3.3b)) or joint separable/isotropic coding gain (i.e., (3.3c)) as the objective function. This implies that good frequency responses tend to happen together with high isotropic coding gain or high joint isotropic/separable coding gain. The preceding coincidence may explain to a certain extent why maximizing the isotropic coding gain or maximizing the joint isotropic/separable coding gain is a good choice for high-performance filter bank design.

3.3.4 Overall Optimal Designs and Coding Results

Having discussed the key design parameters and the best choice of objective function, we are now ready to introduce some optimal filter banks constructed using our proposed method. In particular, five optimal designs are presented. For all of these designs, the objective function (3.3c) was employed, as this was earlier determined to be the best choice. For comparison purposes, we also consider the well-known 9/7 filter bank from JPEG 2000 [20], which we henceforth refer to by the name 9/7-J (in order to distinguish it from other filter banks having the same analysis-filter lengths).

Several characteristics of our five optimal designs as well as the 9/7-J filter bank are shown in Table 3.4. Note that none of our designed filter banks in odd-length analysis/synthesis filter family (3.1a) have exactly-vanishing moments. The lifting coefficients of the 9/7-J filter bank are irrational numbers. In practice, we use finite-precision arithmetic in the performance evaluation of the filter bank. The characteristics of the

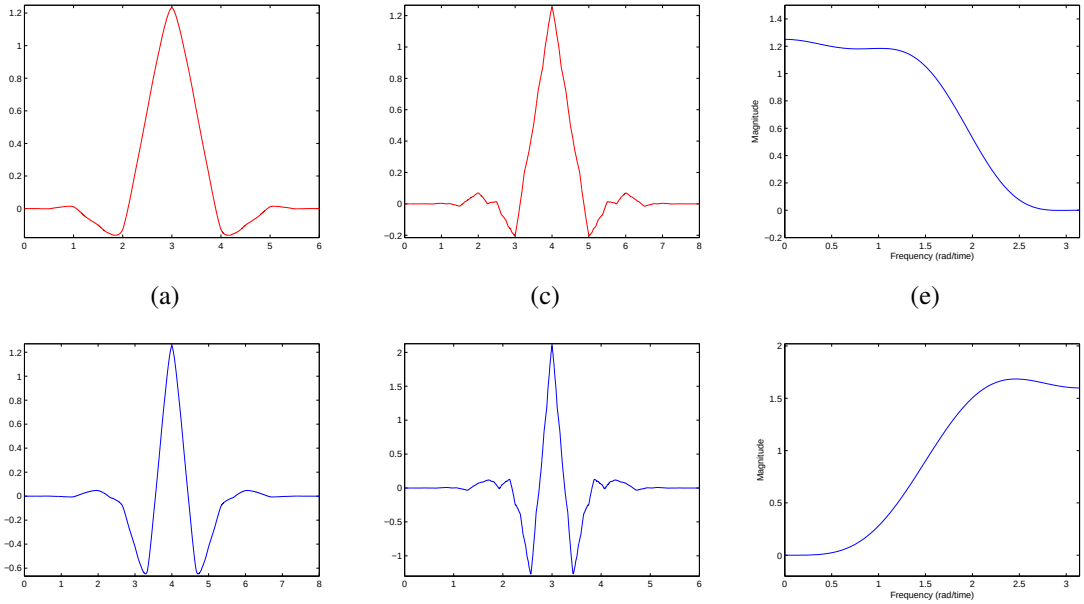
9/7-J filter bank listed in Table 3.4 are associated with the filter bank used in our evaluation. Therefore, the quantities $\hat{h}_0(\pi)$ and $\hat{h}_1(0)$ are slightly deviated from the theoretical value zero. Accordingly, we consider moments less than $2 \cdot 10^{-5}$ to be near-vanishing here. The vectors of independent lifting-filter coefficients of the five optimal designs are given in Table 3.5. The set of independent coefficients $\{a_{k,i}\}$ as given in (3.1) are presented in the increasing order of the lifting filter index k. Coefficients in the same lifting filter are presented in the increasing order of the index i.

The primal scaling, primal wavelet, dual scaling, and dual wavelet functions of the five optimal filter banks are shown in Figures 3.4, 3.5, 3.6, 3.7, and 3.8. Filter banks with longer filters have more degrees of freedom. Their scaling and wavelet functions tend to be smoother. The shapes of the primal (synthesis) scaling and wavelet functions are of great importance. They determine what shapes of functions can be efficiently represented by the wavelet systems as well as what shapes of distortions the representations might cause. Figure 3.9 gives a lossy-compression example for the `hotel` image. Parts of the original image and the reconstructed images at a compression ratio of 64:1 using the 2/6-opt, 17/11, and 13/15 filter banks are presented. The lengths of the analysis lowpass and highpass filters of the 2/6-opt transform are 2 and 6, respectively. The scaling and wavelet functions of the 2/6-opt transform will be shown later in Figure 3.13. From Figure 3.9, we can see that the 17/11 and 13/15 transforms having longer analysis lowpass and highpass filters introduce more pronounced ringing artifacts in the reconstructed images than the filter bank 2/6-opt having shorter analysis lowpass and highpass filters, especially in the regions near the text, where strong edges exist. This behavior can be attributed to the larger supports of the scaling and wavelet functions for the 17/11 and 13/15 transforms. On the other hand, there are some noticeable blocky artifacts above the top of the building in the top left portion of Figure 3.9(b). This fact is due to the relatively rougher primal scaling and wavelet functions associated with the 2/6-opt filter bank. Figures 3.4, 3.5, 3.6, 3.7, and 3.8 also present the magnitude responses of the lowpass and highpass analysis filters of the five optimal filter banks. Due to the constraints on the stopband of the frequency responses of the lowpass and highpass analysis filters, the energy leakage is very small in the stopbands. In the meanwhile, we see that the magnitude responses in the passbands are also relatively smooth and reasonably well behaved. This shows that constraining passbands in the filter-bank design may not be necessary.

To demonstrate the effectiveness of our design method, we now consider the lossy and lossless coding performance of the five filter banks constructed with our method. We start the evaluation by first considering the lossy coding performance of the various filter banks. To evaluate lossy coding performance, each filter bank was used to compress all of the test images at several bit rates. Then, we measured the relative difference Δ between the distortions (in PSNR) obtained with each of our optimal designs and the reference 9/7-J

Table 3.5: The vectors of independent lifting-filter coefficients for the optimal filter banks

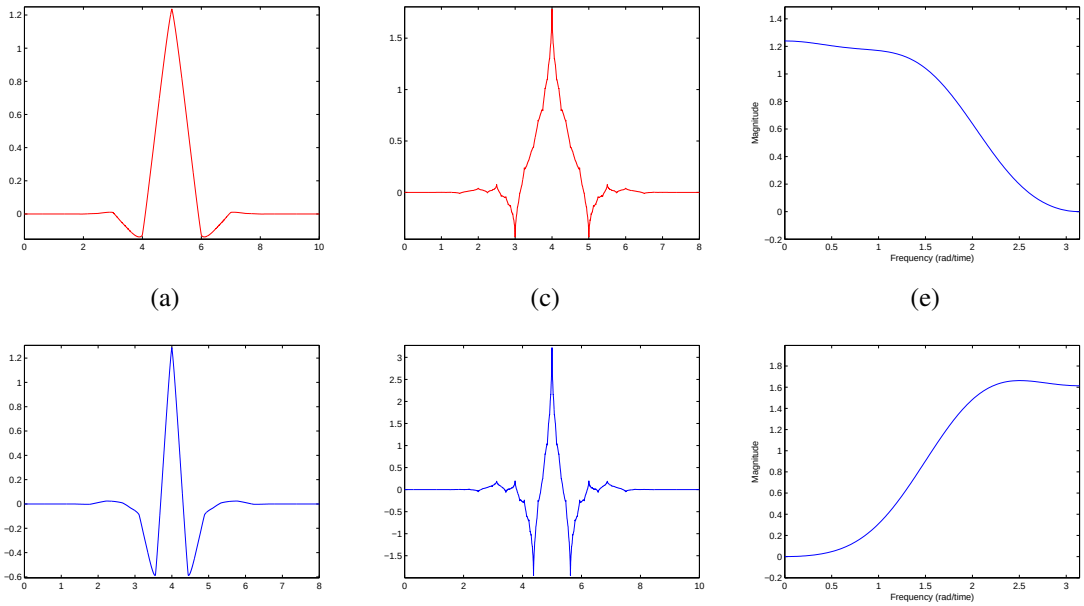
Transform	Lifting Configuration	Vector of Independent Coefficients
9/7	{2,2,2,2}	$\begin{bmatrix} -1.49341357280050 \\ -0.06301133138956 \\ 0.79448237353497 \\ -0.06301133138956 \end{bmatrix}$
9/11	{4,2,2}	$\begin{bmatrix} -0.35956162455574 \\ 0.05272161794337 \\ 0.30985251450888 \\ -0.15585284358476 \end{bmatrix}$
13/11	{4,2,2,2}	$\begin{bmatrix} -0.64730471284755 \\ 0.06342960259470 \\ -0.59194343074078 \\ 0.06996968233842 \\ 1.06861747599803 \end{bmatrix}$
17/11	{2,2,4,4}	$\begin{bmatrix} 0.55635379507590 \\ 0.04422978567149 \\ -0.98714912175476 \\ 0.09713229870681 \\ 0.33896200425457 \\ -0.03928354648785 \end{bmatrix}$
13/15	{6,2,2}	$\begin{bmatrix} -0.31220807547429 \\ 0.06899098759464 \\ 0.00053878746487 \\ 0.33662132370920 \\ -0.19111047469385 \end{bmatrix}$



(a) (c) (e)

(b) (d) (f)

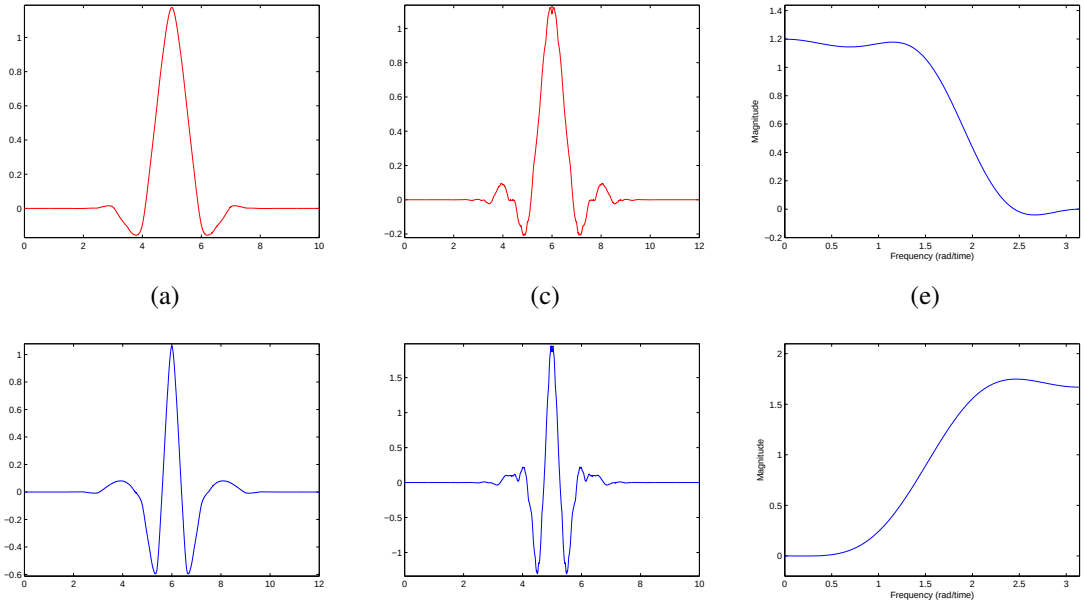
Figure 3.4: The 9/7 transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.



(a) (c) (e)

(b) (d) (f)

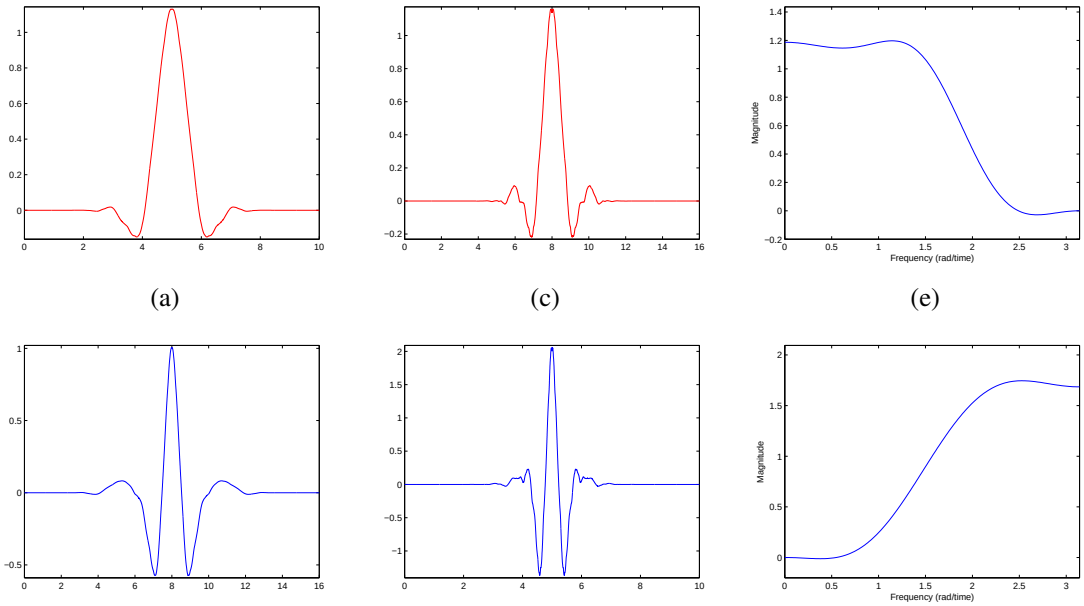
Figure 3.5: The 9/11 transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.



(a) (c) (e)

(b) (d) (f)

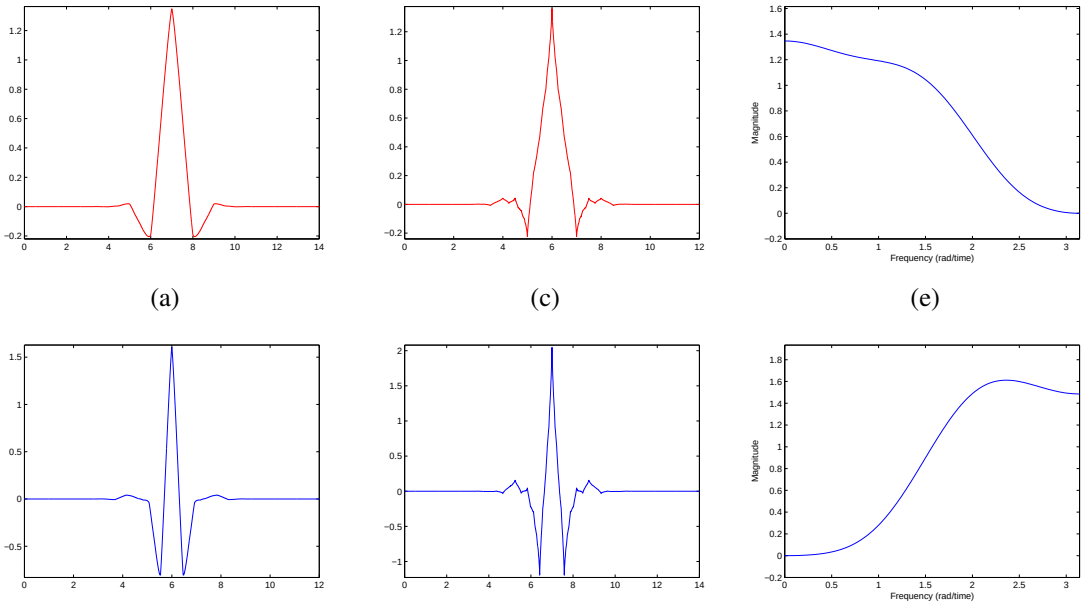
Figure 3.6: The 13/11 transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.



(a) (c) (e)

(b) (d) (f)

Figure 3.7: The 17/11 transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.



(a) (c) (e)

(b) (d) (f)

Figure 3.8: The 13/15 transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.

filter bank, as given by

$$\Delta = \frac{\text{PSNR}_{\text{opt}} - \text{PSNR}_{\text{ref}}}{\text{PSNR}_{\text{ref}}}, \tag{3.16}$$

where PSNR_{opt} and PSNR_{ref} are the PSNRs obtained from the reconstructed images of our optimal design and the reference filter bank, respectively. The results are summarized in statistical form in Table 3.6(a). In particular, we provide the mean and median relative differences of PSNR (with positive values corresponding to our designs outperforming the reference 9/7-J filter bank). As well, we indicate the percentage of cases in which our optimal design outperforms the 9/7-J filter bank. From these results, it is clear that all of our optimal designs outperform the 9/7-J filter bank in the majority of cases. For example, our 9/7 optimal design outperforms the 9/7-J filter bank 87.69% of the time. Our four other designs outperform the 9/7-J filter bank by margins ranging from about 59 to 78%. The above results are extremely encouraging, given that the 9/7-J filter bank is well known for its exceptional lossy-coding performance. In Table 3.6(b), we provide the actual PSNR results obtained for a representative subset of the test images, where the best result for each bit rate is highlighted. From the table, we can see that, even for images with different statistical properties (such as the three images considered here), our optimal designs outperform the 9/7-J filter bank, sometimes by a margin as high as 1.65 dB. Lastly, we would like to note that our optimal designs also lead to good subjective image quality, comparable to that of the 9/7-J filter bank. In Figure 3.10, we provide an example of the lossy

(a)

(b)

(c)

(d)

Figure 3.9: Parts of the `hotel` image. (a) Original. Lossy reconstructions at a compression ratio of 64:1 using the (b) 2/6-opt, (c) 17/11, and (d) 13/15 filter banks.

image reconstructions obtained with the various filter banks. For this image, one can see that the quality of the reconstructions produced by our optimal designs is comparable to that obtained with the 9/7-J filter bank.

As can be seen from Table 3.4, our 9/7 design and the 9/7-J filter bank have the same lifting configuration. While the 9/7-J filter bank has four dual and four primal vanishing moments, our 9/7 design has only two dual and two primal near-vanishing moments and slightly higher isotropic coding gain as well. With our design approach, by reducing the number of constrained moments and relaxing the requirement that moments be exactly zero, we are able to gain additional freedom, which ultimately allows higher-performance filter banks to be constructed. We also note that filter banks with fewer lifting filters can be implemented more efficiently, and potentially introduce less roundoff error than filter banks with more lifting filters. This could be the reason that the 9/11 and 13/15 transforms have good coding performance even if their coding gains are not as high as other transforms listed in Table 3.4. It is also interesting to note that, amongst all of the filter banks that we designed in the family of (3.1b), we were not able to find any that outperforms the 9/7-J filter bank. This might be caused by the fact that (3.1b) is an incomplete parameterization of all PR linear-phase FIR 1-D two-channel filter banks with even-length analysis/synthesis filters while (3.1a) is a complete parameterization of all PR linear-phase FIR 1-D two-channel filter banks with odd-length analysis/synthesis filters (up to a normalization).

In our work, we also evaluated the lossless coding performance of the various filter banks. Each of the filter banks was used to losslessly compress all of the twenty-six test images. The results are shown in Table 3.7. In particular, we provide the **normalized bit rate** (i.e., the reciprocal of compression ratio) for three images as well as the mean taken over all of the twenty-six test images. Evidently, all of our filter banks perform better overall than the 9/7-J filter bank for lossless coding, with the 9/11 design yielding the best results.

In addition to coding performance, computational complexity is another important consideration in practical implementations. Here, we compare the 9/7 and 9/7-J filter banks, both of which have the same lifting configuration. Before evaluating the computational complexity, we need to know what coefficients are used in the implementation. Although in Table 3.5, we have a very large number of precision for the lifting-filter coefficients, in practice, we often quantize the coefficients to numbers with less precision in order to reduce the computational complexity. Since the filter banks need to have high coding gains to perform well, we take the sensitivity of coding gain with respect to each independent lifting-filter coefficient into consideration for quantization, so that the impact of quantization on coding gain is minimized. To obtain a set of reasonable quantized coefficients, we first compute the gradient $\nabla_{\boldsymbol{x}} G(\boldsymbol{x})$ of the coding gain function with respect to the vector $\boldsymbol{x}$ of independent lifting-filter coefficients. Then, we evaluate the vector function $\nabla_{\boldsymbol{x}} G(\boldsymbol{x})$ at the point

Table 3.6: Lossy compression results for the various filter banks. (a) Summary statistical results over all twenty-six test images and five bit rates. (b) Specific results for three images.

(a)

Transform	Mean (%)	Median (%)	Outperform (%)
9/7	0.1488	0.1070	87.69
9/11	0.5371	0.0554	59.23
13/11	0.1863	0.0872	74.62
17/11	0.5804	0.2422	77.69
13/15	0.5936	0.1633	68.46

(b)

		PSNR (dB)					
Image	CR†	9/7	9/11	13/11	17/11	13/15	9/7-J
gold	8	36.88	37.34	36.85	37.17	**37.39**	36.75
	16	33.84	**34.00**	33.76	33.91	33.95	33.75
	32	31.27	**31.35**	31.24	31.32	31.27	31.23
	64	29.15	29.24	29.17	29.17	**29.25**	29.16
	128	27.32	**27.39**	27.35	27.37	27.34	27.32
target	8	41.59	42.92	42.12	42.81	**43.11**	41.46
	16	33.55	33.47	33.83	**34.00**	33.45	33.54
	32	27.19	26.65	27.84	**27.88**	26.84	27.07
	64	22.84	22.35	23.11	**23.19**	22.42	22.70
	128	19.14	18.86	19.35	**19.46**	18.94	19.16
sar2	8	30.35	30.30	30.33	30.33	**30.37**	30.32
	16	26.62	26.59	**26.62**	26.60	26.57	26.61
	32	**24.70**	24.65	24.69	24.69	24.70	24.69
	64	**23.55**	23.52	23.54	23.53	23.54	23.55
	128	22.73	22.70	**22.74**	22.74	22.67	22.73

†compression ratio

(e) (f)

Figure 3.10: Parts of the lossy reconstructions obtained after coding the `gold` image at a compression ratio of 32:1 using the (a) 9/7, (b) 9/11, (c) 13/11, (d) 17/11, (e) 13/15, and (f) 9/7-J filter banks.

Table 3.7: Lossless compression results for the various filter banks

Image	Normalized Bit Rate					
	9/7	9/11	13/11	17/11	13/15	9/7-J
gold	0.5666	**0.5630**	0.5652	0.5657	0.5644	0.5673
target	0.3086	**0.2949**	0.2964	0.3130	0.2975	0.3173
sar2	0.6351	**0.6349**	0.6353	0.6353	0.6352	0.6350
Mean†	0.4874	**0.4771**	0.4787	0.4870	0.4828	0.4880

†mean taken over all twenty-six test images

Table 3.8: The 9/7 and 9/7-J filter banks with quantized lifting-filter coefficients

9/7 transform:
$$\begin{cases} F_0(z) = -\frac{3057}{2048}(1+z) \\ F_1(z) = -\frac{257}{4096}(z^{-1}+1) \\ F_2(z) = \frac{407}{512}(1+z) \\ F_3(z) = \frac{15}{32}(z^{-1}+1) \end{cases}$$

9/7-J transform:
$$\begin{cases} F_0(z) = -\frac{203}{128}(1+z) \\ F_1(z) = -\frac{217}{4096}(z^{-1}+1) \\ F_2(z) = \frac{113}{128}(1+z) \\ F_3(z) = \frac{57}{128}(z^{-1}+1) \end{cases}$$

$\boldsymbol{x}^*$ by which the optimal lifting-filter coefficients are specified. Next, the quantizer step sizes for lifting-filter coefficients are decided roughly proportional to the vector $\frac{1}{\left|\nabla_{\boldsymbol{x}} G(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_i}\right|}$, so that the precisions of quantized lifting-filter coefficients are approximately proportional to their influence on coding gain. Since we want to choose the denominators of the quantized coefficients to be some powers of two, it is only possible to have the precisions of the quantized coefficients roughly proportional to their sensitivity of coding gain. The quantized lifting-filter coefficients for the two filter banks that we compare are given in Table 3.8.

We then evaluate the computational complexity of the filter banks in terms of the number of arithmetic operations. Since multiplications are more costly than addition and shift operations on many hardware and some software platforms, we convert all multiplications to add and shift operations via Booth's algorithm [5]. The computational complexity is represented by the number of adds and shifts needed for each pair of input samples. Table 3.9 shows the computational complexity in terms of the number of adds, shifts, and the total number of arithmetic operations required for each pair of input samples with one level 1-D wavelet

Table 3.9: Computational complexity of the 9/7 and 9/7-J filter banks

Transform	Adds	Shifts	Total
9/7	21	13	34
9/7-J	24	16	40

decomposition. From Table 3.9, we can see that the 9/7 filter bank needs six fewer operations than the 9/7-J filter bank, which makes the 9/7 design even more appealing for practical applications.

3.3.5 Optimal Designs and Coding Results for Different Lifting Configurations

Sometimes, we may need a high-performance filter bank with a certain prescribed lifting configuration. In this section, we will present several optimal designs for different prescribed lifting configurations produced by our proposed design method.

To examine if the proposed method can be broadly applied to design high-performance filter banks in filter-bank families with various structures, we design optimal filter banks with many different lifting configurations and test the coding performance of the optimal filter-bank designs. Of all possible lifting configurations of the form shown in Figure 2.7, we design filter banks belonging to lifting configurations with different numbers of lifting filters, with odd-length analysis/synthesis filters as in (3.1a) and even-length analysis/synthesis filters as in (3.1b), and with $F_0(z)$ in (2.7) being nonzero and zero.

Using our proposed method, several filter banks in different lifting configurations are designed. A previously proposed well-performing filter bank in each lifting configuration in which we are interested is also introduced as a reference filter bank. The characteristics of the optimal filter banks as well as some previously proposed high-performance filter banks having the same lifting configurations are given in Table 3.10, where the optimal filter banks are designated by the suffix "opt", and the reference filter banks are marked by the suffix "ref". As can be seen from Table 3.10, by reducing the number of vanishing moments, we are able to design optimal filter banks with higher coding gains than the reference filter banks in all four lifting configurations. The vectors of independent lifting-filter coefficients of the optimal filter banks are given in Table 3.11. Figures 3.11, 3.12, 3.13, and 3.14 show the primal scaling, primal wavelet, dual scaling, and dual wavelet functions of the designed optimal filter banks. All primal scaling and wavelet functions are smoother than the corresponding dual scaling and wavelet functions to help efficiently represent images, especially smoother images. Figures 3.11, 3.12, 3.13, and 3.14 also show the magnitude responses of the lowpass and highpass analysis filters of the optimal filter banks.

Having produced several sets of filter banks as described above, we now proceed to compare the coding

Table 3.10: Characteristics of improved filter banks in different configurations

Transform	$\{L_k\}$	$G^{\ddagger}_{\mathsf{sep}}$	$G^{\ddagger}_{\mathsf{iso}}$	$b_0^{\dagger}$	$b_1^{\dagger}$	VM*	$\hat{h}_0(0)$	$\hat{h}_0(\pi)$	$\hat{h}_1(0)$	$\hat{h}_1(\pi)$
9/7-opt	{4,2}	15.164	11.731	0.128	0.035	2, 0.055	1	-5.8E-6	8.2E-6	2
9/7-ref	{4,2}	15.145	11.719	0.126	0.032	4, 9	1	0	0	2
13/7-opt	{4,4}	15.357	11.984	0.027	0.030	2, 0.044	1	7.6E-6	4.6E-5	2
13/7-ref	{4,4}	15.363	11.978	0.023	0.032	4, 9	1	0	0	2
2/6-opt	{1,1,3}	14.438	11.821	0.127	0.127	1, 0.055	1	0	0	2
2/6-ref	{1,1,3}	14.429	11.818	0.127	0.136	3, 4.5	1	0	0	2
2/10-opt	{1,1,5}	14.448	11.825	0.127	0.111	1, 0.016	1	0	0	2
2/10-ref	{1,1,5}	14.405	11.757	0.127	0.056	5, 75	1	0	0	2

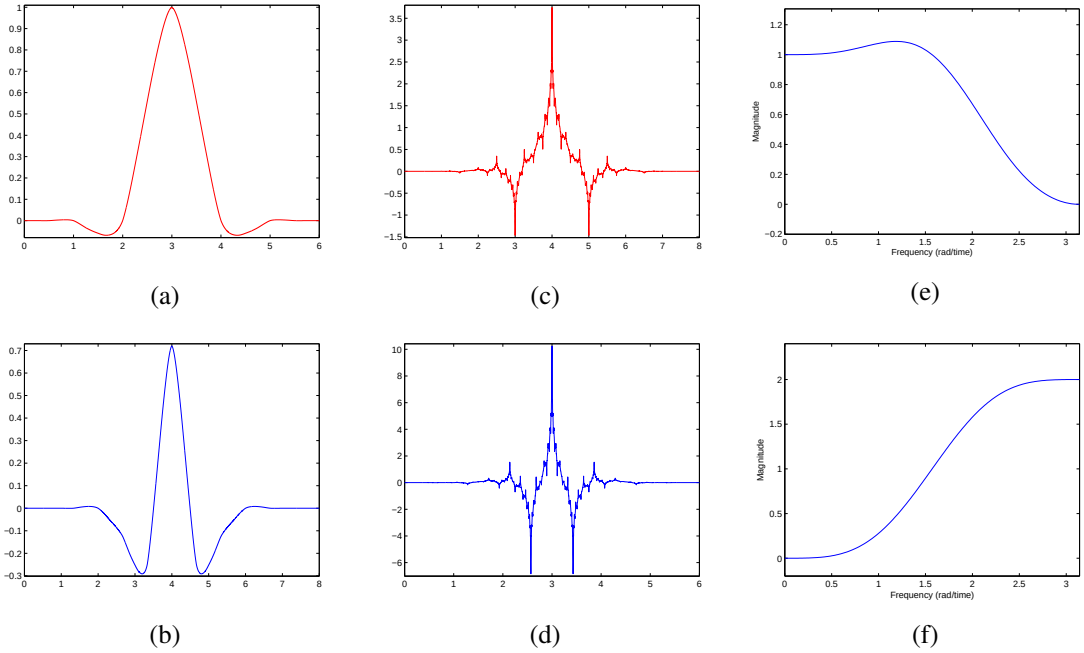
*index and magnitude of the first moment of dual wavelet coefficient sequence with moment magnitude greater than $5 \cdot 10^{-5}$

‡coding gain (in dB) for a five-level decomposition

†stopband energy as defined by (3.4)

Table 3.11: The vectors of independent lifting-filter coefficients for the optimal filter banks

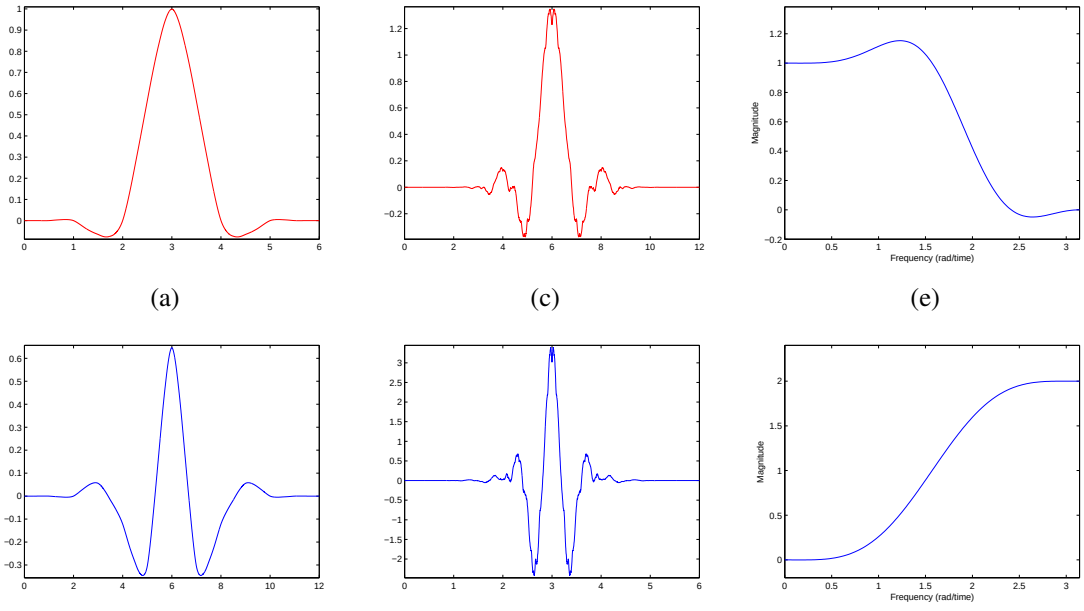
Transform	Lifting Configuration	Vector of Independent Coefficients
9/7-opt	{4,2}	$\begin{bmatrix} -0.55906599795795 \\ 0.05907007571340 \\ 0.25000246638637 \end{bmatrix}$
13/7-opt	{4,4}	$\begin{bmatrix} -0.56524341972549 \\ 0.06526636552708 \\ 0.3058610729295 \\ -0.05585722504449 \end{bmatrix}$
2/6-opt	{1,1,3}	$\begin{bmatrix} 0.261 \end{bmatrix}$
2/10-opt	{1,1,5}	$\begin{bmatrix} 0.274 \\ -0.014 \end{bmatrix}$



(a) (c) (e)

(b) (d) (f)

Figure 3.11: The 9/7-opt transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.



(a) (c) (e)

(b) (d) (f)

Figure 3.12: The 13/7-opt transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.

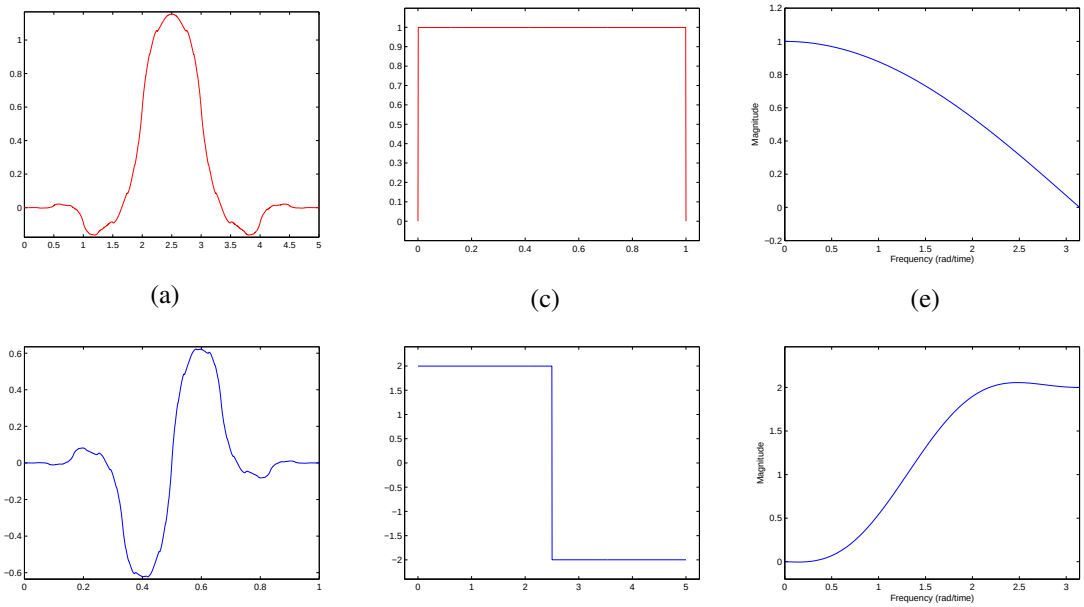


Figure 3.13: The 2/6-opt transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.

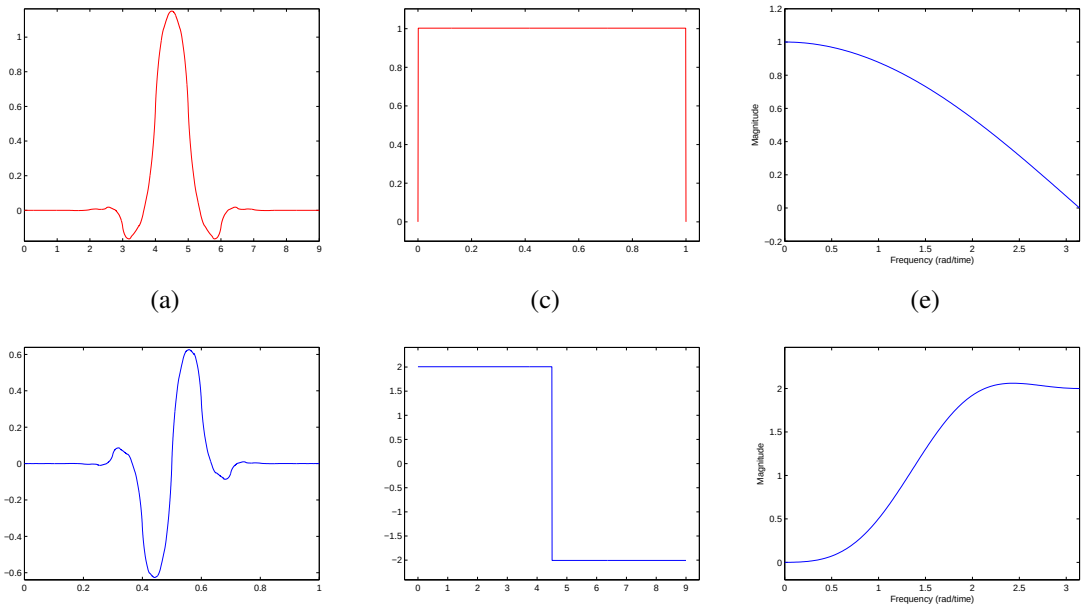


Figure 3.14: The 2/10-opt transform. The (a) primal scaling, (b) primal wavelet, (c) dual scaling, and (d) dual wavelet functions. Magnitude responses of the (e) lowpass analysis and (f) highpass analysis filters.

Table 3.12: Summary statistical lossy compression results for the improved filter banks

Transform	Mean (%)	Median (%)	Outperform (%)
9/7-opt	0.0022	0.0104	60.00
13/7-opt	0.0231	0.0232	64.62
2/6-opt	0.0245	0.0092	64.62
2/10-opt	0.0151	0.0881	62.31

performance of the filter banks associated with each lifting configuration using the EZW codec. To evaluate the coding performance, the lossy compression results of the optimal and reference filter banks over all twenty-six test images and five bit rates are collected. Then, the relative difference between the distortions (in PSNR) obtained with each of our optimal filter banks and the reference filter bank in the same lifting configuration is measured. The results are summarized in statistical form. More precisely, we compute the mean and median relative differences of PSNR (with positive values corresponding to our designs outperforming the reference filter banks) and the percentage of cases in which our optimal designs outperform the reference filter banks. A statistical summary of the lossy compression results obtained from the filter banks from different lifting configurations is given in Table 3.12. These results show that all of the optimal filter banks outperform their corresponding reference filter bank (with the same lifting configuration) by margins ranging from 60 to 64.62%. The proposed algorithm produces improved filter banks for different lifting configurations. The results are very attractive due to the fact that the reference filter banks are well-known for their good performance. In addition to the four lifting configurations presented here, we also considered most lifting configurations resulting filter banks with the lengths of analysis/synthesis filters less than 17. It is worth mentioning that for each and every lifting configuration that we considered, the proposed method never fails to achieve higher-performance designs than the previously proposed filter banks that we considered in the same lifting configuration. Furthermore, results show that filter banks with improved performance can be produced even by solving the optimization problem which considers the lifting-filter coefficients of a previously proposed filter bank in the same lifting configuration as an initial point.

In passing, we note that the coding gains of the optimal filter banks in Table 3.10 are not as high as those of the filter banks in Table 3.4. We also compare the coding performance of the two groups of filter banks using the EZW codec. Both objective and subjective results show that the designed filter banks for the prescribed lifting configurations in this section are inferior to the optimal filter banks proposed in Section 3.3.4. This implies that coding gain is a good prediction of coding performance for filter banks.

3.4 Summary

In this chapter, we proposed a novel method for the design of filter banks for image coding. Our method yields systems with numerous desirable properties such as PR, linear phase, high coding gain, good frequency selectivity, and certain prescribed moment properties. Several examples of filter banks constructed using our method were presented and shown to be highly effective for image coding. In particular, our optimal designs outperformed the well-known 9/7-J filter bank (from the JPEG-2000 standard) for both lossy and lossless compression, an impressive feat given that the 9/7-J filter bank is known for its exceptional lossy coding performance. Furthermore, we also designed filter banks for various lifting configurations using our proposed method. Experimental results showed that the designed filter banks outperformed the previously-known filter banks in their respective lifting configurations.

Chapter 4

An Improved Normal-Mesh-Based Scheme for Image Coding

Overview

Three modifications to the normal-mesh-based image coder of Jansen, Baraniuk, and Lavu are proposed, namely, the use of a data-dependent base mesh, an alternative representation for normal/vertical offsets, and a different scan-conversion scheme based on bicubic interpolation. Experimental results show that our proposed changes lead to improved coding performance in terms of both objective and subjective image quality measures. Some of our results presented in this chapter have also been published in [47].

4.1 Introduction

Because the human visual system (HVS) is sensitive to image edges, it is important for image coding systems to faithfully preserve edges. Many of today's best image coders are based on wavelet transforms. Unfortunately, such coders cannot efficiently represent the geometric features inherent in images (i.e., edges). This has led to an interest in coders that employ geometric representations of images, such as normal (triangle) meshes. Unlike wavelets, meshes are well suited to efficiently capturing the geometric information in images. Recently, an image coder based on normal meshes was proposed by Jansen, Baraniuk, and Lavu [22], which we henceforth refer to as the JBL coder. Let a **horizon class** function be a function consisting of piecewise constant regions separated by smooth contours of discontinuity. The JBL coder performs especially

well in coding images from this horizon class. It has been shown in [22] that under certain conditions, the JBL coder possesses an error-decay rate of $\| f - f_n \|_2 = O(n^{-1})$ when approximating a horizon class function f by a function f_n with n nonzero coefficients. The asymptotic error-decay rate is greater than the rate $\| f - f_n \| = O(n^{-\frac{1}{2}})$ obtained by wavelet approximations of horizon class functions. The fast error-decay rate makes the JBL coder very attractive for coding horizon-class images. Unfortunately, the JBL coder also has some shortcomings that unnecessarily restrict its performance. By addressing these shortcomings, the coder's performance can be improved. Therefore, we propose three modifications to the JBL coder and demonstrate through experimental results that these changes lead to improved coding performance.

The remainder of this chapter is organized as follows. Section 4.2 introduces some necessary background information about normal-mesh subdivision and the JBL coder. Section 4.3 describes our implementation of the JBL scheme. Some additional work involved in our implementation of the JBL coder such as computing bit rates for the transformed data based on an assumed entropy-coding scheme is explained. In Section 4.4 we identify some shortcomings of the JBL coder. This then motivates our proposal of three modifications to the JBL coder, which aims at improving its performance. These modifications are discussed in detail in Section 4.5. Section 4.6 briefly introduces our implementation of the enhanced coder. In Section 4.7, to verify the effectiveness of the three modifications to the JBL coder, results obtained with the original JBL coder and the enhanced coder are compared. Finally, Section 4.8 summarizes the work presented in this chapter.

4.2 Normal-Mesh Subdivision and the JBL Coder

In the sections that follow, we introduce some normal-mesh concepts that are necessary for understanding the JBL coder. Then, the subdivision schemes used in the JBL coder are described in detail. Both normal subdivision and several exceptional subdivision cases for refining two different categories of edges are introduced. A simplified JBL coder, called JBL-S, is also introduced for comparison purposes.

4.2.1 Normal-Mesh Concepts for the JBL Coder

Normal meshes can be used for surface approximation. In order to adopt a normal-mesh representation for images, we explain how to represent grayscale images as surface in 3-D. A grayscale image is a function f of two variables x and y, where x and y correspond to position, and $z = f(x,y)$ corresponds to image intensity. In this way, an image can be viewed as a surface parameterized over the xy plane. Thus, mesh-based techniques for representing surfaces can be used for images. In the case of the JBL coder, a normal (triangle) mesh is employed for this purpose.

For our purposes here, a normal mesh [19] is a multiresolution surface representation that consists of a nested sequence of meshes $\{M_0,\ M_1,\ M_2,\ \cdots\}$, generated by repeated refinement of a base mesh M_0. The base mesh consists of a small number of points from the true surface. The refinement process then generates a finer mesh by adding new points from the true surface to a coarser mesh. This is done in such a way that each new vertex on the finer mesh can be expressed as a displacement from a base point on the coarser mesh in the direction of the base point's surface normal. In other words, the new vertices added during refinement are located where surface normals from base points on the coarser mesh pierce the true surface (i.e., new vertices are located at so-called **piercing points**). The line passing through the base point and along some search direction for locating a piercing point is called a **search line**. Since each base point and its corresponding normal direction are completely determined by the coarser mesh, only a single scalar value (i.e., a **normal offset**) is needed to identify the location of each new vertex on the finer mesh. Thus, a normal mesh can be completely characterized by its base mesh and a set of normal offsets.

As mentioned previously, an image can be represented as a surface parameterized over the *xy* plane. In what follows, we refer to this plane as the **parameter plane**. Since some aspects of the JBL coder are more easily explained in terms of the parameter plane rather than explicit 3-D geometry, we will largely adopt a parameter-plane perspective in our description of this coder. In essence, the JBL coder creates a partitioning of the parameter plane using a triangulation, and then forms an interpolant over each of the resulting triangles in order to construct a surface in 3-D (i.e., the image surface). In what follows, unless otherwise noted, the term "vertex" will always refer to a vertex in the parameter-plane triangulation. Vertices are associated with height (i.e., *z* coordinate) values. In this way, each vertex/height-value pair corresponds to a point in 3-D. With the JBL coder, the three points associated with the vertices of each triangle are used to form a planar interpolant. By combining these interpolants, an image surface in 3-D is formed as illustrated in Figure 4.1. The 3-D surface is a continuous piecewise planar function.

To represent discontinuities, the JBL coder models edges explicitly using the so-called **horizon model**. As a matter of terminology, a contour in the parameter plane that corresponds to a discontinuity contour (i.e., image edge) is called a **horizon**. A vertex that is on a horizon is said to be a **horizon vertex**, and an edge (in the triangulation) with both of its endpoints being horizon vertices is said to be a **horizon edge**. The number of height values associated with a particular vertex, depends on whether the vertex is a horizon vertex. A nonhorizon vertex is associated with only one height value, while a horizon vertex is associated with two, in order to represent the height of the image surface on both sides of the horizon. To distinguish between these two cases, each vertex is associated with a bit, called a **horizon bit**, indicating if the vertex is a horizon vertex. In what follows, we will introduce the JBL subdivision scheme in detail.

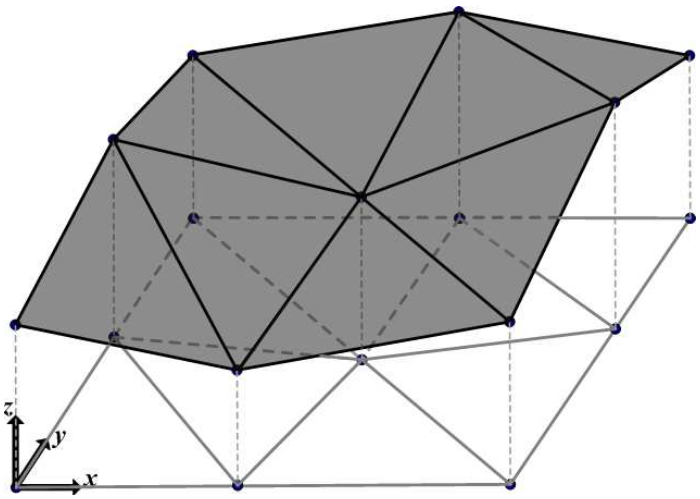


Figure 4.1: A mesh example and its 3-D correspondence.

4.2.2 JBL Subdivision Scheme

Although the JBL coder employs a normal mesh, the base mesh and its subsequent refinement are more easily described in terms of the parameter-plane triangulation (introduced above) rather than directly in terms of the 3-D mesh itself. First, the refinement process requires the notion of a true surface. Since images are essentially assumed to be piecewise constant in [22], the true surface is constructed using piecewise constant interpolation of the original image sample data. Centers of pixels are aligned with the lattice $\frac{1}{2}\mathbb{Z}_{\text{odd}}$, and the width and height of a pixel are one. The base mesh is associated with a particular base (i.e., initial) triangulation of the parameter plane. In the JBL scheme, the base triangulation is chosen to have four vertices, corresponding to the four corner points of the image bounding box. The refinement of the mesh then corresponds to a refinement of the parameter-plane triangulation through the addition of new vertices. In particular, refinement of the triangulation is performed by quaternary subdivision as shown in Figure 2.12 (on page 24), whereby a new vertex is added for each edge in the triangulation, resulting in each triangle being split into four new triangles. Due to the manner in which the refinement of the parameter-plane triangulation is performed (i.e., using numerous normal directions), this entire process can be viewed as the refinement of a normal mesh.

The location of the new vertex added for each edge is determined in one of two ways, depending on whether the edge is a horizon edge. The reason for adding new vertices for horizon and nonhorizon edges in different ways is that, horizon-edge refinement is resulting polyline refinement of horizon edges in the end, while nonhorizon-edge refinement is achieving fast location of horizon vertices. In what follows, we

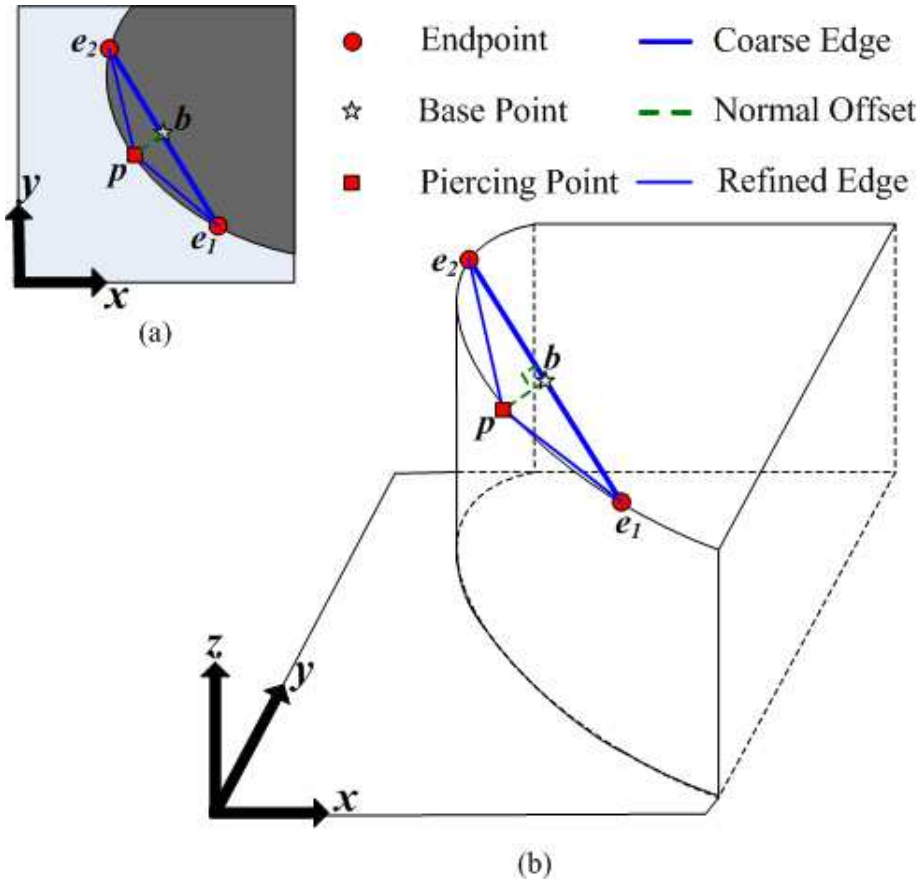


Figure 4.2: Subdivision of a horizon edge along the normal direction. The (a) parameter plane view, and (b) 3-D view.

will introduce the subdivision methods for the two different types of edges. To simplify the explanation that follows, we neglect a few exceptional cases.

Let us first discuss the subdivision of a horizon edge. Since our goal in this case is to construct a refined polyline approximation of the horizon, we would like the piercing point associated with the horizon edge to be a point on a horizon. To perform the subdivision, we first define the base point as the midpoint of the edge. Then, the search line for the new horizon vertex is chosen to be normal to the edge and parallel to the *xy* plane. The new vertex is added where the search line pierces the vertical surface through the horizon. The above process is illustrated in Figure 4.2, where the Figures 4.2(a) and 4.2(b) show the subdivision in the parameter plane and 3-D space, respectively. In the diagram, a filled circle denotes an endpoint associated with a horizon vertex. The thick solid segment $\overline{e_1e_2}$ is a coarse horizon edge. The star b at the middle of the coarse edge represents a base point. The thick dotted segment $\overline{bp}$ is on the search line, which is normal to the coarse edge

$\overline{e_1e_2}$ and parallel to the parameter plane. The filled square p represents a new piercing point on a horizon, the point at which the search line intersects the vertical surface through the horizon. The (signed) length of the thick dotted segment between the base point b and the piercing point p corresponds to the normal offset. The thin solid segments $\overline{e_1p}$ and $\overline{e_2p}$ connecting the endpoints of the coarse edge and the piercing point are refined horizon edges of the coarse horizon edge. The refined edges $\overline{e_1p}$ and $\overline{e_2p}$ form a polyline refinement of the coarse horizon edge $\overline{e_1e_2}$. Having located the piercing point, we also need to determine the horizon-bit and z-coordinate information associated with the newly added vertex. In this case (i.e., subdividing a horizon edge along the normal direction), the piercing point is an intersection of the search line with the vertical surface through the horizon. Therefore, the piercing point is always a point on a horizon. For this reason, no horizon bit is required for the new vertex. Let z_1 and z_2 be the two z coordinates associated with one endpoint of the horizon edge, and z_3 and z_4 be the two z coordinates associated with the other endpoint of the edge. Then, the two z coordinates associated with the new vertex are determined as $\frac{1}{2}\big(\max\{z_1,\, z_2\} + \max\{z_3,\, z_4\}\big)$ and $\frac{1}{2}\big(\min\{z_1,\, z_2\} + \min\{z_3,\, z_4\}\big)$.

Let us now consider the subdivision of a nonhorizon edge. Before proceeding, we need to first determine which of the z coordinates associated with the two endpoints of the nonhorizon edge are appropriate to use for determining the nonhorizon edge in 3-D. If both endpoints of the nonhorizon edge are associated with nonhorizon vertices, the z coordinates of the endpoints are unambiguously determined. Suppose now that one of the endpoints of the nonhorizon edge is associated with a horizon vertex. Let us assume that the horizon vertex is associated with the two z coordinates z_1 and z_2. Since, at most, one vertex of a nonhorizon edge can be a nonhorizon vertex, the other endpoint of the edge must be associated with a nonhorizon vertex. We assume that the nonhorizon vertex is associated with the z coordinate z_3. The z coordinate associated with the horizon vertex is chosen to be whichever of $\{z_1,\, z_2\}$ that is closer to z_3. Then the nonhorizon edge in 3-D is completely determined by its two endpoints and their chosen z coordinates. The base point is determined as the midpoint of this edge. The search line is normal to the edge, through its base point, and in the vertical plane containing the edge. The new point is added where the search line pierces the image surface. This process is illustrated both in the parameter plane and 3-D space in Figures 4.3(a) and 4.3(b), respectively. The unfilled circles e_1 and e_2 are the endpoints of an edge. The thick solid segment $\overline{e_1e_2}$ is a coarse nonhorizon edge. The star b represents the base point of the edge $\overline{e_1e_2}$. The normal search line is perpendicular to $\overline{e_1e_2}$ and in the vertical plane containing $\overline{e_1e_2}$. At the unfilled square p, where the search line intersects the image surface, a new point is added associated with a nonhorizon vertex. Consequently, the two segments $\overline{e_1p}$ and $\overline{pe_2}$ shown by the thin solid segments form a refinement of the coarse nonhorizon edge $\overline{e_1e_2}$. To illustrate the adaptivity of nonhorizon-edge subdivision along the normal direction, Figure 4.3 also shows the refinement

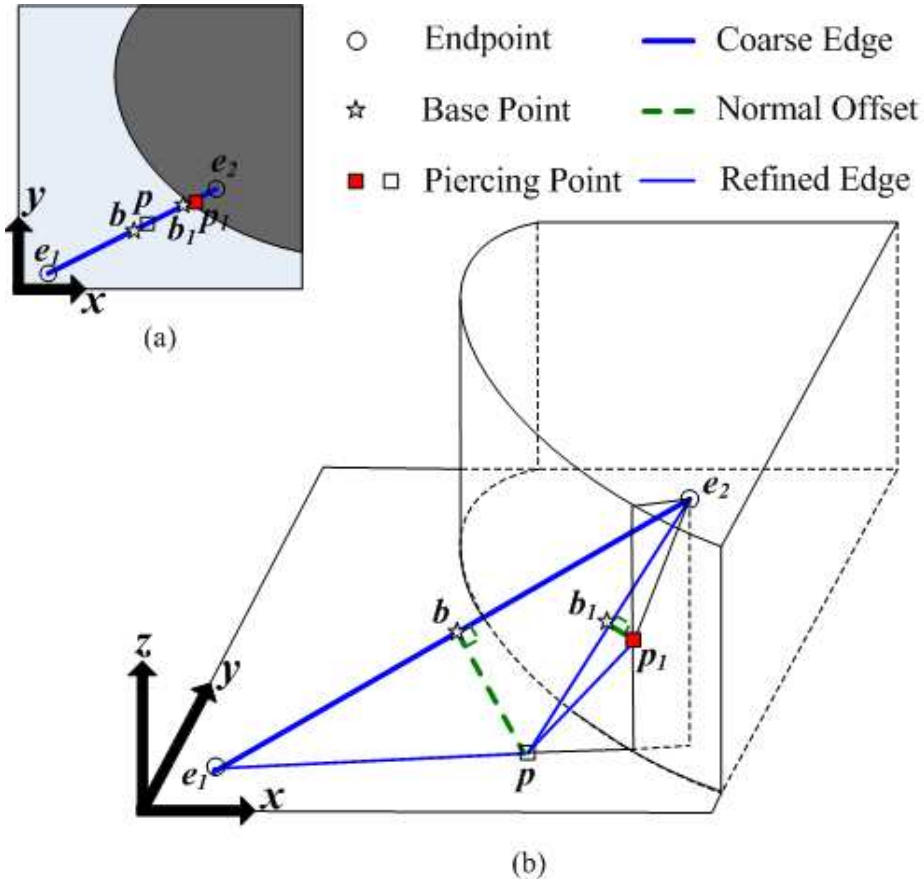


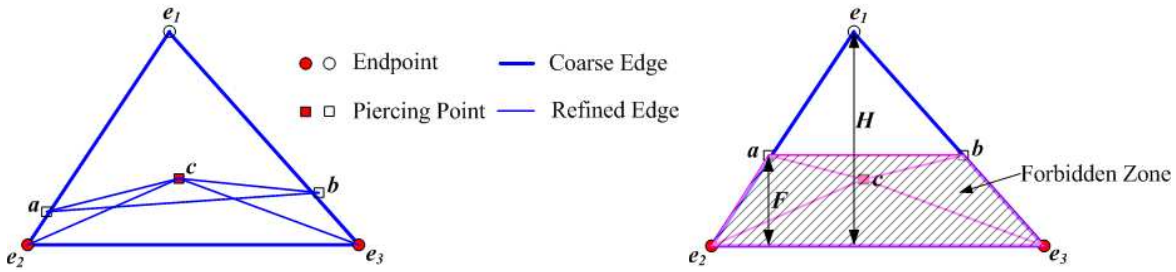
Figure 4.3: Subdivision of a nonhorizon edge along the normal direction. The (a) parameter plane view, and (b) 3-D view.

of a new nonhorizon edge $\overline{pe_2}$ for the next level of subdivision. Through the corresponding base point b_1, the search line is perpendicular to the edge $\overline{pe_2}$. At the filled square p_1, the normal search line intersects the image surface. A piercing point p_1 associated with a horizon vertex on the parameter plane is located during the subdivision. By two iterations of subdivision to the coarse nonhorizon edge $\overline{e_1e_2}$, a new horizon vertex p_1 is found.

The normal search direction used in the subdivision of nonhorizon edges is considered to contribute to the fast location of new horizon vertices, since the normal direction tends to point towards the function discontinuity. In contrast with the subdivision along normal directions, if vertical directions were used instead, new vertices are always added at the midpoints of the coarse edges in the parameter plane. Hence, searching along vertical directions is very difficult to find a jump of a function due to the non-adaptivity of this subdivision process. This fact further explains why the adaptivity of subdivision along normal directions is desirable.

As suggested earlier, some exceptional cases other than subdivision along normal directions occasionally occur during the refinement process. This is because, with the approach described above, the piercing point found in a normal direction does not always lead to a valid triangulation in the parameter plane. Figure 4.4(a) shows an example of overlapping triangles caused by subdivision along normal directions. In this figure, filled and unfilled circles represent horizon and nonhorizon vertices, respectively. As a result, the edge $\overline{e_2e_3}$ is a horizon edge, while edges $\overline{e_1e_2}$ and $\overline{e_1e_3}$ are nonhorizon edges. The normal piercing points are found for all three edges of the triangle. In the parameter plane, vertices a, b, and c are the projections of the piercing points associated with edges $\overline{e_1e_2}$, $\overline{e_1e_3}$, and $\overline{e_2e_3}$, respectively. Quaternary subdivision generates four new triangles, namely $\triangle e_1ab$, $\triangle ae_2c$, $\triangle abc$, and $\triangle bce_3$. Apparently, some of these triangles overlap, such as triangles $\triangle e_1ab$ and $\triangle ae_2c$. During the refinement of the triangle $\triangle abc$ involving a horizon edge $\overline{e_2e_3}$, it is possible that the new piercing points found in normal directions do not lead to a valid triangulation due to overlapping triangles in the parameter plane. To avoid this and other problems, we must, in some exceptional circumstances, include an offset in the vertical direction, in lieu of or in addition to the normal direction. Therefore, an additional value, called the **direction value**, is required for each offset to capture which combination of normal/vertical directions is employed.

We now explain each of the exceptional cases in the refinement process in detail. First, to avoid invalid triangulations during subdivision, a **forbidden zone** is constructed for each triangle having a horizon edge and on the concave side of the horizon contour. This process is illustrated in Figure 4.4(b). The forbidden zone, indicated by the shaded area, is a trapezoidal region in the parameter plane adjacent to segment $\overline{e_2e_3}$ of the polyline approximation of a horizon. The forbidden zone avoids invalid triangulations by restricting new vertices associated with nonhorizon edges of the triangle from being added within the forbidden zone, and



(a) (b)

Figure 4.4: Use of the forbidden zone to avoid overlapping triangles. (a) The situation of overlapping triangles. (b) Use of the forbidden zone to avoid invalid triangulation.

confining new vertices associated with horizon edges of the triangle in the forbidden zone. By introducing the above restrictions, vertices a and b in Figure 4.4(b) must not be in the forbidden zone, while vertex c must be in the forbidden zone. The restricted geometric positions of the vertices in the parameter plane prevent the triangles $\triangle e_1ab$, $\triangle ae_2c$, $\triangle acb$, and $\triangle bce_3$ from overlapping. Therefore, triangulation problems are avoided through the use of forbidden zones. The width of a forbidden zone is carefully defined as follows. Let $\overline{E_2E_3}$ denote a segment in the coarsest polyline approximation of a horizon, and let us assume that the segment $\overline{e_2e_3}$ is produced by iteratively refining $\overline{E_2E_3}$. Let h_0 and h be the lengths of the segments $\overline{E_2E_3}$ and $\overline{e_2e_3}$, respectively. In Figure 4.4(b), let F be the altitude (the distance between the parallel sides) of the trapezoid, and H be the altitude of the $\triangle e_1e_2e_3$ on the edge $\overline{e_2e_3}$. The width F of the forbidden zone is given by $\min\left\{\frac{3h^2}{8h_0}, \frac{H}{2}\right\}$.

Since, in the parameter plane, the horizon-edge subdivision can only be done within the forbidden zone as discussed above, if a piercing point for a horizon edge cannot be found in the normal direction such that the projection of the piercing point in the parameter plane is within the forbidden zone, an alternative search direction is used. The new search line is normal to the parameter plane (i.e., in the vertical direction) through the corresponding base point. A piercing point can always be found in this way. In the case of subdivision of a horizon edge along a vertical direction, a direction value is set to indicate a vertical search direction used in the edge-refinement process.

Having introduced the above exceptional case for horizon-edge subdivision, we now discuss two exceptional cases for nonhorizon-edge subdivision. The nonhorizon-edge subdivision is somewhat more complex than the horizon-edge subdivision. In the parameter plane, the nonhorizon-edge subdivision can only be performed outside the forbidden zone. The first exceptional case for nonhorizon-edge subdivision occurs when the subdivision along the normal direction does not result in a new vertex outside the forbidden zone. In this

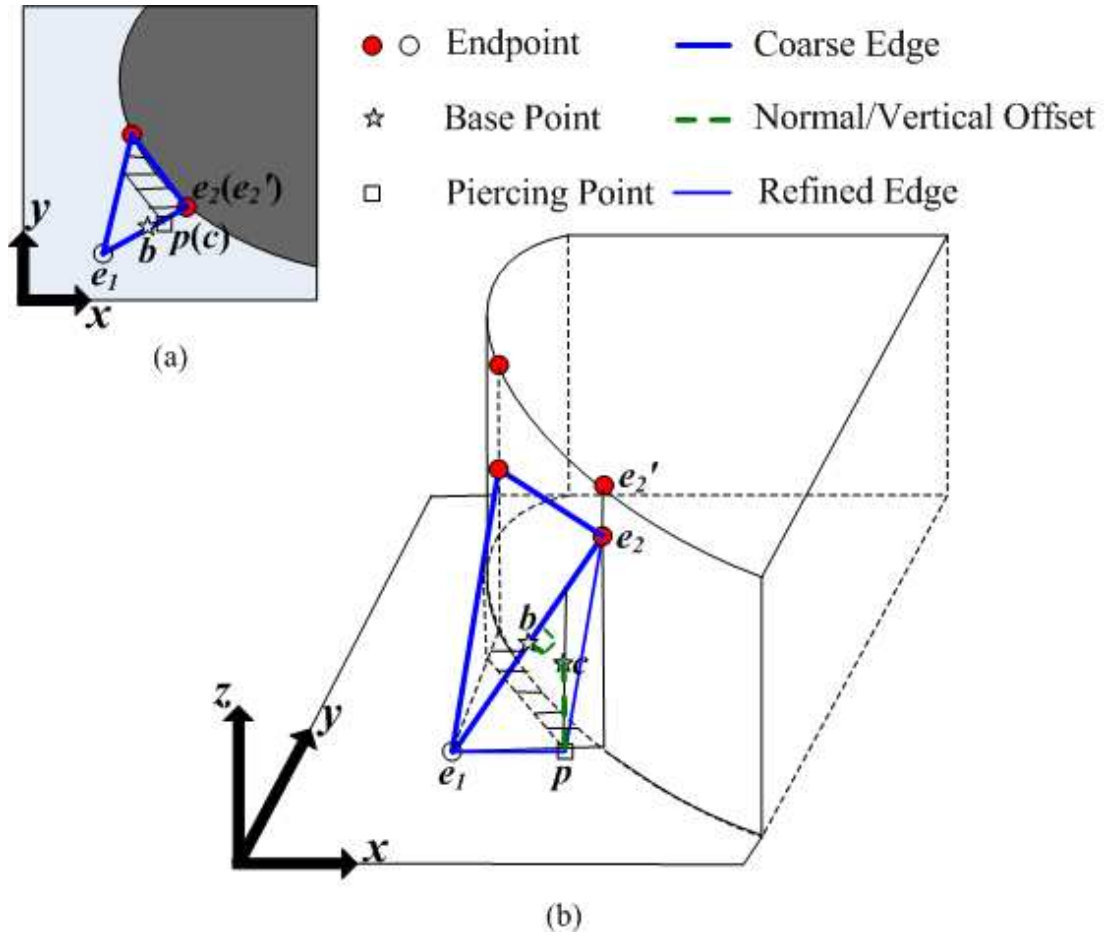


Figure 4.5: Nonhorizon edge subdivision: exceptional case 1. Subdivide first along the normal direction then along the vertical direction. The (a) parameter plane view, and (b) 3-D view.

case, the search along the normal direction stops at the forbidden-zone boundary and then proceeds along the vertical direction (i.e., normal to the parameter plane). The subdivision where we search first along the normal direction then along the vertical direction is demonstrated in Figure 4.5. In the figure, the point e_1 is associated with a nonhorizon vertex, while points e_2 and e_2' are associated with the same horizon vertex. As discussed earlier, whether we should choose the segment $\overline{e_1e_2}$ or $\overline{e_1e_2'}$ as the nonhorizon edge to subdivide depends on whichever z coordinates at e_2 and e_2' is closer to the z coordinate at e_1. From the figure, it is clear that the nonhorizon edge $\overline{e_1e_2}$ is the edge of interest in our subdivision since the z coordinate at e_1 is closer to the z coordinate at e_2 than to the z coordinate at e_2'. The hatched area in the parameter plane denotes the forbidden zone. Then, we search for the piercing point along the normal direction of segment $\overline{e_1e_2}$ from the base point b. Unfortunately, no piercing point is found such that its projection onto the parameter plane is

outside the forbidden zone. The search stops at the point c, where the normal search line intersects the vertical plane through the forbidden-zone boundary. Then, the search line changes to be a vertical line through the transition point c. A piercing point p is found at the intersection of the new search line and the image surface. In this exceptional case, the direction value indicates that the search proceeds in first normal then vertical direction. Since the forbidden zone can be precisely determined from the coarse mesh, the transition point c can be derived based on the forbidden-zone and search-line information. Consequently, the offset in the normal direction (i.e., segment $\overline{bc}$) need not be stored. A vertical offset from the transition point to the piercing point (i.e., segment $\overline{cp}$) is stored and is sufficient to locate the piercing point p.

The second exceptional case for nonhorizon-edge subdivision occurs when the piercing point that is found in the normal direction is coincident with one of the vertices associated with the two endpoints of the edge. In this case, if the coincident vertex is a horizon vertex, and the z coordinate at the piercing point, say z_3, is greater or less than both z coordinates, say z_1 and z_2, at the coincident endpoint, z_1 or z_2 is replaced by z_3 according to the following rules. If $|z_1 - z_3| < |z_2 - z_3|$, the coordinate z_3 replaces z_1, otherwise the coordinate z_3 replaces z_2. In the parameter plane, no new vertex associated with the original edge is added due to the coincidence of the piercing point and the endpoint. Therefore, a piercing point found in a vertical direction (i.e., normal to the parameter plane) through the base point is considered to be the new piercing point associated with the nonhorizon edge. The offset associated with this new piercing point is adding to the list of offsets. A special example of the exceptional subdivision case discussed here is illustrated in Figure 4.6, where the unfilled square p is the new added vertex associated with the coarse edge $\overline{e_1e_2}$. The filled square c is the piercing point in the normal direction. The piercing point c is used only for updating the smaller z coordinate at the endpoint e_2. As shown in the figure, the z coordinate associated with a horizon vertex can be updated during subdivision of a nonhorizon edge along the vertical direction. As a result, the mesh refinement for nonhorizon edges is conducted for both parameter plane position (i.e., x and y coordinates) and intensity value (i.e., z coordinate). In the second exceptional case for nonhorizon-edge subdivision, the direction value is set to indicate a vertical search direction being used in the refinement.

In all of the subdivision cases other than subdivision of horizon edges along normal directions, once a piercing point is found, we determine if the piercing point is associated with a horizon vertex. Then the horizon bit is set depending on whether the associated vertex for the piercing point is on a horizon. The z coordinate associated with the newly added vertex is the z coordinate of the image surface at the piercing point. In all of the edge subdivision cases, once the search line is fixed, an offset can be in one of the two opposite directions starting from the base point. Offsets are signed values in order to distinguish the two directions.

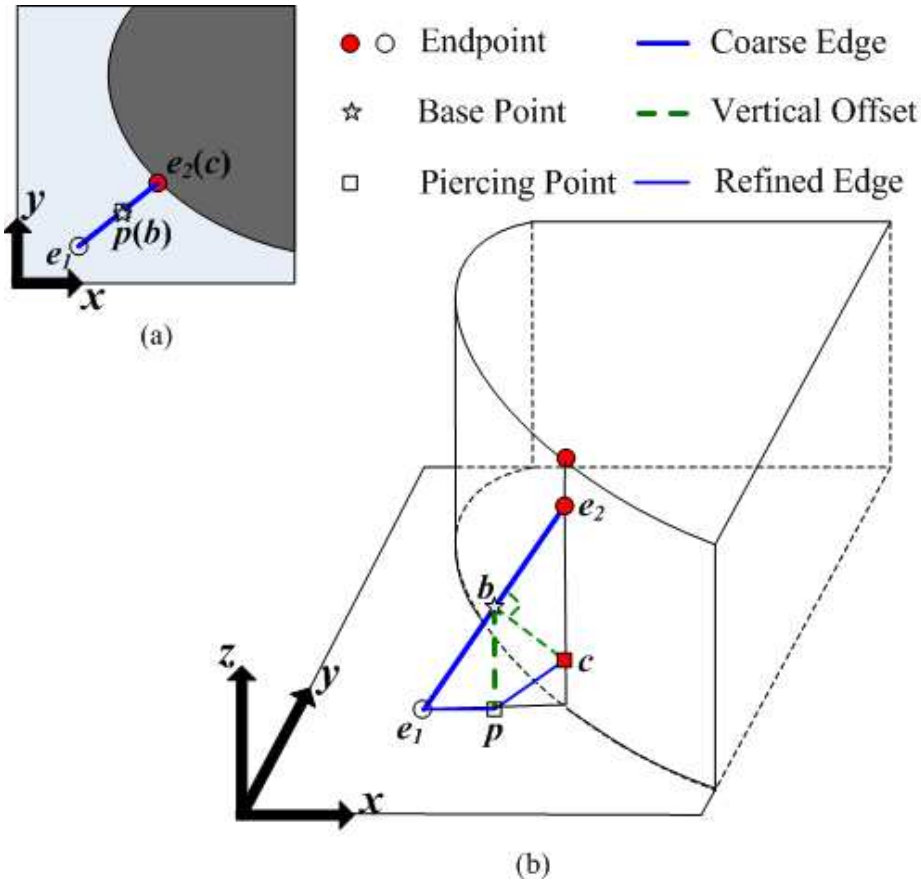


Figure 4.6: Nonhorizon edge subdivision: exceptional case 2. Subdivide along vertical direction. The (a) parameter plane view, and (b) 3-D view.

4.2.3 Scan Conversion

The normal-mesh-based representation produced by the JBL coder is completely characterized by the base mesh, normal/vertical offsets, horizon bits, and direction values. Using this information, the corresponding vertices of the refined mesh can be reconstructed. Since the resulting mesh representation is a surface defined on a continuous domain, a scan-conversion process is needed to convert such a mesh from the continuous domain to points on a raster grid. It is implied in [22] that planar interpolation is being used in the scan conversion by the JBL coder. Then, the interpolated surface is sampled on grids at centers of pixels in the parameter plane. As a result, a discrete signal is produced.

4.2.4 Simplified JBL Coder

Having introduced the JBL coder, we now take a moment to briefly introduce a greatly simplified version of this coder that is implicitly suggested in [22] (in the context of natural images). We refer to this simplified version of the coder as JBL-S herein. In the JBL-S coder, the image surface is constructed using piecewise planar interpolation rather than piecewise constant interpolation. Due to the use of a piecewise planar interpolant, no discontinuities exist in the image surface, and hence no horizons exist either. Thus, the JBL-S coder is essentially the JBL coder without the horizon model. Unlike in the JBL case, horizon bits are not needed since there is no horizon vertex in the JBL-S coder. Furthermore, in the JBL-S coder, subdivision always processes along normal directions, and none of the exceptional cases ever occur. Therefore, direction values are not required either. In the case of the JBL-S coder, the normal-mesh-based image representation is completely characterized by only the base mesh and normal offsets.

4.3 Our Implementation of the JBL Coder

Before proceeding further, we briefly describe our implementation of the JBL coder, which was used as the basis for our work herein. Generally, our implementation is written in MATLAB. The results for the JBL coder presented in [22] are exclusively applied to images with piecewise constant regions, most likely because the horizons in such images can be simply detected by finding intensity differences between neighboring pixels. To apply the JBL coder to general images, where intensity differences of neighboring pixels exist almost everywhere, in our implementation of the JBL coder, we use a Canny edge detector [9] in the encoder to determine horizons. Then, a normal-mesh-based representation is constructed, and the corresponding normal/vertical offsets are then quantized and assumed to be entropy coded. The bit rate is estimated based

Table 4.1: Summary of data for the JBL and JBL-S coders

Method	Data Name	Type	Range of Value
JBL	offset	$\mathbb{R}$	$-a$ to a, where $a = \max(2^P - 1, \frac{\text{WIDTH}-1}{2}, \frac{\text{HEIGHT}-1}{2})$
	direction value	$\mathbb{Z}$	$\{0,1,2\}$
	horizon bit	boolean	$\{0,1\}$
	base mesh (z)	$\mathbb{Z}$	0 to $2^P - 1$
JBL-S	offset	$\mathbb{R}$	$-a$ to a, where $a = \max(2^P - 1, \frac{\text{WIDTH}-1}{2}, \frac{\text{HEIGHT}-1}{2})$
	base mesh (z)	$\mathbb{Z}$	0 to $2^P - 1$

on the assumed coding scheme.

We introduce the quantization scheme used in our implementation in what follows. The normal/vertical offsets are scalar quantized with a separate quantizer being used for the offsets of each subdivision level. Rather than specifying all of the quantizer step sizes individually, all step sizes are computed from a single encoder parameter q. In particular, the step size Δ_k for the offsets of the kth subdivision level (where $k = 0$ corresponds to the base mesh) is chosen as $\Delta_k = q2^{k-1}$ for $k \geq 1$. Here, $k \neq 0$ since no offsets are associated with the base mesh. This choice of Δ_k results in offsets from coarser levels being weighted more heavily than those from finer levels. Such is desirable, since in a normal mesh, errors in the coarser-level offsets introduce considerably more distortion than errors in the finer-level offsets. Distortions of the reconstructed vertices and their height values caused by offset-roundoff errors also propagate from coarser levels to finer levels.

In [22], the authors do not make any attempt to estimate the bit rate required to code the data of the normal-mesh-based representation. In our work, however, this rate is estimated using the data's entropy. To discuss our rate-estimation scheme, some data properties for different data sets need to be addressed first. Table 4.1 identifies the data that needs to be coded for the JBL and JBL-S coders. An offset is a real number. The probability density function of the offsets is symmetric about zero. Hence, only the maximum absolute value needs to be specified in order to identify the range of values. For the JBL and JBL-S coders, offsets have the maximum value of $\max(2^P - 1, \frac{\text{WIDTH}-1}{2}, \frac{\text{HEIGHT}-1}{2})$, where WIDTH, HEIGHT, and P denote the width, height, and number of bits per sample of the original image, respectively.

In our rate-estimation scheme, a Laplacian distribution is used to model the normal/vertical offsets, since the probability density function of the offsets has a symmetrically shaped sharp peak with zero mean. Recall that the probability density function p of a Laplacian distribution is $p(x) = \frac{\lambda}{2}e^{-\lambda|x-\mu|}$, as given by (2.14). In this formula, μ and λ are location and scale parameters, respectively. For a set of offsets, since most data clusters close to the origin, the location parameter μ is zero. To fit the distribution model for a particular set

of offsets and find the scale parameter λ, we measure the variance of the data and then match the variance of the probability density function of the Laplacian distribution to the data. The offset variance σ^2 in terms of λ is given as follows:

$$\begin{aligned}
\sigma^2 &= \int_{-\infty}^{\infty} (x - E(x))^2 p(x) dx \\
&= \int_{-\infty}^{\infty} x^2 p(x) dx \quad \text{(mean of offsets } E(x) \text{ is approximately zero from both theory and experiments)} \\
&= \int_{-\infty}^{\infty} x^2 \tfrac{\lambda}{2} e^{-\lambda|x|} dx \quad \text{(substitution of probability density function)} \\
&= \tfrac{1}{2}\left[e^{\lambda x}(x^2 - \tfrac{2x}{\lambda} + \tfrac{2}{\lambda^2}) \big|_{-\infty}^{0} + e^{-\lambda x}(x^2 + \tfrac{2x}{\lambda} + \tfrac{2}{\lambda^2}) \big|_{0}^{\infty} \right] \\
&= \tfrac{1}{2}\left[\tfrac{2}{\lambda^2} + \tfrac{2}{\lambda^2} \right] \\
&= \tfrac{2}{\lambda^2}.
\end{aligned}$$

Thus, we choose the parameter $\lambda = \frac{\sqrt{2}}{\sigma}$ to fit the Laplacian distribution for a set of offsets. Note that this calculation is performed for offsets from each level of subdivision, because offsets from different levels of subdivision typically have distinct probability density functions. In our work, we choose a parameter λ for offsets from each level of subdivision, and use 32 bits to represent each λ. The entropy H of the quantized normal/vertical-offset data can be estimated as follows:

$$H = -\sum_{k=1}^{n} \frac{\text{NumInRegion}_k}{\text{TotalDataNum}} \log_2 \text{PrOfRegionFromModel}_k, \tag{4.1}$$

where n is the total number of possible distinct values for each quantized offset, NumInRegion_k is the number of quantized offsets that equal the kth possible value, TotalDataNum is the total number of offsets in the data set, and $\text{PrOfRegionFromModel}_k$ is the discrete probability of the kth possible value.

A simple first-order entropy estimate is used for the horizon-bit and direction-value data. The actual first-order probability for each symbol measured from the data set is used to estimate the entropy. Therefore, for the horizon-bit and direction-value data, the general entropy equation (2.13) becomes the following:

$$H = -\sum_{k=1}^{n} \frac{\text{NumInRegion}_k}{\text{TotalDataNum}} \log_2 \frac{\text{NumInRegion}_k}{\text{TotalDataNum}}, \tag{4.2}$$

where n is the total number of possible distinct values for each symbol, NumInRegion_k is the number of symbols that equal the kth possible distinct value, and TotalDataNum is the total number of symbols in the data set. For the horizon-bit and direction-value data, n is 2 and 3, respectively. The actual probabilities $\left\{ \frac{\text{NumInRegion}_k}{\text{TotalDataNum}} \right\}_{k=1}^{n-1}$ for the first $n-1$ possible distinct values are coded as side information. Since $\sum_{k=1}^{n} \left(\frac{\text{NumInRegion}_k}{\text{TotalDataNum}} \right) = 1$, the quantity $\frac{\text{NumInRegion}_n}{\text{TotalDataNum}}$ need not be coded.

For both the JBL and JBL-S coders, given the image size, the information that needs to be coded for the base mesh consists of the z coordinates associated with the four vertices of the image bounding box. The base-mesh data is assumed not to be entropy coded, however. We use P bits to represent each z coordinate, where P denotes the number of bits per sample in the original image.

Having introduced the entropy estimation method for normal/vertical offsets, horizon bits, and direction values, we finally calculate the total number TotalBit of bits used in the coded image as

$$\text{TotalBit} = \sum_{i=1}^{N} (H_i \times \text{TotalDataNum}_i) + \text{SideInfo}, \tag{4.3}$$

where N is the number of different data sets that need to be coded, H_i is the entropy of the ith data set computed by (4.1) or (4.2), TotalDataNum_i is the total number of symbols in the ith data set, and SideInfo is the number of bits used for coding all necessary side information. The side information includes:

- 32 bits representing the parameter λ for offset data in each level of subdivision,
- 32 bits for actual-probability information for horizon-bit data,
- 32×2 bits for actual-probability information for direction-value data,
- 32 bits for the quantizer step size q of the normal/vertical offsets from the coarsest level of subdivision,
- 8 bits for the number of levels of subdivision, and
- 16×2 bits for the width and height of the image.

Since we use only the first-order probability of the data in the entropy estimation, the estimated entropy does not vary with the data-scanning order used in the coding process. Figure 4.7 describes the code stream for the JBL coder. Some side information is coded in the header including the width and height of the image, the total number of levels of subdivision, and the quantizer step size for the coarsest level of offsets. Then, the base-mesh information is coded. Since the base mesh has simply four vertices corresponding to the four corners of the image bounding box, the x and y coordinates of the vertices can be derived from the width and height of the image. Only z coordinates need to be stored, where P bits are used for each coordinate. Finally, the data for each level of subdivision is coded in order of increasing subdivision-level index. For each level of subdivision, the new-vertex information includes offsets, normal bits, direction values, a scale parameter λ for offset data, and the actual probabilities for normal-bit and direction-value data.

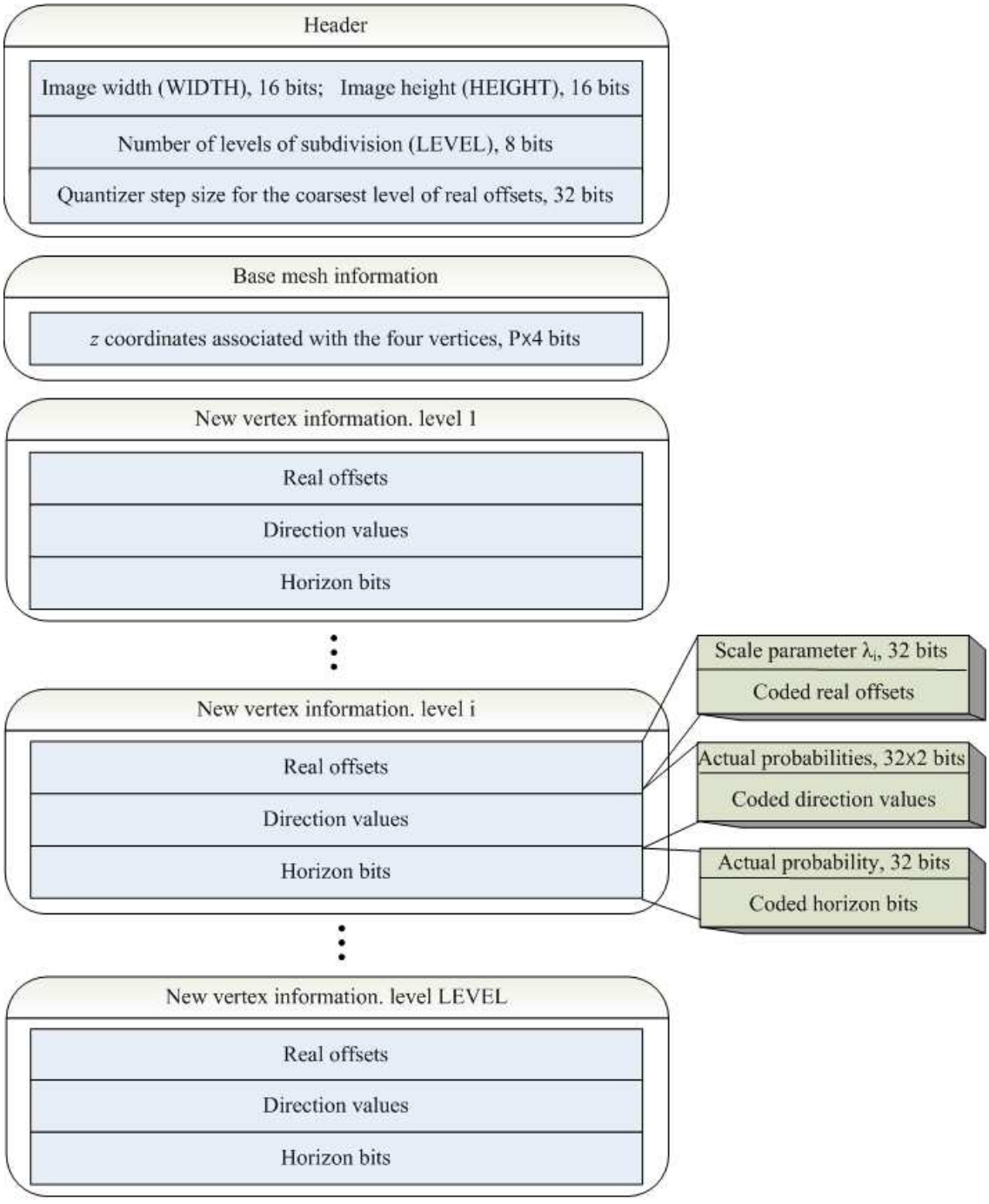


Figure 4.7: Code stream for the JBL coder.

4.4 Shortcomings of the JBL Coder

Having introduced the JBL coder, we now discuss some weaknesses of it. These weaknesses motivate our design of an enhanced coder, so that the enhanced coder can overcome these shortcomings and offer improved image coding performance.

4.4.1 Choice of Base Mesh

The JBL coder is very attractive due to its fast error-decay rate. Unfortunately, the JBL coder solely depends on the adaptivity of the normal subdivision to locate enough horizon vertices to form good polyline approximations of horizons. In so doing, normal subdivision introduces many vertices that are not helpful in improving polyline approximations of horizons. The vertices in the regions far from horizons contribute less to a good surface approximation. Thus, the coder is less efficient, especially for low-bit-rate compression. Figure 4.8 illustrates the ineffectiveness of using a data-independent base mesh to construct a polyline approximation of the horizon. In a good polyline approximation of a horizon, triangle edges need to be avoided from crossing the horizon. Triangle edges tangential to the horizon curve are preferred. Unfortunately, for the simple horizon-class image `circle3`, the JBL coder uses five levels of subdivision and many vertices to only partially construct a polyline approximation of the horizon. In Figure 4.8(f), many vertices are far away from the horizon, and not helpful in the polyline approximation of the horizon. For those vertices that are close to the horizon, some segments connecting them undesirably intersect the horizon. Those segments make the polyline approximation even more inefficient. Instead of completely depending on subdivision to form polyline approximations of horizons, by choosing vertices along the circle explicitly, the triangle edges can also be formed along the circle. Therefore, a much better approximation of the circle than the resulting approximation in Figure 4.8(f) can be produced. There is a good reason to believe that with a data-dependent base mesh, a better approximate of the horizon could be achieved. The ineffectiveness of the data-independent base mesh motivates our first proposed modification to the JBL coder. A modified coder may employ a data-dependent base mesh to form a good polyline approximation of the horizon.

4.4.2 Normal/Vertical Offset Format

The second shortcoming of the JBL coder involves the representation it employs for normal/vertical offsets. An offset measures the distance from a base point to its associated piercing point. Since neither the base point nor the piercing point falls on an integer grid, they can be anywhere on the image surface. The offset measuring the distance between the two points is a real number.

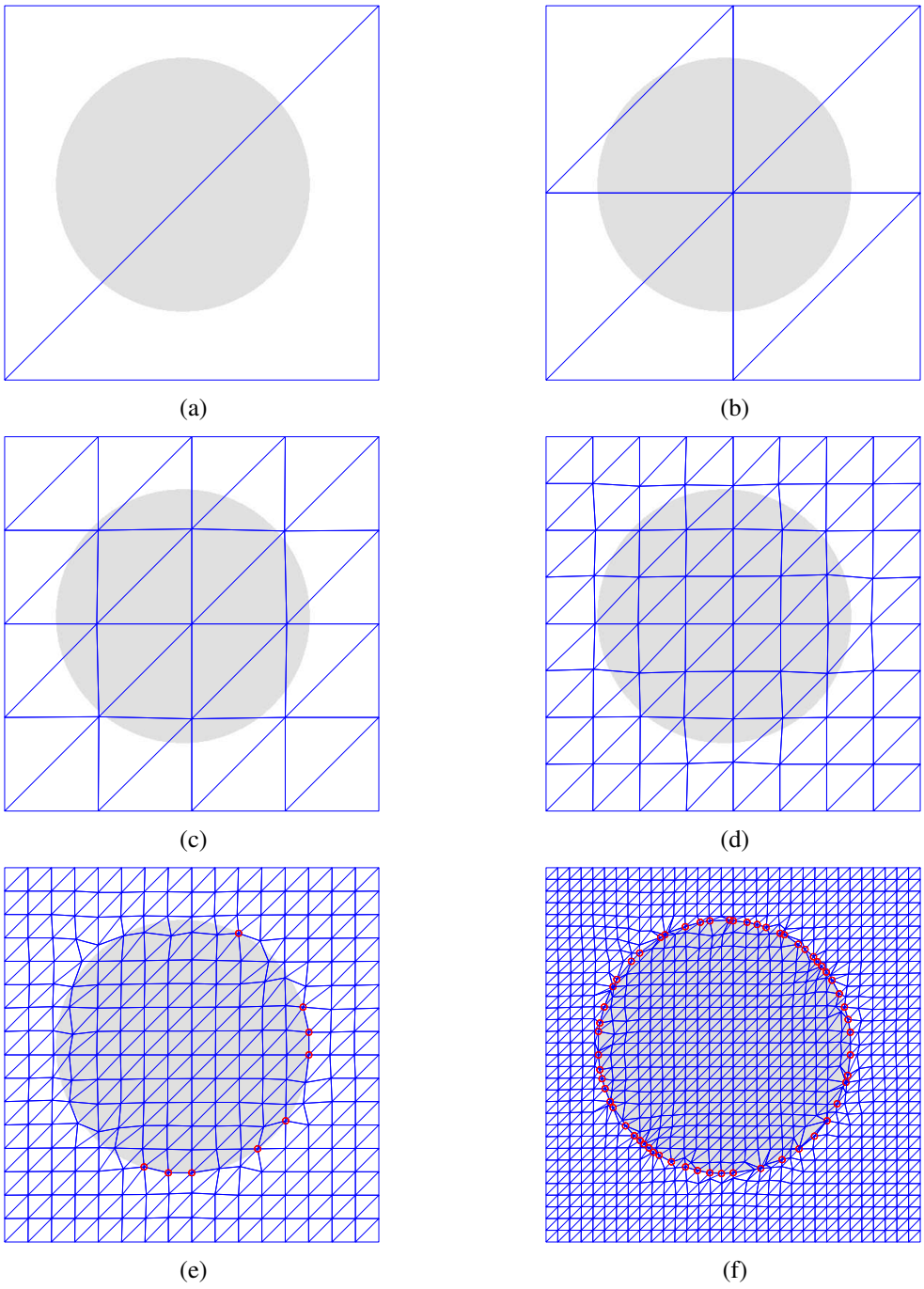


Figure 4.8: Example illustrating the ineffectiveness of a data-independent base mesh. The (a) base mesh for image `circle3`, and the meshes obtained after the (b) first, (c) second, (d) third, (e) fourth, and (f) fifth levels of subdivision.

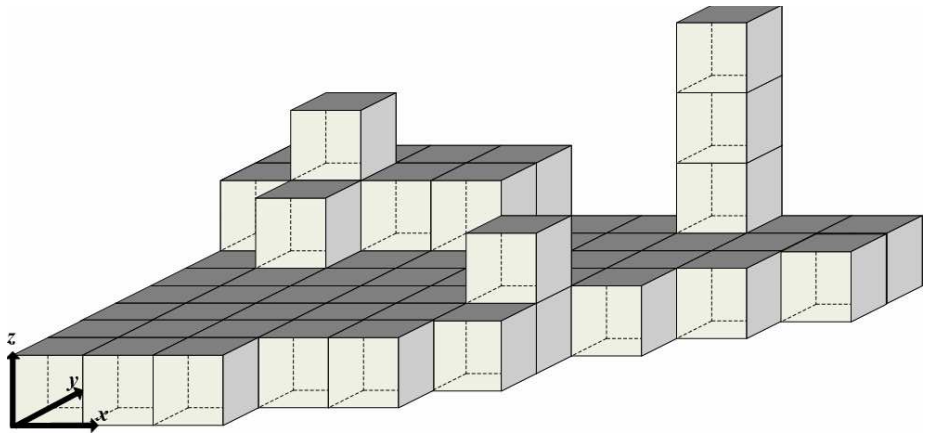


Figure 4.9: Digital image surface model.

By further observation, we notice that the image surface obtained from a digital image is not an arbitrary surface (in 3-D). Due to the piecewise constant interpolation used to generate the image surface, any point on the surface has at least one of its x, y, or z coordinates being integer. Figure 4.9 illustrates a part of a surface converted from a digital image. We observe that each face on the surface is parallel to the xy, yz, or zx plane, and with its z, x, or y coordinate being an integer, respectively. This suggests that the JBL coder can be improved by exploiting this special property of the image surface.

4.4.3 Scan Conversion

The third shortcoming of the JBL coder involves the scheme employed for scan conversion. Ideally, we desire a scan-conversion scheme that preserves both smooth regions in an image and sharp intensity changes along horizons. Piecewise planar interpolation offers smoothness within each interpolated region, but the interpolant at the boundaries of piecewise planar regions is most likely not smooth due to the mismatch in partial derivatives along the boundaries of neighboring regions. A higher-order interpolation scheme should be able to improve the smoothness along the boundaries of neighboring triangular regions. Therefore, there is probably room for improvement in scan-conversion method.

4.5 Proposed Modifications to the JBL Coder

Having identified some shortcomings of the JBL coder, we now propose three modifications to it in order to overcome these weaknesses. As we will later show, each of these changes leads to improved coding performance.

4.5.1 Choice of Base Mesh

We begin our discussion by first introducing our modification to the choice of base mesh. A data-dependent base mesh aims at constructing good polyline approximations of horizons using few vertices. In what follows, we describe how to choose vertices for the base mesh and how to construct a triangulation based on the chosen vertices in order to form good polyline approximations of horizons.

The vertices on horizons are indirectly selected by exploiting edges in images. Horizons of an image surface and edges in the corresponding image are closely related. Therefore, we first select points from centers of pixels on edges. Then we shift these points to their closest horizon vertices and connect the vertices to construct polyline approximations of horizons.

In our scheme, we locate discrete edges in an image using the Canny edge detector. To avoid potential problems in some of the subsequent processing, any intersecting edges are split at their intersections. Then, we consider each new edge separately. Since curvature is usually computed based on parametric functions, such as $c(t) = \big(x(t), y(t)\big)$ for a plane curve, we need to impose some ordering on the pixels of an edge to link them together and form a curve. Each discrete edge in Cartesian form can be parameterized, so that a pixel on the edge is expressed as a function $(x[n], y[n])$ with respect to the discrete variable n. Then the curvature of the parameterized discrete edge is estimated by the method of [28].

We are now ready to discuss point selection along edges in the image. To help construct good polyline approximations of edges, points are selected from pixels on the edges based on the preceding curvature estimates. Such points include the starting point and endpoint of an edge. They also include the points whose associated curvature is beyond a threshold. By including the points associated with large curvature, an edge with sharp cusps is divided into several pieces. This approach solves the potential problem of needing many points to form a good polyline approximation at places with large curvature. Between the selected points (i.e., starting point, endpoint, and points at large curvature), more points are inserted so that the distance between the neighboring points is smaller than the reciprocal of the curvature at the local edge contour. As a result, more points are inserted at the place where the edge bends sharply, while fewer points are inserted at the place where the edge is relatively straight. The above point-insertion scheme collaborating with the carefully defined forbidden zone, which is also impacted by the local curvature of horizons, tries to ensure [22] the location of new horizon vertices during horizon-edge subdivision, so that polyline refinement of horizons is constructed.

Before introducing how to shift the above selected points to vertices on horizons, in what follows, we first describe the process for converting from discrete edges in images to horizon curves in the continuous domain. Discrete images are represented in a discrete domain, while normal-mesh-based representations are

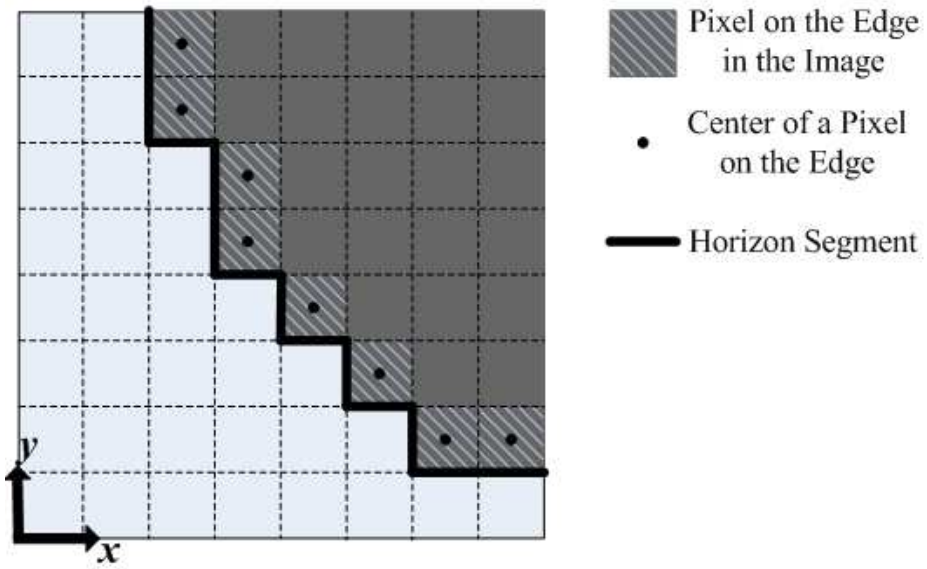


Figure 4.10: Conversion of discrete edges to horizons.

defined on a continuous domain. Therefore, the discrete images and edges in the images need to be converted to surfaces and horizons, respectively, in the continuous domain. Since piecewise constant interpolation is used to convert a digital image to a surface defined on a continuous domain, a horizon point (associated with a large jump in intensity) should be some point in the parameter plane with at least one of its x and y coordinates being an integer (i.e., a point at the boundary of two neighboring pixel areas). Hence, we convert edges to horizons by finding zigzag discontinuity contours of the image surface which are half-pixel away from the center of pixels on edges in the image. Figure 4.10 shows the conversion process from a discrete edge to a horizon. In the figure, each small square represents a pixel area in a digital image. The grid lines in the figure align with integer grids. This horizon-class image consists of two different intensity values. By using an edge detector, the hatched pixels in the figure are determined to be pixels on the edge in the image. The solid dots denote the centers of the pixels on the edge. The thick segments along the boundary of the two regions with different intensities constitute a horizon of the image surface in the continuous domain. Evidently, the horizon is formed by zigzag segments in the continuous domain. As discussed earlier, the points selected for approximating the edge are a subset of the centers of the pixels on the edge. Each selected point along the edge in the image is then shifted to its nearest horizon vertex, where the x and y coordinates are an integer and an element of $\frac{1}{2}\mathbb{Z}_{\text{odd}}$. Figure 4.11 illustrates the shifting process. A selected pixel on the edge in an image is shown in Figure 4.11. Let the central dot be the pixel center, which has the x and y coordinates being elements of $\frac{1}{2}\mathbb{Z}_{\text{odd}}$. The four arrows indicates all four shifting directions and positions where the potential

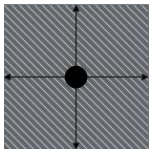


Figure 4.11: Shift a point at a pixel center to its closest horizon vertex.

horizon vertices with the closest distances to the pixel center can be found. If more than one of these four vertices are on the horizon, we shift the point from the center of a pixel to one of these nearest horizon vertices in the following priority (from high to low): the horizon vertices on the left, top, right, and bottom of the point. The resulting vertices shifted from the selected points are equivalent to sampling along the horizon according to the local curvature of the horizon contour.

Having chosen vertices along horizons, we now discuss the construction of base-mesh topology. In the base triangulation, the selected vertices on each horizon curve need to be connected in their associated parameterized order by some constrained segments, so that coarse polyline approximations of horizons are constructed. These connections are needed for fully exploiting the horizon model and realizing polyline refinements by normal subdivision. The selected horizon vertices and the necessary connections together constitute a planar straight line graph (PSLG). Due to possible intersections of the constrained segments, some segments occasionally need to be removed to generate a valid PSLG. A base triangulation for a data-dependent base mesh could be constructed based on the PSLG. To further improve the quality of the base triangulation, obtain a more uniform vertex distribution, and avoid sliver triangles in the parameter plane, we add some extra points (i.e., **Steiner points**) to the PSLG to form a new PSLG. The Steiner points are determined using the Triangle software [38] with the original PSLG (without Steiner points) being the input and the new PSLG (with Steiner points) embedded in a constrained Delaunay triangulation (DT) being the output. The Steiner points generated by the Triangle software have real x and y coordinates. To overcome the inefficiency of coding real coordinates, the coordinates of Steiner points are rounded to the centers of the nearest pixels. In other words, the rounded Steiner points have both x and y coordinates being elements of $\frac{1}{2}\mathbb{Z}_{\text{odd}}$. Clearly, no Steiner points could be a horizon vertex, since none of the coordinates of the Steiner points is an integer. Therefore, the rounding operation also eliminates the demand of storing horizon bits pertaining to the Steiner points. Furthermore, since the rounding of the Steiner point coordinates changes the vertex geometry only very slightly, the good vertex distribution is maintained in the data-dependent base mesh.

A constrained DT based on a PSLG is not unique. In order to guarantee the uniqueness of the base triangulation in the parameter plane, a constrained DT [2] with preferred directions [17] is applied to the new PSLG. Due to the uniqueness of the resulting triangulation, the efforts of storing topology information for

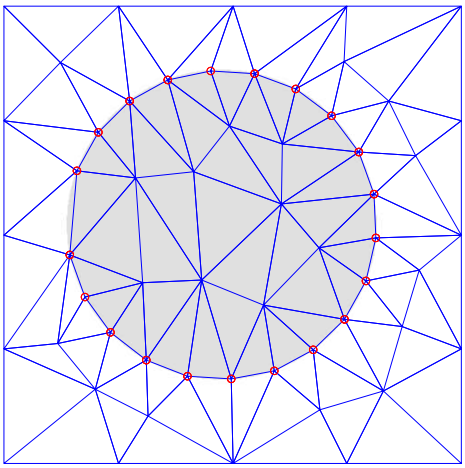


Figure 4.12: Example of a data-dependent base mesh for image `circle3`.

the base triangulation is unnecessary. Therefore, only the PSLG information needs to be stored in order to reproduce the base mesh.

As it turns out, due to the geometry of vertices in our data-dependent base mesh, many constrained edges in a constrained DT are also edges in a DT (i.e., constraints are inactive). The large number of inactive constraints can be partly attributed to the small polyline resolution (i.e., the length of the longest segment in the PSLG) and good distribution of vertices in the PSLG. As an optimization, we only encode the constrained segments that affect resulting a constrained DT. In this way, the information that needs to be coded for the base mesh (more precisely the PSLG) can be significantly reduced.

By carefully choosing a data-dependent base mesh with both good geometry and good topology as explained earlier, superior polyline approximations of horizons are constructed in the base mesh. Figure 4.12 shows a data-dependent base mesh for the image `circle3`. The base mesh contains few vertices, and possesses a good approximation of the horizon. In comparison, we recall the refined meshes for the same image obtained from a data-independent base mesh, which are shown in Figure 4.8. We observe that using the data-independent base mesh, the refined mesh obtained after five levels of subdivision contains much more vertices than the data-dependent base mesh. Furthermore, due to some intersections of segments with the horizon curve, the mesh in Figure 4.8(f) has a poorer polyline approximation of the horizon than the mesh in Figure 4.12.

4.5.2 Normal/Vertical Offset Format

Our second modification to the JBL coder involves the representation of normal/vertical offsets. Since integers can be more efficiently coded than real numbers, we propose to use an integer offset instead of a real offset for locating each piercing point. Recall that an offset is a real value measuring the distance between a base point and its corresponding piercing point. Due to the piecewise constant interpolation process used to generate the image surface, we observe that at least one of the x, y, or z coordinates of each piercing point must be integer. Therefore, along the normal/vertical search line through the base point, all potential piercing points can be identified, by finding all intersections of the search line and the planes of the form $x = c$, $y = c$, and $z = c$, where $c \in \mathbb{Z}$. All potential piercing points are enumerated with an integer index. This index can be used to specify which of the potential piercing points is the actual piercing point. By doing so, a piercing point is explicitly represented by an integer index. The offset data can be more efficiently encoded than if real offsets were used.

Figure 4.13 shows a set of potential piercing points found as intersections of the search line and the planes of the form $x = c$, where $c \in \mathbb{Z}$. In the figure, the segment with unfilled circles as endpoints is a triangle edge in a mesh. The search line is through the star-shaped base point. The search line intersects the set of planes of the form $x = c$, where $c \in \mathbb{Z}$, at the filled circles. Figure 4.14 further illustrates the manner of finding all potential piercing points by locating all three sets of intersections. The intersections of the search line with the three sets of planes of the form $x = c$, $y = c$, and $z = c$, where $c \in \mathbb{Z}$, are denoted by filled circles, filled squares, and unfilled squares, respectively. All potential piercing point are enumerated with an integer index.

4.5.3 Scan Conversion

Instead of using piecewise planar interpolation for scan conversion, a higher-order interpolation scheme should be able to improve the smoothness of interpolant at the boundaries of neighboring triangular regions. Bicubic interpolation [48, p.446-449] offers a higher degree of smoothness than planar interpolation used in the JBL coder. Bicubic interpolation can often be desirable with respect to subjective image quality. Figure 4.15 illustrates bicubic interpolation for a triangular region. In the piecewise constant image, the vertically-hatched triangular region is the region of interest for which we want to determine an interpolant. The proposed method generates an interpolant that passes through the three vertices of the triangle in the mesh, and also has continuous first-order partial derivatives. All direct neighbors to the three vertices of the vertically-hatched triangle are involved in determining the partial derivative information. As shown in the figure, all vertex/height-value pairs in and on the boundary of the diagonally-hatched regions are used

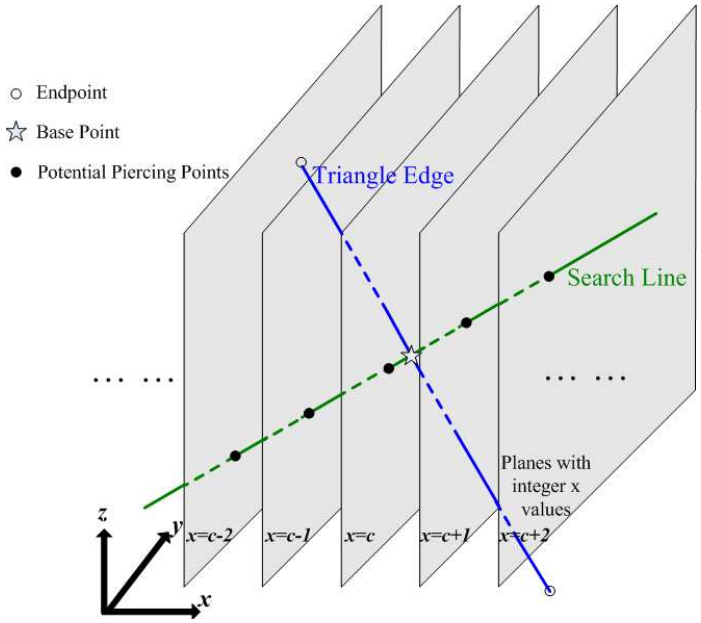


Figure 4.13: Potential piercing points intersected with a set of planes with integer x coordinate.

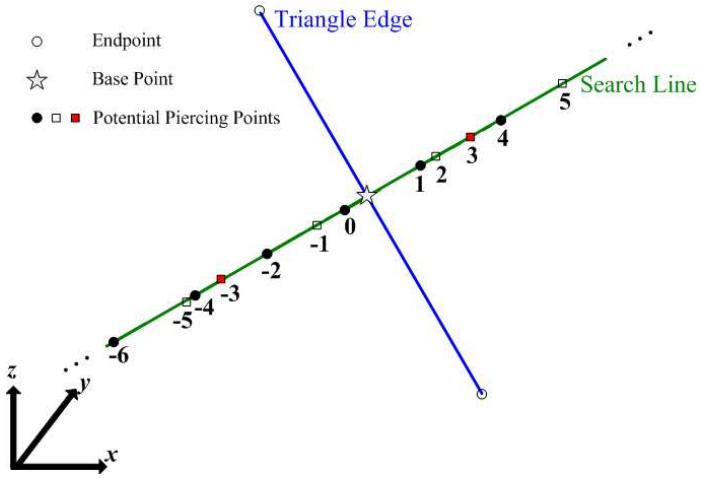


Figure 4.14: All potential piercing points.

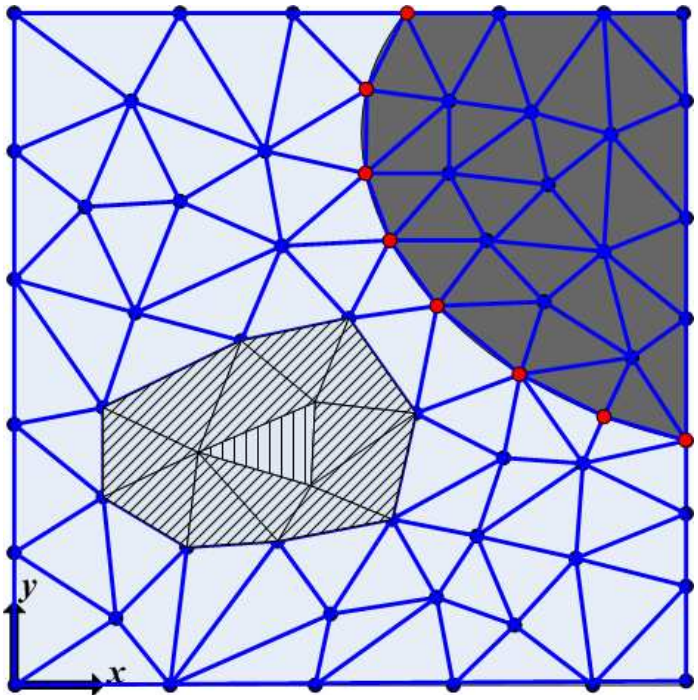


Figure 4.15: Improved scan conversion.

for bicubic interpolation. The interpolant above the vertically-hatched triangular region is a part of the final interpolated surface. By combining all interpolants for the triangle mesh, an image surface is constructed. The bicubic interpolation described above offers smoothness at the boundaries of neighboring triangles. It, however, has the undesired potential effect to smooth the surface discontinuities at horizons.

Since we would like to preserve sharp image edges, we must apply bicubic interpolation carefully, so that blurring of image edges is avoided. In our scheme, when determining the bicubic interpolant for a particular triangle, we use only vertex/height-value pairs from the same side of the horizon as the triangular region of interest. This does not affect interpolation of regions far away from the horizons, while for regions near the horizons, some adaptation needs to be made. As depicted in Figure 4.16, the vertically-hatched triangular region is our region of interest for which we want to determine an interpolant. The diagonally hatched regions straddle the horizon. We use only vertex/height-value pairs in and on the boundary of the diagonally-hatched white-shaded regions for interpolation, since the triangular region of interest is also from the same side of the horizon (i.e., the white-shaded as opposed to gray-shaded region). This adaptation near horizons allows us to avoid unnecessarily smoothing the sharp intensity changes across horizons. Finally, the improved interpolated surface is sampled on grids at centers of pixels in the parameter plane to generate a discrete signal.

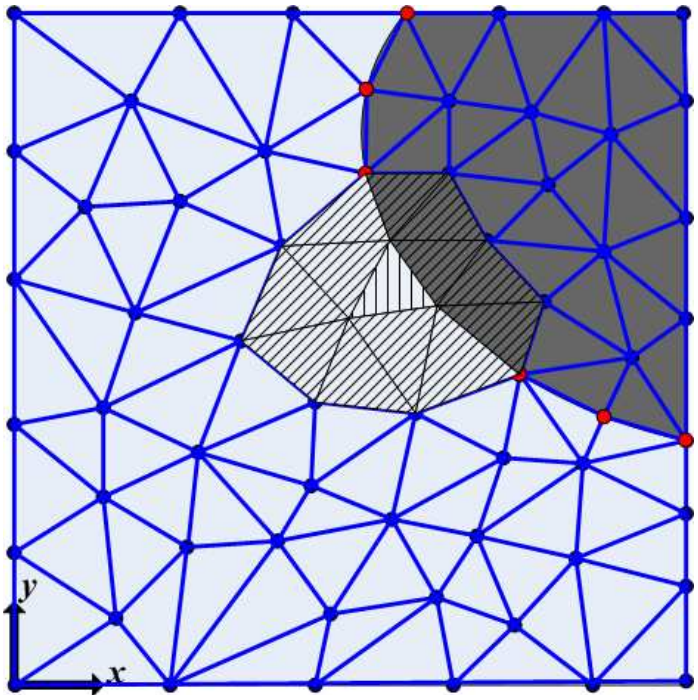


Figure 4.16: Improved scan conversion near a horizon.

4.6 Our Implementation of the Enhanced JBL Coder

Having introduced three modifications to the JBL coder, we briefly describe our implementation of an enhanced JBL coder in this section. Since the implementation of the enhanced coder is based on our implementation of the JBL coder, we only point out the differences between our implementation of the JBL coder and the enhanced version. To help explain the implementation and performance differences between various coders (i.e., the JBL, JBL-S, and enhanced coders), the features of these coders are summarized in Table 4.2. The enhanced coder supports all combinations of data-independent/data-dependent base mesh, real/integer offsets, and planar-based/bicubic-based scan conversion. Table 4.3 shows the data that needs to be coded for the enhanced coder. The real or integer offset data is assumed to be entropy coded, and the entropy can be computed in the manner given by (4.1). For integer offsets, the specified quantizer step size parameter q should be greater or equal to 1. When $q = 1$, the quantization is bypassed, and no information is lost due to the quantization. The entropy of direction-value and horizon-bit data can be estimated using (4.2) as in the JBL coder.

As for the base mesh, the coding of the data-independent base mesh has been discussed previously in Section 4.3. Now we will briefly describe the way in which the data-dependent base mesh is assumed to be

Table 4.2: Summary of features for the various coders

Coder	Image Interpolant	Model	Base Mesh	Offset Format	Scan-Conversion Interpolant
JBL-S	piecewise planar	nonhorizon	data independent	$\mathbb{R}$	piecewise planar
JBL	piecewise constant	horizon	data independent	$\mathbb{R}$	piecewise planar
Enhanced	piecewise constant	horizon	data dependent/independent	$\mathbb{R}$ or $\mathbb{Z}$	piecewise planar/bicubic

Table 4.3: Summary of data for the enhanced coder

Data Name			Type	Range of Value
offset			$\mathbb{R}$ or $\mathbb{Z}$	$-a$ to a, where $a = \max(2^P - 1, \frac{\text{WIDTH}+\text{HEIGHT}}{2})$
direction value			$\mathbb{Z}$	$\{0,1,2\}$
horizon bit			boolean	$\{0,1\}$
base mesh	$x^\dagger$	horizon	$\frac{1}{2}\mathbb{Z}$	0 to WIDTH $-$ 1
		Steiner	$\frac{1}{2}\mathbb{Z}_{\text{odd}}$	
	$y^\dagger$	horizon	$\frac{1}{2}\mathbb{Z}$	0 to HEIGHT $-$ 1
		Steiner	$\frac{1}{2}\mathbb{Z}_{\text{odd}}$	
	z		$\mathbb{Z}$	0 to $2^P - 1$
	constrained segment†		$\mathbb{Z}$	0 to the number of vertices in the base mesh$-$1

† only employed when using data-dependent base mesh

coded for the enhanced coder. The x, y and z coordinates of vertices of the base mesh tend to follow uniform distributions. Hence, we choose to not apply any coding scheme to the coordinates. Let WIDTH, HEIGHT, and P denote the width, height, and the number of bits per sample of the original image, respectively. For the horizon vertices in the base mesh, both of the x and y coordinates are an element of $\frac{1}{2}\mathbb{Z}$. Furthermore, we know that if the x coordinate is an integer, the corresponding y coordinate must be an element of $\frac{1}{2}\mathbb{Z}_{\text{odd}}$, and vice versa. Although the width and height of the image can be represented with $\lceil \log_2 \text{WIDTH} \rceil$ and $\lceil \log_2 \text{HEIGHT} \rceil$ bits, respectively, each x coordinate is stored with one more bit than $\lceil \log_2 \text{WIDTH} \rceil$, and each y coordinate is stored with $\lceil \log_2 \text{HEIGHT} \rceil$ bits. For Steiner points in the base mesh, we use $\lceil \log_2 \text{WIDTH} \rceil$ and $\lceil \log_2 \text{HEIGHT} \rceil$ bits to represent each x and y coordinate, respectively. Since the x and y coordinates of Steiner points are always elements of $\frac{1}{2}\mathbb{Z}_{\text{odd}}$, the actual coordinates of the Steiner points are shifted by $\frac{1}{2}$ along both x and y axes before coding in order to save one bit per coordinate. The four corner points of the image bounding box are always chosen as vertices in the base mesh. Their x and y coordinates can be derived from the size of the image, and therefore do not need to be coded. Each z coordinate in the base mesh is represented using P bits. Figure 4.17 describes the code stream for the enhanced coder. Similar to the code stream for the JBL coder, some additional information is coded in the header. Then, the x, y, and z coordinates of the base mesh are coded as explained earlier. All vertices in the base mesh are numbered by their orders in the code stream. The active constrained line segments in the base mesh are stored by the indices of their endpoints. Finally, the data associated with different levels of subdivision are coded, starting from data associated with the smallest subdivision level index and proceeding to the data associated with the largest subdivision level index.

Note that, since we do not entropy code the base mesh information, there might be more efficient schemes for handling this data. At high bit rates, the base mesh is a small portion of data in comparison with other information that needs to be coded. Therefore, the scheme we use for coding the base mesh is still acceptable but somewhat suboptimal. At low bit rates, the scheme we use for coding the base mesh does not degrade the performance in a significant way, nor make the coder completely useless. Our proposed coder would require fewer bits, however, if the base mesh were coded in a more intelligent way, such as through a differential coding of the vertex coordinates.

4.7 Experimental Results

Earlier, we suggested that our proposed modifications to the JBL coder improve its performance. In the sections that follow, we support our claim through experimental evidence. Although numerous 8-bit grayscale

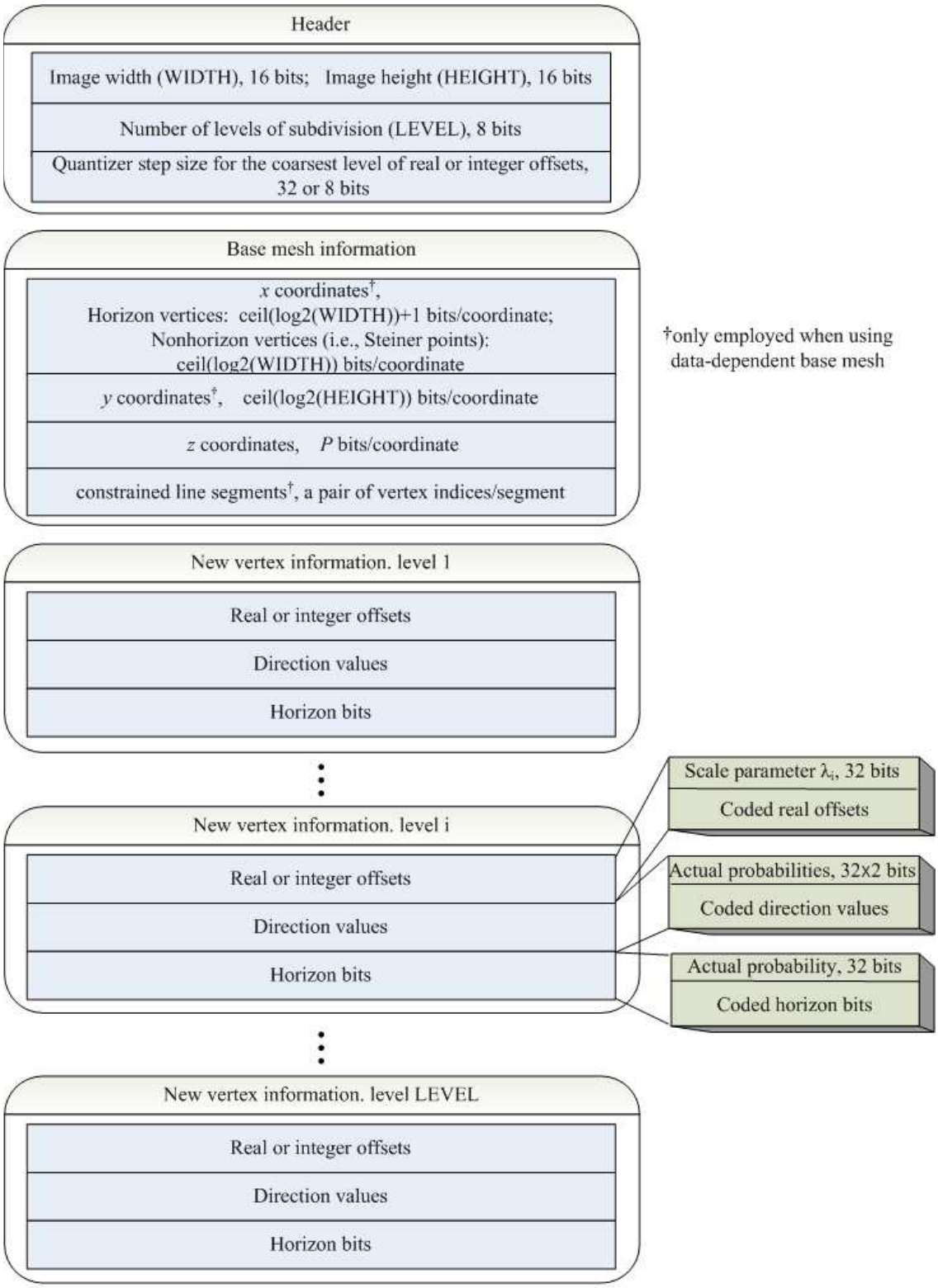


Figure 4.17: Code stream for the enhanced coder.

Table 4.4: Test images for normal-mesh-based coders

Image	Size	Description
arch	512×512	collection of shaded objects
arti	256×256	piecewise constant image with four grey intensities
checkerboard	640×640	checkerboard
circle1	256×256	circle (white-black)
circle2	512×512	circle (white-black)
circle3	512×512	circle (white-grey)
flower	1024×1024	piecewise constant image with black and white
heart	1024×1024	heart with smooth intensity changes
lena	512×512	woman
paw	1024×1024	paw
peppers	512×512	peppers
slope	256×256	regions with linear-intensity changes

test images were employed in our work as listed in Table 4.4, we focus our attention on the results for two representative images herein, namely, the `paw` and `peppers` images. Of the test images that we used, the nonstandard ones are shown in Figure 4.18.

Since we are primarily interested in low bit rates in our work, when we subsequently use qualifiers like "low" or "high" for the bit rate, these qualifiers should be understood in relative terms (i.e., relative to the range of bit rates under consideration in our study). In what follows, we evaluate the performance of each of the three proposed modifications to the JBL coder in turn.

4.7.1 Choice of Base Mesh

To begin, we consider our proposed modification to the JBL coder of employing a data-dependent base mesh. In what follows, we refer to the JBL coder with this change by the name "XA". To assess the value of our proposed change, numerous test images were compressed at various bit rates with both the JBL and XA methods and the results are examined. In what follows, we provide a representative subset of these results for the `paw` and `peppers` images. For reference purposes, we also include results obtained from the JBL-S and (in some cases) JPEG-2000 [20] coders. To maintain a fair comparison for the mesh-based methods, the number of subdivision levels for each method was chosen so that the final-mesh vertex counts would be as

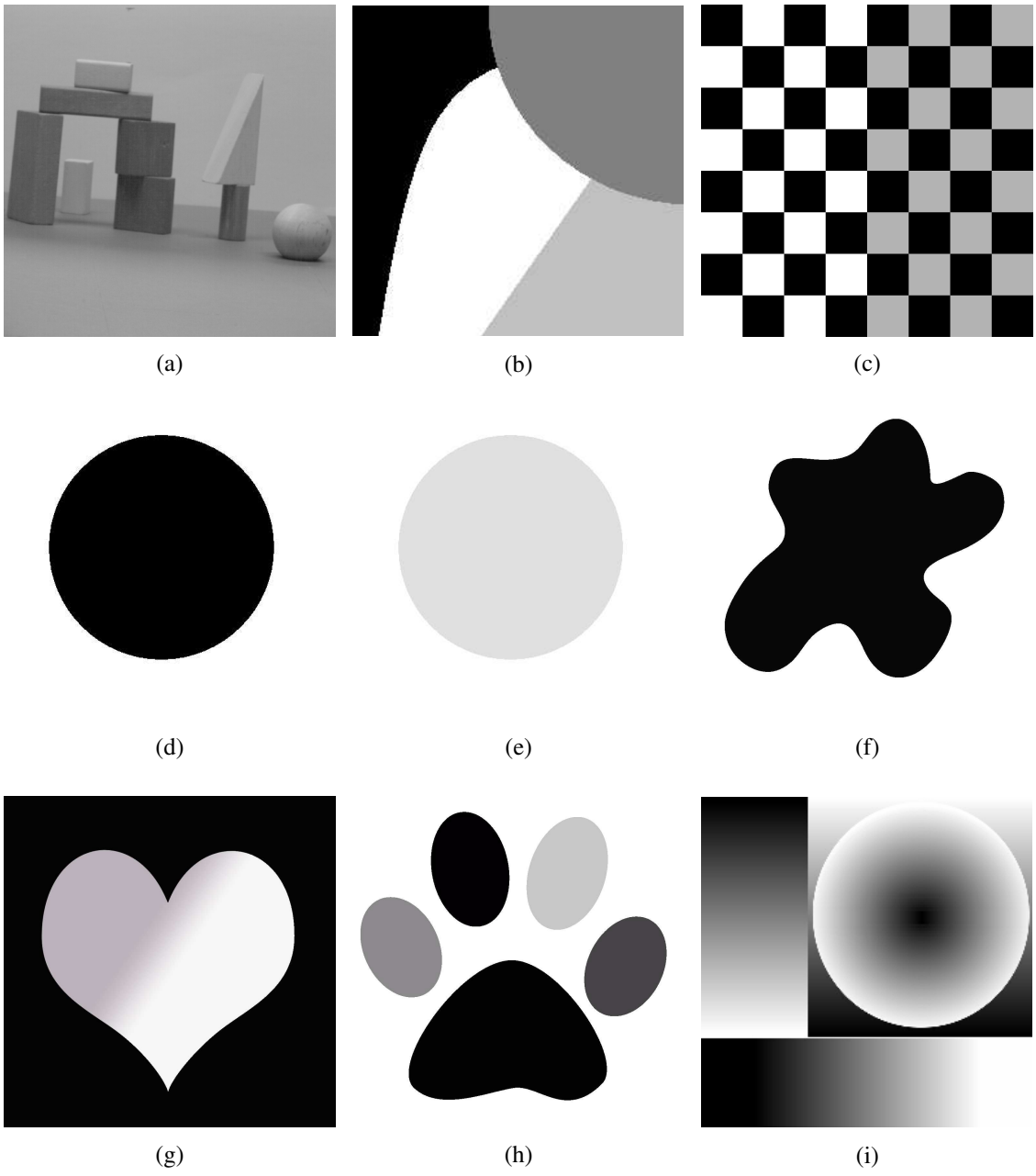


Figure 4.18: Test images for normal-mesh-based coders. The (a) `arch`, (b) `arti`, (c) `checkerboard`, (d) `circle1/circle2`, (e) `circle3`, (f) `flower`, (g) `heart`, (h) `paw`, (i) `slope` images.

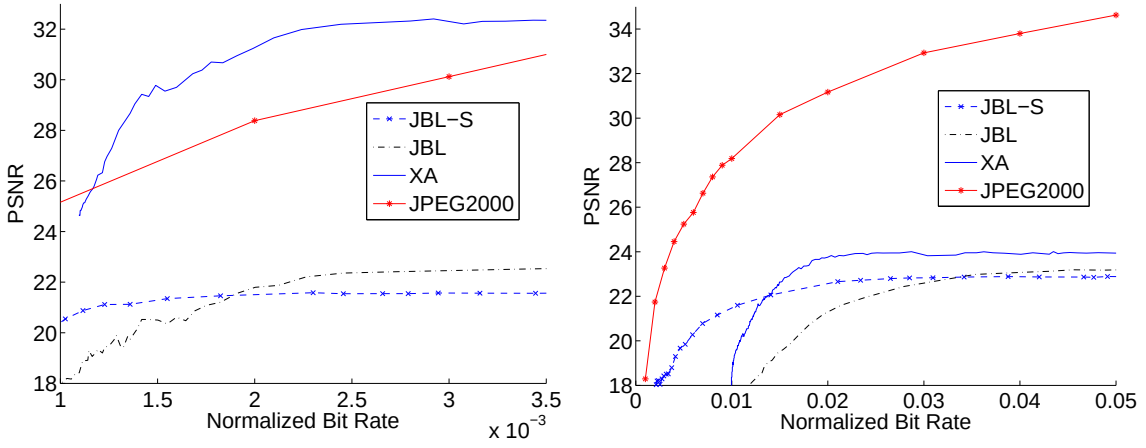


Figure 4.19: Coding performance for the (a) `paw` and (b) `peppers` images using the JBL-S, JBL, XA, and JPEG-2000 methods.

close to one another as possible (without giving an unfair advantage to our XA method). Since these counts can only be controlled very coarsely, it is only possible to have them match to within a factor of about three. More specifically, for the `paw` image, the JBL and JBL-S coders use six levels of subdivision, resulting in 4225 vertices, while the XA coder uses two levels of subdivision, resulting in 3308 vertices. For the `peppers` image, the JBL and JBL-S coders use seven levels of subdivision, resulting in 16641 vertices, and the XA coder uses two levels of subdivision, resulting in 6543 vertices. Now, we examine the results obtained in detail.

The rate-distortion plots obtained for the `paw` and `peppers` images using the various methods are shown in Figure 4.19. From these results, we can see that, at high bit rates, the XA coder outperforms the JBL and JBL-S coders, and the XA coder even outperforms the JPEG-2000 coder in the case of the (synthetic) `paw` image. At low bit rates, however, the XA coder can sometimes perform more poorly than the other three methods due to the rate overhead associated with the base mesh (which is not entropy coded in our scheme). As mentioned earlier, clever schemes for coding the base mesh, however, could reduce this overhead, and significantly improve the coding efficiency of the XA method at low bit rates. From the coding results, we can also see that, at low bit rates, the JBL coder performs worse than the JBL-S coder, while at high bit rates, the JBL coder performs better than or comparable to the JBL-S coder. The superior performance of the JBL coder at high bit rates can be attributed to the higher efficiency of the horizon model, while the inferior performance at low bit rates can be attributed to the overhead of encoding horizon bits and direction values.

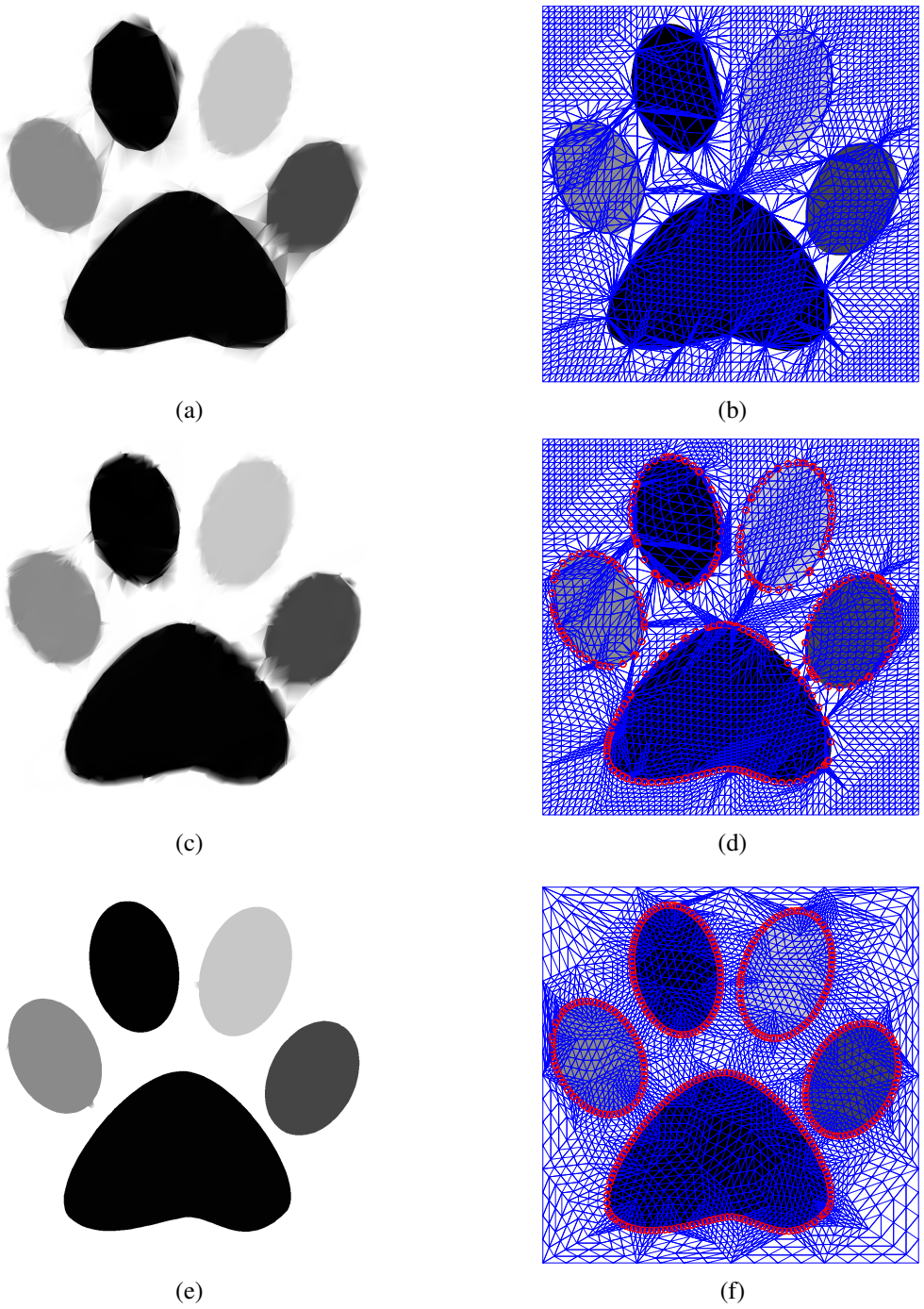
Now, we consider the subjective performance of the various methods. For the case of the `paw` and

peppers images, examples of the obtained reconstructed images are shown in Figures 4.20 and 4.21, respectively. In the first case, the final mesh employed by each method is shown superimposed on the original image with the horizon vertices denoted by circles. Examining the results for the paw image in Figure 4.20, we can see that very significant edge distortions occur in the case of the JBL and JBL-S methods, while the XA scheme has little noticeable distortion. Clearly, the XA method approximates horizons much better than the JBL and JBL-S methods. Examining the reconstructions of the peppers image in Figure 4.21, we can see that the results obtained with the XA method are comparable to those obtained with the JBL and JBL-S coders, in spite of the fact that the JBL and JBL-S methods have meshes with about 2.5 times more vertices than the XA case. On this basis, it is reasonable to conclude that the XA scheme is superior to the JBL and JBL-S schemes.

Let us again consider the subjective results for the paw image, including the final meshes produced by the various methods, as shown in Figure 4.20. By examining the final meshes, we can see why the XA method is able to outperform the JBL and JBL-S methods. The mesh for the JBL-S method has some larger-area triangles that straddle horizons. This leads to very visually disturbing artifacts such as those near the rightmost pad of the paw in Figure 4.20(a). By explicitly modelling horizons, the JBL and XA methods are able to locate horizon vertices faster, and reduce distortions in large regions. Furthermore, the XA method, with a data-dependent base mesh, locates horizon vertices faster and approximates horizons better than the JBL and JBL-S schemes. As an aside, we note that the small triangular teeth occurring in the reconstructed image for the XA coder can be attributed to inaccuracies in the estimation of the location and curvature of horizons.

4.7.2 Real Versus Integer Offsets

Let us now consider our second proposed change to the JBL coder, which is to use integer values rather than real values to represent normal/vertical offsets. To demonstrate the effectiveness of our proposed change, we compare the coding performance obtained with real and integer offsets. Figure 4.22 shows the coding performance achieved when each method is employed in the XA coder (using two levels of subdivision) for the paw image. From the graphs, we can see that, at low to medium bit rates, integer offsets yield better results than real offsets, with the difference being more pronounced at medium rates, while in the high bit-rate case, comparable results are obtained with integer and real offsets. It is worth noting that similar results as above also hold in terms of subjective image quality (i.e., integer offsets are better than or as good as real offsets).



(a) (b)

(c) (d)

(e) (f)

Figure 4.20: Coding example for the `paw` image. Lossy reconstructions obtained at about 400:1 compression using the (a) JBL-S, (c) JBL, and (e) XA methods. The corresponding final meshes employed by the (b) JBL-S, (d) JBL, and (f) XA methods.

(a)

(b)

(c)

(d)

Figure 4.21: Coding example for the `peppers` image. Portions of the (a) original image and the lossy reconstructions obtained at about 29:1 compression using the (b) JBL-S, (c) JBL, and (d) XA methods.

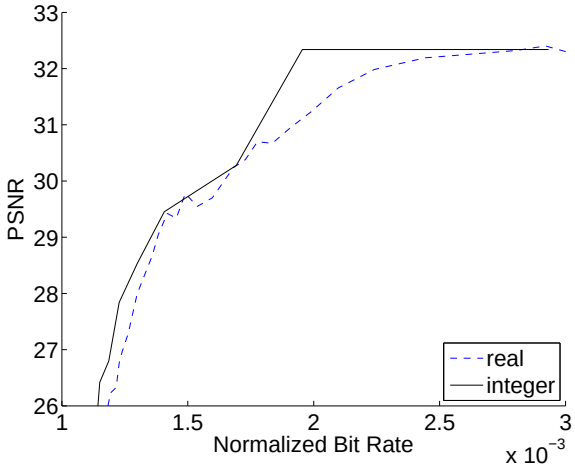


Figure 4.22: Coding performance for the `paw` image using real and integer offsets.

4.7.3 Planar Versus Bicubic Interpolation

Lastly, we consider our third proposed change to the coder, which is to use bicubic instead of planar interpolation for scan conversion. To evaluate the value of the change, both planar and bicubic interpolation were employed in the XA coder to compress the `peppers` image at various bit rates (using two levels of subdivision). The results obtained are shown in Figure 4.23. From these results, we can see that bicubic interpolation outperforms planar interpolation, especially at higher bit rates. In terms of subjective image quality, bicubic interpolation also leads to superior results, as it tends to better preserve smoother regions in images, without destroying sharp intensity changes at horizons.

4.7.4 Discussion of Different Coders for Various Types of Images

In this section, the performance of different coders for various types of images is discussed from several perspectives. For images with piecewise smooth regions, the normal-mesh-based coders code images with high resolution more efficiently than those with low resolution. The difference in coding efficiency might be caused by the difference in accuracy of curvature estimates for curves with distinct curvatures. More specifically, for images with low resolution, the zigzag digital edges tend to have more impact on the curvature estimates than for images with high resolution.

To code images with essentially the same content but different intensity contrast, using a coder with a data-dependent base mesh yields smaller differences in the coding performance than using a coder with a data-independent base mesh. For the coder with a data-independent base mesh, the polyline approximations

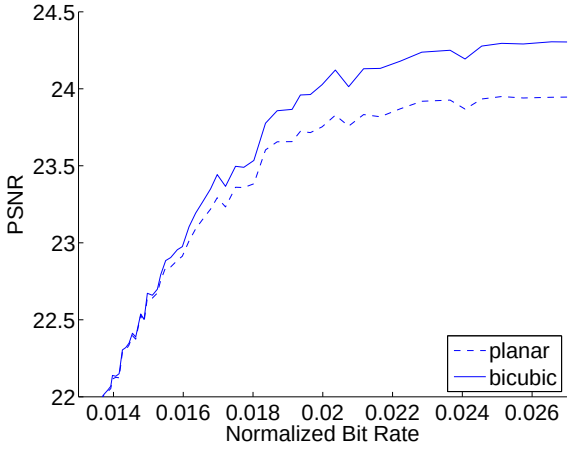


Figure 4.23: Coding performance for the `peppers` image using planar and bicubic interpolation.

of horizons are constructed faster for images with high intensity contrast than images with low intensity contrast. Since the ratio between the height of jump in image surface and the length of triangle edges in a mesh is larger for images with high intensity contrast, the normal subdivision locates horizon vertices faster and construct polyline approximations of horizons better. Figure 4.24 gives a subdivision example using the JBL coder for coding images with the same content but different intensity contrast. Figure 4.24 (a) shows the base mesh superimposed on the test image `circle2`, which has high intensity contrast. Figures 4.24 (b), (c), (d), and (e) are the refined meshes obtained after the first, second, third, and the fourth levels of subdivision of the image `circle2`, with the horizon vertices denoted by small unfilled circles. Figure 4.24 (f) is the refined mesh obtained after the fifth level of subdivision of the image `circle3`, which has the same image content as `circle2` but with low intensity contrast. Examining the meshes for the two images, we can see that a good polyline approximation of horizon is constructed in Figure 4.24(e), while in Figure 4.24(f), some triangle edges cross the horizon and only a portion of the polyline approximation of the horizon is formed. When considering the fact that there is much more vertices in Figure 4.24(f) than in Figure 4.24(e), we can safely conclude that a fast location and better polyline approximation of horizons tend to occur when subdividing image with high intensity contrast with a data-independent normal-mesh-based coder.

For a data-independent base mesh with vertices being the four corner points of the image bounding box, the base triangulation can be formed in one of the two ways, depending on which diagonal of the image bounding box is included in the base triangulation. For interest sake, we compared coding performance for the same image using the two different data-independent base meshes. We observe that the two different base meshes generate distinct subjective results. After several levels of subdivision, many triangles in the final

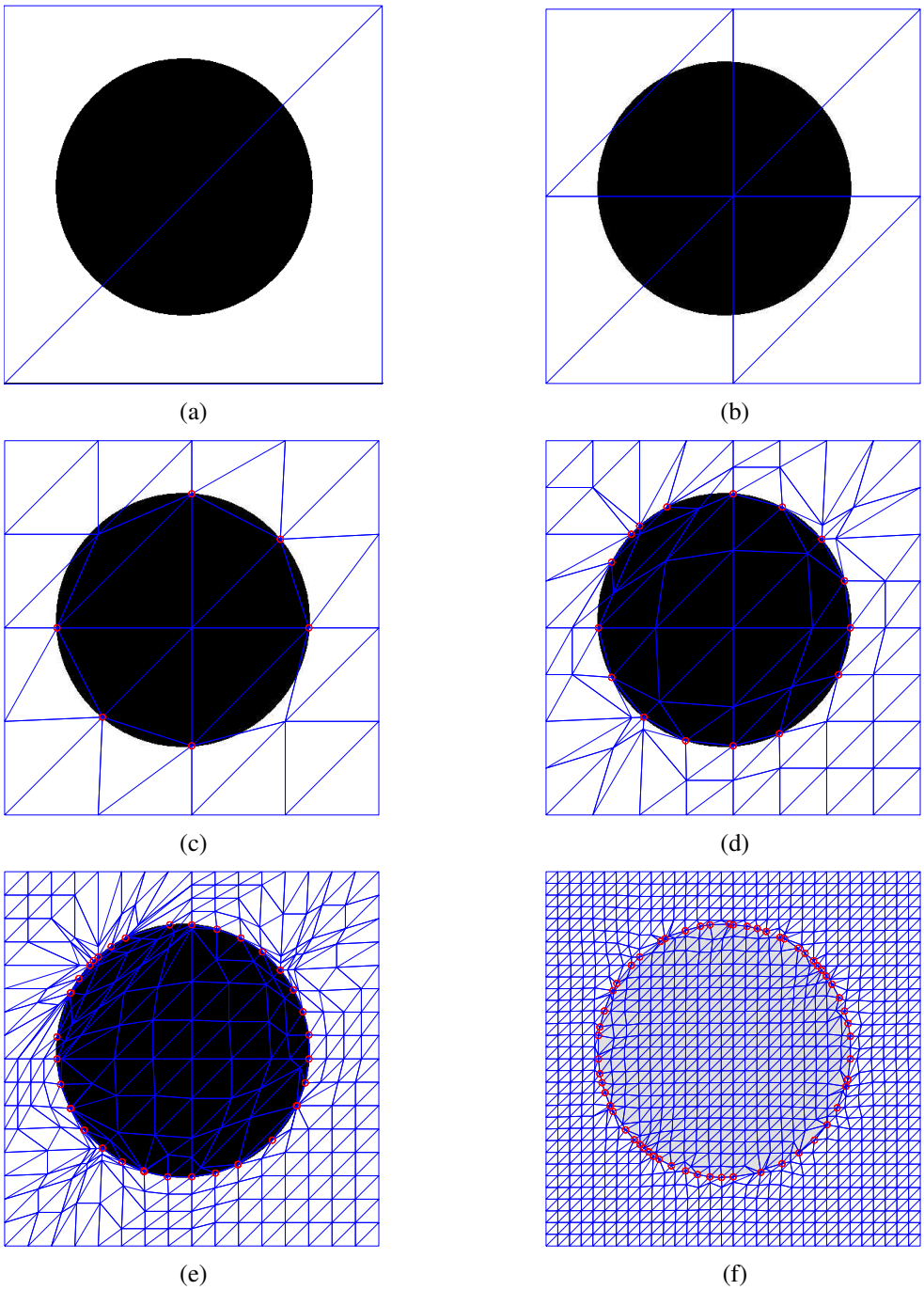
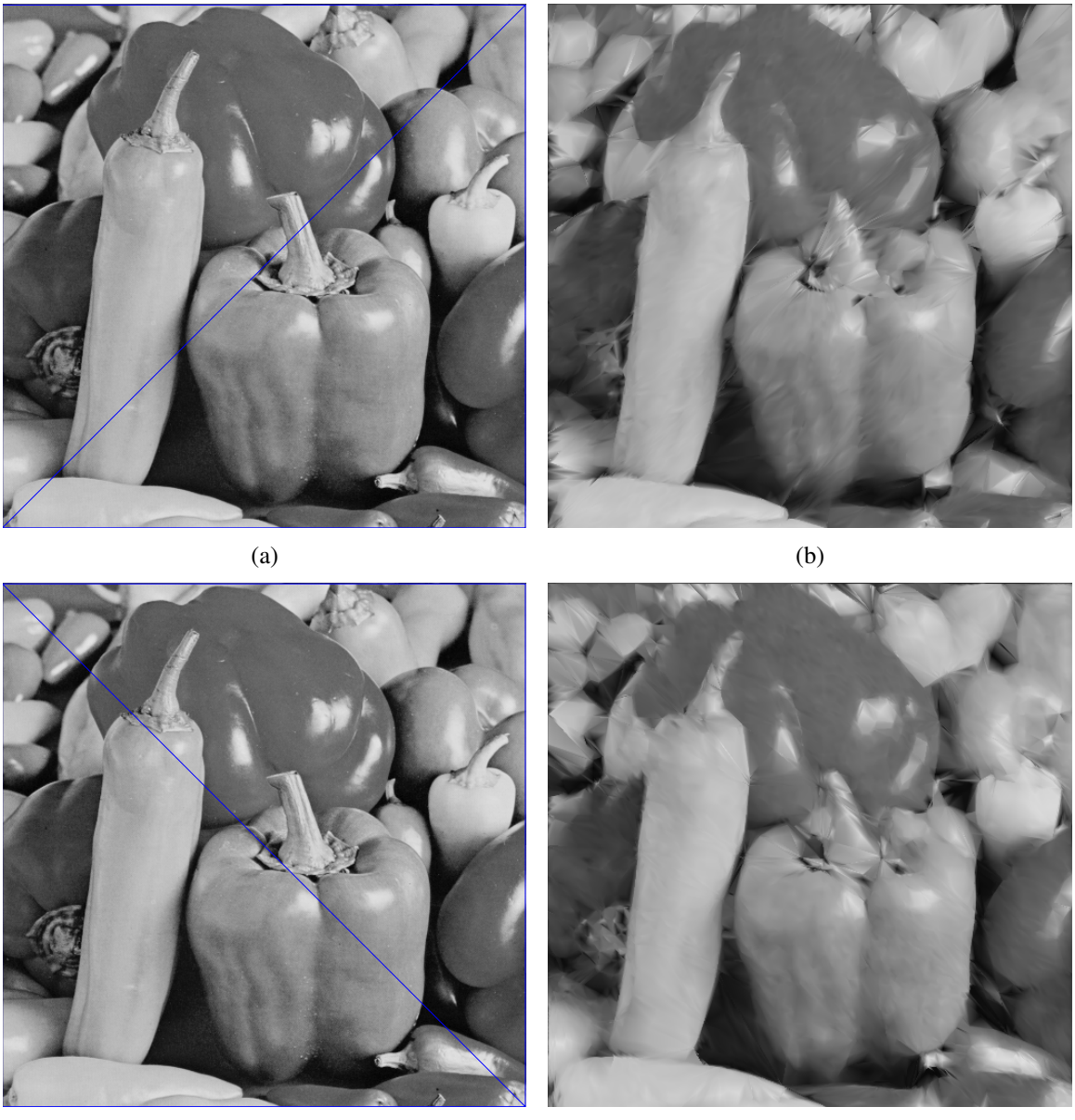


Figure 4.24: Example of the JBL coder for images with different intensity contrast. The (a) base mesh, meshes obtained after the (b) first, (c) second, (d) third, and (e) fourth levels of subdivision of the image `circle2`. The (f) mesh obtained after the fifth level of subdivision of the image `circle3`.

mesh tend to be thin, long, and roughly parallel to the connected diagonal of the image bounding box in the base triangulation, and hence result in noticeable artifacts in a certain direction. Figures 4.25 (a) and (c) show two base triangulations of the `peppers` image superimposed on the original images. The resulting image representations, after seven levels of subdivision starting from the two different base meshes, are shown in Figures 4.25 (b) and (d), respectively. We observe many annoying distortions along the diagonal directions of the base meshes for both reconstructed images.

4.8 Summary

In this chapter, we proposed three modifications to the JBL coder and demonstrated through experimental results that these changes lead to improved coding performance. For example, we showed that good data-dependent base meshes can help to locate horizons faster and preserve edges better. Also, we showed that using a normal/vertical-offset representation based on integers (instead of real numbers) yields superior performance. Finally, we demonstrated that, by exploiting horizon information, bicubic interpolation can be made to provide smoother reconstructed images while still maintaining sharp edges.



(a) (b)

(c) (d)

Figure 4.25: Example of the JBL coder for different base triangulations for `peppers` image. The (a) base triangulation and (b) its associated normal-mesh representation. Another (c) base triangulation and (d) its associated normal-mesh representation.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis, we have studied high-performance image coding systems from two distinct perspectives. In particular, we proposed an optimization-based method for the design of high-performance filter banks for image coding in the first part of the thesis, while in the second part, we proposed several modifications to a normal-mesh-based image coding system that lead to improved performance. The main contributions of the thesis are summarized in what follows.

As mentioned above, in the first part of our work, we proposed an optimization-based method for the design of filter banks. This scheme allows us to design filter banks having numerous desirable properties for image coding (i.e., PR, linear phase, high coding gain, good frequency selectivity, and certain prescribed moment properties). Furthermore, this scheme offers an effective tradeoff among some of the desirable properties. Two different autocorrelation function models for images were used for computing coding gain. We discussed which model leads to the best filter banks. Then the optimal choices of design parameters were discussed. Several examples of filter banks constructed using our proposed method were presented, and shown to be highly effective for image coding. In particular, our optimal designs outperform the well-known 9/7-J filter bank (from the JPEG-2000 standard) for both lossy and lossless compression. This is a very impressive result as the 9/7-J filter bank is well known for its exceptional lossy coding performance. Furthermore, several filter banks from different lifting configurations were designed using our proposed method. Experimental results showed that the designed filter banks in each lifting configuration outperform the other well-known filter banks from the same lifting configuration.

In the second part of our work, we proposed three modifications to the JBL coder and demonstrated through experimental results that each of these changes leads to improved coding performance. More specifically, we showed that good data-dependent base meshes help to locate horizons faster and preserve edges better. Also, we demonstrated that using integers instead of real numbers to represent normal/vertical offsets yields superior performance. Finally, we showed that, by exploiting horizon information, bicubic interpolation can be made to provide smoother reconstructed images while still maintaining sharp edges.

5.2 Future Research

Although this thesis has made some contributions to the subband-based and normal-mesh-based image coding systems, further work in this area could still be beneficial. Some future research areas are discussed in what follows.

As discussed earlier, the maximization of the separable, isotropic, and joint separable/isotropic coding gains is considered in our optimal filter bank design method. From our experimental results, we found that having the highest separable coding gain is less important than having the highest isotropic coding gain. Therefore, an objective function that combines the separable and isotropic coding gains, with more weight on the isotropic coding gain, might be a better choice for filter bank design. The optimal way to combine the two coding gains would be worthwhile to investigate. Another possible scheme for improving the filter-bank design method might be to develop some more sophisticated image model other than using the separable/isotropic models for the image autocorrelation function. Alternatively, an effective image classification method can be applied. Optimal filter banks are designed specifically for each image category, so that images belonging to this category can be coded very efficiently using the corresponding filter bank design. In so doing, images are likely to be coded more efficiently.

In our work on the JBL coder, we use one z coordinate to represent the image intensity at a nonhorizon point, and two z coordinates to represent the image intensity at a horizon point, where a jump in intensity is present. When horizon contours intersect, there can be more than two different intensity values associated with the intersection point. Therefore, to better represent the image surface in 3-D, we may need more than two z coordinates at such intersection points. Most likely, four z coordinates would be needed at an intersection of horizon contours.

In our current scheme, when a horizon edge is subdivided along its normal direction, the normal offset only indicates the x and y coordinates of the piercing point. The associated z coordinates of the piercing point are estimated based on the z coordinates of the two endpoints of the horizon edge. This estimation

method may not pose any significant problems when we focus only on lossy compression, especially when the intensity along the horizon is a relatively smooth function. The estimation of the z coordinates, however, makes lossless compression rather inefficient. Therefore, using some extra information to specify the differences between the exact and estimated z coordinates at the intersection of horizons would improve the coding efficiency.

Since edge distortion is generally quite noticeable, we prefer to construct polyline refinements of edges during subdivision in our normal-mesh-based scheme to reduce edge distortion. In order to realize iterative polyline refinement, it is necessary to locate a new horizon vertex when subdividing each horizon edge. Occasionally, locating all necessary horizon vertices is difficult due to possible mismatch between the forbidden zone restriction and the local geometry (i.e., curvature) of the horizon. The mismatch is most likely caused by the inexact digital edge curvature estimates, based on which some horizon vertices are inaccurately selected in the base mesh. Therefore, either some relaxation scheme when computing the forbidden zone or a better curvature-computation method may help to avoid the above polyline-refinement problem.

In many applications, it is useful for image coding systems to offer region of interest coding, spatial scalability, and quality scalability. Although we did not attempt to develop coders with the preceding functionalities in this thesis, the normal-mesh-based image coder does have the potential to support such functionalities. Since the normal-mesh-based coder yields polyline approximation of horizons, the resulting mesh can partition the whole image into several visually meaningful regions. Further refinement for each region can be performed independently. This feature is quite attractive for region of interest coding. By using more levels of subdivision, regions of interest can be coded with higher resolution than other parts of the image. In addition, by reordering the transformed data in the code stream, spatial scalability and quality scalability can be supported by the normal-mesh-based coder.

Normal-mesh-based image coders are most effective for images consisting of piecewise smooth regions. Many natural images contain some texture information, however. An efficient scheme for coding general images might be to separate the texture part and cartoon part (i.e., piecewise smooth part) of the image, and code each of these parts separately. The cartoon part can be efficiently coded using our enhanced normal-mesh-based image coder, while methods such as wavelets and curvelets may be effective for coding the texture part.

In the entropy coding of the transformed data of the normal-mesh-based methods, the quantizer step sizes of normal/vertical offset data vary from different levels of subdivision. The quantizer step sizes for offsets from adjacent levels of subdivision differ by a multiplicative factor of two in our coder. By optimizing this factor, we can possibly obtain better bit allocation and, therefore, achieve superior compression results.

In our normal-mesh-based work, we considered piecewise constant and piecewise planar interpolation when converting from a digital image to a surface. A better solution might be to use piecewise planar interpolation except at horizons. Along horizons, we still allow vertical jumps to exist. In so doing, the image surface is smooth in smooth regions, and jumps at horizons. This new surface model may fit better for real images, and consequently lead to better image-coding results.

In this thesis, we have focused exclusively on grayscale images. A thorough consideration of color images would also certainly be a worthwhile endeavor.

5.3 Closing Remarks

Both subband-based and normal-mesh-based image coding schemes are studied in this thesis. By exploiting the techniques presented herein, image coders with improved performance can be developed. Through our work, we have helped to advance the state of the art in both subband-based and normal-mesh-based image coding systems. As a result, the work presented in this thesis is useful to numerous image coding applications.

Bibliography

[1] M. D. Adams. ELEC 545 project: A wavelet-based lossy/lossless image compression system, April 1999.

[2] M. D. Adams. Triangulations and signal approximation. A Talk in Digital Signal Processing Group Meeting, University of Victoria, April 2007.

[3] M. D. Adams and R. K. Ward. Symmetric-extension-compatible reversible integer-to-integer wavelet transforms. *IEEE Transactions on Signal Processing*, 51(10):2624–2636, October 2003.

[4] A. Antoniou and W.-S. Lu. *Practical Optimization: Algorithms and Engineering Applications*. Springer, 1st edition, 2007.

[5] A. D. Booth. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240, 1951.

[6] C. M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Applied and Computational Harmonic Analysis*, 3:337–357, October 1996.

[7] A. M. Bruckstein, M. Elad, and R. Kimmel. Down-scaling for better transform compression. In *Scale-Space '01: Proceedings of the Third International Conference on Scale-Space and Morphology in Computer Vision*, pages 123–136, London, UK, 2001. Springer-Verlag.

[8] E. Candès and D. Donoho. *Curvelets: A Surprisingly Effective Nonadaptive Representation of Objects with Edges. In Curves and Surfaces*. Vanderbilt University Press, Nashville, TN, USA, 1999.

[9] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[10] Y. Chen, M. D. Adams, and W.-S. Lu. Design of optimal quincunx filter banks for image coding. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 2041–2044, May 2006.

[11] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45(5):485–560, 1992.

[12] E. A. B. da Silva. *Wavelet Transforms for Image Coding*. Ph.D. thesis, Department of Electronic Systems Engineering, University of Essex, UK, 1995.

[13] M. N. Do and M. Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12):2091–2106, December 2005.

[14] D. Donoho and X. Huo. Combined image representation using edgelets and wavelets. In *Wavelet Applications in Signal and Image Processing VII, in Proceedings SPIE*, volume 3813, pages 468–476, Denver, Colorado, October 1999.

[15] D. L. Donoho. Wedgelets: nearly minimax estimation of edges. *Annals Statistics*, 27(3):859–897, 1999.

[16] D. L. Donoho. Orthonormal ridgelets and linear singularities. *Journal on Mathematical Analysis of the Society for Industrial and Applied Mathematics*, 31(5):1062–1099, 2000.

[17] C. Dyken and M. S. Floater. Preferred directions for resolving the non-uniqueness of Delaunay triangulations. *Computational Geometry: Theory and Applications*, 34(2):96–101, 2006.

[18] A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[19] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder. Normal meshes. In *Computer Graphics Proceedings*, pages 95–102, 2000.

[20] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.

[21] ISO/IEC JTC 1/SC 29/WG 1 N 545. *JPEG-2000 test images*, July 1997.

[22] M. Jansen, R. Baraniuk, and S. Lavu. Multiscale approximation of piecewise smooth two-dimensional functions using normal triangulated meshes. *Applied and Computational Harmonic Analysis*, 19(1):92–130, 2005.

[23] J. Katto and Y. Yasuda. Performance evaluation of subband coding and optimization of its filter coefficients. In *Proc. of SPIE Visual Communications and Image Processing*, volume 1605, pages 95–106, November 1991.

[24] P. Lancaster and M. Tismenetsky. *The Theory of Matrices: with Applications*. Academic Press, San Diego, CA, USA, 2nd edition, 1985.

[25] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 248:193–228, November 1998.

[26] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[27] H. S. Malvar. *Signal Processing with Lapped Transforms*. Artech House, Norwood, MA, USA, 1992.

[28] M. Marji and P. Siy. A new algorithm for dominant points detection and polygonization of digital curves. *Pattern Recognition*, 36(10):2239–2251, October 2003.

[29] M. Nelson. *The Data Compression Book*. Henry Holt and Co., New York, NY, USA, 1991.

[30] E. L. Pennec and S. Mallat. Sparse geometric image representation with bandelets. *IEEE Transactions on Image Processing*, 14(4):423–438, April 2005.

[31] K. R. Rao and P. Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic Press Professional, San Diego, CA, USA, 1990.

[32] O. Rioul. A discrete-time multiresolution theory. *IEEE Transactions on Signal Processing*, 41(8):2591–2606, August 1993.

[33] A. Roorda. *Encyclopedia of Imaging Science and Technology*, volume 1. John Wiley and Sons, New York, 1st edition, 2002.

[34] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[35] A.-M. Sevcenco and W.-S. Lu. Combined adaptive and averaging strategies for JPEG-based low bit-rate image coding. In *20th Canadian Conference on Electrical and Computer Engineering*, pages 168–171, Vancover, Canada, April 2007.

[36] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[37] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[38] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, May 2002.

[39] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley College, 2nd edition, 1996.

[40] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

[41] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200, 1996.

[42] Y. Tsaig, M. Elad, P. Milanfar, and G. H. Golub. Variable projection for near-optimal filtering in low bit-rate block coders. *IEEE Transactions on Circuits and Systems for Video Technology*, 15:154–160, January 2005.

[43] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[44] M. H. Wright. Interior methods for constrained optimization. *Acta Numerica*, 1:341–407, 1992.

[45] S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[46] D. Xu and M. D. Adams. Design of high-performance filter banks for image coding. In *Proceedings of IEEE International Symposium on Signal Processing and Information Technology*, pages 868–873, Vancouver, BC, Canada, August 2006.

[47] D. Xu and M. D. Adams. An improved multiscale normal-mesh-based image coder. In *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 50–53, Victoria, BC, Canada, August 2007.

[48] T. Y. Yang. *Finite Element Structural Analysis*. Prentice Hall, Englewood Cliffs, 1986.