The Continuous Boundary Local Trigonometric Transform

Brons Larson

May 31, 2002

The Continuous Boundary Local Trigonometric Transform

BY

BRONS MICHAEL LARSON

B.A. (Revelle College - UCSD) 1991

M.S. (San Diego State University) 1993

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Applied Mathematics

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Committee in Charge

2002

Brons Michael Larson
June 2002
Applied Mathematics

The Continuous Boundary Local Trigonometric Transform

Abstract

The Local Trigonometric Transform (LTT) provides a nice tool for localizing both a signal and its frequency content. But there are certain properties of this algorithm that make it unattractive for various applications. For example, instability of the folding process on or near an edge can cause increased edge effect which translates recursively to all lower levels of the LTT. This can result in inefficient signal representation, improper segmentation and incorrect analysis of the signal.

In this thesis, some of these disadvantages of the LTT are examined, and a new approach to localized trigonometric analysis is proposed, the Continuous Boundary Local Trigonometric Transform (CBLTT), which attempts to correct these and other shortcomings. The main difference between the LTT and the CBLTT is that the former projects the signal onto smooth overlapping basis functions, whereas the latter decomposes the signal into regions that are assumed to be independent of one another. Each independent subspace then undergoes an invertible, nonlinear transformation to reduce edge effect, and it is immediately projected onto an orthogonal basis; e.g., Fourier basis, Sine/Cosine basis, Wavelet basis. As a tensor product, it can be applied to multi-dimensional data: signals, images, video. Since this new approach can be used to efficiently segment the frequency domain of a signal, it also gives rise to a new version of the Brushlet Transform.

In order to reduce edge effect and increase numerical stability, many variations of this new scheme are derived and presented, and their strengths and weaknesses are examined and contrasted. In addition, their usefulness in various applications, ranging from signal segmentation to compression, are also explored.

To Susan

Acknowledgments

I would like to start by thanking the U.C. Davis Mathematics faculty for providing such a warm and stimulating environment so conducive to learning. The many challenging courses that I took, and the many interesting discussions that I had with faculty members helped to lay a solid foundation upon which to base my career in Mathematical Research.

As for my thesis committee, it was a great pleasure to have worked with such distinguished scientists: Professor Naoki Saito, Professor Thomas Strohmer and Professor Bruno Olshausen. Their diverse scientific backgrounds and high calibre research skills were instrumental in my development of the CBLTT.

To Professor Thomas Strohmer, I offer warm thanks for his helpful advice in many areas of Digital Signal and Image Processing. His knowledge of the Gabor Transform assisted in my understanding of localized image analysis.

I offer special thanks to Professor Bruno Olshausen for his helpful conversations and many interesting seminars from Neuroscience. The insight that he provided into the human visual system aided me greatly in my research.

But most of all I would like to thank my advisor, Professor Naoki Saito, whose enthusiasm, expertise, and many interesting projects inspired my current research. It is because of his encouragement and guidance, that I have persevered and completed my Ph.D. For all of his support, I will be forever indebted to him[1].

[1]This research was partially funded by Dr. Saito's grants, NSF DMS-99-78321, and ONR YIP N00014-00-1-0469

Contents

List of Tables

List of Figures

Chapter 1

Introduction

It is a well known fact that the visual world is full of textures and structures which can be described as pseudo-periodic. From man-made objects such as shingle roofs, brick walls or striped clothing, to natural or biological objects such as trees, grass or blown sand, these pseudo-periodicities abound at many scales [9]. It is also well known from Neuroscience that the mammalian visual system filters visual information in a localized, patch-by-patch manner [17]. The main benefit of this type of decomposition is that the visual information is more efficiently represented. Because of this, much recent effort has been devoted to developing mathematical models and transform algorithms that mimic this type of behavior [18], since decreasing the representation cost of a dataset can result in improved analysis and compression. Some of the more successful approaches have been the Discrete Cosine Transform (DCT) used in JPEG (JPEG-DCT), the Wavelet Transform (WT), Wavelet Packets (WP), Local Trigonometric Transforms (LTT), and the Brushlet Transform (BT) ([14], [47], [53]).

The main advantage of these methods is that they provide a good tool for localizing both a signal and its frequency content. In addition, the Local Fourier Transform (LFT) and Brushlet Transform are equipped with a phase. But all of these approaches suffer from a crucial problem; that is, how to effectively treat the initial signal to provide boundary conditions which match those of bases associated with each particular transform. Without any information outside of the original interval, there lacks an effective way to both provide proper boundary conditions matching the associated transform bases, and also recover the original signal from the transformed coefficients in a stable manner. Furthermore, many of these methods operate by folding, or mixing, information between adjacent intervals. Instability of this folding process on or near an edge can increase edge effects which translate recursively to all lower levels of the transform. This can result in inefficient representation, improper segmentation and incorrect analysis of the signal.

It is with this in mind that a new approach to localized analysis is proposed, the Continuous Boundary Local Trigonometric Transform (CBLTT). Closely related to the Local Trigonometric Transform, it attempts to correct these and other shortcomings of the LTT. The main difference between the LTT and the CBLTT is that the former projects the signal onto smooth overlapping basis functions, whereas the latter decomposes the signal into disjoint regions. Each disjoint subspace then undergoes an invertible, nonlinear transformation to reduce edge effect, and it is immediately projected onto an orthonormal basis; e.g., Fourier basis, Sine/Cosine basis, Wavelet basis. As a tensor product, it can be applied to multi-dimensional data: signals, images, video. And since this new approach can be used to efficiently segment the frequency domain of a signal, it also gives rise to a new version of the Brushlet Transform.

In order to reduce edge effects and increase numerical stability, many variations of this new scheme are derived and presented, and their strengths and weaknesses are examined and contrasted. In addition, their usefulness in various applications, ranging from signal segmentation to compression, are also explored. It is shown that for compression, this approach can outperform some of the more popular approaches such as the LTT, LFT, BT and WP for certain signals. And for segmentation, the method shows much promise. The overall goal of this thesis is to show that the CBLTT, or some variation of it, is a viable alternative to signal analysis and compression.

The contents of this thesis are organized as follows. Chapter 2 starts with a brief introduction to localized analysis, with descriptions of the numerical methods mentioned above: JPEG-DCT, Wavelets, Wavelet Packets, the Local Trigonometric Transform and the Brushlet Transform. It contains a review of the basic concepts and definitions that will be used throughout the remaining chapters. Chapter 3 illustrates some of the shortcomings of the LTT and BT, while Chapter 4 introduces the Continuous Boundary Local Trigonometric Transform and all of its variations. Proofs for the existence of solutions, and control of the balance between spatial and frequency localization are also presented here. Chapter 5 shows some of the results from segmentation and compression. These ideas are then extended to higher dimensions in Chapter 6. Lastly, Chapters 7 and 8 offer some concluding remarks and describe further research directions. Also included are three appendixes which complement the thesis. The first of these, Appendix A, lists some of the abbreviations, along with their definitions, which are used throughout the thesis. The second, Appendix B, offers supplementary formulas, definitions and proofs. Lastly, Appendix C, contains image compression results which are analyzed in Chapter 6.

Chapter 2

Background Material

In this chapter, some of the more popular methods for localized analysis are described: the local trigonometric transform, the local Fourier transform, the brushlet transform, wavelets, wavelet packets and JPEG-DCT. Since the CBLTT is closely related to these schemes, their descriptions are fundamental in the development of the CBLTT. The concepts and definitions presented here will be used throughout the remainder of this thesis.

2.1 Discrete Trigonometric Transforms

This section starts by defining variations of the Discrete Cosine Transform (DCT) and the Discrete Sine Transform (DST), key components of the LTT, JPEG-DCT and CBLTT. Following this is a brief description of JPEG-DCT.

2.1.1 Discrete cosine/sine transforms

A key ingredient of the LTT transform is the Discrete Cosine Transform type-IV (DCT-IV) and the Discrete Sine Transform type-IV (DST-IV). These transforms are closely related to the standard DST-II and DCT-II (used in JPEG), in that they have fast algorithms associated with them, but they differ primarily in the symmetry of the transform basis functions at the boundaries of their intervals. To see this, consider the following. The 1D-DCT-II of a function x_n for $n = 0, \ldots, N$ is defined as [2]

$$C_k^{II} = \sqrt{\frac{2}{N+1}} c_k \sum_{n=0}^{N} x_n \cos \frac{(n+\frac{1}{2})k\pi}{N+1}, \quad \text{for } k = 0, \ldots, N, \tag{2.1}$$

where

$$c_k = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } k = 0, \\ 1, & \text{if } k = 1, \ldots, N, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Its inversion is given by

$$x_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N} c_k C_k^{II} \cos \frac{(n+\frac{1}{2})k\pi}{N+1}, \quad \text{for } n = 0, \ldots, N. \tag{2.3}$$

Similarly, the 1D-DST-II is defined as ([72], p.84)

$$S_k^{II} = \sqrt{\frac{2}{N+1}} c_{k+1} \sum_{n=0}^{N} x_n \sin \frac{(n+\frac{1}{2})(k+1)\pi}{N+1}, \quad \text{for } k = 0, \ldots, N, \tag{2.4}$$

and its inversion is

$$x_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N} c_{k+1} S_k^{II} \sin \frac{(n+\frac{1}{2})(k+1)\pi}{N+1}, \quad \text{for } n = 0, \ldots, N. \tag{2.5}$$

Now, replacing k with $k+\frac{1}{2}$ in the argument of the cosine on the right hand side of Equations (2.1) and (2.3) yields the formulas for DCT-IV. And replacing $k+1$ with $k+\frac{1}{2}$ in the argument of the sine on the right hand side of Equations (2.4) and (2.5) yields the formulas for DST-IV. They are

$$C_k^{IV} = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N} x_n \cos \frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1}, \quad \text{for } k = 0, \ldots, N, \tag{2.6}$$

which has inversion given by

$$x_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N} C_k^{IV} \cos \frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1}, \quad \text{for } n = 0, \ldots, N, \tag{2.7}$$

and

$$S_k^{IV} = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N} x_n \sin \frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1}, \quad \text{for } k = 0, \ldots, N, \tag{2.8}$$

which has the following inverse

$$x_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N} S_k^{IV} \sin \frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1}, \quad \text{for } n = 0, \ldots, N. \tag{2.9}$$

This simple modification completely changes the symmetry behavior of the basis functions at the boundaries of the range of n. For example, looking at the basis functions over the expanded range of $-N-1 \leq j \leq 2N+1$ shows that the DCT-II bases are *even* with respect to $-\frac{1}{2}$ and $N+\frac{1}{2}$ (Equation (2.3)), while the DST-II bases are *odd* (Equation (2.5)). On the other hand, the DCT-IV bases are even at the left side with respect to $-\frac{1}{2}$ and odd at the right side with respect to $N+\frac{1}{2}$ (Equation (2.7)). DST-IV has the exact opposite polarity as DCT-IV. Its bases are odd at the left hand side and even on the right (Equation (2.9)). Because of this property, basis functions which are even at the left boundary and odd at the right are said to have *cosine polarity*, whereas basis functions which are odd at the left boundary and even at the right are said to have *sine polarity*. Figure 2.1 illustrates these differences with sample basis functions from each transform. For the 2D versions of these transforms, see Appendix B.1.

2.1.2 JPEG-DCT

One of the most popular image compression schemes has been, and still is, the JPEG[1] standard which was a collaboration between the ISO[2] and the CCITT[3]. The details of JPEG – as well as its successor, JPEG2000, which is based on current wavelet technology – can be found at http://www.jpeg.org, although a brief summary of JPEG is given below.

[1]Joint Photographic Experts Group
[2]International Standardization Organization
[3]Consultative Committee of the International Telephone and Telegraph

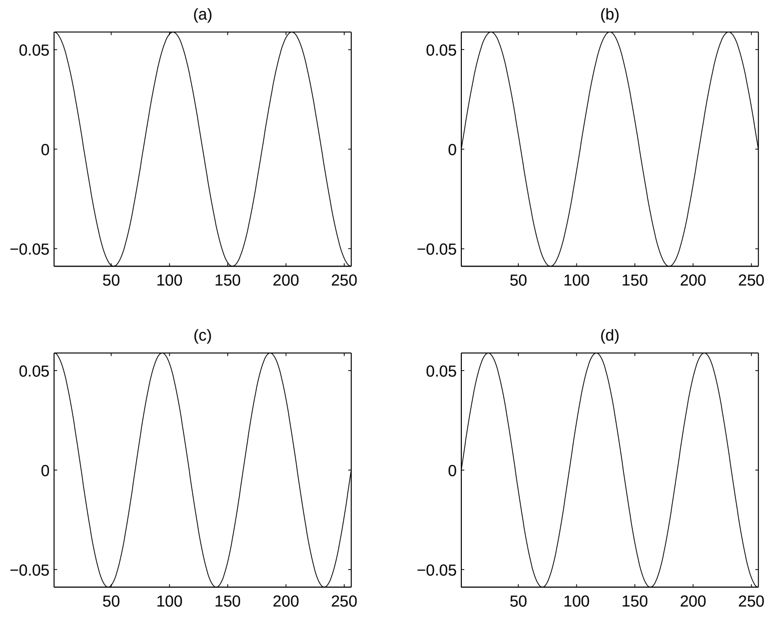

Figure 2.1: Sample basis functions of the (a) DCT-II, (b) DST-II, (c) DCT-IV, and (d) DST-IV illustrating the various boundary polarities.

JPEG consists of three different coding systems: (1) a lossy *baseline coding system* based on the DCT-II; (2) an *extended coding system* which provides greater compression, higher precision, or progressive reconstruction; and (3) a lossless *independent coding system*. Only the first system will be described here and will be used for comparison purposes in later chapters.

In this system, the compression is performed in three steps: DCT-II computation, quantization, and variable-length coding. Again, only the first of these is of particular interest here since all of the algorithms presented in this paper are based on *transform coding* and can also be quantized and variable-length coded in a similar manner to JPEG. In JPEG, the signal is first subdivided into small blocks of 8×8 pixels and processed from left to right and top to bottom. The 64 pixels within each subimage are then level shifted from p-bit unsigned integers in the range $[0, 2^p - 1]$ to $p-1$-bit signed integers in the range $[-2^{p-1}, 2^{p-1} - 1]$. Then the 2D DCT-II of each subimage is taken, quantized, and reordered in a zig-zag pattern. Since the DC components – the DCT-II coefficients with zero frequency – are a measure of the average value of image samples, there is usually a strong correlation between the DC coefficients of adjacent blocks. Hence, the resulting DC values are difference coded relative to the previous block's DC component; i.e., rather than storing the value of the DC component for a particular block, its difference from the DC component of the previous block is stored. This tends to be a crucial step since the DC coefficient frequently contains a significant portion of the total image energy. This is also the reason that the original image is level shifted prior to taking the DCT-II. Finally, the AC coefficients (non-DC components) are Huffman coded according to a predefined table.

One reason that JPEG is based on DCT-II is because DCT-II is a fast transform, $O(M \log_2(M))$ for a signal of length $M = 2^j$, where $j \in \mathbb{N}$. Another reason is that DCT-II provides very good compaction of highly correlated data. This excellent compression is due, in part, to the fact that DCT-II approximates the Karhunen-Loève transform of a first-order stationary Markov process (see [40], p.151-154, for more details). But it is also due to the even boundary polarity of its basis functions, as was seen in the previous section. This latter property causes the magnitude of the Fourier coefficients to decay at a rate similar to $O(\frac{1}{k^2})$ rather than $O(\frac{1}{k})$.

2.2 Local Trigonometric Transforms

In this section, the construction of the local trigonometric basis is reviewed using the notation of Wickerhauser [72] and the organization of Saito [61]. It should be noted that the local trigonometric basis is not actually a basis, but rather a term to describe any of the bases associated with the LCT, LST or the LFT, since their basis functions can be written in terms of sines or cosines. This section starts by defining the smooth orthogonal projection, which is the foundation upon which the local cosine/sine basis and the local Fourier basis are constructed. Following this is a description of the orthogonal periodization operator, key to the local Fourier transform. The section concludes by detailing the construction of the localized dictionary and the best-basis algorithm.

2.2.1 Smooth orthogonal projection

One of the primary objectives of localized analysis is the ability to segment a signal into localized regions and analyze of each of these regions using some basis. An immediate obstacle of this approach is the fact that the boundary conditions within each region do not always coincide with the boundary conditions of the basis of the particular analyzing operator. For example, suppose a given function $x(t) \in L^2(\mathbb{R})$ is to be split into pieces, each of which is supported on an interval $I_k = (\alpha_k, \alpha_{k+1})$, where $\bigcup_{k\in\mathbb{Z}} I_k = \mathbb{R}$, and the I_k's are disjoint. If each of these disjoint intervals is projected onto a new basis, such as $\{\mathbf{1}_{I_k} C_{m,k}\}_{m\in\{0\}\cup\mathbb{N}}$ where $C_{m,k}(t) \triangleq \frac{\sqrt{2}}{|I_k|} \cos\left[\frac{2m+1}{2|I_k|}\pi(t-\alpha_k)\right]$ which has cosine polarity boundary conditions, there is no guarantee the signal segments supported on each I_k will also have cosine polarity boundary conditions. Consequently, this boundary discrepancy will be perceived as a signal discontinuity causing a large number of transform coefficients to have significant energy. If quantization errors are introduced into these transformed coefficients, or if any coefficients are discarded for compression purposes, and signal reconstruction is performed, errors in the reconstructed signal will appear and will be enhanced at the locations of these false discontinuities; i.e., at the boundaries of the subspaces. This effect is known as the *edge effect* or *blocking effect.* This same problem can occur when using the LFT or Wavelets which register an artificial discontinuity at the subspace boundaries if the signals are not periodic within each interval. Thus, the following operator was devised to circumvent this problem, allowing analysis of a signal in a localized, patch by patch manner.

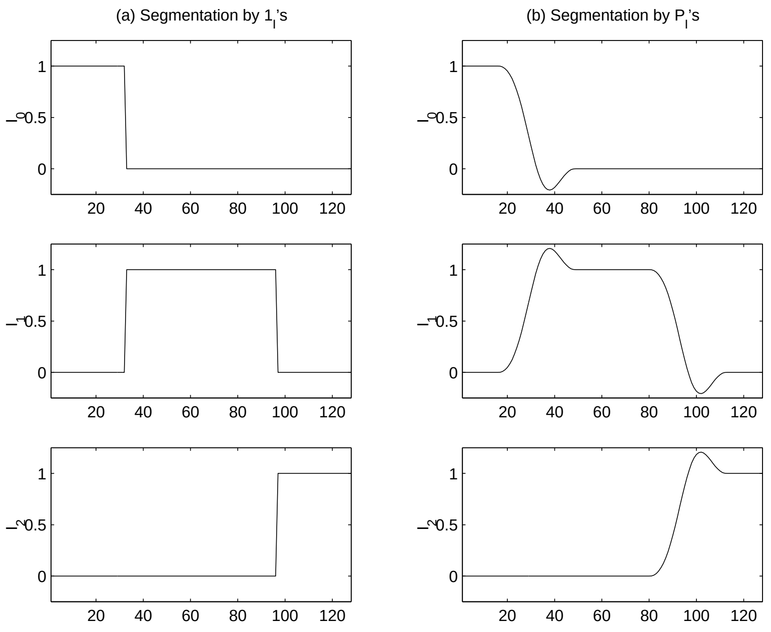

Figure 2.2: Splitting the constant function into three intervals, by the characteristic functions (a), and by the smooth orthogonal projections (b).

Suppose again that $x(t)$ is to be split into pieces, but this time each of these pieces is supported not only on an interval I_k, but also on each of its neighboring intervals, I_{k-1} and I_{k+1}, where $\bigcup_{k\in\mathbb{Z}} I_k = \mathbb{R}$, and the I_k's are disjoint. Define a sequence of subspaces Ω_k associated with the intervals I_k, and impose orthogonality among Ω_k, i.e., $L^2(\mathbb{R}) = \bigoplus_{k\in\mathbb{Z}} \Omega_k$, To achieve this goal, the following *smooth orthogonal projector* from $L^2(\mathbb{R})$ into $\Omega_k = P_{I_k}L^2(\mathbb{R})$ was introduced (see [3]):

$$P_{I_k}x(t) \stackrel{\Delta}{=} U^*_{I_k}\mathbf{1}_{I_k}U_{I_k}x(t). \tag{2.10}$$

This operator consists of three operators: U_{I_k} (*unitary folding operator*), $\mathbf{1}_{I_k}$ (restriction operator, i.e., $\mathbf{1}_{I_k}x(t) = x(t)$ if $t \in I_k$, $= 0$ otherwise), and $U^*_{I_k}$ (*unitary unfolding operator*, the adjoint of U_{I_k}). As will be shown next, P_{I_k} is a smoother version of sharp segmentation $\mathbf{1}_{I_k}$. Figure 2.2 shows the application of P_{I_k} to the constant function $x(t) \equiv 1$.

To better understand P_{I_k}, the unitary folding operator U_{I_k} needs to be defined for the interval I_k. Start by letting $I_k = (\alpha_k, \alpha_{k+1})$. Then

$$U_{I_k} x(t) \triangleq U(r_k, \alpha_k, \epsilon_k) U(r_{k+1}, \alpha_{k+1}, \epsilon_{k+1}) x(t), \tag{2.11}$$

where $U(r, \alpha, \epsilon)$ is a unitary folding operator associated with the region $t \in (\alpha - \epsilon, \alpha + \epsilon)$ and is defined as

$$U(r, \alpha, \epsilon)x(t) \triangleq \begin{cases} r\left(\frac{t-\alpha}{\epsilon}\right) x(t) + r\left(\frac{\alpha-t}{\epsilon}\right) x(2\alpha - t) & \text{if } \alpha < t < \alpha + \epsilon, \\ r\left(\frac{\alpha-t}{\epsilon}\right) x(t) - r\left(\frac{t-\alpha}{\epsilon}\right) x(2\alpha - t) & \text{if } \alpha - \epsilon < t < \alpha, \\ x(t) & \text{otherwise.} \end{cases} \tag{2.12}$$

In other words, U_{I_k} mixes, or folds, information from $(\alpha - \epsilon, \alpha)$ into $(\alpha, \alpha + \epsilon)$ and vice-versa. The region of the signal upon which U_{I_k} operates, or acts, is known as the *action region*. So the action region above is $t \in (\alpha - \epsilon, \alpha + \epsilon)$. In addition, the function $r(t)$ above is called a *rising cutoff function*, which is a smooth version (e.g., $r \in C^d(\mathbb{R})$ with $d \in \mathbb{N}$) of the Heaviside step function satisfying the following condition:

$$|r(t)|^2 + |r(-t)|^2 = 1 \quad \text{for all } t \in \mathbb{R}, \quad \text{and} \quad r(t) = \begin{cases} 0 & \text{if } t \leq -1, \\ 1 & \text{if } t \geq 1. \end{cases} \tag{2.13}$$

A typical example of a $C^m(\mathbb{R})$ rising cutoff function is the following iterated sine function:

$$r_{[m+1]}(t) \triangleq r_{[m]}(\sin \frac{\pi}{2} t) \tag{2.14}$$

where

$$r_{[0]}(t) = \begin{cases} 0, & \text{if } t \leq -1, \\ \sin[\frac{\pi}{4}(1+t)], & \text{if } |t| < 1, \\ 1 & \text{if } t \geq 1. \end{cases} \tag{2.15}$$

Many other possible choices of rising cutoff functions exist, each with varying properties (see [51]). Figure 2.3 shows a typical folding operator for an interval in matrix form. As can be seen, the folding operator $U(r, \alpha, \epsilon)$ makes a signal *locally even* for the region $(\alpha, \alpha + \epsilon)$ and *locally odd* for the region $(\alpha - \epsilon, \alpha)$. Because of this polarity, U is said to be a cosine polarity folding operator.

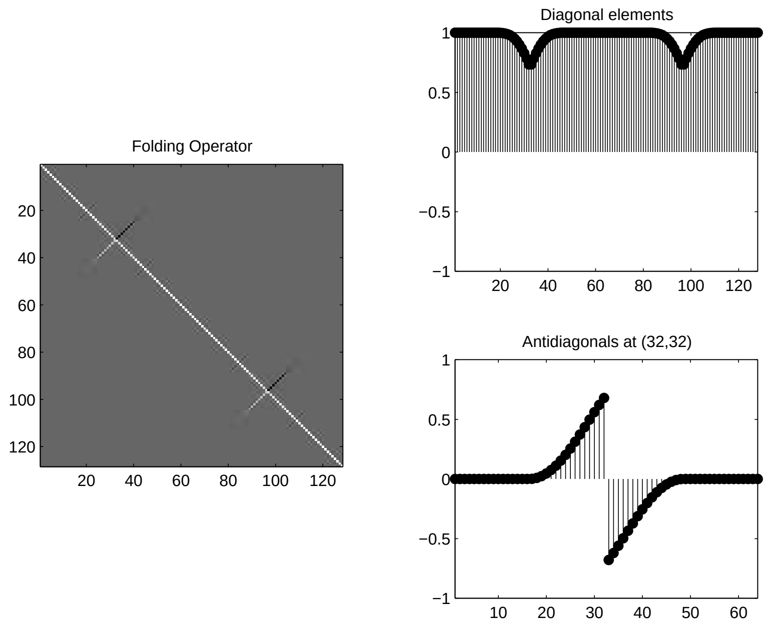

Figure 2.3: A typical folding operator U_I. In this case, $I = (31.5, 95.5)$, and $\epsilon = 16$ at both the left and right borders of the interval. The unfolding operator U_I^* is simply a transposition of U_I in this case.

The adjoint operator of $U(r, \alpha, \epsilon)$ is called the unitary unfolding operator:

$$U^*(r,\alpha,\epsilon)x(t) \triangleq \begin{cases} r\left(\frac{t-\alpha}{\epsilon}\right)x(t) - r\left(\frac{\alpha-t}{\epsilon}\right)x(2\alpha - t) & \text{if } \alpha < t < \alpha + \epsilon, \\ r\left(\frac{\alpha-t}{\epsilon}\right)x(t) + r\left(\frac{t-\alpha}{\epsilon}\right)x(2\alpha - t) & \text{if } \alpha - \epsilon < t < \alpha, \\ x(t) & \text{otherwise.} \end{cases} \tag{2.16}$$

and it makes a signal locally odd for the region $(\alpha, \alpha+\epsilon)$ and locally even for the region $(\alpha - \epsilon, \alpha)$. U^* is therefore known as a sine polarity operator.

One of the main benefits of this folding operation is now immediately realized. Applying U_{I_k} to a signal has the effect of forcing the boundary conditions to match those of a cosine polarity basis, such as $\{\mathbf{1}_{I_k} C_{m,k}\}_{m \in \{0\} \cup \mathbb{N}}$ defined earlier. While applying $U_{I_k}^*$ to a signal forces the boundaries to match those of a sine polarity basis. Thus, edge effect can be reduced if the signal is preprocessed with the appropriate folding operator. It should be noted that both $U = U(r, \alpha, \epsilon)$ and $U^* = U^*(r, \alpha, \epsilon)$ are unitary isomorphisms of $L^2(\mathbb{R})$ since $U^*Ux(t) = UU^*x(t) = x(t)$ for $t \neq \alpha$, and $Ux(t) = U^*x(t) = x(t)$ if $|t - \alpha| \geq \epsilon$.

For the operator U_{I_k} defined in (2.11), it is necessary that the two action regions around α_k and α_{k+1} of I_k do not interfere, i.e., $\alpha_k + \epsilon_k < \alpha_{k+1} - \epsilon_{k+1}$. The size of the action regions $2\epsilon_k$ around the boundary α_k, and the rising cutoff function $r_k(t)$, can be either dependent or independent of I_k. If ϵ_k is independent of I_k and is held constant for all k, then U and U^* are known as *fixed folding* and *fixed unfolding* operators respectively. If ϵ_k is dependent on the I_k, then U and U^* are known as *multiple folding* and *multiple unfolding* operators since multiple rising cutoff functions are used for different I_k. If $\epsilon_k = \frac{|I_k|}{2}$, then the terms *full folding* and *full unfolding* are used to describe U and U^*. Since the action regions are required to be disjoint of one another, then the choice of ϵ_k determines the smallest size of the interval partitions; that is, $|I_k| \geq \epsilon_{k+1}+\epsilon_k$. This is important because the choice of ϵ_k determines the resolution, or depth, of the LTT decomposition described later. Some clever choices of ϵ_k and r_k, dependent on I_k, have been devised which lead to better time-frequency localization schemes based on multiple folding [24] and the time-frequency local cosines [69].

With the folding operators well defined, there are some computational issues that must also be taken into consideration when discrete versions of these operators are used in practice. First, is the fact that the various rising cutoff functions with different ϵ_k should be precomputed once and stored for later use. Second, a decision must be made whether to fold at *gridpoint* locations or between gridpoints; i.e., at *midpoints*. If midpoint folding is employed, then the r_k consist of discrete points indexed from $-R \leq k \leq R-1$ where R is the radius of the rising cutoff function. In this case, cosine polarity folding of a function x_n for $n = 0, \ldots, R-1$ takes the form

$$\tilde{x}_n = r_n x_n + r_{-n-1} x_{-n-1}, \tag{2.17}$$

$$\tilde{x}_{N-n} = r_n x_{N-n} - r_{-n-1} x_{N+n+1} \tag{2.18}$$

where $\tilde{x}_n$ represents the folded signal.

It is straight forward to implement this version of folding. But if folding occurs at gridpoints instead, then an odd number of discrete values indexed from $-R \leq k \leq R$ for r_k are used. Some caution must be taken to ensure the existence of the unfolding operator, U^*. For example, if cosine folding is employed then the gridpoint folding operations become

$$\tilde{x}_n = r_n x_n + r_{-n} x_{-n}, \tag{2.19}$$

$$\tilde{x}_{N-n} = r_n x_{N-n} - r_{-n} x_{N+n}. \tag{2.20}$$

Now if the center rising cutoff value, r_0, is used to fold the right hand side of the interval, Equation (2.20), then $\tilde{x}_N = r_0 x_N - r_0 x_N = 0$. Inverting this yields $x_0 = r_0 \tilde{x}_0 + r_0 \tilde{x}_0 = 0$ which is incorrect; hence, inversion is undefined. The solution to this dilemma is to fold the right hand side at the x_{N+1} location instead of at x_N. With this convention, Equation (2.20) now becomes

$$\tilde{x}_{N-n} = r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2}, \quad \text{for } n = 0, \ldots, R-2, \tag{2.21}$$

This means that the middle value, $x_{\frac{N+1}{2}}$, will never be altered by the gridpoint folding operation, even if full folding is employed.

With the folding and unfolding operations well defined, they can now be used to construct the LTT. This is described next.

2.2.2 Local cosine/sine bases

For the local cosines, let $C_{m,k}(t) \triangleq \sqrt{\frac{2}{|I_k|}} \cos\left[\pi\left(m + \frac{1}{2}\right)\frac{t-\alpha_k}{|I_k|}\right]$. Then the set $\{\mathbf{1}_{I_k} C_{m,k}\}_{m \in \{0\} \cup \mathbb{N}}$ forms an orthonormal basis of $L^2(I_k)$. And from this, the *local cosine functions* (see [3]) are defined to be $\phi_{m,k}(t) \triangleq U^*_{I_k} \mathbf{1}_{I_k} C_{m,k}$. It can easily be shown that the set $\{\phi_{m,k}\}_{m \in \{0\} \cup \mathbb{N}}$ forms an orthonormal basis of $\Omega_k = P_{I_k} L^2(\mathbb{R})$ which implies that $\{\phi_{m,k}\}_{m \in \{0\} \cup \mathbb{N}, k \in \mathbb{Z}}$ forms an orthonormal basis of $L^2(\mathbb{R})$. Proving that $\{\phi_{m,k}\}_{m \in \{0\} \cup \mathbb{N}}$ is an orthonormal basis of Ω_k can be accomplished by using the following facts: 1) $\Omega_k = P_{I_k} L^2(\mathbb{R}) = U^*_{I_k} \mathbf{1}_{I_k} U_{I_k} L^2(\mathbb{R})$ is isomorphic to $U^*_{I_k} \mathbf{1}_{I_k} L^2(\mathbb{R}) = U^*_{I_k} L^2(I_k)$ because U_{I_k} is a unitary isomorphism on $L^2(\mathbb{R})$, and 2) $\{\mathbf{1}_{I_k} C_{m,k}\}_{m \in \{0\} \cup \mathbb{N}}$ is an orthonormal basis of $L^2(I_k)$.

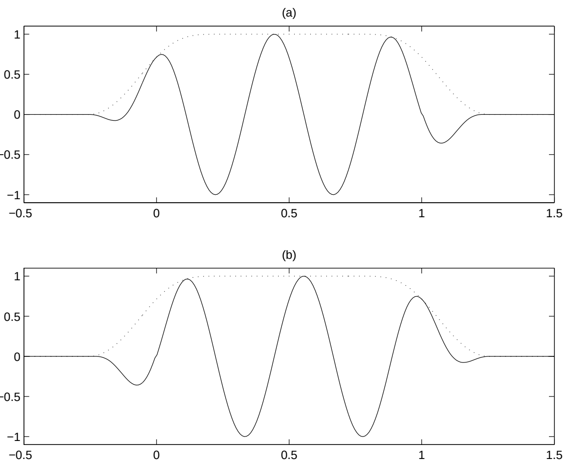

Figure 2.4: (a) A sample local cosine function along with its window, and (b) a sample local sine function along with its window.

Using these facts, analysis and synthesis are simple:

$$\begin{aligned} x(t) &= \sum_k \sum_m < x, U^*_{I_k} \mathbf{1}_{I_k} C_{m,k} > U^*_{I_k} \mathbf{1}_{I_k} C_{m,k}(t) \\ &= \sum_k \sum_m < U_{I_k} x, C_{m,k} > U^*_{I_k} \mathbf{1}_{I_k} C_{m,k}(t). \end{aligned} \tag{2.22}$$

In other words, the expansion coefficients can be computed as $< U_{I_k} x, C_{m,k} >$. These are simply the expansion coefficients of the smoothly folded cosine polarity function $U_{I_k} x(t)$ with respect to the orthonormal basis $\{C_{m,k}\}$. For the discrete case, DCT-IV is employed. For synthesis, applying U_{I_k} to both sides of Equation (2.22) yields

$$U_{I_k} x(t) = \sum_m < U_{I_k} x, C_{m,k} > \mathbf{1}_{I_k} C_{m,k}(t). \tag{2.23}$$

This is just synthesis using the coefficients $< U_{I_k} x, C_{m,k} >$ and the basis $\mathbf{1}_{I_k} C_{m,k}(t)$. For the discrete case, the inverse DCT-IV is simply applied to these coefficients, followed by applying $U^*_{I_k} \mathbf{1}_{I_k}$ to give $U^*_{I_k} \mathbf{1} U_{I_k} x(t) = P_{I_k} x(t)$. And since $L^2(\mathbb{R}) = \bigoplus_{k \in \mathbb{Z}} P_{I_k} L^2(\mathbb{R})$, then

$$x(t) = \sum_{k \in \mathbb{Z}} P_{I_k} x(t). \tag{2.24}$$

Replacing $C_{m,k}(t)$ with $S_{m,k}(t) \triangleq \sqrt{\frac{2}{|I_k|}} \sin\left[\pi\left(m+\frac{1}{2}\right)\frac{t-\alpha_k}{|I_k|}\right]$, yields the *local sines*, $\psi_{m,k}(t) \triangleq U_{I_k}\mathbf{1}_{I_k}S_{m,k}$. Figure 2.4 shows an example of the local cosine basis $\phi_4(t)$ and the local sine basis $\psi_4(t)$ when $I = [0,1]$ and $r = r_{[1]}$ with $\epsilon = 0.25$, along with their envelopes, or windows.

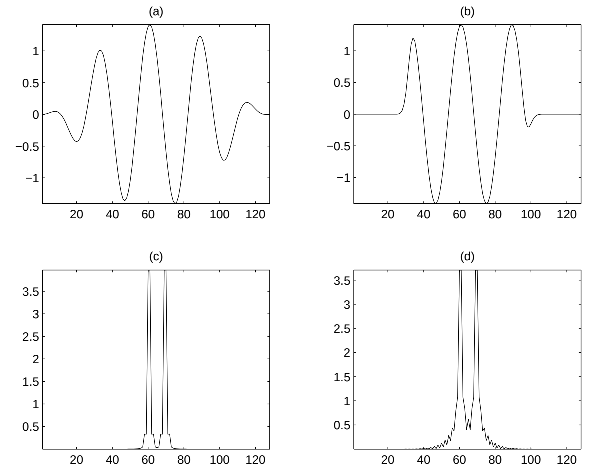

Figure 2.5: Time-frequency localization property of the local cosine function ϕ_4 using (a) $r = r_{[1]}$, and (b) $r = r_{[5]}$. The magnitude of their corresponding Fourier transforms (with DC component shifted to the center of each plot) using (c) $r = r_{[1]}$, and (d) $r = r_{[5]}$.

One of the immediate benefits of the local cosines and local sines is the fact that they are well-localized in both time and frequency. In time, their position uncertainty Δx is simply the width of the window's support; i.e., $\Delta x = ||[\alpha_k - \epsilon_k, \alpha_k + \epsilon_k]|| = \epsilon_{k+1} - \epsilon_k$. In frequency, they consists of two modulated bumps centered at $n + \frac{1}{2}$ and $-n + \frac{1}{2}$ with frequency uncertainty $\Delta\xi$ equal to $\hat{b}_k$, the Fourier transform of the window $b_k(t) \triangleq r_k\left(\frac{t-\alpha_k}{\epsilon_k}\right) r_{k+1}\left(\frac{t-\alpha_{k+1}}{\epsilon_{k+1}}\right)$. It should be noted that both these uncertainties are dependent upon the relative steepness of the window; that is, $\Delta x = \Delta x(b)$ and $\Delta\xi = \Delta\xi(b)$. These two values are inversely proportional to each other and are related by the following inequality known as the Heisenberg uncertainty product (see [72], p.122-124)

$$\Delta x(b) \cdot \Delta\xi(b) \geq \frac{1}{\sqrt{6}}. \tag{2.25}$$

Thus, as the window becomes steeper, the time uncertainty decreases but the frequency uncertainty increases and vice-versa. Figure 2.5 illustrates this using a local cosine function.

The relatively small value of the right hand side of Inequality (2.25) guarantees that the bases will remain reasonably localized in both time and frequency. Even so, there still exists the fact that the local cosines and local sines have two modulated bumps in frequency rather than one bump. This can be a detriment for certain applications such as signal analysis where a specific frequency often is requested. In addition, the local cosines and local sines lack a phase factor. Because of this, it is only natural to try to extend the above idea to build basis functions from windowed exponentials which have the majority of their energy centered around one frequency. In two dimensions, the phase of the exponential provides key information about the direction of the pattern when describing images. Unfortunately, an obstruction stands in the way. According to the Balian-Low Theorem (see [14], p.108) if $G = \{g(t-n)e^{2\pi imt} : n, m \in \mathbb{Z}\}$ is an orthonormal basis, then the Heisenberg product of g is infinite. Various approaches have been derived to circumvent this obstacle. One such idea constructs Wilson bases that use cosines and sines rather than exponentials [49]. In some respect, Wilson bases can be viewed as a special case of the LTT. But since they reduce blocking effect without the use of folding operators, they are actually closer to a local DCT. A comparison between Wilson bases and the CBLTT will be presented in a future paper.

Fortunately, through careful construction of the window, the Balian-Low obstruction can be avoided when using complex exponentials, allowing the use of smooth orthonormal basis functions which have the majority of their energy localized in the positive part of the frequency spectrum. This idea is described next.

2.2.3 Smooth orthogonal periodization

A modification to the smooth orthogonal projection P_{I_k} provides a method for smoothly restricting a function to an interval and periodizing it, thus permitting it to be expanded into a periodic basis with minimal edge effect; e.g., the local Fourier or Wavelet basis. This modification, the *smooth orthogonal periodization*, is defined as follows:

$$T_{I_k} x(t) \triangleq W^*_{I_k} \mathbf{1}_{I_k} U_{I_k} x(t). \tag{2.26}$$

In order to understand this operator, first define an *I-periodic extension* (or *I-periodization*) x_I of a function $x \in L^2(I)$ as

$$x_I(t) \triangleq \sum_{k \in \mathbb{Z}} x(t - k|I|). \tag{2.27}$$

Then, the operator T_{I_k} maps an $L^2_{loc}(\mathbb{R})$ function into an I_k-periodic extension of an $L^2(I_k)$ function. Moreover, T_{I_k} preserves the smoothness of a function: if $x \in C^d(\mathbb{R})$, then $T_{I_k} x$ has an I_k-periodic extension that also belongs to $C^d(\mathbb{R})$. The *periodized unfolding operator* $W^*_{I_k}$ in Equation (2.26) is defined as:

$$\begin{aligned} W^*_{I_k} x(t) &= W^*(r_k, I_k, \epsilon_k) x(t) \\ &= \begin{cases} r_k(\frac{t-\alpha}{\epsilon_k}) x(t) - r_k(\frac{\alpha_k - t}{\epsilon_k}) x(\alpha_k + \alpha_{k+1} - t) & \text{if } \alpha_k < t < \alpha_k + \epsilon_k, \\ r_k(\frac{\alpha_{k+1}-t}{\epsilon_k}) x(t) + r_k(\frac{t-\alpha_{k+1}}{\epsilon_k}) x(\alpha_k + \alpha_{k+1} - t) & \text{if } \alpha_{k+1} - \epsilon_k < t < \alpha_{k+1}, \\ x(t) & \text{otherwise.} \end{cases} \end{aligned} \tag{2.28}$$

This is the adjoint operator of the *periodized folding operator* W_{I_k}:

$$\begin{aligned} W_{I_k} x(t) &= W(r_k, I_k, \epsilon_k) x(t) \\ &= \begin{cases} r_k(\frac{t-\alpha}{\epsilon_k}) x(t) + r_k(\frac{\alpha_k - t}{\epsilon_k}) x(\alpha_k + \alpha_{k+1} - t) & \text{if } \alpha_k < t < \alpha_k + \epsilon_k, \\ r_k(\frac{\alpha_{k+1}-t}{\epsilon_k}) x(t) - r_k(\frac{t-\alpha_{k+1}}{\epsilon_k}) x(\alpha_k + \alpha_{k+1} - t) & \text{if } \alpha_{k+1} - \epsilon_k < t < \alpha_{k+1}, \\ x(t) & \text{otherwise.} \end{cases} \end{aligned} \tag{2.29}$$

It is easy to show that both W_I and W^*_I are isomorphisms of $L^2(\mathbb{R})$ as well as $L^2(I)$. It should be noted that the action regions and rising cutoff functions used in $W^*_{I_k}$ and U_{I_k} can be chosen differently. Figure 2.6 shows the periodized

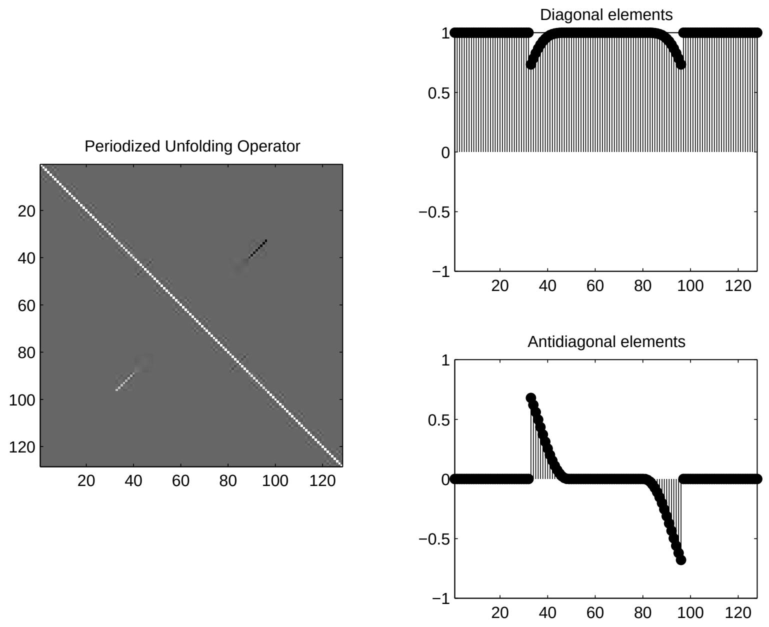

Figure 2.6: The periodized unfolding operator W_I^*. The size of the action region is the same as Figure 2.3. The periodized folding operator W_I is simply a transposition of W_I^* in this case.

unfolding operator W_I^* for the same interval as Figure 2.3. Figure 2.7 contrasts T_I with P_I.

To better clarify the difference between T_I and P_I, these operators are applied to a constant function and a linear function, and the results are shown in Figure 2.8. Mathematically, the exact relationship between T_I and P_I is summarized in the following equations:

$$T_I^* T_I = P_I, \quad T_I T_I^* = \mathbf{1}_I \tag{2.30}$$

which are easily derived from the definitions (2.10) and (2.26). For an I-periodic extension of a function supported on I, T_I simply restricts such extension to the interval I, i.e., does the role of $\mathbf{1}_I$, as shown in Figure 2.8. However, its adjoint $T_I^* = U_I^* \mathbf{1}_I W_I : L^2(I) \to U_I^* L^2(I) \simeq \Omega_k$, plays a key role for an I-periodic extension of a function as follows:

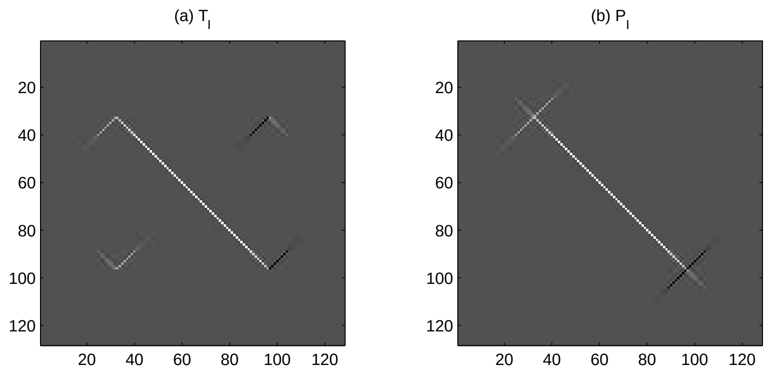

Figure 2.7: (a) The smooth orthogonal periodization operator T_I, and (b) the smooth orthogonal projector P_I.

Proposition 2.2.1. *Let $x_I(t)$ be an I-periodic extension of $x \in L^2(I)$. Then,*

$$T_I^* x_I(t) = P_I x_I(t).$$

Proof. Lemma 4.7 of Wickerhauser [72] states that if W_I and U_I share the same action regions and the rising cut-off function, then for any $x \in L^2(\mathbb{R})$ we have

$$W_I \mathbf{1}_I x = \mathbf{1}_I U_I (\mathbf{1}_I x)_I,$$

where $(\mathbf{1}_I x)_I$ is an I-periodic extension of the restriction $\mathbf{1}_I x$. Therefore, if we start with the I-periodized function x_I instead of x above, we immediately have $W_I x_I = U_I x_I$. Using this fact, it is easy to derive

$$T_I^* x_I = U_I^* \mathbf{1}_I W_I x_I = U_I^* \mathbf{1}_I U_I x_I = P_I x_I.$$

□

Now for a general function $x \in L^2(\mathbb{R})$, using Equation (2.30), the smooth orthogonal projection is realized in *two steps*:

$$P_I x = T_I^* T_I x. \tag{2.31}$$

In the first step, $T_I x$ makes $x(t)$ smoothly localized and I-periodic, which allows $T_I x$ to be expanded into a periodic basis. Then, in the second step, $T_I^*(T_I x)$ is a

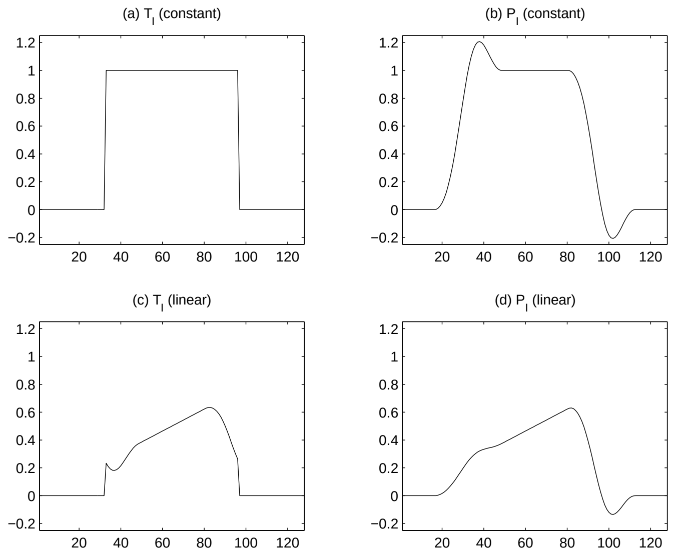

Figure 2.8: The operators in action. (a) T_I applied to the constant function 1. (b) P_I applied to the constant function. (c) T_I applied to a linear function. (d) P_I applied to the linear function. Note that if the original function is periodic, T_I does simple restriction.

mapping into $\Omega_I = P_I L^2(\mathbb{R})$. This is the primary difference from the construction of the local cosine functions, which is realized by just *one step*.

2.2.4 Local Fourier basis

It is now possible to construct smooth, localized, orthonormal bases using the smooth orthogonal periodization process of the previous subsection. Let $\mathbb{R} = \bigcup_{k\in\mathbb{Z}} I_k$ where all the I_k's have disjoint action regions. Let $\{e_{m,k}(t) : m \in \mathbb{Z}\}$ be a periodic orthonormal basis of $L^2(I_k) = \mathbf{1}_{I_k} L^2(\mathbb{R})$ with period $|I_k| = \alpha_{k+1} - \alpha_k$ (a typical example is the complex exponentials, $e_{m,k}(t) = (1/\sqrt{|I_k|})\mathrm{e}^{2\pi \mathrm{i} m(t-\alpha_k)/|I_k|}$). Then

Theorem 2.2.2 (Wickerhauser ([72], p.133-134)). *The set $\{T^*_{I_k} e_{m,k}\}_{m\in\mathbb{Z}}$ is an orthonormal basis of $\Omega_k = P_{I_k} L^2(\mathbb{R})$, and the set $\{T^*_{I_k} e_{m,k}\}_{(m,k)\in\mathbb{Z}^2}$ is an orthonormal basis of $L^2(\mathbb{R})$.*

From this theorem, it is easy to do both analysis and synthesis in a manner similar to that of the local cosines and local sines:

$$\begin{aligned} x(t) &= \sum_k \sum_m < x, T^*_{I_k} e_{m,k} > \; T^*_{I_k} e_{m,k}(t) \\ &= \sum_k \sum_m < T_{I_k} x, e_{m,k} > \; T^*_{I_k} e_{m,k}(t). \end{aligned} \tag{2.32}$$

That is, the expansion coefficients are found using inner products $< T_{I_k} x, e_{m,k} >$. These are simply the expansion coefficients of the smoothly periodized function $T_{I_k} x(t)$ with respect to the periodic orthonormal basis $\{e_{m,k}\}$. For the discrete version using the complex exponentials $e_{m,k}$, the FFT algorithm is employed. For synthesis, applying T_{I_k} on both sides of Equation (2.32) with Equation (2.30) leads to

$$T_{I_k} x(t) = \sum_m < T_{I_k} x, e_{m,k} > \; \mathbf{1}_{I_k} e_{m,k}(t). \tag{2.33}$$

This is just Fourier synthesis using the coefficients $< T_{I_k} x, e_{m,k} >$ and the basis $\mathbf{1}_{I_k} e_{m,k}(t)$. For the discrete setting, the inverse FFT is simply applied to these coefficients. This is followed by applying $T^*_{I_k}$ so that $T^*_{I_k} T_{I_k} x(t) = P_{I_k} x(t)$. Because $L^2(\mathbb{R}) = \bigoplus_{k \in \mathbb{Z}} P_{I_k} L^2(\mathbb{R})$, summing yields

$$x(t) = \sum_{k \in \mathbb{Z}} P_{I_k} x(t). \tag{2.34}$$

As was mentioned earlier, the above constructions allows for a smooth orthonormal basis which evades the Balian-Low obstruction. To illustrate this, let $I = [-1, 1]$, $\epsilon = 1$ and $r = r_{[m]}$. Apply P_I to a complex exponential to get

$$\begin{aligned} P_I e^{2\pi imt} &= \underbrace{r(1-t)^2 r(t+1)^2}_{b_+} e^{2\pi ikt} \\ &\quad + \underbrace{[r(t+1)r(-t-1)e^{-4\pi im} - r(t-1)r(1-t)e^{4\pi ik}]}_{b_-} e^{2\pi i(-k)t} \quad (2.35) \\ & \qquad (2.36) \end{aligned}$$

It can be shown that $||b_+||^2 \to 2$ and $||b_-||^2 \to 0$ as $m \to \infty$ (see [72] p.126, or Section 4.4.1 for the details).

Now, with the local Fourier functions well defined, it should be noted that as long as the action regions of the intervals $\{I_k\}$ do not interfere with each other, then each split $\{I_k\}_{k\in\mathbb{Z}}$ of the time axis $\mathbb{R}$ leads to an orthonormal basis. This naturally leads to the concept of a *dictionary of orthonormal* bases or a *time-frequency dictionary.*

2.2.5 Local trigonometric and local Fourier dictionaries

Recursively partitioning the time axis into a binary tree structured set of intervals, a notion of the *dictionary of orthonormal bases* or *time-frequency dictionary* is created (see [13], [72] p.126-130 and p.274-276). Using the same idea, it is easy to construct the *local trigonometric dictionary* and the *local Fourier dictionary.* Their construction is briefly described below.

Define $I_{0,k} = [k, k+1)$ so that $\mathbb{R} = \bigcup_{k\in\mathbb{Z}} I_{0,k}$. Then recursively split the intervals at their midpoints. After the jth recursion, each interval is of the form $I_{j,k} \triangleq [k/2^j, (k+1)/2^j)$, $k \in \mathbb{Z}$. Clearly, for each $j \in \mathbb{Z}$, $\mathbb{R} = \bigcup_{k\in\mathbb{Z}} I_{j,k}$ and $I_{j,k} = I_{j+1,2k} \cup I_{j+1,2k+1}$. For practical purposes, assume that all of the signals are of compact support and have been mapped to $I_{0,0} = [0,1)$; that is, all signals are in $L^2(I_{0,0})$. In practice, the recursion needs to be stopped at a certain level $J \in \mathbb{N}$. Then, the dyadic intervals $\{I_{j,k}\}$, $j = 0, 1, \ldots, J$, $k = 0, 1, \ldots, 2^j - 1$ can be readily arranged as a binary tree with the root node $I_{0,0}$. Associated with each interval $I_{j,k}$ is a subspace $\Omega_{j,k} = P_{I_{j,k}} L^2(I_{0,0})$ with $\Omega_{0,0} = L^2(I_{0,0})$. This set of tree-structured subspaces (with the localized orthonormal basis functions at each subspace) is called the *local trigonometric dictionary* or the *local Fourier dictionary* depending on the basis expansion employed. This dictionary contains a huge number (more than $2^{2^{J-1}}$) of orthonormal bases since each cover of $I_{0,0}$ by a subset of $\{I_{j,k}\}$ corresponds to one orthonormal basis.

For discrete and finite dimensional versions of this dictionary, a set of discrete signals is sampled on the regular grid in the interval $I_{0,0}$ with n time samples. Then, this version of the dictionary consists of a redundant number (e.g., $n \log n$) of basis vectors with specific characteristics in scale, position, and frequency. These basis vectors are organized as a binary tree in a *hierarchical* manner ranging from very localized spikes to global oscillations on $I_{0,0}$ with different frequencies. Therefore, pattern analysis and interpretation tasks using this dictionary become more intuitive than using the standard basis or the DCT/DST or discrete Fourier basis on the interval $I_{0,0}$. Decomposing or reconstructing a signal using this dictionary is fast, e.g., $O(n[\log n]^2)$ for LCT, LST and LFT, thanks to the efficiency of the FFT algorithm.

In order to select a good basis out of so many possible bases, a numerical criterion is needed to evaluate its effectiveness for a specific purpose. Once this is defined, a bottom-up procedure is used to efficiently search for a good basis by optimizing this criterion. This divide-and-conquer (or split-and-merge) algorithm is called the *best-basis* algorithm. Therefore, this dictionary provides a flexible, hierarchical, and computationally efficient set of feature extractors.

It should be noted that the most straightforward extension of the above construction to higher dimensions can be easily achieved by the appropriate tensor products, which is described in Chapter 6. The local Fourier dictionary for images is particularly attractive because it contains the basis vectors with *oblique* oscillations, which the usual wavelet packets and local cosine/sine dictionaries do not have. In addition, as was seen in the previous section, the local Fourier bases can have the majority of their energy localized around one frequency. To achieve better frequency localization properties, the local Fourier transform can be used to segment the frequency domain instead of the space domain. If this is the case, then the dual of the LFT arises and is know as the *brushlet* transform (see [53]). The brushlet bases are efficient for capturing oriented textured patterns, and its details are discussed next using the description of Meyer and Coifman [53].

2.3 Brushlets

The *Brushlet dictionary* contains a large number of reasonably localized complex exponential functions which contain only one peak in frequency. The decomposition is accomplished by applying the *local Fourier transform* directly to the Fourier domain. Representations of a function using this dictionary are similar to wavelet decompositions, with the added benefit of a phase. Since brushlets are derived from successive applications of the Fourier transform and the local Fourier transform, the phase of the two-dimensional brushlet is similar to the phase of the two-dimensional Fourier transform, providing key information about the orientation of the corresponding brushlet. This orientation capability, coupled with the hierarchical decomposition of the brushlet transform, means that brushlets are well suited for textured analysis by providing very precise representation of an image using a dictionary containing bases with all possible directions, scales, frequencies and locations. In this section, we review the construction of the standard one and two dimensional brushlet dictionary as described by Meyer and Coifman [53], and compare and contrast alternate versions of the algorithm.

2.3.1 One-dimensional brushlets

As mentioned in the previous paragraph, the standard brushlet dictionary is obtained by simply applying the local Fourier transform to the frequency domain. That is, if $x(t) \in L^2(\mathbb{R})$, $\hat{x}$ is the Fourier transform of x, and $u_{m,k} = T^*_{I_k} e_{m,k}(\xi)$ are the local Fourier bases functions, then the brushlet coefficients $\hat{x}_{m,k}$ are given by

$$\hat{x}(\xi) = \sum_{m,k \in \mathbb{Z}^2} \hat{x}_{m,k} u_{m,k}(\xi). \tag{2.37}$$

Now since the Fourier transform is a unitary isomorphism of $L^2(\mathbb{R})$, a new orthonormal basis $\{w_{m,k}\}_{(m,k)\in\mathbb{Z}^2}$ of this space can be derived by taking the inverse

Fourier transform of the above equation:

$$x(t) = \sum_{m,k \in \mathbb{Z}^2} \hat{x}_{m,k} w_{m,k}(t). \tag{2.38}$$

Thus the brushlet basis, $\{w_{m,k}\}_{(m,k)\in\mathbb{Z}^2}$, is defined to be the inverse Fourier transform of $\{u_{m,k}\}_{(m,k)\in\mathbb{Z}^2}$; i.e., $w_{m,k}(t) = \check{u}_{m,k}(t)$.

To better understand the structure of these brushlet bases, the following construction of $\{u_{m,k}\}_{(m,k)\in\mathbb{Z}^2}$ is useful. Although different rising cutoff functions can be used at the end of each interval, for simplicity assume that the bump functions and windowing functions are symmetric about the origin; that is, let v be the bump function supported on $[-\epsilon, \epsilon]$

$$v(\xi) = r\left(\frac{\xi}{\epsilon}\right) r\left(-\frac{\xi}{\epsilon}\right) \tag{2.39}$$

and let b_n be the windowing function supported on $[-|I_k|/2-\epsilon, |I_k|/2+\epsilon]$

$$b_k(\xi) = \begin{cases} r^2(\frac{\xi-|I_k|/2}{\epsilon}) & \text{if } \xi \in [-|I_k|/2-\epsilon, -|I_k|/2+\epsilon], \\ 1 & \text{if } \xi \in [-|I_k|/2+\epsilon, |I_k|/2-\epsilon], \\ r^2(\frac{|I_k|/2-\xi}{\epsilon}) & \text{if } \xi \in [|I_k|/2-\epsilon, |I_k|/2+\epsilon]. \end{cases} \tag{2.40}$$

It can then be shown that each $u_{m,k}$ supported in $[\alpha_k-\epsilon, \alpha_{k+1}+\epsilon]$ can be expressed as

$$\begin{aligned} u_{m,k}(\xi) &= b_k(\xi - c_k)e_{m,k}(\xi) \\ &\quad + v(\xi-\alpha_k)e_{m,k}(2\alpha_k-\xi) - v(\xi-\alpha_{k+1})e_{m,k}(2\alpha_{k+1}-\xi) \end{aligned} \tag{2.41}$$

where c_k is the midpoint of the interval I_k. Taking the inverse Fourier transform of this yields

$$w_{m,k}(t) = \frac{1}{\sqrt{|I_k|}} e^{2i\pi c_k t}\left\{(-1)^m \check{b}_k\left(t - \frac{m}{|I_k|}\right) - 2i\sin(\pi|I_k|t)\check{v}\left(t + \frac{m}{|I_k|}\right)\right\}. \tag{2.42}$$

Now as the support of v decreases, the tails of $b_k(\xi)$ become shorter and steeper; i.e., $\lim_{\epsilon\to 0} b_k(\xi) = \chi_{I_k}$. Introducing notation illustrating the window's dependence on ϵ yields

$$v_\sigma(\xi) = v(|I_k|\xi) \tag{2.43}$$

and

$$b_\sigma(\xi) = b_k(|I_k|\xi) \tag{2.44}$$

where $\sigma = \epsilon/|I_k|$. With this notation, Equation (2.42) becomes

$$\begin{aligned} w_{m,k}(t) &= \sqrt{|I_k|}e^{2i\pi\alpha_k t}e^{i\pi|I_k|t} \\ &\cdot \left\{(-1)^m \check{b}_\sigma(|I_k|t - m) - 2i\sin(\pi|I_k|t)\check{v}_\sigma(|I_k|t + m)\right\}. \end{aligned} \tag{2.45}$$

From this expression it is evident that $w_{m,k}$ is very similar to a wavelet, with I_k and m acting as the scaling factor and translation index respectively. The major difference, though, is that the $w_{m,k}$ are complex valued functions with a phase. More precisely, the brushlet is composed of two terms, localized about m/I_k and $-m/I_k$, each oscillating at the frequency c_k. But since $|\check{v}_\sigma(t))| = |\check{v}(t/I_k)/|I_k|| \leq \sigma$, then the second term in Equation (2.45) can be made arbitrarily small. However, as σ tends to zero, the first term no longer remains localized (the curse of the Balian-Low Theorem (see [72], p.122-126)).

2.3.2 Two-dimensional brushlets

The bi-dimensional brushlet transform is a powerful tool for analysis because its phase provides valuable information about the orientation of the brushlet. Constructing this orthonormal windowed basis of $L^2(\mathbb{R}^2)$ is a very straightforward task. Simply define two partitions of $\mathbb{R}$: $\cup_{k=-\infty}^{k=\infty} I_k$ and $\cup_{l=-\infty}^{l=\infty} J_l$ where $I_k = [\alpha_k, \alpha_{k+1}[$ and $J_l = [\beta_l, \beta_{l+1}[$. If these are used to create rectangular partitions of $\mathbb{R}^2$, $I_k \otimes J_l$, then the separable tensor products of $w_{m,k}$ and $w_{n,l}$ form an orthonormal basis of $L^2(\mathbb{R}^2)$:

$$\begin{aligned} w_{m,k}(t_1) \otimes w_{n,l}(t_2) &= \sqrt{|I_k||J_l|}e^{2i\pi(c_k t_1 + d_l t_2)} \\ &\cdot \left\{(-1)^m \check{b}_\sigma(I_k t_1 - m) - 2i\sin(\pi I_k t_1)\check{v}_\sigma(I_k t_1 + m)\right\} \\ &\cdot \left\{(-1)^n \check{b}_\sigma(J_l t_2 - n) - 2i\sin(\pi J_l t_2)\check{v}_\sigma(J_l t_2 + n)\right\} \end{aligned} \tag{2.46}$$

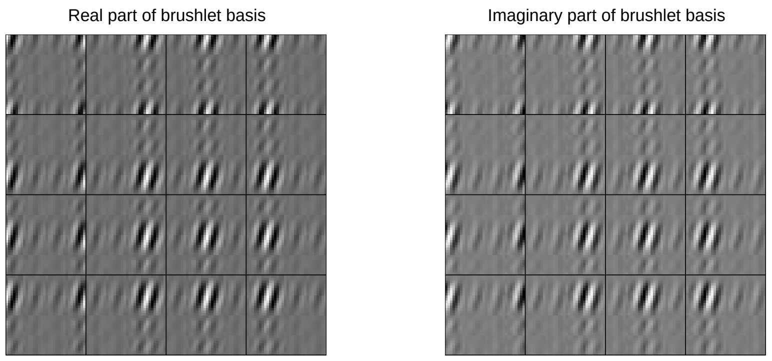

Figure 2.9: Some standard brushlet basis functions created with a sharp rising cutoff function ($\epsilon = 0$). Although there is only one bump in frequency, the sharp window produces a ringing effect which causes the bases to have global support. Notice, though, how the brushlets behave like oriented wavelets; i.e., they are shifted versions of the same pattern.

This tensor product is an oriented pattern oscillating with the frequency (c_k, d_l) at two spatial locations: $(m/|I_k|, n/|J_l|)$ and $(-m/|I_k|, -n/|J_l|)$. Here c_k is the center of I_k, and d_l is the center of J_l. But like the one-dimensional brushlet, the steepness of the rising cutoff functions can be chosen in order to localize the two-dimensional brushlet around one spatial location, $(m/|I_k|, n/|J_l|)$ as can be seen in Figure 2.9. It should be noted that biorthogonal brushlets, which allow better time-frequency localization, are also possible using a different folding procedure (see [51]).

2.3.3 The brushlet dictionary

The brushlet dictionary is constructed in the same manner described in Section 2.2.5. Using the best basis algorithm, the Fourier space, with DC component shifted to the center, is recursively split into dyadic subspaces, and the brushlet coefficients are computed for each subspace. Based on this set of coefficients, a cost is assigned to each node of the resulting tree, and the child nodes of the tree are pruned if their cost is greater than the cost of their immediate parent. This process is repeated, starting at the bottom of the tree and working up to the top level, resulting in a globally optimal tree.

Performing a level one brushlet decomposition of the surf image, Figure 2.10, results in the coefficient table shown in Figure 2.11. For illustrative purposes, only the imaginary part is shown. From this table, the orientation analysis capabilities of the brushlet are clearly exhibited. As can be seen, the upper left quadrant of the Fourier plane contains textures whose structure is oriented along the $\frac{\pi}{4}$ direction: the surfers going right on the wave, the upper rightmost surfer behind the wave. In the upper right window, textures oriented along the $\frac{3\pi}{4}$ direction are present: the surfers going left, the bottom surfer paddling up the wave.

2.3.4 Real-valued brushlets

For certain applications, such as compression, it is beneficial to decompose real-valued data sets into purely real basis functions. For example, if a real-valued signal is decomposed using the standard complex brushlet transform, at least *two* complex transform coefficients will be needed to represent the signal. If a real-valued brushlet transform can be used instead, it is possible to represent the same signal using only *one* real-valued transform coefficient. It is therefore important to have real versions of the brushlet transform. There are many ways to accomplish this, two of which are described below.

The first involves simply using a real-valued Fourier transform (FFTR) in place of the complex Fourier transform at each step of the brushlet decomposition. That is, first apply FFTR to the global signal and then use a real-valued local Fourier transform (LFTR) to segment the frequency domain (let this method be denoted by FFTR$\rightarrow$LFTR). For inversion, the transformed coefficients are first inverted via the inverse LFTR to get the global frequency content, and then inverse FFTR is performed to retrieve the spatial coefficients. The only consideration is the ordering of the coefficients in the frequency space. The

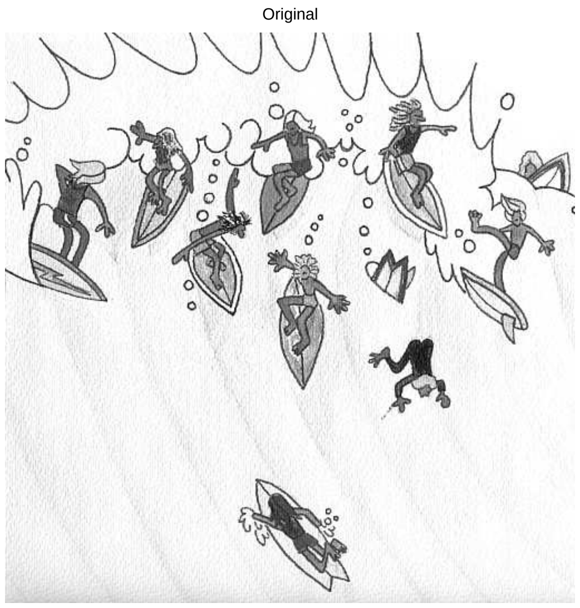

Figure 2.10: Original surf image.

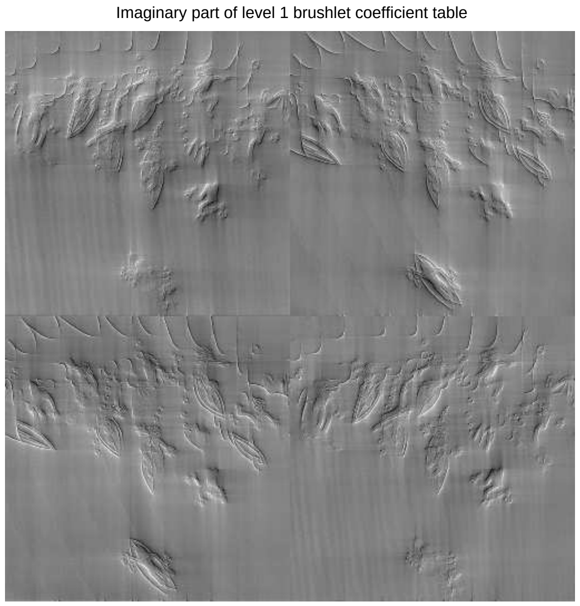

Figure 2.11: Imaginary part of the level 1 brushlet coefficient table. The anti-symmetry comes from the fact that the input image is purely real. The upper left and bottom right quadrants contain textures oriented along the $\frac{\pi}{4}$ direction. The upper right and lower left quadrants have textures oriented along the $\frac{3\pi}{4}$ direction.

following conversion between the complex Fourier transform and the real Fourier transform makes for a natural ordering of the Fourier coefficients.

In one dimension, start with the unitary discrete Fourier transform of a signal x of length M:

$$\begin{aligned}
\sqrt{M}x_k &= \sum_{m=0}^{M-1} \hat{x}_m e^{2\pi i \frac{mk}{M}} \\
&= \sum_{m=0}^{M-1} \hat{x}_m \left[\cos\left(2\pi\frac{mk}{M}\right) + i\sin\left(2\pi\frac{mk}{M}\right)\right] \\
&= \hat{x}_0 + \sum_{m=1}^{\frac{M}{2}-1} \hat{x}_m \left[\cos\left(2\pi\frac{mk}{M}\right) + i\sin\left(2\pi\frac{mk}{M}\right)\right] \\
&\quad + (-1)^k \hat{x}_{\frac{M}{2}} + \sum_{m=\frac{M}{2}+1}^{M-1} \hat{x}_m \left[\cos\left(2\pi\frac{mk}{M}\right) + i\sin\left(2\pi\frac{mk}{M}\right)\right] \\
&= \hat{x}_0 + \sum_{m=1}^{\frac{M}{2}-1} (\hat{x}_m + \hat{x}_{M-m}) \cos\left(2\pi\frac{mk}{M}\right) \\
&\quad + (-1)^k \hat{x}_{\frac{M}{2}} + i \sum_{m=\frac{M}{2}+1}^{M-1} (\hat{x}_m - \hat{x}_{M-m}) \sin\left(2\pi\frac{mk}{M}\right) \\
&= \sum_{m=0}^{\frac{M}{2}} \sqrt{2}\mathrm{Re}(\hat{x}_m)\cdot\sqrt{2}\cos\left(2\pi\frac{mk}{M}\right) - \sum_{m=\frac{M}{2}+1}^{M-1} \sqrt{2}\mathrm{Im}(\hat{x}_m)\cdot\sqrt{2}\sin\left(2\pi\frac{mk}{M}\right) (*)
\end{aligned}$$

It should be noted that in $(*)$, every $\sqrt{2}$ should be replaced by 1 when $m = 0$ and $m = M/2$. With this form, the coefficients can be easily organized as follows:

$$\left\{\hat{x}_0, \sqrt{2}\mathrm{Re}(\hat{x}_1), \ldots, \sqrt{2}\mathrm{Re}(\hat{x}_{\frac{M}{2}-1}), \hat{x}_{\frac{M}{2}}, -\sqrt{2}\mathrm{Im}(\hat{x}_{\frac{M}{2}+1}), \ldots, -\sqrt{2}\mathrm{Im}(\hat{x}_{M-1})\right\}. \quad (2.47)$$

The corresponding basis functions are

$$\sqrt{\frac{1}{M}}, \left\{\sqrt{\frac{2}{M}}\cos\left(2\pi\frac{(1)k}{M}\right)\right\}_{k=0}^{M-1}, \ldots, \left\{\sqrt{\frac{2}{M}}\cos\left(2\pi\frac{(\frac{M}{2}-1)k}{M}\right)\right\}_{k=0}^{M-1}, \frac{(-1)^k}{\sqrt{M}},$$
$$\left\{\sqrt{\frac{2}{M}}\sin\left(2\pi\frac{(\frac{M}{2}+1)k}{M}\right)\right\}_{k=0}^{M-1}, \ldots, \left\{\sqrt{\frac{2}{M}}\sin\left(2\pi\frac{(M-1)k}{M}\right)\right\}_{k=0}^{M-1} \quad (2.48)$$

In two dimensions, the same procedure can be repeated for a $M \times N$ matrix x. The details are provided in Appendix B.2, but the results are as follows:

$$\begin{aligned}
\sqrt{MN}x_{k,l} &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{x}_{m,n} e^{2\pi i\left(\frac{mk}{M}+\frac{nl}{N}\right)} \\
&= \sum_{m=0}^{\frac{M}{2}}\sum_{n=0}^{\frac{N}{2}} \underbrace{\sqrt{2}\mathrm{Re}(\hat{x}_{m,n})}_{\dagger} \cdot \overbrace{\sqrt{2}\cos\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger} (\star) \\
&\quad + \sum_{m=1}^{\frac{M}{2}-1}\sum_{n=\frac{N}{2}+1}^{N-1} \underbrace{\sqrt{2}\mathrm{Re}(\hat{x}_{m,n})}_{\dagger} \cdot \overbrace{\sqrt{2}\cos\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger} \\
&\quad - \sum_{m=\frac{M}{2}+1}^{M-1}\sum_{n=0}^{\frac{N}{2}} \underbrace{\sqrt{2}\mathrm{Im}(\hat{x}_{m,n})}_{\dagger} \cdot \overbrace{\sqrt{2}\sin\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger} \\
&\quad - \sum_{m=\frac{M}{2}}^{M}\sum_{n=\frac{N}{2}+1}^{N-1} \underbrace{\sqrt{2}\mathrm{Im}(\hat{x}_{m,n})}_{\dagger} \cdot \overbrace{\sqrt{2}\sin\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger} (\star\star)
\end{aligned}$$

As in the one dimensional case, every $\sqrt{2}$ in $(\star)$ should be replaced by 1 when (m,n) equals $(0,0)$, $(\frac{M}{2},0)$, $(0,\frac{N}{2})$, or $(\frac{M}{2},\frac{N}{2})$. Also, when $m = M$ in $(\star\star)$, $m = 0$ should be used instead. With this indexing, the coefficients (†) associated with the basis functions (‡) can easily be organized in a manner similar to the standard complex FFT coefficients. This ordering is illustrated in Figure 2.12 with the DC component located in the upper left corner and the Nyquist frequency located in the center.

Although it is not necessary, it may be desirable to shift the DC component to the center of the Fourier coefficient table so that it is surrounded by circular frequency bands of increasing frequency as they move away from the DC component. This ordering is shown in Figure 2.13. With this arrangement, the

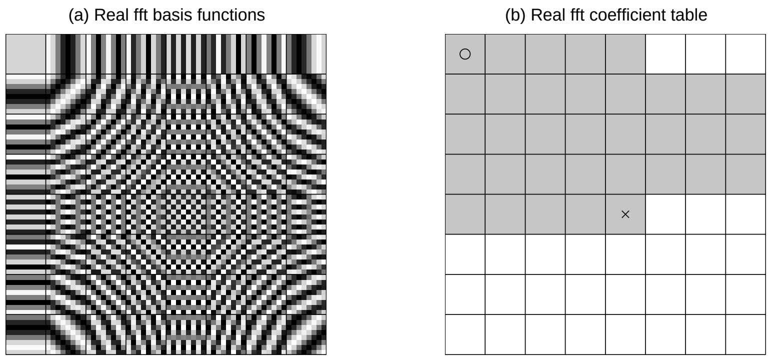

Figure 2.12: (a) Basis functions of the real-valued FFT transform. The DC component is located in the upper left corner (marked with a ○ in (b)), and the Nyquist component is in the center (marked with a × in (b)). (b) The shaded region corresponds to the coefficients associated with the cosine basis, and the non shaded region corresponds to the location of the coefficients of the sine basis.

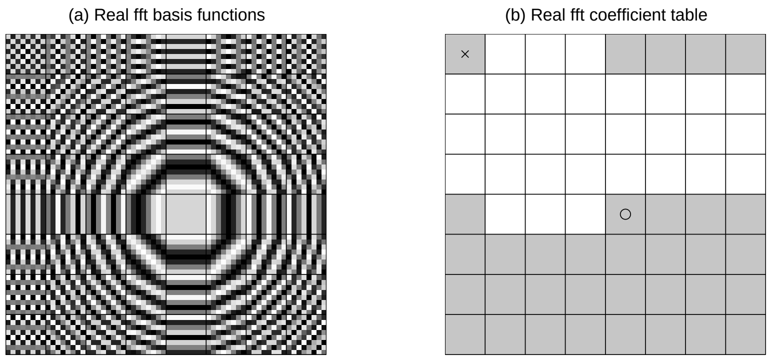

Figure 2.13: (a) Basis functions of the real-valued FFT transform. The DC component has been shifted to the center (marked with a ○ in (b)) and the Nyquist component has been shifted to the upper left corner (marked with a × in (b)). (b) The shaded region corresponds to the coefficients associated with the cosine basis, and the non shaded region corresponds to the location of the coefficients of the sine basis. Notice the circular bands of increasing frequency as the distance from the DC component increases.

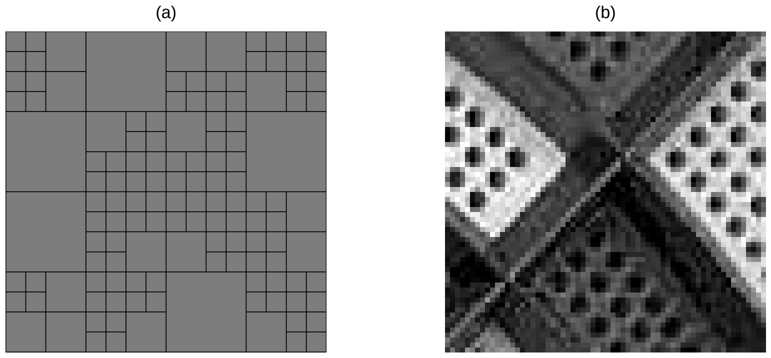

Figure 2.14: (a) Partition patterns in the 2D Fourier domain (with the DC component shifted to the center) associated with the real-valued brushlet transform of the image shown in (b).

underlying structure of a signal can be very easily recognized from the brushlet partition pattern that arises from a best basis approach. For example, looking at Figure 2.14(a), the small partitions lying along the diagonals correspond to large scale patterns that are oriented along the $\frac{\pi}{4}$ and $\frac{3\pi}{4}$ directions in the spatial domain. And the large partitions that lie along the edge of Figure 2.14(a) are part of a high frequency band, which correspond to small scale patterns such as circles at all positions within the spatial domain. As can be seen, these observations correspond exactly with the original textured image shown in Figure 2.14(b). In addition, this arrangement of real-valued brushlet coefficients allows for easy comparison with the standard complex brushlet decomposition.

Another version of the real-valued brushlet is one that uses local cosines (LCT) or local sines (LST) to segment the frequency domain of the FFTR; this method shall be denoted by FFTR→LCT/LST. It should be noted that it is also possible to use the LCT/LST to segment the frequency domain of the DCT or DST (denoted as DCT/DST→LCT/LST), but since none of the methods based on this approach (i.e., DCT/DST→LCT/LST) have a phase factor, this defeats one of the main purposes for creating brushlets in the first place; brushlets were originally created to have good frequency localization *and* oblique oscillations. But since the real-valued FFT and the LCT/LST both are orthogonal transforma-

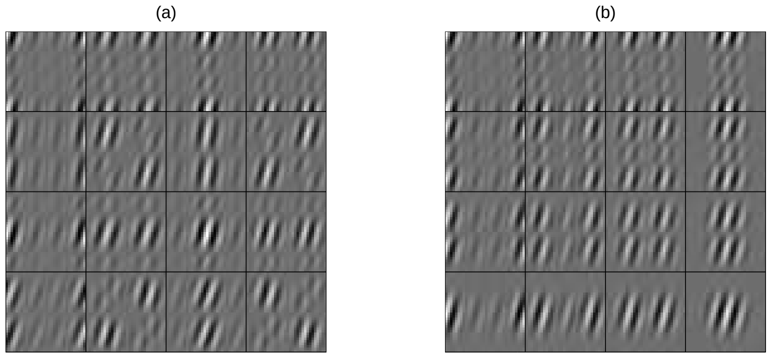

Figure 2.15: Sample basis functions of the real brushlet transform. (a) the FFTR→LFTR version with two localized, oriented bumps, and (b) the FFTR→LCT/LST version with four localized, oriented bumps.

tions, then FFTR→LCT/LST can simply be viewed as two successive rotations of the data. And since the LCT/LST does not have any phase, then its rotation does not destroy the phase produced by the real-valued FFT. Hence, the FFTR→LCT/LST version preserves the two most desirable properties of the original brushlet transform.

The real-valued brushlet bases differ from the standard brushlet bases in that they have four oriented, oscillating patterns located symmetrically about the center of the signal rather than two complex patterns. But in the FFTR→LFTR version of the real brushlet transform, two of these bumps can be suppressed if a steep rising cutoff function is employed (see Equation (2.45)). However, this is not true for the FFTR→LCT/LST brushlet bases (see Figure 2.15). In this case, the absence of a phase in the LCT/LST causes four oriented and localized bumps to always be present.

But for real signals and images, the real valued brushlet can often be a more useful tool than the standard brushlet. For example, in terms of representation cost, any real signal will require a minimum of two standard brushlet coefficients to code it because of its complex basis functions, whereas it is possible to only require one real valued brushlet coefficient to code the same signal.

2.4 Wavelets

The wavelet transform (WT) (see [14] p.167-185, [47] p.241-262, [62] and [11] ch.2) is similar to the brushlet transform in that it smoothly segments the frequency domain. The main difference, though is that it lacks a phase, causing poor angular resolution. In this respect, it is often considered the dual of the LCT/LST.

The mechanics of the discrete transform are as follows. First, a pair of finite length (L) lowpass filters ($\{h_k\}_{k=0}^{L-1}$) and highpass filters ($\{g_k\}_{k=0}^{L-1}$) decompose the time axis into low and high frequency parts by simultaneously performing convolution and subsampling. Letting H and G be the operators associated with these filters, they are constructed so as to satisfy the following orthogonality conditions for perfect reconstruction:

$$\begin{aligned} &HG^* = GH^* = 0, \\ &H^*H = G^*G = I, \\ &g_k = (-1)^k h_{L-1-k} \end{aligned} \tag{2.49}$$

Here, H^* and G^*, are the adjoint operators of H and G respectively, and I is the identity. The filter pairs that satisfy Equations (2.49) are known as *quadrature mirror filters*. After decomposing the top level, the high frequency coefficients are saved and the process is iterated on the low frequency components until the bottom level is reached. At this point, both the high and low frequency components are stored. In essence, this algorithm dyadically segments the frequency domain into finer and finer low frequency regions.

For example, if $x \in \ell^2(N)$, then the above process will first split x into two subsequences, Hx and Gx, both of length $N/2$. Then, the low frequency part, Hx will be split dyadically again into two subsequences, H^2x and GHx each of length $N/4$. Repeating this process $J = \log_2(N)$ times yields the N discrete wavelet coefficients $(Gx, GHx, GH^2x, \ldots, GH^Jx, H^{J+1}x)$. The corresponding bases functions, $w_{j,k}$, at scale j and position k are constructed by substituting

$(GH^j x)_l = \delta_{l,k}$, where $\delta_{l,k}$ is the Kronecker delta function, in the following synthesis algorithm. Starting with the coarsest scale components, $H^{J+1}x$ and $GH^J x$, compute

$$H^J x = (H^* H^{J+1} + G^* GH^J)x \tag{2.50}$$

and iterate until x is reconstructed. If $(GH^j x)_l = \delta_{l,k}$ is used instead, then the wavelet transform can be expressed simply as

$$\begin{array}{lll} (1) & \text{vanishing moments:} & \sum_{l=0}^{N-1} l^m w_{j,k}(l) = 0 \text{ for } m = 0, \ldots, M-1, \\ (2) & \text{regularity:} & \|w_{j,k}(l+1) - w_{j,k}(l)\| \leq c2^{-j\alpha}, \text{ for some } \alpha > 0 \\ (3) & \text{compact support:} & w_{j,k}(l) = 0 \text{ for } l \notin [2^j k, 2^j k + (2^j - 1)(L-1)]. \end{array} \tag{2.51}$$

In first of these conditions (1), the size of M determines the level of compression of the smooth parts of the signal. In second condition (2), the larger $\alpha > 0$ is, the smoother the basis functions become. The last condition (3), is needed for efficient numerical computation. As for the computational complexity, both analysis and synthesis require $O(N)$ operations.

A result similar to the Balian-Low theorem also exists for Wavelets; it states roughly that a wavelet constituting orthonormal bases cannot have exponential decay in both time and frequency [5]. As an example, take the following two well known wavelet bases: the Shannon wavelets, which are the difference between two sinc functions, and the Haar wavelets, which have the shortest possible filter length ($L = 2$). The two transforms associated with these bases clearly lie at opposite ends of the time-frequency plane; i.e., Shannon wavelets have good frequency localization but poor spatial localization, whereas the Haar wavelets have poor frequency localization but good spatial localization.

2.5 Wavelet Packets

There are many instances where the wavelet transform does not provide an efficient representation of a dataset because of its inability to adequately partition the high frequency axis. With this in mind, *wavelet packets* (see [72] chapter 7, and [13]) were devised. The decomposition begins in the same manner as the wavelet transform by splitting the frequency domain into low and high frequency components, Hx and Gx, respectively. But, then the high frequency components are further segmented so that the second level becomes, H^2x, GHx, HGx, G^2x. Repeating this process J times produces an overcomplete set of JN expansion coefficients giving rise to more than $2^{2^{J-1}}$ possible orthonormal bases. Choosing an efficient basis from this dictionary is done via the *best basis* algorithm described in Section 2.2.5 and Section 2.3.3 along with an appropriate information criterion. It should be noted that the computational cost of the best-basis algorithm is $O(N \log_2 N)$, for both analysis and synthesis.

Chapter 3

Problems With The LTT

In this chapter, some of the problems and shortcomings of the local trigonometric transform, the brushlet transform and the wavelet transforms from Chapter 2 are presented. These descriptions provide motivation for the Continuous Boundary Local Trigonometric Transform which is introduced in Chapter 4 and described in detail.

3.1 Inconsistencies

Some of the shortcomings of the methods named above first presented themselves to the author while working on a particular problem: the statistics of natural scenes. This type of analysis has gained recent popularity in the field of neuroscience ([6], [34], [25], [26], [54], [55], [56]), where the main focus is in better understanding the driving force behind the evolution of the mammalian primary visual cortex; that is, immersed in a natural environment, whether or not the receptive fields of simple cells of mammals self-organize into edge detectors. The basic approach in studying this has been to search overcomplete sets of bases to find ones that provide the *sparsest* or *least statistically dependent* representation. The main drawback, though, is the use of neural nets which are cost prohibitive

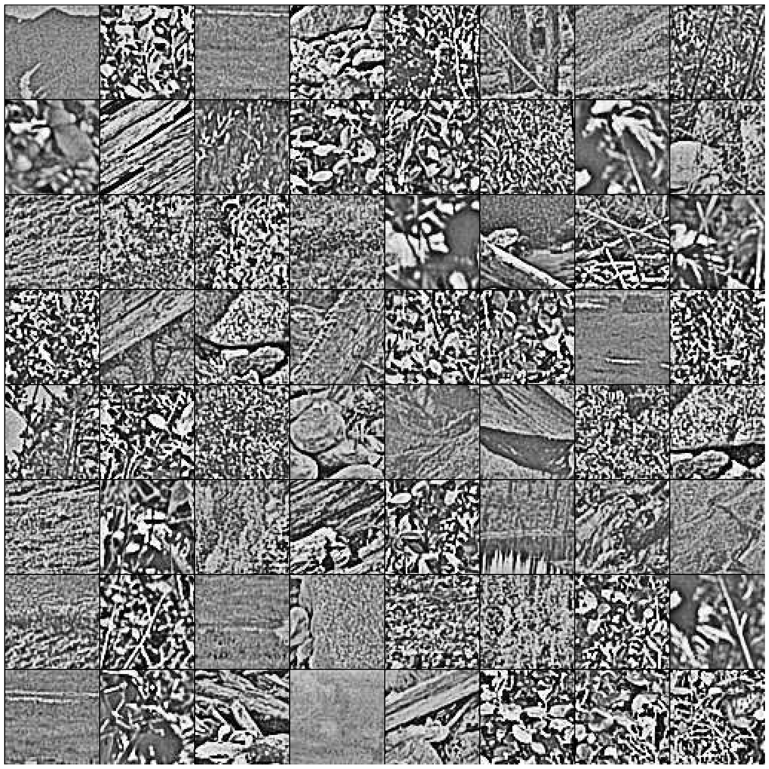

Figure 3.1: Sample of 64x64 pixel random patches drawn from natural scenes. A whitening filter was applied to each scene to mimic a process that occurs at the retinal level of the mammalian visual system. The natural scenes dataset was provided by Dr. Bruno Olshausen.

for large scale experiments. Thus, most of the experiments have been constrained to small 16x16 pixel image patches randomly drawn from a set of natural scenes (see Figure 3.1)

To overcome this problem, and to further analyze a class of similar images, the above experiments were repeated by Saito and the author [63] using fast algorithms which have finite basis dictionaries: local cosines and sines, local Fourier, brushlets and wavelet packets. The hope was to add insight and better understand the effects of image patch size, orthonormality, overcompleteness, basis orientation, sparsity and statistical independence in the formation of edge detectors. More details can be found in Saito [60] and Bénichou-Saito [7]. In mathematical terms, the approach is defined as follows.

Let X be a vector representing an image patch drawn randomly from a set of natural scenes, and let Y be the vector of expansion coefficients relative to a basis Φ, i.e., $Y = \Phi^{-1}X$. Then the following rule is used to search for the optimum basis via the joint best basis algorithm ([13])

$$\min_{\Phi \in \mathfrak{D}} E_X\{S(Y)\} \tag{3.1}$$

where E_X is the expectation operator, and $S(Y)$ measures either sparsity or statistical independence; i.e., $S(y) = ||Y||_p$ with $0 < p \leq 1$ for sparsity, and $S(Y) = \sum_i H(Y_i)$ for statistical dependence among the Y_i, where $H(Y_i)$ is the differential entropy of Y_i. It should be noted that $\lim_{p\downarrow 0} = ||Y||_p^p = \lim_{p\downarrow 0} \sum_i |Y_i|^p = ||Y||_0 = \#\{i : Y_i \neq 0\}$. Hence, the sparsity measure ℓ^p with $0 < p \leq 1$ is a more stable approximation of true sparsity. In addition, the reason $S(Y) = \sum_i H(Y_i)$ measures statistical dependence among the Y_i is because the mutual information of Y is

$$I(Y) = \int f_Y(y_1, \ldots, y_n) \log \frac{f_y(y_1, \ldots, y_n)}{\prod_{i=1}^n f_{Y_i}(y_i)} dy_1 \cdots dy_n = -H(Y) + \sum_{i=1}^n H(Y_i), \tag{3.2}$$

where f_Y is a joint probability density function of Y, and f_{Y_i} is a marginal pdf found by integration the joint pdf with respect to all but Y_i. Thus, the more dependent the Y_i's are, the larger $I(Y)$ becomes. And as long as the basis, Φ belongs to the set of volume preserving, linear invertible transformation, then $H(Y) = H(X)$; i.e, joint entropy is preserved. The computational cost of this approach is only $O(n[\log n]^2)$, much faster than the neural net approach.

Now, as stated above, some of the shortcomings of these localized transforms first presented themselves while working on this project. One of the main problems was the lack of any consistent results upon which to draw conclusions. Thousands of tests were performed using patch sizes ranging from 16×16 to 128×128. Variable rising cutoff function widths were tested in fixed folding and multiple folding schemes. Many different entropy estimates were tried, as well as

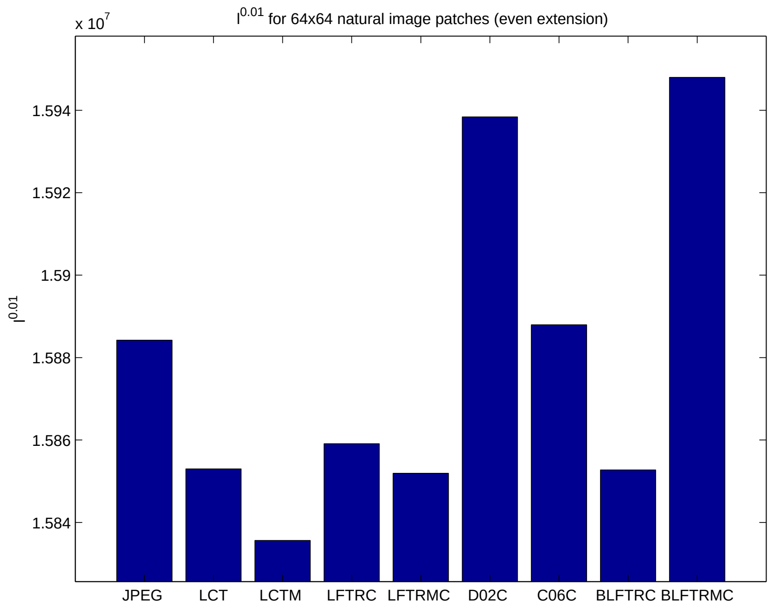

Figure 3.2: Representation cost using the sparsity measure, $\ell^{0.01}$, of various methods. Notice the wide range for the brushlet transform. With a sharp window (BLFTRC) the method achieved one of the sparsest representations, while a smooth windowed brushlet (BLFTRMC) was the least sparse.

many different ℓ^p norms. And throughout it all, no stable results were readily observed when using LFT, LCT, LST or brushlets. The partition patterns changed drastically for the different rising cutoff functions tested. And the overall representation costs varied wildly as well. As can be seen in Figure 3.2, there was a wide range in the representation cost of the real-valued brushlet transform when a sharp window was used (BLFTRC) versus a smooth window (BLFTRMC). The window width caused the method to range between the highest cost among the other methods, to one of the lowest costs. This type of behavior, along with the unpredictable test results, illustrated the impact that the folding operation had on the data. *The folding operation itself, rather than the dataset, seemed to be the dominating factor in the optimum basis partition pattern and overall representation cost.*

3.2 Instability Of The Folding Process

One of the main drawbacks of the LTT derives from the instability of the folding procedure on or near a jump in the signal. Discontinuities at folding locations cause improper boundary conditions within a subspace, often times resulting in increased representation cost. Furthermore, once a subspace fails to achieve the proper boundary conditions, the problem repeats itself recursively in children subspaces, resulting in improper or unpredictable partitioning (see Figure 3.3). In other words, it is the folding operation itself, rather than the dataset, that is the dominating factor in the resulting basis partition pattern and overall representation cost. In addition, since there does not exist an attractive, invertible folding process for the top level, then this problem is present in every level of the decomposition.

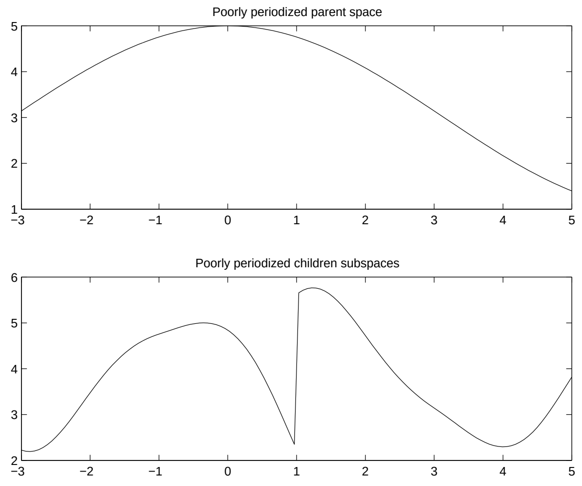

Figure 3.3: If the parent space is not properly periodized (top figure), then the children subspaces will not properly periodize (bottom figure). This error propagates recursively to lower levels of the decomposition.

Evidence of this behavior can also be seen when switching between cosine and sine polarities. Using the same dataset, completely different basis partition patterns arise when the polarity is switched. Figure 3.4 illustrates this fact. The folding operation seems to be the determining factor in the choice of basis (or partitioning), rather than the underlying structure in the image.

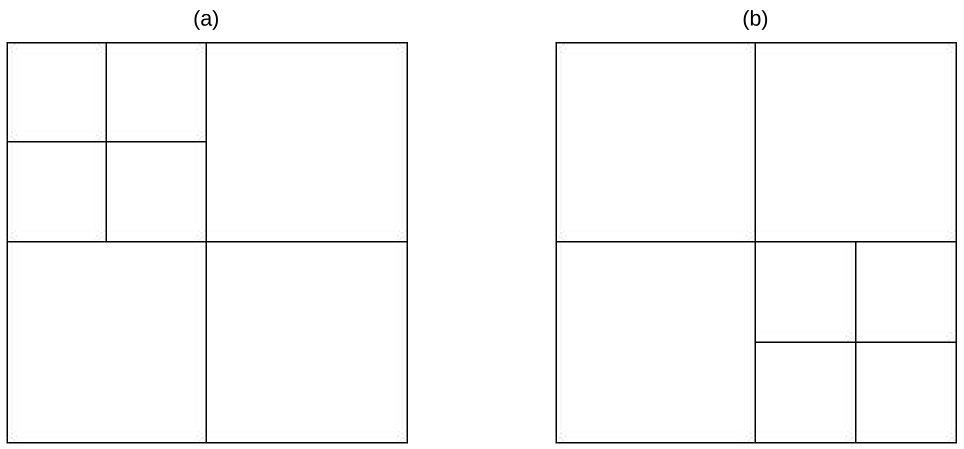

Figure 3.4: Basis partition patterns of the same image using the real valued local Fourier transform with (a) sine polarity and (b) cosine polarity.

A more drastic illustration is found by applying the LTT to the image in Figures 3.5 and 3.6. The partition pattern chosen is not based on the underlying structure in the image. As the image is shifted and rotated, and different ℓ^p norms are employed, the resulting segmentations are not consistent with the image details. The discrepancies are even worse when the LFT is used.

The better understand the cause of this behavior, the following lemmas are needed:

Lemma 3.2.1 (Wickerhauser ([72], p.110)). *Suppose $r \in C^d(\mathbb{R})$ for some $0 \le d \le \infty$. If $x \in C^d(\mathbb{R})$, then Ux has d continuous derivatives in $\mathbb{R}\backslash\{0\}$, and for all $0 \le n \le d$ there exist limits $[Ux]^{(n)}(0+)$ and $[Ux]^{(n)}(0-)$ which satisfy the following conditions:*

$$\begin{array}{rcl} \lim_{t\to 0+}[Ux]^{(n)}(t) & = & 0 \quad \textit{if } n \textit{ is odd}, \\ \lim_{t\to 0-}[Ux]^{(n)}(t) & = & 0 \quad \textit{if } n \textit{ is even}. \end{array} \tag{3.3}$$

For $I = (\alpha_0, \alpha_1)$

Lemma 3.2.2 (Wickerhauser ([72], p.120)). *If $x \in C^d(\mathbb{R})$ then $W_I x \in C^d(\mathbb{R}\backslash\{\alpha_0, \alpha_1\})$, has one-sided limits $[W_I x]^{(n)}(\alpha_0+)$ and $[W_I x]^{(n)}(\alpha_1-)$ for all*

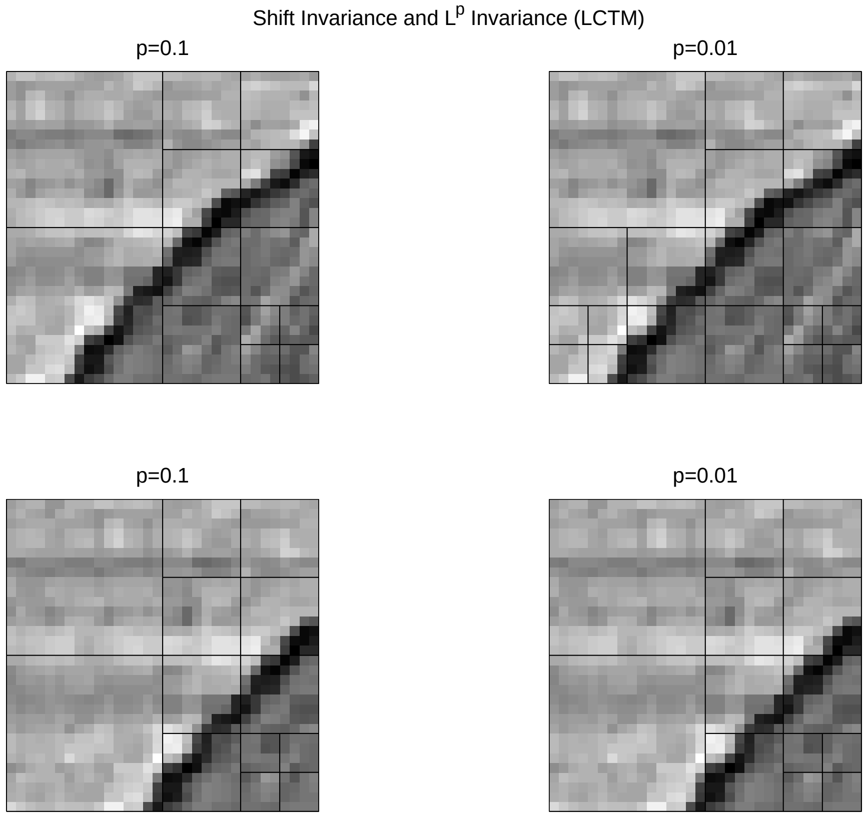

Figure 3.5: Comparing the left two plots to the right two plots shows the instability of the partition pattern under two different sparsity measures ($\ell^{0.1}$ versus $\ell^{0.01}$) using LCT with multiple folding (LCTM). Comparing the top two plots to the bottom two plots shows the lack of robust partitioning under a shift using LCTM.

$0 \le n \le d$, *and satisfies the following conditions:*

$$\begin{array}{rcll} \lim_{t\to\alpha_0+}[W_I x]^{(n)}(t) & = & 0 & \textit{if } n \textit{ is odd;} \\ \lim_{t\to\alpha_1-}[W_I x]^{(n)}(t) & = & 0 & \textit{if } n \textit{ is even.} \end{array} \tag{3.4}$$

Conversely, if $x \in C^{d(I)}$ *with one-sided limits* $x^{(n)}(\alpha_0+)$ *and* $x^{(n)}(\alpha_1-)$ *for all* $0 \le n \le d$ *which satisfy*

$$\begin{array}{rcll} \lim_{t\to\alpha_0+} x^{(n)}(t) & = & 0 & \textit{if } n \textit{ is odd;} \\ \lim_{t\to\alpha_1-} x^{(n)}(t) & = & 0 & \textit{if } n \textit{ is even,} \end{array} \tag{3.5}$$

then $W_I^* x$ *satisfies the equation*

$$\lim_{t\to\alpha_0+}[W_I^* x]^{(n)}(t) \;=\; \lim_{t\to\alpha_1-}[W_I^* x]^{(n)}(t), \quad \textit{for all } 0 \le n \le d. \tag{3.6}$$

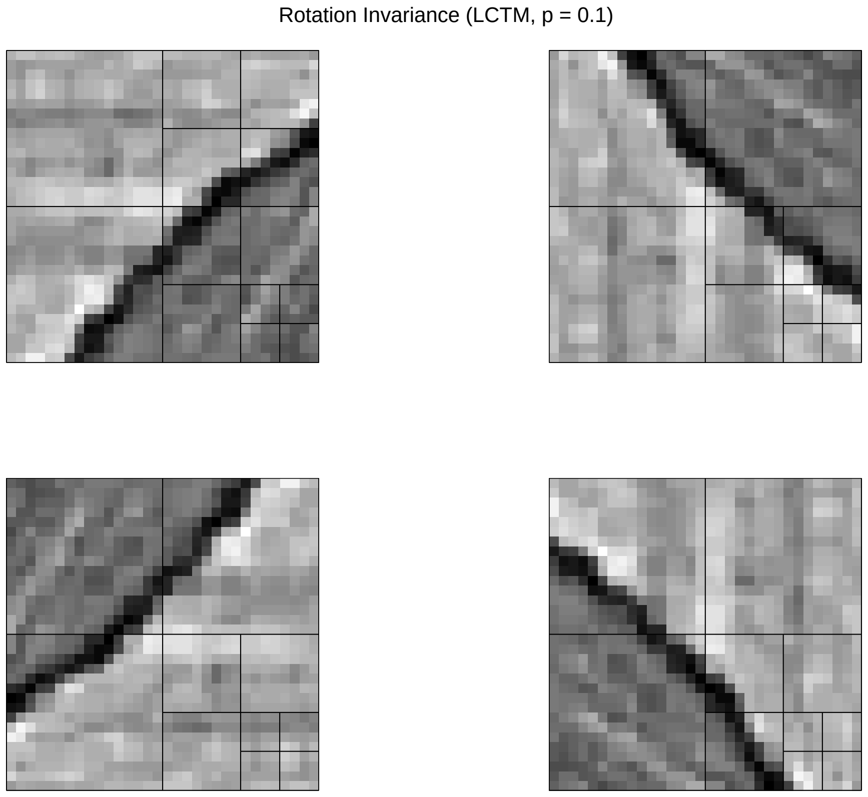

Figure 3.6: Non robust partition patterns under a rotation of the image using LCTM.

Thus $W_I^* \mathbf{1}_I x$ *has a continuous periodic extension in* $C^d(\mathbb{R})$.

So suppose that $x \in C^d([0,1])$ and $x(0) \neq x(1)$; that is, x does not have a continuous periodic extension. Assume that full cosine polarity folding is used along with LCT or LFT. Now the mechanics of the these transforms are as follows. At the top level of the decomposition, periodic folding is first applied to the boundary of the signal; i.e., $W_{[0,1]}x$. But since x does not satisfy the condition of Lemma 3.2.2, namely $x \notin C^d(\mathbb{R})$ for any d, Equations (3.4) will not hold true for any d. In the next step of the process, folding is performed at the midpoint of the signal; $\tilde{x}^{LCT}_{[0,1]} = U_{\frac{1}{2}} W_{[0,1]} x$. According to Lemma 3.2.1, Equations (3.3) will be valid for all d. What this means is that the left half interval will have boundary conditions matching those of the LCT basis functions for level 1, but the right half interval will not. Similarly, if LFT is used instead of LCT, then periodized unfolding is needed to periodize the two half intervals; i.e., $\tilde{x}^{LFT}_{[0,\frac{1}{2}]} = W^*_{[0,\frac{1}{2}]}U_{\frac{1}{2}}W_{[0,1]}x$ and $\tilde{x}^{LFT}_{[\frac{1}{2},1]} = W^*_{[\frac{1}{2},1]}U_{\frac{1}{2}}W_{[0,1]}x$. But after applying this final step, it is clear that conditions (3.5) are only satisfied for $d = 0$ on $\tilde{x}^{LFT}_{[0,\frac{1}{2}]}$. On $\tilde{x}^{LFT}_{[\frac{1}{2},1]}$, they will not be satisfied for any d. In other words, regardless of the method used, LCT or LFT, if the parent interval does not have the proper boundary conditions, then the left child interval will have the proper boundary conditions, but the right child interval will not.

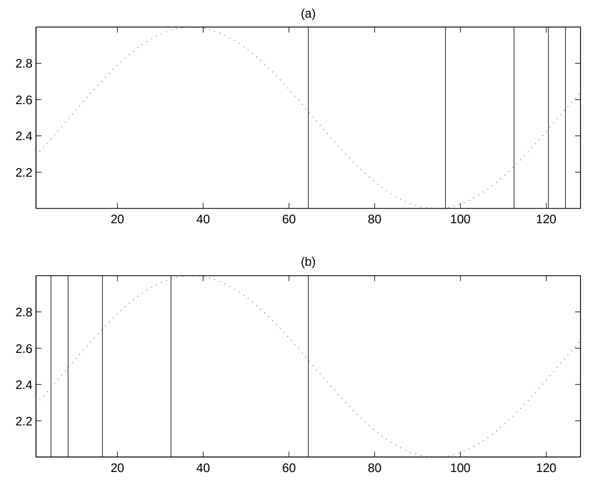

Figure 3.7: Basis partition patterns of the same signal using the real-valued local Fourier transform. The patterns were chosen using (a) cosine polarity and (b) sine polarity.

Repeating the process in a recursive manner yields the same results for the right child subspace for either method, LCT or LFT. Since $\tilde{x}_{[\frac{1}{2},1]}$ does not have the proper boundary conditions, then just like its parent interval, its left child interval will have the proper boundary conditions while its right child interval will not. On the other hand, since $\tilde{x}_{[0,\frac{1}{2}]}$ has the proper boundary conditions for $d = 0$, then its two children will also have continuous extensions in C^0. This pattern repeats itself recursively for all children intervals so that at each level of the decomposition, every interval has the proper boundary conditions except for the one farthest to the *right.* Hence, when the best basis algorithm is employed, it tries to minimize the effect of this rightmost discontinuous interval by confining it to the smallest subspace possible: the one at the bottom level of the decomposition. This leads to partition patterns like those of figure 3.7(a). If, instead, sine polarity is used, then similar results appear, but in reverse order; that is, all of the children intervals have the proper boundary conditions in C^0 except for the one farthest to the *left.* This gives rise to partition patterns like those seen in figure 3.7(b).

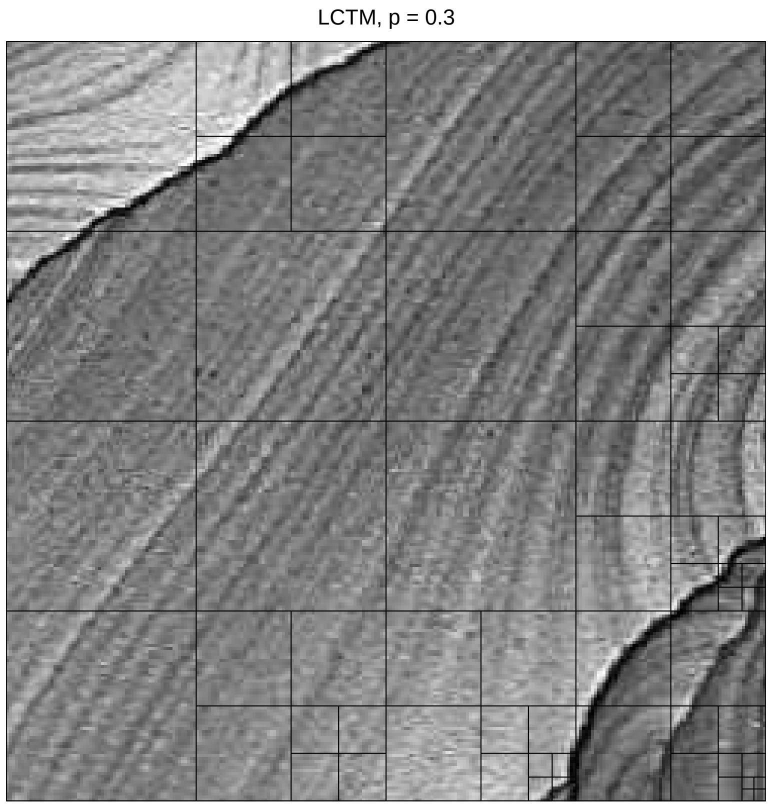

Figure 3.8: Basis partition patterns of an image using the local Cosine transform. Notice that the partition pattern found in Figure 3.4(b) appears in many locations throughout the image.

Since this problem can occur whenever folding takes place at a discontinuity, then these same patterns can appear at any level and location within the decomposition. Figure 3.8 illustrates this. Notice that the partition pattern found in Figure 3.4(b) appears in many locations throughout the image. It is therefore very important to try to reduce this artifact, thus allowing the underlying structure within the signal to dictate the partitioning, rather than the folding procedure.

3.3 Mixing Information Across Boundaries

Another problem arises because information within subspaces is folded across boundaries; that is, basis functions have global support. Although this can produce a desirable effect when reconstructing compressed signals by reducing the blocking effect between subspace boundaries, it also can be a detriment to various applications such as texture segmentation and signal analysis. This becomes particularly problematic for the LFT and BT since they require the additional "periodized unfolding" operations (equation (2.28)) which are not needed for the LCT.

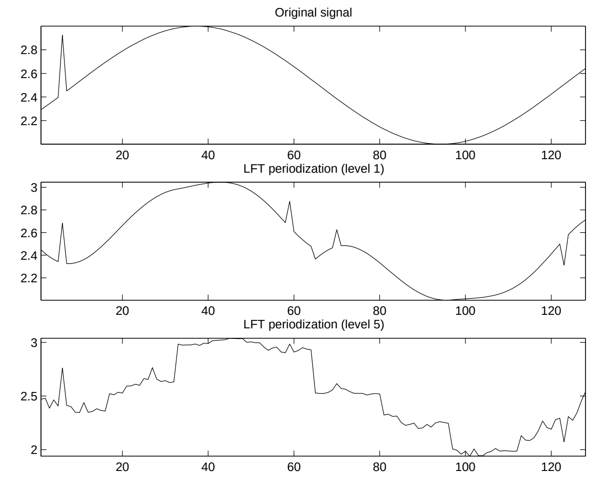

Figure 3.9: LFT periodization causes the sharp bump in the parent space to be mixed into many locations of the children subspaces.

As can be seen in Figure 3.9, a structure such as the sharp bump which is localized to one region of the parent space, is mixed within all children subspaces when using LFT periodization. By the time the bottom level of the decomposition is reached, the bump is mixed to every region of the signal; in essence, almost all recognizable structure has been lost due to the mixing of information across subspace boundaries.

In response to the problems described above, an approach was devised to remedy these and other shortcomings. In particular, the algorithm satisfies the following constraints:

- It is invertible;
- It operates on all subspaces including the top level;
- It is stable, preserving continuity at the subspace boundaries;
- It is computationally efficient;
- It is an isometry for use in the best basis algorithm.

One possible solution is presented in the next section.

Chapter 4

The Continuous Boundary Local Trigonometric Transform

In this chapter, the Continuous Boundary Local Trigonometric Transform is presented. The derivation includes the earliest approaches, and illustrates how their shortcomings led to the much improved later versions, which will be described in Chapter 5. Many alternate approaches are described in Appendix B.4.

4.1 Reduction of Edge Effect

The main goal of the CBLTT is to stabilize the folding process by forcing continuity at the location of folding; i.e., at the boundaries of the subspaces in the hierarchy of the LTT decomposition. To achieve this goal, each subspace is considered to be disjoint of one another so that no information is mixed between adjacent intervals during the folding procedure, and an artificial extension is created in a continuous fashion at the boundary of each subspace. It is because of this forced continuity at the boundary that this algorithm derives its name (CBLTT). Although there exists an infinite number of possible continuous extensions, the minimum requirement is to create one with certain properties that make for stable inversion. The reason for this requirement is that after folding is performed in the forward direction, for compression purposes the folded extension is discarded. Thus, inverse folding cannot be performed unless an inverse formula can be derived from the structure of the artificial extension. As will be seen in the remainder of the thesis, finding a stable inverse is not a simple manner.

To begin with, an inverse does not always exist in the continuous case. To see this, and for simplicity, start with an even extension; that is, let the artificial extension be created by reflecting the data within the subspace about the subspace boundaries. Folding is then performed at each subspace boundary, and the folded extensions are discarded. The result is a windowed signal which has the proper boundary conditions for the LCT or LST, depending on the polarity. If the subspace is to be periodized instead (as a preprocessing step for LFT), then periodized unfolding, W^*, should be applied to the result.

Mathematically this approach is formulated as follows. Let $x(t) \in C^d[0,1]$ for some d. Let $x(0) \neq x(1)$; i.e., there is a discontinuity at the boundary if the function is periodized. Let $0 \leq \epsilon \leq \frac{1}{2}$ where ϵ is the radius of the rising cutoff function. Then the even extension is defines as $x(t) = x(-t)$ for $t \in [-\epsilon, 0]$, and $x(t) = x(2-t)$ for $t \in [1, 1+\epsilon]$. Using Equation (2.12), folding becomes

$$\begin{aligned}
\tilde{x}(t) &= \begin{cases} r\left(\frac{t}{\epsilon}\right)x(t) + r\left(\frac{-t}{\epsilon}\right)x(-t) & \text{if } 0 < t < \epsilon, \\ x(t) & \text{if } \epsilon \leq t \leq 1-\epsilon, \\ r\left(\frac{1-t}{\epsilon}\right)x(t) - r\left(\frac{t-1}{\epsilon}\right)x(2-t) & \text{if } 1-\epsilon < t < 1, \end{cases} \\
&= \begin{cases} \left[r\left(\frac{t}{\epsilon}\right) + r\left(\frac{-t}{\epsilon}\right)\right]x(t) & \text{if } 0 < t < \epsilon, \\ x(t) & \text{if } \epsilon \leq t \leq 1-\epsilon, \\ \left[r\left(\frac{1-t}{\epsilon}\right) - r\left(\frac{t-1}{\epsilon}\right)\right]x(t) & \text{if } 1-\epsilon < t < 1, \end{cases} \qquad (4.1)
\end{aligned}$$

where $\tilde{x}$ denotes the folded function. From Equation (4.1), the formula for inversion becomes

$$x(t) = \begin{cases} \frac{\tilde{x}(t)}{r\left(\frac{t}{\epsilon}\right)+r\left(\frac{-t}{\epsilon}\right)} & \text{if } 0 < t < \epsilon, \\ \tilde{x}(t) & \text{if } \epsilon \leq t \leq 1-\epsilon, \\ \frac{\tilde{x}(t)}{r\left(\frac{1-t}{\epsilon}\right)-r\left(\frac{t-1}{\epsilon}\right)} & \text{if } 1-\epsilon < t < 1. \end{cases} \qquad (4.2)$$

As can be seen by Equation (4.2), when $t = 1$, $x(1) = \frac{\tilde{x}(1)}{r(0)-r(0)}$ which is undefined. Thus, inversion is impossible unless the value of the right hand boundary is stored. Luckily, though, this obstacle can be circumvented when the problem is cast in a discrete setting. Not only does it become possible to find an inverse, but is it is also possible to find stable inverse formulas. Because of this fact, the formulation of the CBLTT is continued in a discrete setting, referring to the continuous case only when it adds insight into the understanding of the algorithm.

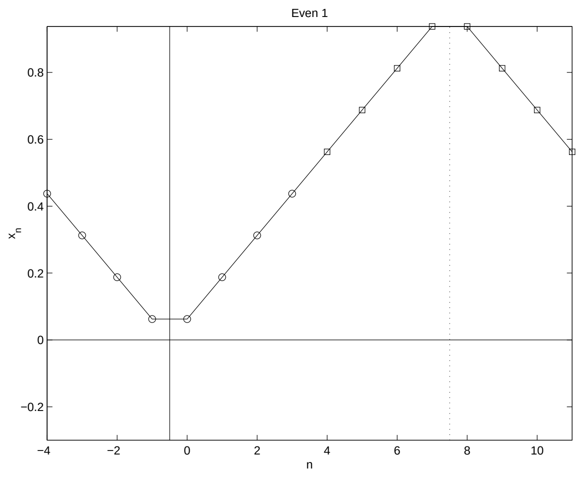

Figure 4.1: Example of an even extension of a linear function with midpoint folding. In this example, $N = 7$.

Repeating the above derivation with discrete variables produces the following. Let x_n be a function of $N+1$ discrete points indexed from 0 to N (see Figure 4.1). Let $x_0 \neq x_N$, i.e., there is a discontinuity at the boundary if the function is simply periodized. For simplicity, let R be the radius of the rising cutoff function such that $R \leq \frac{N+1}{2}$, and *let* $n = 0, \ldots, R-1$ *in all of the following formulas.*

Then one version of the discrete continuous even extension can be constructed according to Figure 4.1 using midpoint folding. With this arrangement, the values of the extension to the left of x_0 are given by

$$x_{-n-1} = x_n \tag{4.3}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} = x_{N-n}. \tag{4.4}$$

Using Equation (2.12), folding at the left hand edge is defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n-1} x_{-n-1} \\ &= r_n x_n + r_{-n-1} x_n \\ &= (r_n + r_{-n-1}) x_n \end{aligned} \tag{4.5}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1} x_{N-n} \\ &= (r_n - r_{-n-1}) x_{N-n}. \end{aligned} \tag{4.6}$$

For inversion, simply solve for x_n in Equation (4.5) to get

$$x_n = \frac{\tilde{x}_n}{r_n + r_{-n-1}} \tag{4.7}$$

and for x_{N-n} in Equation (4.6) to get

$$x_{N-n} = \frac{\tilde{x}_{N-n}}{r_n - r_{-n-1}}. \tag{4.8}$$

For future reference, CBLFT methods based on even extensions will be denoted as CELFT. If the letter M or G is included, this denotes midpoint folding or gridpoint folding respectively. For example, CEMLFT stands for the Continuous Even extension Midpoint folding Local Fourier Transform.

One of the first problems that arises is that this is not a unitary transformation. To see this, let U_E be the even extension folding operation defined by

Equations (4.5) and (4.6); that is $\tilde{\mathbf{x}} = U_E\mathbf{x}$ where

$$U_E = \begin{bmatrix} r_0 + r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{R-1} + r_{-R} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{R-1} - r_{-R} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 - r_{-1} \end{bmatrix}.$$

Now, since U_E is diagonal, it will be an unitary transformation as long as its diagonal entries lie on the unit circle. Clearly this is not the case for the above matrix. Hence, it is not a unitary transformation; in particular, it is not an isometry. This latter condition is necessary for use in the best basis algorithm to facilitate a comparison between parent and children subspaces. Now, the folding operation, U, is itself unitary, preserving energy of the original signal. But when artificial extensions are introduced at the boundary, energy is added to, or removed from, the subspace. After folding, there is no guarantee that the change in energy of the left and right parts of the subspace cancel each other out. Consequently, the following simple idea is proposed to guarantee an isometry.

- Starting with one half of the subspace, extend the signal as an even reflection about the boundary and perform odd polarity folding;
- Discard the folded extension;
- At the other boundary, extend the signal as an even reflection and add a certain amount, say s, to the signal and extension;
- Perform even polarity folding at this location;
- Subtract s from the signal and extension so that continuity is preserved throughout the interval;

- Discard the folded extension;
- Perform periodized unfolding if required (for CBLFT).

Figure 4.2 illustrates this idea.

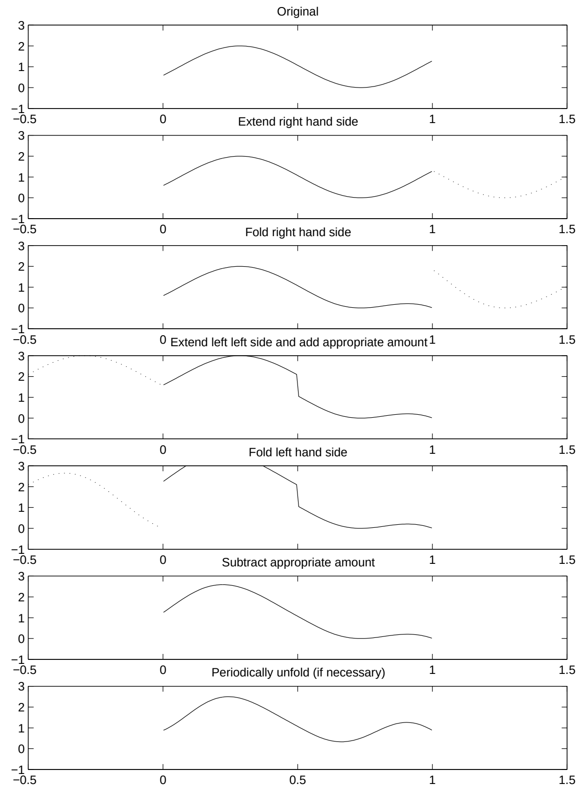

Figure 4.2: Isometric folding with an even extension.

Mathematically, the generalized isometric folding formula for the left half interval can be found using Equation (4.5) with the addition and subtraction of $s = s(x_n)$ to get

$$\begin{aligned}\tilde{x}_{s,n} &= r_n(x_n + s) + r_{-n-1}(x_{-n-1} + s) - s \\ &= r_n x_n + r_{-n-1}x_{-n-1} + s(r_n + r_{-n-1} - 1) \\ &= \underbrace{(r_n + r_{-n-1})x_n}_{\tilde{x}_n} + s\underbrace{(r_n + r_{-n-1} - 1)}_{\lambda_n}. \qquad (4.9)\end{aligned}$$

where $\tilde{x}_{s,n}$ stands for the folded result with amplitude shift. Thus, this is just the standard folding operation, with the added term, $s\lambda_n$. The fact that s must be computed from x_n in order to guarantee the isometry is addressed in section 4.3.

For inversion, solving for x_n in the previous formula yields the generalized isometric inversion formula for the left half interval. It is

$$x_n = \underbrace{\frac{\tilde{x}_{s,n}}{r_n + r_{-n-1}}}_{z_n} - s \underbrace{\frac{\lambda_n}{r_n + r_{-n-1}}}_{\alpha_n} . \tag{4.10}$$

Notice that this is just the standard inversion operation of Equation (4.7) applied to $\tilde{x}_{s,n}$ with the added term $-s\alpha_n$.

4.2 Initial Drawbacks of the CBLTT

One of the first noticeable drawbacks of the CBLTT is the lack of C^1 continuity at the subspace boundaries between the signal and the even extension. This translates into a cusp at the boundary when viewed as a periodic function (see the bottom plot in Figure 4.2). As was mentioned in Section 2.2.3, the amount of smoothness the function has at the folding location is preserved by the folding and periodization operators. So it would be better to design C^1 extensions.

Another problem is that a real-valued constant, s, does not always exist. Although complex constants are possible, there are many times when it is not desirable to allow complex shifts to occur. For all of the examples and experiments in this paper, only real-valued signals are used; hence, only real-valued shifts make sense.

A third drawback is that this method is not linear. To see this, consider the following proposition. For simplicity, the proposition has been generalized for the continuum case, although the results hold true for the discrete version as well.

Proposition 4.2.1. *Let $s_x = s(x) \in \mathbb{R}$ and let U be the unitary folding operator* *defined by Equation* (2.16)*. Then* $U_{s_x}x(t) \triangleq U[x(t)+s_x]-s_x$ *is not a linear operator.*

Proof. Let $z(t) = ax(t) + by(t)$ where $x(t) \in L^2(\mathbb{R})$ and $y(t) \in L^2(\mathbb{R})$, and $a \in \mathbb{R}$ and $b \in \mathbb{R}$ are scalars. Then

$$\begin{aligned}
U_{s_z}z(t) &= U_{s_z}[ax(t)+by(t)] \\
&= U[ax(t)+by(t)+s_z]-s_z \\
&= aUx(t)+bUy(t)+Us_z-s_z \\
&= [aUx(t)+aUs_z-as_z]+[bUy(t)+bUs_z-bs_z]+(1-a-b)(Us_z-s_z) \\
&= a\{U[x(t)+s_z]-s_z\}+b\{U[y(t)+s_z]-s_z\}+(1-a-b)(Us_z-s_z) \\
&= aU_{s_z}x(t)+bU_{s_z}y(t)+(1-a-b)U_{s_z}0 \\
&\neq aU_{s_z}x(t)+bU_{s_z}y(t) \text{ for all } s_z.
\end{aligned}$$

To see why $U_{s_z}0 \neq 0$ for every s_z, simply combine Equations (4.9) and (2.12) to get

$$U_{s_z}(r,\alpha,\epsilon)x(t) \triangleq \begin{cases} \left[r\left(\frac{t-\alpha}{\epsilon}\right)+r\left(\frac{\alpha-t}{\epsilon}\right)\right]x(t) & \\ \quad +s_z\left[r\left(\frac{t-\alpha}{\epsilon}\right)+r\left(\frac{\alpha-t}{\epsilon}\right)-1\right] & \text{if } \alpha<t<\alpha+\epsilon, \\ \left[r\left(\frac{\alpha-t}{\epsilon}\right)-r\left(\frac{t-\alpha}{\epsilon}\right)\right]x(2\alpha-t) & \text{if } \alpha-\epsilon<t<\alpha, \\ x(t) & \text{otherwise.} \end{cases}$$

Thus if $x(t) \equiv 0$, then

$$\begin{aligned}
U_{s_z}(r,\alpha,\epsilon)0 &= \begin{cases} s_z\left[r\left(\frac{t-\alpha}{\epsilon}\right)+r\left(\frac{\alpha-t}{\epsilon}\right)-1\right] & \text{if } \alpha<t<\alpha+\epsilon \\ 0 & \text{otherwise} \end{cases} \\
&\neq 0 \text{ unless } s_z = 0.
\end{aligned}$$

□

This makes it difficult to analyze and construct basis functions.

Another problem is the instability of the inversion formula. According to Equation (4.8), the reconstructed signal depends on values divided by small numbers near the odd parity boundary; recall that $0 < r_n - r_{-n-1} \ll 1$ for values of n

near 0. So reconstruction errors due to thresholding or quantization are enhanced. This can be seen in Figure 4.3 which shows a subspace being compressed at the rate of 4:1 while using CBLFT. Compression was achieved by simple thresholding; that is, only the most energetic 25 percent of the transformed coefficients were retained for this example.

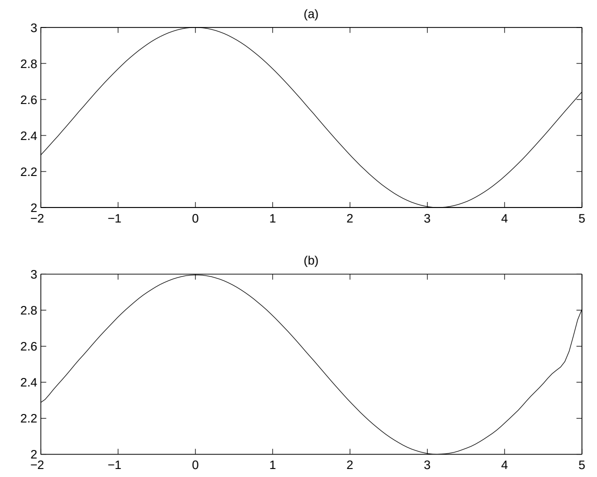

Figure 4.3: (a) Original signal. (b) Compression results of 4:1 using just the top level CBLFT. Notice that the left half interval exhibits stable reconstruction, whereas the right half interval is unstable.

The last issue involves the computation of the constant, s. One method of evaluation involves searching for the shift via some rootfinding scheme such as the Bisection Method. But this can be a costly operation with potentially slow convergence; for example, the Bisection Method has only a linear convergence rate. It would therefore be better to derive an explicit formula for the constant, s. Each of these issues is addressed in the next sections, starting with this last issue of finding an explicit formula for s.

4.3 An Explicit Shift Formula

With isometric folding well defined, a formula for finding an explicit value of s can be easily derived by using the notation from the previous section along with the definition of an isometry:

$$\begin{aligned}
0 &= \textstyle\sum_{n=0}^{R-1}\left[(\tilde{x}_n + s\lambda_n)^2 + \tilde{x}_{N-n}^2\right] - \sum_{n=0}^{R-1}\left(x_n^2 + x_{N-n}^2\right)\\
\Leftrightarrow 0 &= \textstyle\sum_{n=0}^{R-1}\left(\tilde{x}_n^2 + 2s\tilde{x}_n\lambda_n + s^2\lambda_n^2 + \tilde{x}_{N-n}^2 - x_n^2 - x_{N-n}^2\right)\\
\Leftrightarrow 0 &= \textstyle s^2\sum_{n=0}^{R-1}\lambda_n^2 + 2s\sum_{n=0}^{R-1}\tilde{x}_n\lambda_n + \sum_{n=0}^{R-1}\left(\tilde{x}_n^2 + \tilde{x}_{N-n}^2 - x_n^2 - x_{N-n}^2\right)\\
\Leftrightarrow s &= \frac{-\sum_{n=0}^{R-1}\tilde{x}_n\lambda_n \pm \sqrt{\left(\sum_{n=0}^{R-1}\tilde{x}_n\lambda_n\right)^2 - \sum_{n=0}^{R-1}\lambda_n^2\cdot\sum_{n=0}^{R-1}\left(\tilde{x}_n^2+\tilde{x}_{N-n}^2-x_n^2-x_{N-n}^2\right)}}{\sum_{n=0}^{R-1}\lambda_n^2}. \qquad (4.11)
\end{aligned}$$

In a similar fashion, the formula for isometric inversion can be easily computed to give

$$s = \frac{\sum_{n=0}^{R-1}\alpha_n z_n \pm \sqrt{\left(\sum_{n=0}^{R-1}\alpha_n z_n\right)^2 - \sum_{n=0}^{R-1}\alpha_n^2\cdot\sum_{n=0}^{R-1}\left(z_n^2+z_{N-n}^2-\tilde{x}_{s,n}^2-\tilde{x}_{s,N-n}^2\right)}}{\sum_{n=0}^{R-1}\alpha_n^2}. \qquad (4.12)$$

where z_n and α_n are given by Equation (4.10), and z_{N-n} is simply the results of applying Equation (4.8) to $\tilde{x}_{s,n}$.

It should be noted that most of the values in Equations (4.11) and (4.12) are constants that can be precomputed only once for each interval; hence, there is little computational overhead involved with these isometric operators. To be exact, notice that λ_n and $\sum_{n=0}^{R-1}\lambda_n^2$ can both be precomputed once and stored for use in all subsequent calls to the folding and unfolding operators. Therefore, the computational cost is found from the remaining terms to be:

$$\sum_{n=0}^{R-1}\tilde{x}_n\lambda_n \quad : \quad R \text{ mults} + R-1 \text{ adds} = 2R-1 \text{ flops}$$

$$\sum_{n=0}^{R-1}(\tilde{x}_n^2 + \tilde{x}_{N-n}^2 + x_n^2 + x_n^2) \quad : \quad 4R \text{ mults} + 4R-1 \text{ adds} = 8R-1 \text{ flops}$$

Adding these together along with 6 additional flops needed to compute Equation (4.11) yields a total cost of $10R + 4$ flops for each interval. Even if full folding is employed, where $R = \frac{N}{2}$, this is still an $O(N)$ operation. Thus DCT, DST and FFT are still the most costly parts of the procedure.

4.4 The Continuous Odd Extension

In response to the cusp problem described in Section 4.2, the idea of adding C^1 continuity is natural. One obvious choice of an extension is to use an odd function which is shifted to force continuity. Mechanically, the procedure operates in exactly the same manner as the even extension, simply with the new continuous odd extension substituted. See Figure 4.4. Mathematically, things become a bit

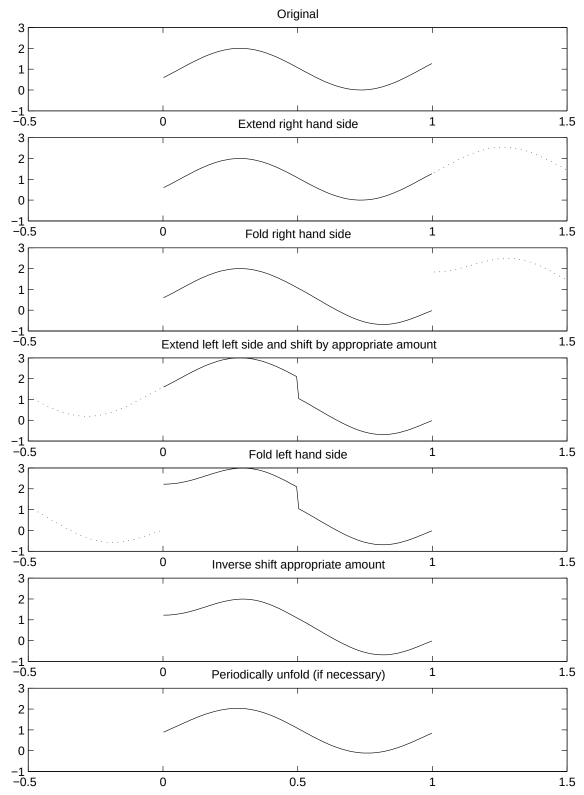

Figure 4.4: Isometric folding with a continuous odd extension.

more challenging, though, especially for the inversion formulas. To start with, let $n = 0, \ldots, R-1$ in all of the formulas in this section. Then the values for the extension to the left of x_0 are given by

$$\begin{aligned} x_{-n-1} &= x_0 - (x_n - x_0) \\ &= 2x_0 - x_n \end{aligned} \tag{4.13}$$

and the values of the extension to the right of x_N are

$$\begin{aligned} x_{N+n+1} &= x_N + (x_N - x_{N-n}) \\ &= 2x_N - x_{N-n}. \end{aligned} \tag{4.14}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n-1} x_{-n-1} \\ &= r_n x_n + r_{-n-1}(2x_0 - x_n) \\ &= (r_n - r_{-n-1})x_n + 2r_{-n-1}x_0 \end{aligned} \tag{4.15}$$

and at the right hand edge as

$$\begin{aligned} \tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1}(2x_N - x_{N-n}) \\ &= (r_n + r_{-n-1})x_{N-n} - 2r_{-n-1}x_N. \end{aligned} \tag{4.16}$$

Now, for the isometric version, the isometric folding formula for the left half interval, $\tilde{x}_{s,n}$, can be found using Equation (4.11) with the following values of $\tilde{x}_n$ and λ_n

$$\begin{aligned} \tilde{x}_{s,n} &= r_n(x_n + s) + r_{-n-1}(x_{-n-1} + s) - s \\ &= r_n x_n + r_{-n-1}x_{-n-1} + s(r_n + r_{-n-1} - 1) \\ &= \underbrace{r_n x_n + r_{-n-1}(2x_0 - x_n)}_{\tilde{x}_n} + s \underbrace{(r_n + r_{-n-1} - 1)}_{\lambda_n}. \end{aligned} \tag{4.17}$$

where $\tilde{x}_{s,n}$ stands for the folded result with shift. Thus, this is just the standard folding operation, with the added term, $s\lambda_n$.

Solving for x in Equations (4.15) and (4.16) yields the following formula for inverse folding on the left half interval

$$x_n = \frac{\tilde{x}_n - 2r_{-n-1}x_0}{r_n - r_{-n-1}} \tag{4.18}$$

and on the right half interval

$$x_{N-n} = \frac{\tilde{x}_{N-n} + 2r_{-n-1}x_N}{r_n + r_{-n-1}}. \tag{4.19}$$

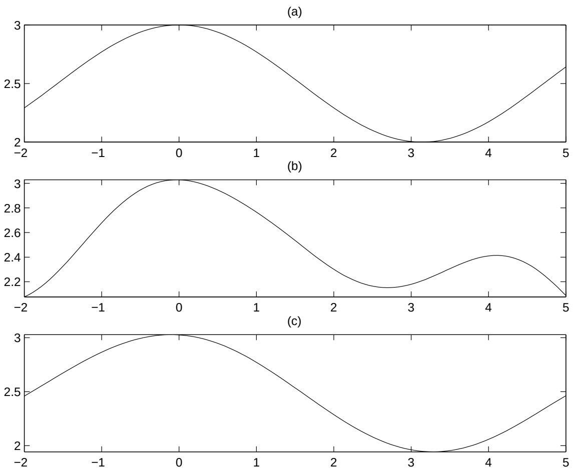

Figure 4.5: Periodization comparison. (a) Original signal, (b) CELFT periodization, (c) COLFT periodization.

Now this approach yields implicit inversion formulas dependent on x_0 and x_N. Luckily explicit formulas can be easily found for x_0 and x_N and they are

$$x_0 = \frac{\tilde{x}_0}{r_0 + r_{-1}} \tag{4.20}$$

and

$$x_N = \frac{\tilde{x}_N}{r_0 - r_{-1}}. \tag{4.21}$$

Substituting these results into Equations (4.18) and (4.19) produces the following generalized inversion formulas for the left half interval

$$x_n = \frac{\tilde{x}_n - \frac{2r_{-n-1}\tilde{x}_0}{r_0+r_{-1}}}{r_n - r_{-n-1}} \tag{4.22}$$

and for the right half interval

$$x_{N-n} = \frac{\tilde{x}_{N-n} + \frac{2r_{-n-1}\tilde{x}_N}{r_0 - r_{-1}}}{r_n + r_{-n-1}}. \tag{4.23}$$

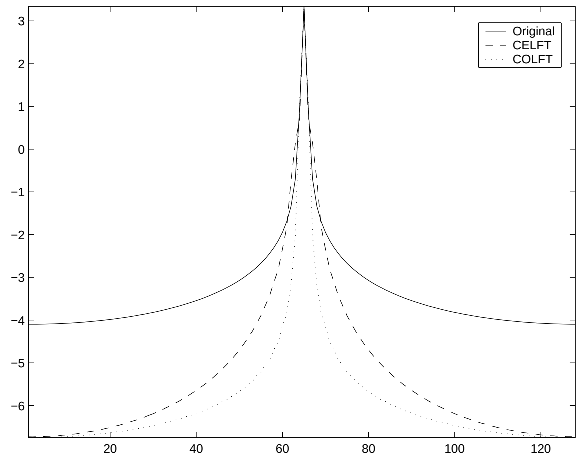

Figure 4.6: Spectrum of the functions in Figure 4.5.

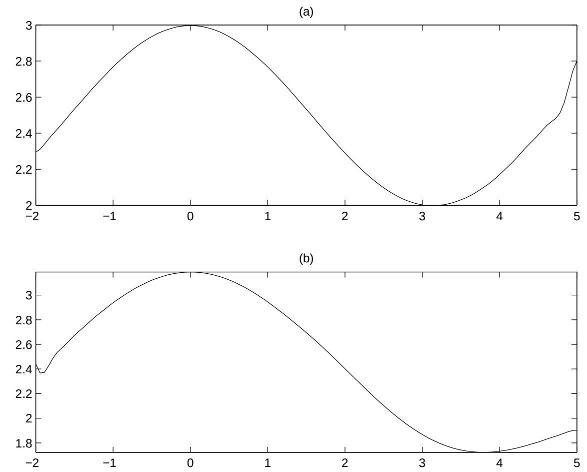

Figure 4.7: 4:1 compression using (a) CELFT and (b) COLFT.

For the isometry, the generalized isometric inversion formula for the left half interval is found using Equation (4.17) to be

$$x_n = \underbrace{\frac{\tilde{x}_{s,n} - \frac{2r_{-n-1}\tilde{x}_{s,0}}{r_0 + r_{-1}}}{r_n - r_{-n-1}}}_{z_n} + s\underbrace{\frac{\lambda_n}{r_n - r_{-n-1}}}_{\alpha_n}. \tag{4.24}$$

Using both z_n and α_n in Equation (4.12) will give the value of the isometric shift. So, once again, inversion is obtained by simply applying the standard unfolding operation of Equation (4.22) to $\tilde{x}_{s,n}$, and adding the term, $s\alpha_n$.

For simplicity, let the CBLFT based on this continuous odd folding scheme be called COLFT. Figure 4.5 shows the results of periodizing the function $x(t) = \frac{\cos(t)+1}{2} + 2$ for $-2 \leq t \leq 5$ using the COLFT and the CELFT with $r = r_{[1]}$. Clearly, the continuous odd extension does a much nicer job of periodizing the signal by eliminating the cusp. And looking at Figure 4.6, use of the continuous odd extension provides better frequency localization than use of the continuous even extension. As a matter of fact, the sparsity measure with ℓ^1-norm is 37.02 for the original, 36.17 for CELFT periodization and 33.26 for COLFT periodization. But when the signals are compressed, the COLFT doesn't perform better than CELFT as can be seen in Figure 4.7. Errors at both boundaries are enhanced in the COLFT due to the fact that *both* boundary values are divided by small numbers in the inversion formulas (see Equations (4.22) and (4.23)).

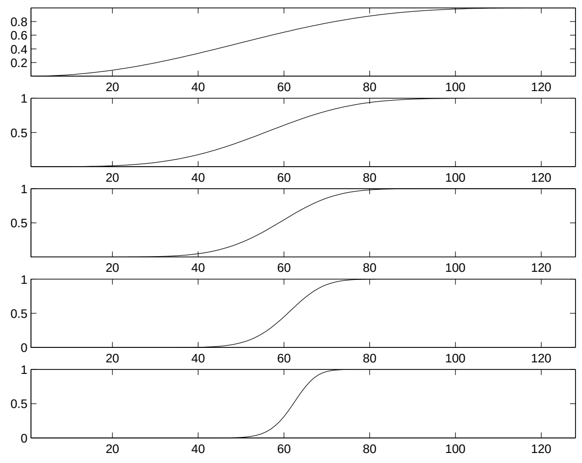

Figure 4.8: Rising cutoff functions of various steepness. From top to bottom, $m = 1$ to 5 using Equation (2.14).

Thus, higher stability could be achieved if the distance between r_n and r_{-n-1} could be increased when n is near 0. One simple method is to use a steeper

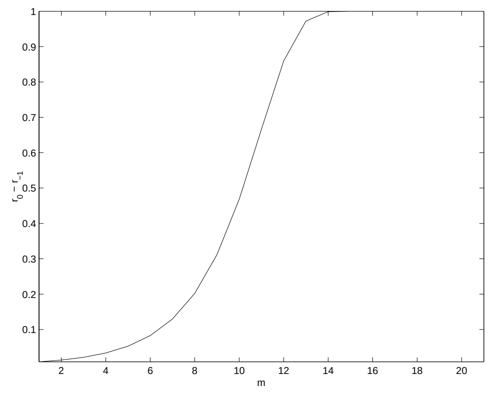

Figure 4.9: The value of $r_0 - r_{-1}$ for various m in Equation (2.14).

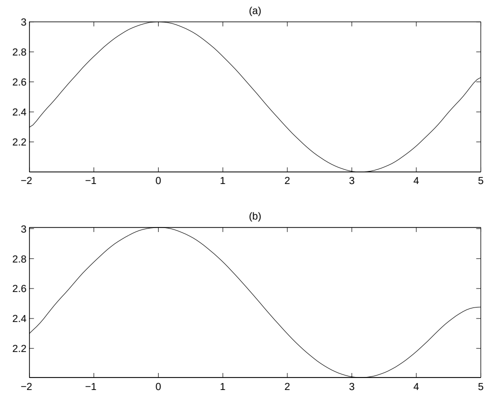

Figure 4.10: 4:1 compression using Equation (2.14) and (a) CELFT with $m = 3$, (b) COLFT with $m = 5$. Notice the improvement over the results in Figure 4.7

rising cutoff function (rcf); that is, to choose an rcf that has higher degrees of continuous derivatives near its boundaries. One possibility is to increase m in Equation (2.14). Some plots of Equation (2.14) with various values of m are shown in Figure 4.8. The corresponding distance between r_0 and r_{-1} are shown in Figure 4.9. Figure 4.10 shows the improvement in compression when using these different rcfs.

Another method is to multiply a taper function and the extension in order to increase the difference between r_n and r_{-n-1}. To do this, define a function $\tau(t) \in C^d(\mathbb{R})$ with $d \geq 0$ such that $\tau(0) = 1$, $\tau(1) = 0$ and $\tau'(0) = \tau'(1) =$

0. Simple examples are polynomials fitted with these boundary conditions, or functions similar to the rcfs of Equation (2.14):

$$\tau(t) \triangleq r_{[m+1]}(y) = r_{[m]}(\sin \frac{\pi}{2} y) \tag{4.25}$$

where

$$r_{[0]}(y) = \begin{cases} 1, & \text{if } y \leq -1, \\ \sin[\frac{\pi}{4}(1-y)], & \text{if } |y| < 1, \\ 0 & \text{if } y \geq 1 \end{cases} \tag{4.26}$$

and

$$y(t) = 2t - 1. \tag{4.27}$$

Mathematically, the folding procedure becomes

$$\begin{aligned} U(r,\tau,\epsilon)x(t) &\triangleq \begin{cases} r\left(\frac{t}{\epsilon}\right)x(t) + r\left(\frac{-t}{\epsilon}\right)[\tau(\frac{t}{\epsilon})x(-t)] & \text{if } 0 < t < \epsilon, \\ r\left(\frac{1-t}{\epsilon}\right)x(t) - r\left(\frac{t-1}{\epsilon}\right)[\tau(\frac{1-t}{\epsilon})x(2-t)] & \text{if } 1-\epsilon < t < 1, \\ x(t) & \text{otherwise} \end{cases} \\ &= \begin{cases} r\left(\frac{t}{\epsilon}\right)x(t) + \tilde{r}\left(\frac{-t}{\epsilon}\right)x(-t) & \text{if } 0 < t < \epsilon, \\ r\left(\frac{1-t}{\epsilon}\right)x(t) - \tilde{r}\left(\frac{t-1}{\epsilon}\right)x(2-t) & \text{if } 1-\epsilon < t < 1, \\ x(t) & \text{otherwise.} \end{cases} \end{aligned} \tag{4.28}$$

This is equivalent to using the standard folding function along with the following modified rising cutoff function

$$\tilde{r}(t) = \begin{cases} 0, & \text{if } t \leq -1, \\ \tau(-t)r(t), & \text{if } -1 < t \leq 0, \\ r(t), & \text{if } 0 < t < 1, \\ 1 & \text{if } t \geq 1. \end{cases} \tag{4.29}$$

A plot of $\tilde{r}$ with a certain τ is shown in Figure 4.11. Figure 4.12 illustrates the improvement in compression achieved when using a taper function along with $r = r_{[5]}$ and COLFT. The ℓ^2 error between the original signal and the reconstructed signal was 0.024 when using a taper function for this example, and 0.088 without one.

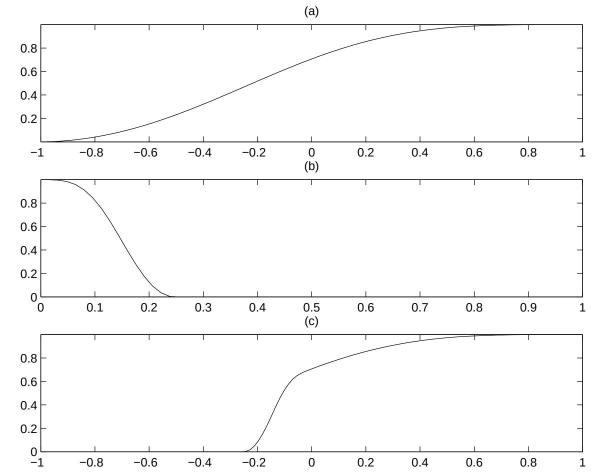

Figure 4.11: Example of how a taper function, τ, affects a rising cutoff function. (a) A typical RCF, (b) τ and (c) $\tilde{r}$.

4.4.1 Mixing of Information

Another issue is the amount of information that is mixed within each subspace because of CBLTT windowing. Recall Figure 3.9 which shows the sharp bump mixed within the subspace. This mixing can affect the frequency localization properties of the CBLFT. To better understand this, consider the continuous odd extension as an example. This type of extension can be split into two parts; that is, the sum of an odd extension and a piecewise constant function. Figure 4.13 illustrates the idea. To be more precise, let $I = [0, 1]$, let $r_{[m]}$ be the rising cutoff function defined by Equation 2.14, let $x(t) \in L^2(I)$ be a function with continuous odd extensions (top plot in Figure 4.13), let $f(t)$ be the same function with odd extension (middle plot in Figure 4.13), and let $g(t)$ be the piecewise constant function (bottom plot in Figure 4.13). That is, $x(t) = f(t) + g(t)$. Now, due to linearity of the periodization operator, $T_I x(t) = T_I f(t) + T_I g(t)$, evaluating each

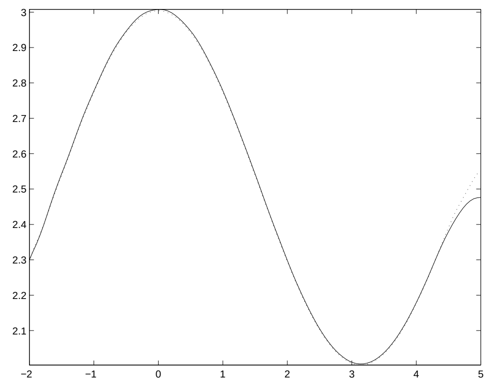

Figure 4.12: Results of 4:1 compression using a continuous odd extension along with the taper function, τ, shown in Figure 4.11(b). The solid line shows the results when no taper function was employed, and the dotted line shows the results when a taper function was used.

of these parts independently yields

$$\begin{aligned} U_I g(t) &= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) g(t) + r_{[m]}\left(\frac{-t}{\epsilon}\right) g(-t) & \text{if } 0 < t < \epsilon \\ g(t) & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) g(t) - r_{[m]}\left(\frac{t-1}{\epsilon}\right) g(2-t) & \text{if } 1-\epsilon < t < 1 \end{cases}, \\ &= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) (0) + r_{[m]}\left(\frac{-t}{\epsilon}\right) [2x(0)] & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) (0) - r_{[m]}\left(\frac{t-1}{\epsilon}\right) [2x(1)] & \text{if } 1-\epsilon < t < 1 \end{cases}, \\ &= \begin{cases} 2r_{[m]}\left(\frac{-t}{\epsilon}\right) x(0) & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ -2r_{[m]}\left(\frac{t-1}{\epsilon}\right) x(1) & \text{if } 1-\epsilon < t < 1 \end{cases}. \end{aligned} \tag{4.30}$$

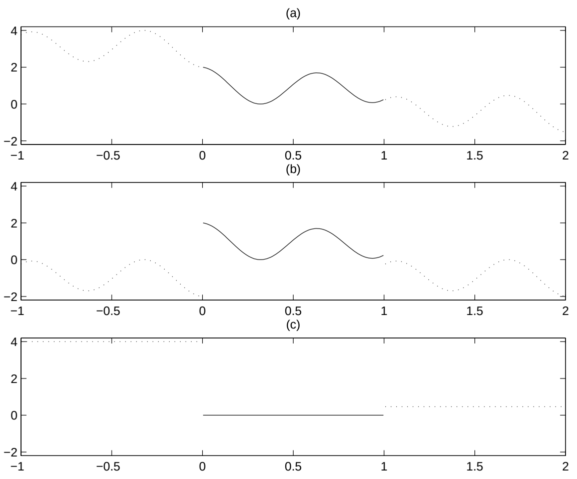

Figure 4.13: The continuous odd extension (top figure) split into the sum of an odd extension (middle figure) and a piecewise constant function (bottom figure).

Therefore,

$$
\begin{aligned}
T_I g(t) &= W_I^\star \mathbf{1}_I U_I g(t) \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) U_I g(t) - r_{[m]}\left(\frac{-t}{\epsilon}\right) U_I g(1-t) & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) U_I g(t) + r_{[m]}\left(\frac{t-1}{\epsilon}\right) U_I g(1-t) & \text{if } 1-\epsilon < t < 1 \end{cases}, \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right)\left[2r_{[m]}\left(\frac{-t}{\epsilon}\right) x(0)\right] & \\ \quad -r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[-2r_{[m]}\left(\frac{-t}{\epsilon}\right) x(1)\right] & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right)\left[-2r_{[m]}\left(\frac{t-1}{\epsilon}\right) x(1)\right] & \\ \quad +r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[2r_{[m]}\left(\frac{t-1}{\epsilon}\right) x(0)\right] & \text{if } 1-\epsilon < t < 1 \end{cases}, \\
&= \begin{cases} 2r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right) x(0) + r_{[m]}\left(\frac{-t}{\epsilon}\right) x(1)\right] & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ 2r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[r_{[m]}\left(\frac{t-1}{\epsilon}\right) x(0) - r_{[m]}\left(\frac{1-t}{\epsilon}\right) x(1)\right] & \text{if } 1-\epsilon < t < 1 \end{cases} \qquad (4.31)
\end{aligned}
$$

In a similar fashion,

$$
\begin{aligned}
U_I f(t) &= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) f(t) + r_{[m]}\left(\frac{-t}{\epsilon}\right) f(-t) & \text{if } 0 < t < \epsilon \\ f(t) & \text{if } \epsilon < t < 1-\epsilon \ , \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) f(t) - r_{[m]}\left(\frac{t-1}{\epsilon}\right) f(2-t) & \text{if } 1-\epsilon < t < 1 \end{cases} \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) x(t) + r_{[m]}\left(\frac{-t}{\epsilon}\right) [-x(t)] & \text{if } 0 < t < \epsilon \\ x(t) & \text{if } \epsilon < t < 1-\epsilon, \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) x(t) - r_{[m]}\left(\frac{t-1}{\epsilon}\right) [-x(t)] & \text{if } 1-\epsilon < t < 1 \end{cases} \\
&= \begin{cases} \left[r_{[m]}\left(\frac{t}{\epsilon}\right) - r_{[m]}\left(\frac{-t}{\epsilon}\right)\right] x(t) & \text{if } 0 < t < \epsilon \\ x(t) & \text{if } \epsilon < t < 1-\epsilon \ . \\ \left[r_{[m]}\left(\frac{1-t}{\epsilon}\right) + r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] x(t) & \text{if } 1-\epsilon < t < 1 \end{cases} \qquad (4.32)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
T_I f(t) &= W_I^{\star} \mathbf{1}_I U_I f(t) \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) U_I f(t) - r_{[m]}\left(\frac{-t}{\epsilon}\right) U_I f(1-t) & \text{if } 0 < t < \epsilon \\ f(t) & \text{if } \epsilon < t < 1-\epsilon \ , \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) U_I f(t) + r_{[m]}\left(\frac{t-1}{\epsilon}\right) U_I f(1-t) & \text{if } 1-\epsilon < t < 1 \end{cases} \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right) - r_{[m]}\left(\frac{-t}{\epsilon}\right)\right] x(t) & \\ \quad -r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right) + r_{[m]}\left(\frac{-t}{\epsilon}\right)\right] x(1-t) & \text{if } 0 < t < \epsilon \\ x(t) & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right) + r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] x(t) & \\ \quad +r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right) - r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] x(1-t) & \text{if } 1-\epsilon < t < 1 \end{cases} \qquad (4.33)
\end{aligned}
$$

Now to see the effects of this folding procedure, let $x(t) = e^{2\pi ibt}$ where $b > 0$. Then Equations (4.31) and (4.33) become

$$
\begin{aligned}
T_I g(t) &= \begin{cases} 2r_{[m]}\left(\frac{-t}{\epsilon}\right) r_{[m]}\left(\frac{t}{\epsilon}\right) + 2r_{[m]}^2(-t) e^{2\pi ib} & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ 2r_{[m]}^2(t-1) - 2r_{[m]}\left(\frac{t-1}{\epsilon}\right) r_{[m]}\left(\frac{1-t}{\epsilon}\right) e^{2\pi ib} & \text{if } 1-\epsilon < t < 1 \end{cases} \qquad (4.34) \\
&= a_1(t)
\end{aligned}
$$

and

$$\begin{aligned} T_I f(t) &= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right)-r_{[m]}\left(\frac{-t}{\epsilon}\right)\right]e^{2\pi ibt} \\ \quad -r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right)+r_{[m]}\left(\frac{-t}{\epsilon}\right)\right]e^{2\pi ib}e^{2\pi i(-b)t} & \text{if } 0<t<\epsilon \\ e^{2\pi ibt} & \text{if } \epsilon<t<1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right)+r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right]e^{2\pi ibt} \\ \quad +r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right)-r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right]e^{2\pi ib}e^{2\pi i(-b)t} & \text{if } 1-\epsilon<t<1 \end{cases} \\ &= a_2(t)e^{2\pi ibt}+a_3(t)e^{2\pi i(-b)t}. \end{aligned} \tag{4.35}$$

The energy of the coefficient function, $a_2(t)$, in front of $e^{2\pi ibt}$ in Equation (4.35) indicates the relative strength of the positive frequency components of $T_I x(t)$. Similarly, the energy of the coefficient function, $a_3(t)$, in front of $e^{2\pi i(-b)t}$ in Equation (4.35) indicates the relative strength of the negative frequency components of $T_I x(t)$. The function $a_1(t)$ is the part that actually performs the periodization by forcing the endpoints of $x(t)$ to meet in a continuous fashion; for example, checking the endpoints in Equations (4.34) and (4.35) shows that $T_I x(0) = T_I x(1) = 1$ for all m. $a_1(t)$ is analyzed in depth in the next chapter.

To see the effect that the rising cutoff function has on this process, take limits as $m \to \infty$ of Equations (4.34) and (4.35). To do this, first notice that every term in these two equations is a combination of terms $r_{[m]}\left(\frac{t}{\epsilon}\right)$ and $r_{[m]}\left(\frac{-t}{\epsilon}\right)$ on $0<t<\epsilon$ (note that $r_{[m]}\left(\frac{1-t}{\epsilon}\right)|_{t\in(1-\epsilon,1)} = r_{[m]}\left(\frac{t}{\epsilon}\right)|_{t\in(0,\epsilon)}$ and $r_{[m]}\left(\frac{t-1}{\epsilon}\right)|_{t\in(1-\epsilon,1)} = r_{[m]}\left(\frac{-t}{\epsilon}\right)|_{t\in(0,\epsilon)}$). So start by analyzing the following:

$$\begin{aligned} \lim_{m\to\infty} r_{[m]}(t) &= \lim_{m\to\infty} \sin\left[\frac{\pi}{4}\left(1+\underbrace{\sin\left(\frac{\pi}{2}\sin\left(\frac{\pi}{2}\left(\cdots\left(\sin\left(\frac{\pi}{2}t\right)\right)\right)\right)\right)}_{m \text{ times}}\right)\right] \\ &= \sin\left[\frac{\pi}{4}\left(1+\lim_{m\to\infty}\underbrace{\sin\left(\frac{\pi}{2}\sin\left(\frac{\pi}{2}\left(\cdots\left(\sin\left(\frac{\pi}{2}t\right)\right)\right)\right)\right)}_{*}\right)\right]. \end{aligned}$$

Now $(*)$ in the previous equation can be written as

$$t_{[m+1]} = \sin\left(\frac{\pi}{2}t_{[m]}\right) \tag{4.36}$$

where $t_{[0]} \in (0, \epsilon)$. To find its limit as $m \to \infty$, first note that $\sin\left(\frac{\pi}{2}t\right) \leq 1$. Second, let $h(t) = \sin\left(\frac{\pi}{2}t\right) - t$. Then $h(t)$ has three roots, namely $t = 0$, and $t = \pm 1$. Picking any point, t^* in the interval $(0, 1)$ reveals that $h(t^*) > 0$ which means that $h(t) > 0$ for all $t \in (0, 1)$. Thus, Equation (4.36) is monotonically increasing in the interval $(0, 1)$. Consequently, $\lim_{m\to\infty} t_{[m]} = 1$ for all $t_{[0]} \in (0, 1)$ since any monotonically increasing bounded function converges to its least upper bound. Therefore,

$$\lim_{m\to\infty} r_{[m]}\left(\frac{t}{\epsilon}\right) = \sin\left[\frac{\pi}{4}(1+1)\right] = 1 \text{ for all } t \in (0, \epsilon) \tag{4.37}$$

and

$$\lim_{m\to\infty} r_{[m]}\left(\frac{-t}{\epsilon}\right) = \sin\left[\frac{\pi}{4}(1-1)\right] = 0 \text{ for all } t \in (0, \epsilon). \tag{4.38}$$

Plugging Equations (4.37) and (4.38) into Equations (4.34) and (4.35) gives the following results

$$\begin{aligned} T_I g(t) &= \begin{cases} \underbrace{2r_{[m]}\left(\frac{-t}{\epsilon}\right) r_{[m]}\left(\frac{t}{\epsilon}\right)}_{\to 0 \text{ as } m\to\infty} + \underbrace{2r_{[m]}^2(-t)}_{\to 0 \text{ as } m\to\infty}\ e^{2\pi i b} & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ \underbrace{2r_{[m]}^2(t-1)}_{\to 0 \text{ as } m\to\infty} - \underbrace{2r_{[m]}\left(\frac{t-1}{\epsilon}\right) r_{[m]}\left(\frac{1-t}{\epsilon}\right) e^{2\pi i b}}_{\to 0 \text{ as } m\to\infty} & \text{if } 1-\epsilon < t < 1 \end{cases} \\ &= 0 \text{ as } m \to \infty \end{aligned} \tag{4.39}$$

and

$$
\begin{aligned}
T_I f(t) &= \begin{cases}
\underbrace{r_{[m]}\left(\frac{t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right) - r_{[m]}\left(\frac{-t}{\epsilon}\right)\right]}_{\to 1 \text{ as } m\to\infty} e^{2\pi ibt} \\
\quad - \underbrace{r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{t}{\epsilon}\right) + r_{[m]}\left(\frac{-t}{\epsilon}\right)\right] e^{2\pi ib}}_{\to 0 \text{ as } m\to\infty} e^{2\pi i(-b)t} & \text{if } 0 < t < \epsilon \\
e^{2\pi ibt} & \text{if } \epsilon < t < 1-\epsilon \\
\underbrace{r_{[m]}\left(\frac{1-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right) + r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right]}_{\to 1 \text{ as } m\to\infty} e^{2\pi ibt} \\
\quad + \underbrace{r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right) - r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] e^{2\pi ib}}_{\to 0 \text{ as } m\to\infty} e^{2\pi i(-b)t} & \text{if } 1-\epsilon < t < 1
\end{cases} \\
&= e^{2\pi ibt} \text{ as } m \to \infty. \qquad (4.40)
\end{aligned}
$$

As can be seen by these equations, folding with a continuous odd extension has the effect of adding an extra bump in frequency. But luckily, the amount of energy associated with the bump at $-b$ can be suppressed by increasing the steepness of the rising cutoff function. Table 4.1 shows the amounts of energy in $a_1(t)$, $a_2(t)$ and $a_3(t)$ associated with the six rising cutoff functions $r_{[m]}$ for $m = 0, \ldots, 5$. Thus, as m increases, energy becomes localized around the single positive frequency, b. It should be noted that the energy of $a_3(t)$ is much smaller than that of $a_1(t)$, indicating that the energy associated with the bump at $-b$ can be suppressed without compromising the effects of periodization. Of course, m should not be too large, or else the effects of periodization will be eliminated altogether; hence, there is a tradeoff in the choice of m. This result reinforces the use of steeper rising cutoff functions that was proposed in the last section. But, if possible, the amount of mixing should be eliminated all together. One solution is derived next.

Table 4.1: Energies of a_1, a_2 and a_3 associated with $r_{[m]}$ for $m = 0, \ldots, 5$.

m	$\|\|a_1\|\|^2$ for $r_{[m]}$	$\|\|a_2\|\|^2$ for $r_{[m]}$	$\|\|a_3\|\|^2$ for $r_{[m]}$
0	0.7267	0.8183	0.1817
1	0.4968	0.8758	0.1242
2	0.3271	0.9182	0.0818
3	0.2112	0.9472	0.0528
4	0.1352	0.9662	0.0338
5	0.0861	0.9785	0.0215

4.5 The Continuous Periodic Extension

The following scheme was devised to satisfy the conditions outlined in Section 3.3, focusing on the information mixing problem. The basic idea is to use a periodic extension of each subspace, while also forcing continuity at the boundary. Figure 4.14 illustrates the approach. In a similar manner as before, the right half of the signal is extended in a continuous periodic fashion, and folding is performed at the right hand boundary. The extension is discarded and the energy of the folded right half is computed to find how much energy has been lost or gained. This is then repeated for the opposite half, but with a temporary amplitude shift in both the signal and extension prior to folding. The value of the shift, which depends on the signal, can be precomputed and is used to preserve the isometry. Immediately afterwards, periodized unfolding is performed to periodize the signal on that interval, as well as undo any mixing which occurred during the folding procedure. In other words, a slightly modified version of periodized folding (W_s) is first applied which folds, or mixes, information between the left and right halves of the folded signal in a manner similar to regular periodized folding W. When periodized unfolding is then applied, W^*, it has the effect of reversing the mixing effect while periodizing the signal at the same time. The motivation for this idea comes from the fact that $W^*W = \mathbf{1}$; hence, $W^*W_s \approx \mathbf{1}$. Figure 4.15 shows the results of this periodization process. Notice that the continuous periodic extension successfully periodizes the two children subspaces, while also minimizing the

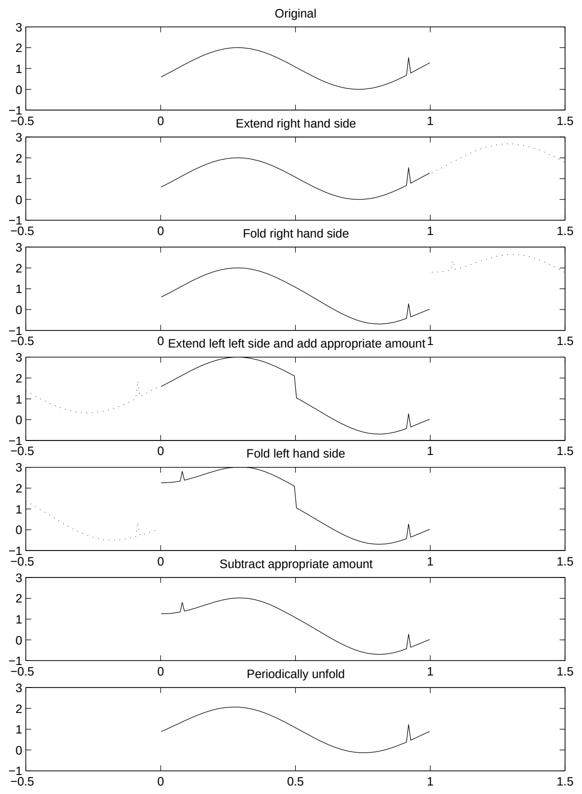

Figure 4.14: Isometric folding using a continuous periodic extension. Notice that both the left and right half intervals are extended in a continuous periodic manner, but the left half is temporarily shifted prior to folding in order to preserve the isometry.

mixing of information.

Mathematically, the approach for folding can be formulated as follows. Using the arrangement shown in Figure 4.16 the values of the extension to the left of x_0 are given by

$$x_{-n-1} = x_{N-n} + x_0 - x_N \tag{4.41}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} = x_n + x_N - x_0. \tag{4.42}$$

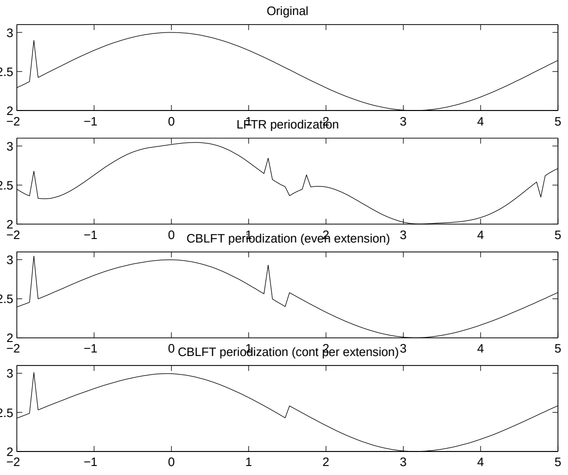

Figure 4.15: A comparison of various methods of periodization. The top figure shows the original function with a sharp bump at one location. The second figure shows the effects of splitting the space into two children via the conventional local Fourier transform. The third figure shows splitting via an even extension. The fourth figure shows the results of splitting the space using the continuous periodic extension.

Using Equation (2.12), folding at the left hand edge is then defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n-1} x_{-n-1} \\ &= r_n x_n + r_{-n-1}(x_{N-n} + x_0 - x_N) \qquad (4.43)\end{aligned}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1}(x_n + x_N - x_0). \qquad (4.44)\end{aligned}$$

Now, for the isometric version, the folding formula for the left half interval, $\tilde{x}_{s,n}$, can be found using Equation (4.11) along with the following values of $\tilde{x}_n$ and λ_n

$$\begin{aligned}\tilde{x}_{s,n} &= r_n(x_n + s) + r_{-n-1}(x_{-n-1} + s) - s \\ &= r_n x_n + r_{-n-1} x_{-n-1} + s(r_n + r_{-n-1} - 1) \\ &= \underbrace{r_n x_n + r_{-n-1}(x_{N-n} + x_0 - x_N)}_{\tilde{x}_n} + s\underbrace{(r_n + r_{-n-1} - 1)}_{\lambda_n}. \qquad (4.45)\end{aligned}$$

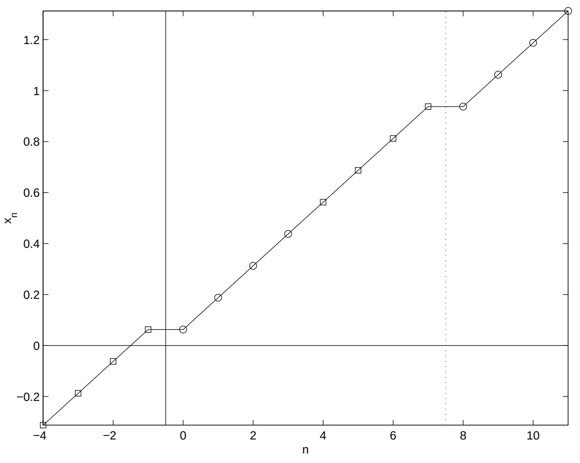

Figure 4.16: Example of a continuous periodic extension of a linear function with midpoint folding[72]. In this example, $N = 7$.

where $\tilde{x}_{s,n}$ stands for the folded result with shift. Thus, this is just the standard folding operation, with the added term, $s\lambda_n$.

Solving for x in Equations (4.43) and (4.44) yields the following formula for inverse folding on the left half interval

$$x_n = \left[\tilde{x}_n - r_{-n-1}(x_{N-n} + x_0 - x_N)\right]/r_n \tag{4.46}$$

and on the right half

$$x_{N-n} = \left[\tilde{x}_{N-n} + r_{-n-1}(x_n + x_N - x_0)\right]/r_n. \tag{4.47}$$

Now, similar to many of the methods described in Appendix B.4, this approach yields implicit inversion formulas. Represented as a linear system, it can be written as

$$\begin{bmatrix} r_0 + r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-2} & r_1 & 0 & 0 & 0 & 0 & r_{-2} & -r_{-2} \\ \downarrow & 0 & \searrow & 0 & 0 & \swarrow & 0 & \downarrow \\ r_{-\frac{N+1}{2}} & 0 & 0 & r_{\frac{N-1}{2}} & r_{-\frac{N+1}{2}} & 0 & 0 & -r_{-\frac{N+1}{2}} \\ r_{-\frac{N+1}{2}} & 0 & 0 & -r_{\frac{N+1}{2}} & r_{\frac{N-1}{2}} & 0 & 0 & -r_{-\frac{N+1}{2}} \\ \downarrow & 0 & \swarrow & 0 & 0 & \searrow & 0 & \downarrow \\ r_{-2} & -r_{-2} & 0 & 0 & 0 & 0 & r_1 & -r_{-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 - r_{-1} \end{bmatrix} x = \tilde{x}. \tag{4.48}$$

In this system, it is assumed that $R = \frac{N+1}{2}$, although this is not necessary.

In order to invert this, sparse matrix routines can be employed, but for computational speed it is better to exploit the structure of the array and find an explicit solution. To do this, it is necessary to recover x_0 and x_N (see Equations (4.43) and (4.44)). They are easily found from the first and last rows of Equation (4.48) to be

$$x_0 = \frac{\tilde{x}_0}{r_0 + r_{-1}} \tag{4.49}$$

and

$$x_N = \frac{\tilde{x}_N}{r_0 - r_{-1}}. \tag{4.50}$$

Using these results, and exploiting the symmetry in Equation (4.48), generalized inversion formulas for $x_1, \ldots, x_{N-1}$ are found by simultaneously solving a system of equations derived from rows n and $N-n$ for $n = 1, \ldots, R-2$ in Equation (4.48). These two equations have two unknowns, and so the unique solution is found to be

$$\begin{aligned} x_n &= [\tilde{x}_n - r_{-n-1}(x_{N-n} + x_0 - x_N)]/r_n \\ &= (\tilde{x}_n - r_{-n-1}\{[\tilde{x}_{N-n} + r_{-n-1}(x_n + x_N - x_0)]/r_n + x_0 - x_N\})/r_n \\ &= [r_n\tilde{x}_n - r_{-n-1}\tilde{x}_{N-n} + r_{-n-1}(r_{-n-1} - r_n)(x_0 - x_N)]/\underbrace{(r_n^2 + r_{-n-1}^2)}_{=1} \end{aligned} \tag{4.51}$$

and

$$\begin{aligned} x_{N-n} &= [\tilde{x}_{N-n} + r_{-n-1}(x_n + x_N - x_0)]/r_n \\ &= (\tilde{x}_{N-n} + r_{-n-1}\{[\tilde{x}_n - r_{-n-1}(x_{N-n} + x_0 - x_N)]/r_n + x_N - x_0\})/r_n \\ &= [r_n\tilde{x}_{N-n} + r_{-n-1}\tilde{x}_n - r_{-n-1}(r_{-n-1} + r_n)(x_0 - x_N)]/\underbrace{(r_n^2 + r_{-n-1}^2)}_{=1}. \end{aligned} \tag{4.52}$$

For the isometry, the generalized isometric inversion formula for the left half interval is found using Equation (4.45) to be

$$x_n = [\tilde{x}_{s,n} - r_{-n-1}(x_{N-n} + x_0 - x_N)]/r_n - s\lambda_n/r_n. \tag{4.53}$$

Adding this to Equations (4.49) , (4.50) (4.51) and (4.52) yields the following isometric inversion formulas

$$x_0 = \underbrace{\frac{\tilde{x}_{s,0}}{r_0 + r_{-1}}}_{z_0} - s\,\underbrace{\frac{\lambda_0}{r_0 + r_{-1}}}_{\alpha_0} \tag{4.54}$$

$$\begin{aligned} x_n &= \underbrace{r_n \tilde{x}_{s,n} - r_{-n-1}\tilde{x}_{s,N-n} + r_{-n-1}(r_{-n-1} - r_n)\left(\frac{\tilde{x}_{s,0}}{r_0 + r_{-1}} - \frac{\tilde{x}_{s,N}}{r_0 - r_{-1}}\right)}_{z_n} \\ &\quad -s\underbrace{\left(r_n\lambda_n + r_{-n-1}(r_{-n-1} - r_n)\frac{\lambda_0}{r_0 + r_{-1}}\right)}_{\alpha_n} \end{aligned} \tag{4.55}$$

and

$$\begin{aligned} x_{N-n} &= \underbrace{r_n \tilde{x}_{s,N-n} + r_{-n-1}\tilde{x}_{s,n} - r_{-n-1}(r_{-n-1} + r_n)\left(\frac{\tilde{x}_{s,0}}{r_0 + r_{-1}} - \frac{\tilde{x}_{s,N}}{r_0 - r_{-1}}\right)}_{z_{N-n}} \\ &\quad -s\underbrace{\left[r_{-n-1}\left(\lambda_n + (r_{-n-1} - r_n)\frac{\lambda_0}{r_0 + r_{-1}}\right)\right]}_{\beta_n} \end{aligned} \tag{4.56}$$

for $n = 1, \ldots, R-1$. Notice that this is just the standard inversion operation of Equations (4.51) and (4.52) applied to $\tilde{x}_{s,n}$, with the added terms α_n and β_n (also note that $\beta_0 = 0$ since Equation (4.50) is unaffected by the constant s).

Now that the inversion formulas are well defined, they too need to be used in a formula for finding an explicit shift value, s. Using the above notation along with the definition of an isometry yields a formula similar to Equation (4.12)

$$\begin{aligned} s &= \frac{\sum_{n=0}^{R-1}(\alpha_n z_n + \beta_n z_{N-n})}{\sum_{n=0}^{R-1}(\alpha_n^2 + \beta_n^2)} \\ &\quad \pm \frac{\sqrt{\left(\sum_{n=0}^{R-1}(\alpha_n z_n + \beta_n z_{N-n})\right)^2 - \sum_{n=0}^{R-1}(\alpha_n^2+\beta_n^2)\cdot\sum_{n=0}^{R-1}\left(z_n^2 + z_{N-n}^2 - \tilde{x}_{s,n}^2 - \tilde{x}_{s,N-n}^2\right)}}{\sum_{n=0}^{R-1}(\alpha_n^2+\beta_n^2)}. \end{aligned} \tag{4.57}$$

As shown in the analysis following Equation (4.12), most of the values in Equation (4.57) are constants that can be precomputed only once; hence, there is little computational overhead involved with this isometric operator (it is an $O(N)$ operation).

4.5.1 Mixing of Information

One of the primary motivations for the continuous periodic extension is minimizing the mixing of information within each interval due to folding and unfolding. To see the benefit achieved with this extension, simply repeat the analysis performed in Section 4.4.1, but for this new extension.

Start by splitting the extension into two parts, the sum of a periodic extension and a piecewise constant function. Figure 4.17 illustrates this. As before, let

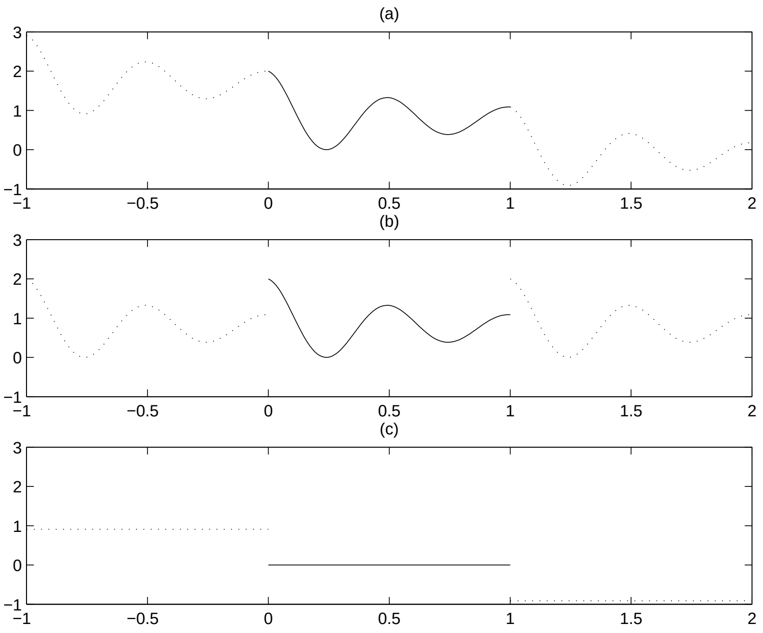

Figure 4.17: (a) The continuous periodic extension split into the sum of (b) a periodic extension and (c) a piecewise constant function.

$I = [0, 1]$, let $r_{[m]}$ be the rising cutoff function defined by Equation 2.14, let $x(t) \in L^2(I)$ be a function with continuous periodic extensions (top plot in Figure 4.17), let $f(t)$ be the same function with periodic extension (middle plot in Figure 4.17), and let $g(t)$ be the piecewise constant function (bottom plot in Figure 4.17). That is, $x(t) = f(t) + g(t)$. Also, define $\Delta x \triangleq x(1) - x(0)$. Now, since $T_I x(t) =$

$T_I f(t) + T_I g(t)$ then each of the parts can be analyzed independently

$$\begin{aligned}
U_I g(t) &= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) g(t) + r_{[m]}\left(\frac{-t}{\epsilon}\right) g(-t) & \text{if } 0 < t < \epsilon \\ g(t) & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) g(t) - r_{[m]}\left(\frac{t-1}{\epsilon}\right) g(2-t) & \text{if } 1-\epsilon < t < 1 \end{cases}, \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right)(0) + r_{[m]}\left(\frac{-t}{\epsilon}\right)[-\Delta x] & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right)(0) - r_{[m]}\left(\frac{t-1}{\epsilon}\right)[\Delta x] & \text{if } 1-\epsilon < t < 1 \end{cases}, \\
&= \begin{cases} -\Delta x r_{[m]}\left(\frac{-t}{\epsilon}\right) & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ -\Delta x r_{[m]}\left(\frac{t-1}{\epsilon}\right) & \text{if } 1-\epsilon < t < 1 \end{cases}. \qquad (4.58)
\end{aligned}$$

Therefore,

$$\begin{aligned}
T_I g(t) &= W_I^\star \mathbf{1}_I U_I g(t) \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right) U_I g(t) - r_{[m]}\left(\frac{-t}{\epsilon}\right) U_I g(1-t) & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right) U_I g(t) + r_{[m]}\left(\frac{t-1}{\epsilon}\right) U_I g(1-t) & \text{if } 1-\epsilon < t < 1 \end{cases}, \\
&= \begin{cases} r_{[m]}\left(\frac{t}{\epsilon}\right)\left[-\Delta x r_{[m]}\left(\frac{-t}{\epsilon}\right)\right] - r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[-\Delta x r_{[m]}\left(\frac{-t}{\epsilon}\right)\right] & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ r_{[m]}\left(\frac{1-t}{\epsilon}\right)\left[-\Delta x r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] + r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[-\Delta x r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] & \text{if } 1-\epsilon < t < 1 \end{cases}, \\
&= \begin{cases} \Delta x r_{[m]}\left(\frac{-t}{\epsilon}\right)\left[r_{[m]}\left(\frac{-t}{\epsilon}\right) - r_{[m]}\left(\frac{t}{\epsilon}\right)\right] & \text{if } 0 < t < \epsilon \\ 0 & \text{if } \epsilon < t < 1-\epsilon \\ -\Delta x r_{[m]}\left(\frac{t-1}{\epsilon}\right)\left[r_{[m]}\left(\frac{1-t}{\epsilon}\right) + r_{[m]}\left(\frac{t-1}{\epsilon}\right)\right] & \text{if } 1-\epsilon < t < 1 \end{cases}. \qquad (4.59)
\end{aligned}$$

As for the other term, $T_I f(t) = W_I W_I^* f(t) = f(t)$. Thus one of the immediate benefits of using the continuous periodic extension is realized; unlike the continuous odd extension, no extra bump in frequency will be introduced when using the continuous periodic extension.

This result also illustrates the manner in which the CBLFT periodizes a signal. In essence, CPLFT simply adds a scalar multiple of the function $h(t) = T_I g(t)$ to the input signal $x(t)$. A plot of this function is shown in Figure 4.18. Understanding the effects of $h(t)$ is the subject of the next section.

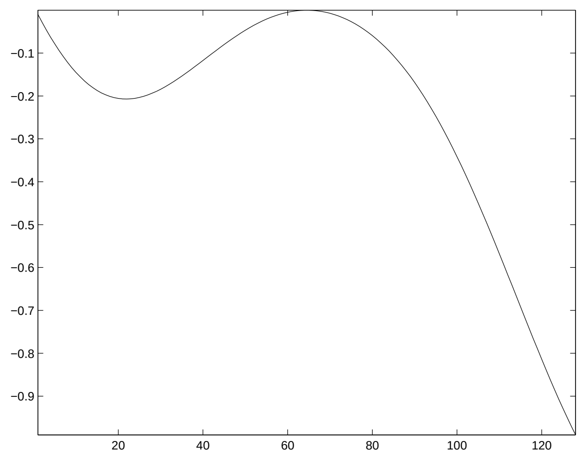

Figure 4.18: The function $h(t) = T_I g(t)$ from Equation (4.59) which periodizes a signal using a continuous periodic extension.

4.5.2 Effects of Continuous Periodization on the Fourier Coefficient Decay Rate

As defined in the previous section, the addition of a scalar multiple of $h(t) = T_I g(t)$ to a function has the effect of periodizing that function. The manner in which it achieves this can be illustrated by simply taking the Fourier transform of $h(t)$. Although similar analysis for the COLFT is possible, the fact that the periodization process for the COLFT cannot be split nicely into two parts like the CPLFT makes Fourier analysis too cumbersome to be included in this thesis. Consequently, the following analysis is only performed for the CPLFT case.

For illustrative simplicity, let $r(t) = r_{[0]}(t) = \sin\left[\frac{\pi}{4}(1+t)\right]$ and $x(t) = t^n$. If $F_b(x)$, $b \in \mathbb{Z}$, is the b^{th} Fourier coefficient of a function x, then $F_b[T_I x(t)] = F_b[f(t)] + \Delta x F_b[h(t)]$ where

$$\begin{aligned} F_0[h(t)] &= -2\epsilon \int_0^1 r(t)r(-t)dt \\ &= -2\epsilon \int_0^1 \cos\left(\frac{\pi}{2}t\right) dt \\ &= -\frac{4\epsilon}{\pi} \end{aligned} \tag{4.60}$$

and

$$\begin{aligned} F_b[h(t)] &= \int_0^1 h(t)e^{-2\pi ibt}dt \\ &= \frac{2\epsilon\cos(2\pi b\epsilon)}{\pi(16b^2\epsilon^2-1)} - \frac{i}{2\pi b}\left[1+\frac{\cos(2\pi b\epsilon)}{b(16b^2\epsilon^2-1)}\right]. \end{aligned} \tag{4.61}$$

Repeating this for $f(t) = t^n$ gives

$$\begin{aligned} F_{b,n}[f(t)] &= \int_0^1 t^n e^{-2\pi ibt}dt \\ &= -\frac{e^{-2\pi ibt}}{2\pi ib}t^n\bigg|_0^1 + \frac{n}{2\pi ib}\int_0^1 t^{n-1}e^{-2\pi ibt}dt \\ &= \frac{1}{2\pi ib}\left\{nF_{b,n-1}[f(t)]-1\right\} \\ &= \frac{n}{2\pi ib}\left(\frac{n-1}{2\pi ib}\left(\frac{n-2}{2\pi ib}\left(\dots\left(\frac{F_{b,0}[f(t)]}{2\pi ib}\right.\right.\right.\right. \\ &\qquad \left.\left.\left.\left.-\frac{1}{2\pi ib}\right)\dots\right)-\frac{1}{2\pi ib}\right)-\frac{1}{2\pi ib}\right)-\frac{1}{2\pi ib} \\ &= -\frac{n!}{1!(2\pi ib)^n}-\frac{n!}{2!(2\pi ib)^{n-1}}-\dots-\frac{n!}{(n-1)!(2\pi ib)^2}-\frac{1}{2\pi ib} \\ &= -\sum_{k=1}^{n}\frac{n!}{(n-k+1)!(2\pi ib)^k} \\ &= i\sum_{k=0}^{\lfloor\frac{n}{2}\rfloor}\frac{(-1)^k n!}{(n-2k)!(2\pi b)^{2k+1}} - \sum_{k=1}^{\lfloor\frac{n}{2}\rfloor}\frac{(-1)^k n!}{(n-2k+1)!(2\pi b)^{2k}} \end{aligned} \tag{4.62}$$

where $F_{b,0}[f(t)] = 0$ for $b \neq 0$. As can be seen, for any n, the dominant factor in Equation (4.62) is $\frac{i}{2\pi b} = O(\frac{1}{b})$. Similarly, the dominant factor in Equation (4.61) is $-\frac{i}{2\pi b}$. Therefore, these two terms will cancel each other out causing $F_{b,n}[T_I x(t)] = O\left[\left(\frac{1}{b}\right)^2\right]$. The details of Equation (4.61) can be found in Appendix B.3

A side note that arises from this example is the fact that a linear function can also be used to periodize a signal. For example, if $h(t) = t$, then according to Equation (4.62), its Fourier decay is dominated by the factor $\frac{i}{2\pi b}$. So this too would cancel out the dominating term for $F_{b,n}[f(t)]$ causing $F_{b,n}[T_I x(t)] =$

$O\left[\left(\frac{1}{b}\right)^2\right]$. But there are some drawbacks to this approach. First is the fact that this linear method lacks the flexibility of the varying action region widths associated with the CBLTT; it cannot be implemented in a method similar to fixed folding. Second is the fact that it lacks the diversity of all of the different extensions of CBLTT, including all of the different rising cutoff functions. Continuity at the boundary after application is always limited to C^0, whereas CBLTT has the potential of increasing the smoothness at the boundary after folding by using different extensions and different rising cutoff functions. Third is the fact that it also cannot be used to provide LCT and LST boundary conditions. But most important is the fact that the inverse is highly unstable. For example, the transformed values are given by $\tilde{x}_n = x_n - (x_N - x_0)\frac{n}{N+1}$ and the inverse is $x_n = \tilde{x}_n + [(N+1)\tilde{x}_N - (n+1)\tilde{x}_0]\frac{n}{N+1}$. So if errors, e_n, are introduced into the $\tilde{x}_n$, for example by compression, letting $\tilde{y}_n = \tilde{x}_n + e_n$ gives the following

$$\begin{aligned} y_n &= \tilde{y}_n + [(N+1)\tilde{y}_N - (n+1)\tilde{y}_0]\frac{n}{N+1} \\ &= \tilde{x}_n + e_n + [(N+1)(\tilde{x}_N + e_N) - (n+1)\tilde{x}_0]\frac{n}{N+1} \\ &= \tilde{x}_n + [(N+1)\tilde{x}_N - (n+1)\tilde{x}_0]\frac{n}{N+1} + e_n + ne_N \\ &= x_n + e_n + ne_N. \end{aligned} \tag{4.63}$$

Hence, as n increases, the error also increases. As will be seen in the following chapter, this instability can be decreased when using CBLTT.

It should be noted that numerous other approaches to reduce edge effect for the Fourier transform have been devised in the past; for example, the Blackman-Harris window [33], the multitaper method ([58], p.331-374), and the Webber window[1][71]. The basic idea behind these methods is that they simply multiply the signal by a window which has enough decay at the boundary to cause the windowed signal to have boundary values near zero. In doing this, the windowed

[1]The image amplitude of a NxN pixel square is rolled off softly to zero over a range of $\frac{1}{10}N$ pixels, by modulation with a circular window boundary of diameter roughly $\frac{9}{10}N$ pixels. As a function of the 2D pixel coordinate, $\mathbf{i}$, the modulation function is $\mathrm{erf}\left[\frac{2}{10}\left(\frac{9}{10}N - 2|\mathbf{i}|\right)\right]$.

signal can become periodic with C^0 continuity. Caveats to this approach are the inability to stably invert this transform, as well as the loss of information near the boundary. This loss of boundary information is especially profound in the two dimensional case near the diagonals (see Figure 4.19). In addition to this, there also is the fact that these approaches are not isometric and therefore cannot be used in a best basis setting.

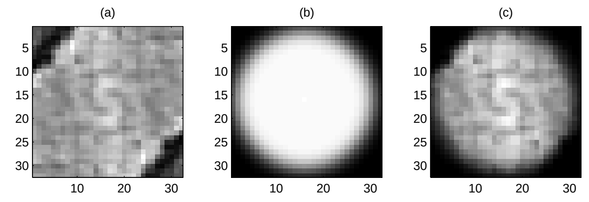

Figure 4.19: (a) Original 2D signal (b) 2D Webber window, (b) Application of the 2D Webber window. Notice the loss of information at the boundaries, especially near the diagonals.

4.6 Existence of a Real-Valued Constant for Isometric Folding and Unfolding

In light of these isometric formulas (Equations (4.11), (4.12) and (4.57)), it is important to know whether a real-valued constant, s, always exists. The answer is no, even for nonnegative functions. To illustrate this, take Equation (4.11). A simple counterexample is $x = (1, 0, \ldots, 0)$ where $N = 15$ and $R = 8$. The problem arises because the right half interval extension is nonpositive, causing the energy of the folded right half to *increase*; that is, $\sum_{n=0}^{R-1} \left(\tilde{x}_{N-n}^2 - x_{N-n}^2\right) \geq 0$. In response to this result, it is natural to wonder whether it is possible to place a condition on x to cause Equation (4.11) to always be satisfied with a real-valued s. The following theorem answers this question.

Theorem 4.6.1. *For all real-valued functions $x \in [0,1]$ and rising cutoff functions* (2.14)*, there exists a value $C \in \mathbb{R}$ such that for $y = x + C$ and $C \geq 0$, a solution to Equation* (4.11) *can always be found for some $s = s(y) \in \mathbb{R}$.*

Proof. (*i*) $\sum_{n=0}^{R} \lambda_n^2 \neq 0$; i.e., the denominator of Equation (4.11) is nonzero. For this to be true, the following equivalent relationships need to be satisfied.

$$\begin{array}{crcll}
 & \sum_{n=0}^{R-1} \lambda_n^2 & \neq & 0 & \\
\Leftrightarrow & \lambda_n & \neq & 0 & \text{for some } n \in [0, R-1] \\
\Leftrightarrow & r_n + r_{N-n} & \neq & 1 & \text{for some } n \in [0, R-1] \\
\Leftrightarrow & r_n + \sqrt{1-r_n^2} & \neq & 1 & \text{for some } n \in [0, R-1] \\
\Leftrightarrow & \sqrt{1-r_n^2} & \neq & 1 - r_n & \text{for some } n \in [0, R-1] \\
\Leftrightarrow & 1-r_n^2 & \neq & 1 - 2r_n + r_n^2 & \text{for some } n \in [0, R-1] \\
\Leftrightarrow & 2r_n^2 - 2r_n & \neq & 0 & \text{for some } n \in [0, R-1] \\
\Leftrightarrow & r_n(r_n - 1) & \neq & 0 & \text{for some } n \in [0, R-1](\star)
\end{array}$$

Now, using the fact that $\frac{1}{\sqrt{2}} < r_n < 1$,$\forall\ n \in [0, R-1]$ (recall that $r_0 > \frac{1}{\sqrt{2}}$ for midpoint folding), then $r_n(r_n - 1) < 0$, $\forall\ n \in [0, R-1]$ and $(\star)$ is satisfied;

(*ii*) $\left(\sum_{n=0}^{R-1} \tilde{y}_n \lambda_n\right)^2 - \sum_{n=0}^{R-1} \lambda_n^2 \cdot \sum_{n=0}^{R-1} \left(\tilde{y}_n^2 + \tilde{y}_{N-n}^2 - y_n^2 - y_{N-n}^2\right) \geq 0$ $(*)$; i.e., the discriminant in Equation (4.11) is nonnegative.

Without loss of generality, assume that cosine polarity folding is used. Then, as stated above, the main problem arises if the energy of the right half interval increases after folding, instead of decreasing. But there is another way for Equation (4.11) to fail; even if the energy of the right half interval decreases with folding, if the energy of the left half interval increases too much, then it may not be possible to amplitude shift the left half enough to cause the pre-folded and post-folded signal to be isometric. So it is necessary to find out if $\sum_{n=0}^{R-1}(\tilde{y}_{N-n}^2 - y_{N-n}^2) = 0$ and $\sum_{n=0}^{R-1}(\tilde{y}_n^2 - y_n^2) = 0$. If these can both be satisfied for some D, then choosing $C \geq D$ will cause $\sum_{n=0}^{R-1} \left(\tilde{y}_n^2 + \tilde{y}_{N-n}^2 - y_n^2 - y_{N-n}^2\right) \leq 0$, satisfying $(*)$.

Starting with the right half, notice that $\sum_{n=0}^{R-1} \left(\tilde{y}_{N-n}^2 - y_{N-n}^2\right)$ will be maximized when y_{N-n}^2 is a minimum, and the right hand extension is also minimized. This occurs when $y_0 = D + 1, y_n = D$ for $n = 1, \ldots, R-1$, and $y_{N-n} = D$ for

$n = 0, \ldots, R-1$, where $D \geq 0$ and $D \in \mathbb{R}$. Using this yields

$$\begin{aligned}
0 &= \sum_{n=0}^{R-1} \left(\tilde{y}_{N-n}^2 - y_{N-n}^2\right) \\
\Leftrightarrow 0 &= \sum_{n=0}^{R-1} \left[r_n y_{N-n} - r_{-n-1}(y_n + y_N - y_0)\right]^2 - \sum_{n=0}^{R-1} y_{N-n}^2 \\
\Leftrightarrow 0 &= D^2(r_0 - r_{-1})^2 + \sum_{n=1}^{R-1} \left[D r_n - (D-1) r_{-n-1}\right]^2 - D^2 R \\
\Leftrightarrow 0 &= D^2 \sum_{n=0}^{R-1} \left[(r_n - r_{-n-1})^2 - 1\right] + 2D \sum_{n=1}^{R-1} \left[r_{-n-1}(r_n - r_{-n-1})\right] + \sum_{n=1}^{R-1} r_{-n-1}^2 \\
\Leftrightarrow D &= \frac{\sum_{n=1}^{R-1} r_{-n-1}(r_n - r_{-n-1})}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} \pm \\
&\quad \frac{\sqrt{\left[\sum_{n=1}^{R-1} r_{-n-1}(r_n - r_{-n-1})\right]^2 + 2\sum_{n=0}^{R-1} r_n r_{-n-1} \cdot \sum_{n=1}^{R-1} r_{-n-1}^2}}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} \\
&= \frac{\sum_{n=1}^{R-1} r_{-n-1}(r_n - r_{-n-1})}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} \pm \\
&\quad \frac{\sqrt{\left(\sum_{n=1}^{R-1} r_n r_{-n-1}\right)^2 + \left(\sum_{n=1}^{R-1} r_{-n-1}^2\right)^2 + 2 r_0 r_{-1} \cdot \sum_{n=1}^{R-1} r_{-n-1}^2}}{2\sum_{n=0}^{R-1} r_n r_{-n-1}}. \qquad (4.64)
\end{aligned}$$

Since the denominator and everything within the square root of Equation (4.64) is positive, then a real value for D exists. An upper bound for D can be found by rearranging Equation (4.64)

$$\begin{aligned}
D &\leq \frac{1}{2}\left\{1 - \frac{\sum_{n=1}^{R-1} r_{-n-1}^2}{\sum_{n=0}^{R-1} r_n r_{-n-1}} + \sqrt{\left(1 - \frac{\sum_{n=1}^{R-1} r_{-n-1}^2}{\sum_{n=0}^{R-1} r_n r_{-n-1}}\right)^2 + 2\frac{\sum_{n=1}^{R-1} r_{-n-1}^2}{\sum_{n=0}^{R-1} r_n r_{-n-1}}}\right\} \\
&= \frac{1 - A + \sqrt{1 + A^2}}{2} \qquad (4.65)
\end{aligned}$$

where

$$A = \frac{\sum_{n=1}^{R-1} r_{-n-1}^2}{\sum_{n=0}^{R-1} r_n r_{-n-1}}. \qquad (4.66)$$

Now, since $r_n > r_{-n-1} > 0$ for all $n \in [0, R-1]$, then $0 \leq A \leq 1$. This implies that $D \leq 1$. Thus, for any $x \in [0,1]$, and for all values of $C \geq 1$, $\sum_{n=0}^{R-1} \left(\tilde{y}_{N-n}^2 - y_{N-n}^2\right) < 0$.

In a similar manner, the above argument can be repeated for the left half interval, but using the function that maximizes $\sum_{n=0}^{R-1} (\tilde{y}_n^2 - y_n^2)$, namely $y_n = D+1$ for $n = 0, \ldots, R-1$, $y_{N-n} = D+1$ for $n = 1, \ldots, R-1$, and $y_N = D$. The results are

$$\begin{aligned}
D &= -\frac{\sum_{n=1}^{R-1} r_{-n-1}(r_n + r_{-n-1})}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} \pm \\
&\quad \frac{\sqrt{\left[\sum_{n=1}^{R-1} r_{-n-1}(r_n + r_{-n-1})\right]^2 - 2\sum_{n=0}^{R-1} r_n r_{-n-1} \cdot \sum_{n=1}^{R-1} r_{-n-1}^2}}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} - 1 \\
&= -\frac{\sum_{n=1}^{R-1} r_{-n-1}(r_n + r_{-n-1})}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} \pm \\
&\quad \frac{\sqrt{\left(\sum_{n=1}^{R-1} r_n r_{-n-1}\right)^2 + \left(\sum_{n=1}^{R-1} r_{-n-1}^2\right)^2 - 2r_0 r_{-1} \cdot \sum_{n=1}^{R-1} r_{-n-1}^2}}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} - 1. \qquad (4.67)
\end{aligned}$$

As before, the denominator is positive. And since the quantity within the square root of Equation (4.67) is positive because of the following relationship

$$\begin{aligned}
&\left(\sum_{n=1}^{R-1} r_n r_{-n-1}\right)^2 + \left(\sum_{n=1}^{R-1} r_{-n-1}^2\right)^2 - 2r_0 r_{-1} \cdot \sum_{n=1}^{R-1} r_{-n-1}^2 \\
&\geq \left(\sum_{n=1}^{R-1} r_n r_{-n-1}\right)^2 + \left(\sum_{n=1}^{R-1} r_{-n-1}^2\right)^2 - 2\sum_{n=1}^{R-1} r_n r_{-n-1} \cdot \sum_{n=1}^{R-1} r_{-n-1}^2 \\
&= \left(\sum_{n=1}^{R-1} r_n r_{-n-1} - \sum_{n=1}^{R-1} r_{-n-1}^2\right)^2 \\
&\geq 0 \qquad (4.68)
\end{aligned}$$

then a real value for D exists. Using this result in Equation (4.67) yields a lower

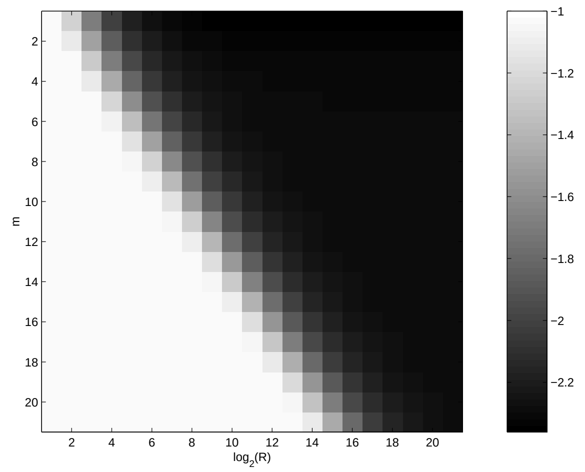

Figure 4.20: Values of D found using Equation (4.67) with $2^0 \leq R \leq 2^{20}$ and $0 \leq m \leq 20$.

bound for the larger of the two roots

$$\begin{aligned} D &\geq -\frac{\sum_{n=1}^{R-1} r_{-n-1}(r_n + r_{-n-1}) + \sqrt{\left(\sum_{n=1}^{R-1} r_n r_{-n-1} - \sum_{n=1}^{R-1} r_{-n-1}^2\right)^2}}{2\sum_{n=0}^{R-1} r_n r_{-n-1}} - 1 \\ &= -\frac{\sum_{n=1}^{R-1} r_{-n-1}^2}{\sum_{n=0}^{R-1} r_n r_{-n-1}} - 1 \\ &= -A - 1. \end{aligned} \tag{4.69}$$

Thus the larger of the two roots can be bounded below by $D \geq -2$. What this means is that the left half can always be shifted so that its energy before and after folding can be the same.

Thus, as long as $C > 1$, then $s = s(y)$ will always exist. And the lowest value that the left half will ever be shifted is -2. $\square$

In practice, different bounds on D can be found. For instance, some values of D computed from Equations (4.64) and (4.67) are shown in Figures 4.21 and 4.20 respectively. As can be seen in Figure 4.21, as R and m increase, D converges to a number close to 0.8144. And according to Figure 4.20, D converges to a

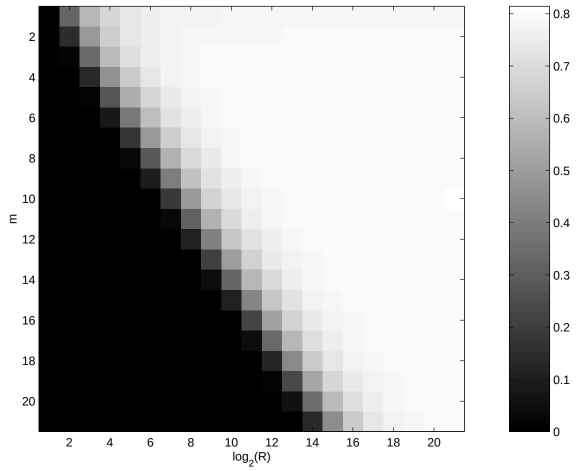

Figure 4.21: Values of D found using Equation (4.64) with $2^0 \leq R \leq 2^{20}$ and $0 \leq m \leq 20$.

value close to -1.2097. Since these values are for large R and m, then choosing $C > 0.8144$ will be adequate for most practical purposes.

Chapter 5

Basis and Scaling

To better understand the dynamics of the CBLTT, and to design extensions that give better results, it would be nice to be able to physically construct the bases. But, as was shown in Section 4.2, these isometric folding and unfolding operators are not linear. This makes analysis difficult. The complexity of the isometric operators, and the fact that a real-valued amplitude shift does not always exist without pretreating the data by the addition of a constant, motivates a different approach.

5.1 Isometry and Scaling

In designing a new isometric folding approach, the biggest constraint, besides minimizing the complexity of the inversion formulas, is maintaining the odd polarity boundary condition after folding. Recall that for cosine folding, the right hand boundary is the case in point. After folding occurs, and the extension is discarded, no operation should be applied that moves this boundary off of zero. One possible way to circumvent this problem while still providing the isometry is to globally *scale* the data instead of adding a constant to it. That is,

- Extend both sides of each subspace in a continuous fashion,
- Perform standard folding at each boundary,
- Discard the extensions,
- Globally scale the result by a value, p, to preserve the isometry,
- Periodize (if specified).

See Figure 5.1.

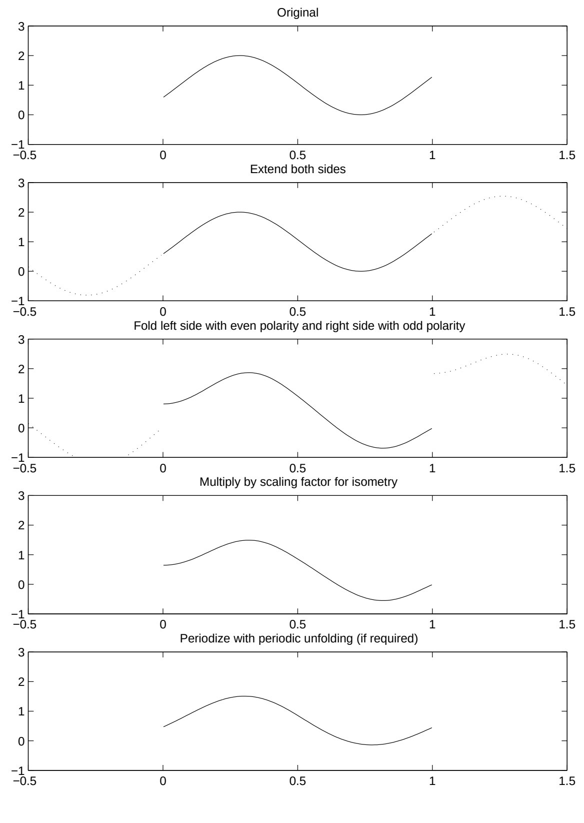

Figure 5.1: Isometric folding via *scaling*.

One of the immediate benefits of this new approach is the simplification of the isometric operators. Instead of *adding* the constant into the equations, all that must be done is to take the original signal, x_n, along with the folded or periodized signal, $\tilde{x}_n$, and use the following formula to find a value, p, by which to multiply or *scale* $\tilde{x}_n$

$$\begin{aligned} 0 &= \sum_{n=0}^{N}(\tilde{x}_{p,n}^2 - x_n^2) \\ \Leftrightarrow 0 &= \sum_{n=0}^{N}(p^2\tilde{x}_n^2 - x_n^2) \\ \Leftrightarrow p &= \sqrt{\frac{\sum_{n=0}^{N} x_n^2}{\sum_{n=0}^{N} \tilde{x}_n^2}}. \end{aligned} \tag{5.1}$$

The inverse is just as simple:

$$\begin{aligned} 0 &= \sum_{n=0}^{N}(\tilde{x}_{p,n}^2 - x_n^2) \\ \Leftrightarrow 0 &= \sum_{n=0}^{N}(\tilde{x}_{p,n}^2 - p^2 z_n^2) \\ \Leftrightarrow p &= \sqrt{\frac{\sum_{n=0}^{N} \tilde{x}_{p,n}^2}{\sum_{n=0}^{N} z_n^2}} \end{aligned} \tag{5.2}$$

where z_n is the result of applying the standard continuous unfolding routines to $\tilde{x}_{p,n}$; see Equations (4.10), (4.24), (4.55), (4.56) as well as those in Appendix B.4.

The second benefit is that, even though the process is not linear, the relationship between the transform coefficients and the basis functions can be easily computed. To see this, consider the following. Let $p_{\mathbf{x}} = p[\tilde{T}(\mathbf{x})] \in \mathbb{R}$, $\tilde{p}_{\mathbf{x}} = \tilde{p}[\tilde{T}^{-1}(\mathbf{x})] \in \mathbb{R}$ and define $F(\mathbf{x}) \overset{\Delta}{=} p_{\mathbf{x}} W^* \tilde{U}\mathbf{x}$ where $\tilde{U}$ and $\tilde{T} = W^*\tilde{U}$ are the folding and periodization operators associated with CBLFT respectively. Let B be a matrix whose columns, $\mathbf{b}_n$, are the real-valued Fourier basis vectors, and let $\mathbf{c}$ be the corresponding real-valued Fourier coefficient vector associated with

$F(\mathbf{x})$. Then $\tilde{\mathbf{b}}_n = F^{-1}(\mathbf{b}_n) = \tilde{p}_{\mathbf{b}_n}\tilde{U}^{-1}W\mathbf{b}_n$ are the CBLFT basis vectors and

$$\begin{array}{rrcl} & F(\mathbf{x}) & = & B\mathbf{c} \\ \Rightarrow & p_{\mathbf{x}}W^*\tilde{U}\mathbf{x} & = & B\mathbf{c} \\ \Rightarrow & p_{\mathbf{x}}\mathbf{x} & = & \tilde{U}^{-1}WB\mathbf{c} \\ \Rightarrow & p_{\mathbf{x}}\mathbf{x} & = & c_0\tilde{U}^{-1}W\mathbf{b}_0 + \ldots + c_N\tilde{U}^{-1}W\mathbf{b}_N \\ \Rightarrow & p_{\mathbf{x}}\mathbf{x} & = & \frac{c_0}{\tilde{p}_{\mathbf{b}_0}}F^{-1}(\mathbf{b}_0) + \ldots + \frac{c_N}{\tilde{p}_{\mathbf{b}_N}}F^{-1}(\mathbf{b}_N) \\ \Rightarrow & \mathbf{x} & = & \frac{c_0}{p_{\mathbf{x}}\tilde{p}_{\mathbf{b}_0}}\tilde{\mathbf{b}}_0 + \ldots + \frac{c_N}{p_{\mathbf{x}}\tilde{p}_{\mathbf{b}_N}}\tilde{\mathbf{b}}_N. \end{array} \tag{5.3}$$

This new realization adds much insight into the performance of the methods when

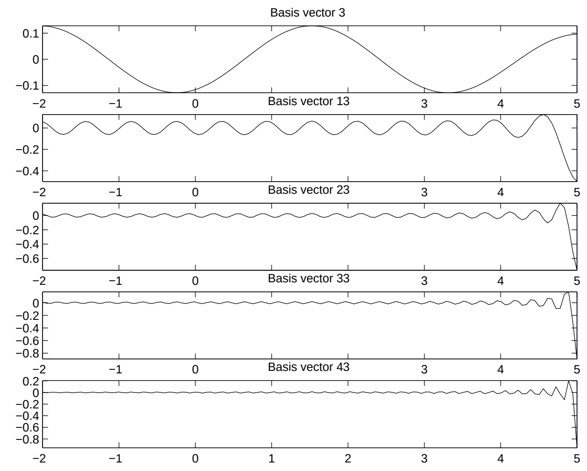

Figure 5.2: Some top level CELFT basis functions. Notice how the right hand side behaves wildly as the frequency increases.

used for compression. For example, several top level CELFT basis vectors for a particular function are seen in Figure 5.2. Notice how the right hand side of some of these basis vectors behave in an erratic manner. If the coefficients associated with these basis vectors are altered or discarded for any reason, reconstruction errors become amplified near the right hand boundary. The reason for this is related to the instability of the CELFT inverse periodization formula for the right hand boundary

$$x_{N-n} = \frac{r_n\tilde{x}_{N-n} - r_{-n-1}\tilde{x}_n}{r_n - r_{-n-1}}. \tag{5.4}$$

Clearly, values near the boundary will be amplified since $r_n - r_{-n-1} \approx 0$ for $n = 0$, as was discussed in Section 4.2. In particular, the larger the difference between x_{N-n} and x_n, the larger the numerator in Equation (5.4) will be. This result, in turn, will be amplified by the denominator. Now, for the real-valued discrete Fourier basis, the cosine terms will have nearly equal boundary values for low frequencies; e.g., for the DC component, $b_0(0) = b_0(N)$. But as the frequency increases, the distance between left and right endpoints also increase; e.g., for the Nyquist frequency, $b_{\frac{N}{2}}(0) = -b_{\frac{N}{2}}(N)$. Because of this, values near the right hand boundary of the CELFT basis vectors will increase as the frequency increases. As a consequence of this added energy to the right hand side of the higher frequency basis vectors, they must be scaled more in order to preserve the isometry. Thus, $\tilde{p}_{\mathbf{b}}$ is larger for higher frequency CELFT basis functions (see Figure 5.3). Note that the sine bases have boundary values that are near zero for both low and high frequencies, but not for the middle frequencies. Hence, $\tilde{p}_{\mathbf{b}}$ is larger for middle frequency CELFT basis functions. This result has a direct

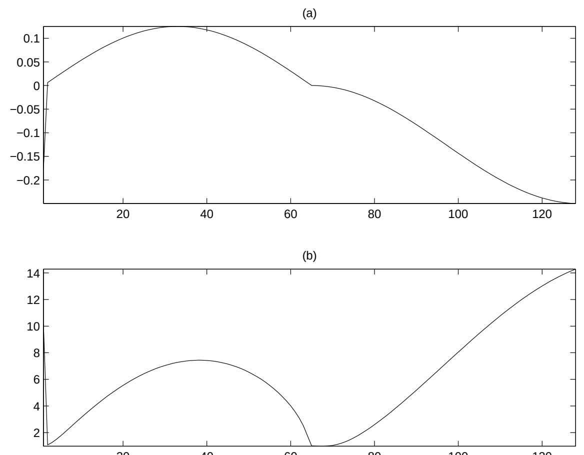

Figure 5.3: (a) $x_N - x_0$ for the real-valued DFT basis vectors, and (b) $\tilde{p}_{\mathbf{b}}$ for the CELFT basis vectors, where the values associated with the DC component have been shifted to the center, and the Fourier cosine components are on the right half, whereas the Fourier sine components are on the left half. Notice that the two graphs are directly related to one another.

effect on the reconstruction of a signal. For example, from Equation (5.3) and Figure 5.3 it is clear that the coefficients associated with high frequency cosine basis vectors are weighted higher than the low frequency ones. Thus, thresholding these coefficients has a more drastic effect on the signal reconstruction since these high frequency basis vectors have the large bump at the right boundary. Since the high frequency Fourier coefficients tend to be smaller than the low frequency ones, they are the ones that are usually thresholded first, thus leading to increased edge effect when using CELFT. It should be noted that reconstruction errors also occur from thresholding sine basis coefficients, but the effect is not as strong since the high frequency sine basis vectors do not exhibit the erratic behavior near the right boundary.

With this in mind, it makes sense to design methods which do not exhibit this type of behavior. Some simple modifications to the extensions can be derived by using different sampling points and modified extensions. See Figure 5.4. Each of these methods can also be used with gridpoint folding, thus doubling the number of possible approaches. The details of each method is described in Appendix B.4. Although these changes are minor, the resulting schemes can have drastically different attributes. As an example, take the Even1 extension shown in Figure 5.4, but apply gridpoint folding (see Figure 5.5). With this arrangement, the values for the extension to the left of x_0 are given by

$$x_{-n-1} = x_n \tag{5.5}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} = x_{N-n}. \tag{5.6}$$

Thus folding at the left boundary is defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= r_n x_n + r_{-n} x_{n-1}\end{aligned} \tag{5.7}$$

and at the right boundary as

$$\begin{aligned}\tilde{x}_{N-n} &= r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2} \\ &= r_{n+1} x_{N-n} - r_{-n-1} x_{N-n-1}.\end{aligned} \tag{5.8}$$

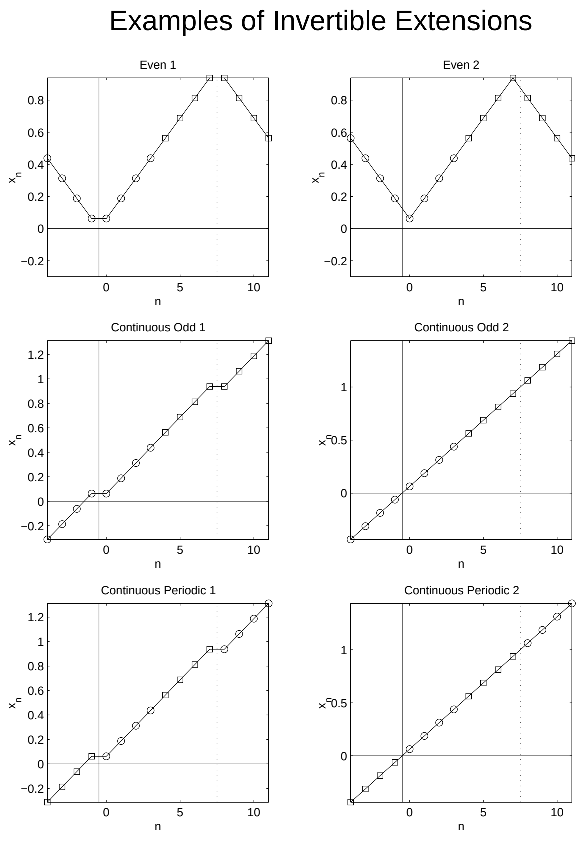

Figure 5.4: Some simple variations to the extensions applied to a linear function, $x(t) = t$ as an example. The vertical lines show the midpoint folding locations. It should be noted that both midpoint and gridpoint folding can be applied to each of these extensions. Although these are minor changes, the resulting methods can be drastically different.

Using these results along with Equation (5.1) provides the isometry.

Solving for x in Equations (5.7) and (5.8) yields the following formula for inverse folding at the left boundary

$$x_n = \frac{\tilde{x}_n - r_{-n} x_{n-1}}{r_n} \tag{5.9}$$

and at the right boundary

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1} x_{N-n-1}}{r_{n+1}}. \tag{5.10}$$

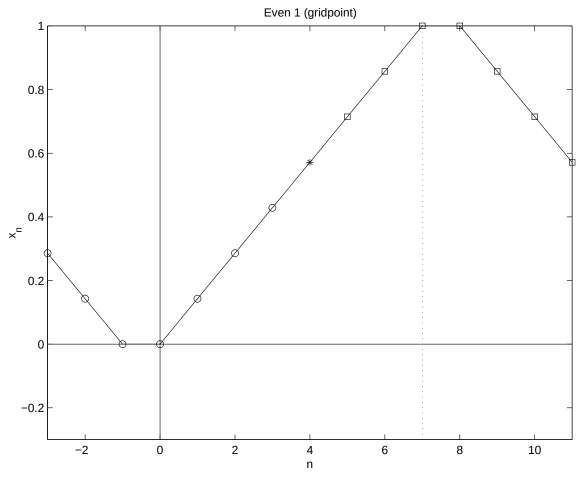

Figure 5.5: The Even1 gridpoint extension. Notice that an odd number of points are used for folding, compared to an even number used for midpoint folding. Thus, as was stated in Section 2.2.1, even for full folding, there remains one value that is untouched by the gridpoint folding procedure (denoted by an $*$ in the above figure).

Now this approach yields implicit inversion formulas. In particular, the x_{n-1} term in Equation (5.9) does not exist when $n = 0$. But luckily, the structure of the even extension can be exploited to yield $x_{-1} = x_0$ so that the inversion formula for $n = 0$ becomes $x_0 = \frac{\tilde{x}_0}{2r_0}$. This can then be used to compute x_1, which can then be used to compute x_2, and so on in a recursive manner.

In a similar fashion, when $n = 0$ the x_{N-n-1} term in Equation (5.10) always exists since it is never touched by the gridpoint folding procedure (the $*$ in Figure 5.5). Hence, all values of x_{N-n} can be recovered in a recursive manner.

But it is this very fact that the two boundary values of the subspace, x_0 and x_N, are recovered in a stable manner, and the fact that both inversion formulas do not involve division by small values, that this method achieves much better results. To be more specific, repeating the error analysis of Section 4.5.2 yields the following for CE1MLFT (Even1 Midpoint)

$$\begin{aligned} y_{N-n} &= \frac{\tilde{y}_{N-n}}{r_n - r_{-n-1}} \\ &= \frac{\tilde{x}_{N-n}}{r_n - r_{-n-1}} + \frac{e_{N-n}}{r_n - r_{-n-1}} \\ &= x_{N-n} + \frac{e_{N-n}}{r_n - r_{-n-1}} \end{aligned} \tag{5.11}$$

and the following for CE1GLFT (Even1 Gridpoint)

$$\begin{aligned} y_{N-n} &= \frac{\tilde{y}_{N-n} + r_{-n-1} y_{N-n-1}}{r_{n+1}} \\ &= \frac{\tilde{x}_{N-n} + e_{N-n} + r_{-n-1}(x_{N-n-1} + e_{N-n-1})}{r_{n+1}} \\ &= \frac{\tilde{x}_{N-n} + r_{-n-1} x_{N-n-1}}{r_{n+1}} + \frac{e_{N-n} + r_{-n-1} e_{N-n-1}}{r_{n+1}} \\ &= x_{N-n} + \frac{e_{N-n} + r_{-n-1} e_{N-n-1}}{r_{n+1}}. \end{aligned} \tag{5.12}$$

Only the analysis of the right half interval is shown since the instability is only

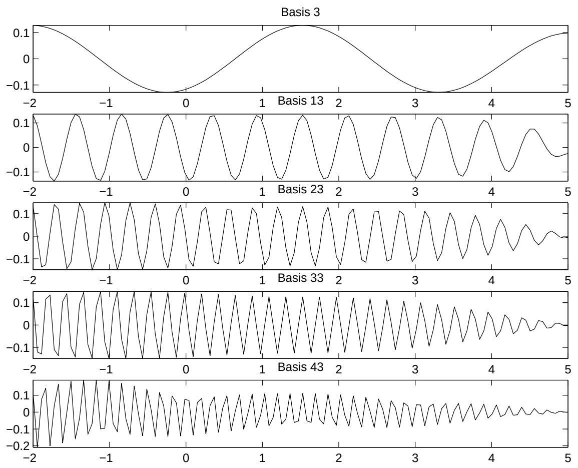

Figure 5.6: Sampling of top level CE1GFLT basis vectors. Notice how the right hand side does not explode as was seen in the CE1MLFT bases.

associated with this boundary. Looking at Equation (5.11) reveals the instability of the method; as n decreases, $r_n - r_{-n-1}$ also decreases causing the error to increase near the right boundary. Although this problem is similar to that of the periodization scheme based on linear component subtraction described in Section 4.5.2, the CBLTT is more stable because the reconstruction error at x_n is only dependent upon e_n, whereas the former's error at x_n is a combination of e_n and ne_N. Furthermore, Equation (5.12) shows the improved stability of the Even1 Gridpoint method. As can be seen, the error is always divided by large numbers, r_{n+1}. And since the error at each step is dependent on error at all previous steps, and since the error at each step can be either positive or negative, then these errors can cancel each other out causing the overall error to be suppressed near the boundary. Furthermore, since r_{-n-1} is small, then the error from each previous location is suppressed.

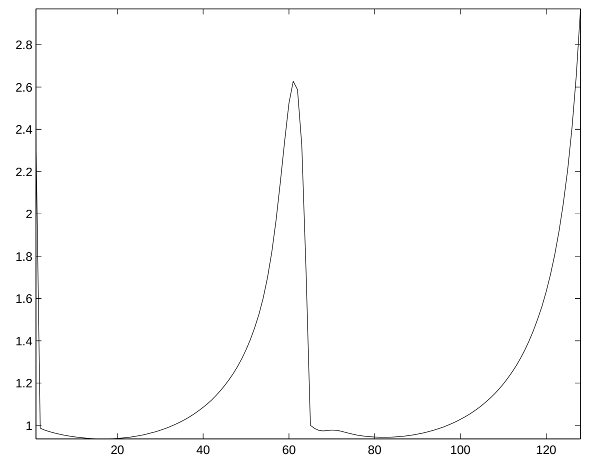

Figure 5.7: $p_{\mathbf{b}}$ for the CELFT (Even1 Gridpoint) basis functions, $\mathbf{b}$, where the value associated with the DC component has been shifted to the center. Notice the difference in magnitudes of these values versus those associated with the CELFT (Even1 Midpoint) shown in Figure 5.3(b).

As shown in Figure 5.6, the resulting basis vectors do not exhibit unstable behavior near the boundaries like those of the Even1 midpoint method. In addition, for this method, the values of $p_{\mathbf{b}}$ from Equation (5.3) are not very large (see Figure 5.7), especially when compared with those of CE1MLFT plotted in Figure 5.3. As a matter of fact, many of the values are less than 1 indicating that they actually *suppress* the effects of thresholding the corresponding basis

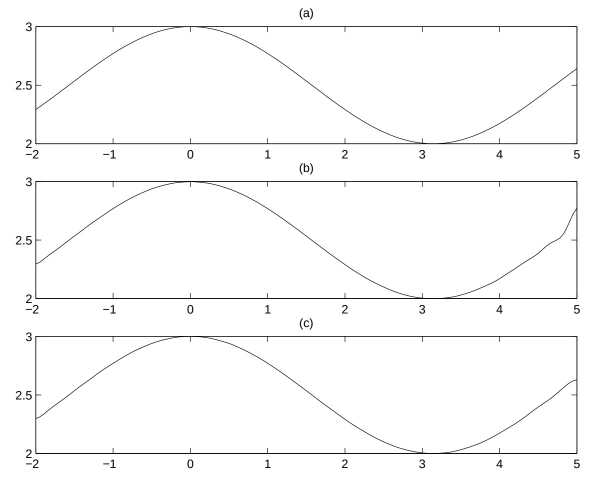

Figure 5.8: 4:1 compression results. (a) Original function, (b) CE1LFT Midpoint folding, (c) CE1LFT Gridpoint folding

coefficients.

In terms of compression, compare the two methods for the top level reconstruction in Figure 5.8. It is apparent that the search for new extensions which give rise to stable inversion formulas is needed. As stated before, a host of possibilities are derived in Appendix B.4.

5.2 Comparison Criteria

In order to facilitate a fair comparison of these methods, and to help design future schemes, some comparison criteria need to be established. As will be shown, the following three criteria provide much insight into the performance of a particular approach, and allow for a fair ranking of the various methods. They are

- the level of continuity at the subspace boundaries;
- the condition number of the folding matrix;
- the envelope, or window, of the basis functions; that is, do the bases become

unstable near the boundaries?

5.2.1 Sparsity criterion

The first bullet, the level of continuity, is mainly concerned with the effects of the periodization operator and its ability to remove boundary effect while minimizing the introduction of artificial structure. The primary measure for this is sparsity of the CBLTT coefficients. For example, the Even1 Midpoint and Even2 Gridpoint extensions both produce periodic functions, but clearly the Even1 Midpoint extension is smoother than the Even2 Gridpoint because it produces less of a jump at the boundary as can be seen in Figure 5.9. The resulting sparsity estimates

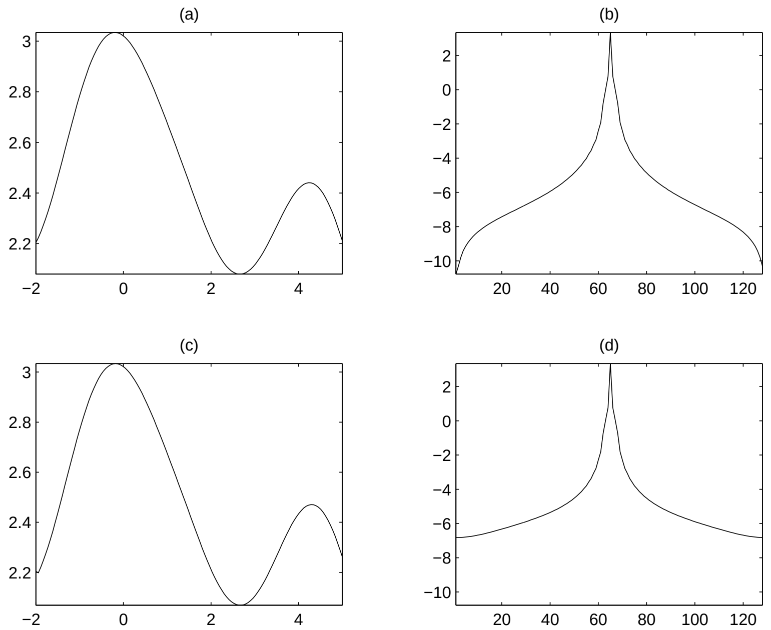

Figure 5.9: Periodization of the function shown in Figure 5.8(a) using (a) CE1MLFT folding and (c) CE2GLFT folding. (c) and (d) are their corresponding Fourier coefficients. Notice that (c) has a larger jump at its boundary than (a) when viewed as a periodic functions. This translates into faster decaying tails in the frequency domain as illustrated in (b) and (d).

are $\ell^{0.1}_{CE1MLFT}(x) = 69.82$ and $\ell^{0.1}_{CE2GLFT}(x) = 76.87$. Table 5.1 lists the sparsity values for many of the methods described so far, as well as those appearing in Appendix B.4.

Table 5.1: Sparsity estimates ($\ell^{0.1}$) of various CBLFT methods for the function in Figure 5.8 when $R = 64$. Note that *CO* and *CP* stand for the Continuous Odd Extension and Continuous Periodic Extension, respectively.

Method	$\ell^{0.1}$
No folding	89.33
Even1 Mid	69.82
Even1 Grid	71.28
Even2 Mid	69.39
Even2 Grid	76.87
CO1 Mid	70.37
CO1 Grid	70.30
CO2 Mid	48.39
CO2 Grid	44.27
CP1 Mid	69.96
CP1 Grid	69.95
CP2 Mid	55.00
CP2 Grid	55.96

5.2.2 Condition Numbers

The second means of comparison, the condition number of the folding procedure, is a measure of the numerical stability of the inverse operator. Even if the forward direction periodizes a signal well, unpredictable and often erroneous results are possible if a large condition number is detected. The condition number can be calculated separately for the left and right half intervals in the following manner. Folding and unfolding for each side of the interval can be written as a matrix, and its condition number can be calculated. It should be noted that the only difference between the standard folding and unfolding operators, Equations (2.12) and (2.16), and the corresponding isometric ones defined by Equations (5.1) and

(5.2) is the scaling. Multiplication of a matrix by a scalar does not affect the condition number of the matrix; hence, for simplicity, only the nonscaled folding matrix is shown and is used in the calculation.

For example, the folding matrix for the Even1 Midpoint extension for $N = 8$ is

$$\begin{bmatrix} r_0+r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_1+r_{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_2+r_{-3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_3+r_{-4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_3-r_{-4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r_2-r_{-3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_1-r_{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0-r_{-1} \end{bmatrix} x = \tilde{x}. \tag{5.13}$$

For the Even1 Gridpoint method, its folding matrix is

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-1} & r_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{-2} & r_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{-3} & r_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -r_{-3} & r_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -r_{-2} & r_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -r_{-1} & r_1 \end{bmatrix} x = \tilde{x}. \tag{5.14}$$

The condition numbers of each matrix for $N = 128$ are listed in Table 5.2. It should be noted that when computing the condition number for one side, the other side is assumed to be known. For example, for the left half interval of the above method, the condition number of the following matrix was computed

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-1} & r_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{-2} & r_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{-3} & r_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the right half interval, the following matrix was used

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -r_{-3} & r_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -r_{-2} & r_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -r_{-1} & r_1 \end{bmatrix}.$$

And for the condition number of both sides, the matrix in Equation (5.14) was used. As can be seen, the condition number for the right half interval of the Even1 Midpoint folding is much larger than for the left half interval. This causes the overall condition number to also be large. This means that inversion of the right hand side is less stable than inversion of the left hand side. But the Even1 Gridpoint method does not suffer from this same problem. The condition number of each of its sides, as well as the overall condition number are all relatively small. Consequently, this results in improved compression performance over the Even1 Midpoint method. This is consistent with the basis vectors shown in Figures 5.2 and 5.6, as well as the compression results seen in Figure 5.8.

Table 5.2: Condition number of the folding matrices for the left half interval (Left), right half interval (Right), and both intervals together (Both). Notice that the Even1 Midpoint right hand boundary has a much higher value than the boundaries. This causes unfavorable reconstruction results.

Even1	Left	Right	Both
Midpoint	1.41	73.36	103.75
Gridpoint	8.12	7.18	8.34

Table 5.3 lists some of these values for many of the methods described so far, as well as those appearing in Appendix B.4. For the most part, these results are in line with what was predicted. For instance, upon inspection of the inversion formulas for the Even1 Midpoint method, Equations (4.7) and (4.8), it is clear

Table 5.3: Condition number of the folding matrices for the left half interval (Left), right half interval (Right), and both sides together (Both) when $N = 128$ and $R = 64$. Note that *CO* and *CP* stand for the Continuous Odd Extension and Continuous Periodic Extension, respectively.

Method	Left	Right	Both
Even1 Mid	1.41	73.36	103.75
Even1 Grid	8.12	7.18	8.34
Even2 Mid	7.60	7.60	7.60
Even2 Grid	1.41	5.49	5.58
CO1 Mid	217.69	1718.80	1726.70
CO1 Grid	81.54	6761.20	6800.80
CO2 Mid	24.44	$1.16 \cdot 10^{17}$	$3.55 \cdot 10^{16}$
CO2 Grid	316.95	$4.18 \cdot 10^{16}$	$6.10 \cdot 10^{16}$
CP1 Mid	18.16	1021.70	1651.50
CP1 Grid	17.77	702.61	1371.70
CP2 Mid	15.84	1455.00	$4.21 \cdot 10^{16}$
CP2 Grid	18.77	702.61	$6.90 \cdot 10^{16}$

that the large condition number associated with folding at the right boundary is a direct result of the $\frac{1}{r_0 - r_{-1}}$ term associated with x_N. When looking at the inversion formulas for the other Even extension folding methods (Equations (5.7) and (5.8) for the Even1 Gridpoint, Equations (B.16) and (B.17) for the Even2 Midpoint, and Equations (B.25) and (B.17) for the Even2 Gridpoint), they are all devoid of small denominators, even for the x_N term. This causes their condition numbers to be smaller, reflecting their stable inversion formulas.

But even though inspection of inversion formulas can indicate whether a method will work or not, it will not tell *how well* it will work, or *how well* it compares to other approaches. Case in point, take the Continuous Odd1 Midpoint inversion formulas given by Equations (4.22) and (4.23). As can be seen, both formulas contain $\frac{1}{r_0 - r_{-1}}$ terms. It is unclear from inspection which is more stable. But as can be seen by their condition numbers in Table 5.3, folding at the left boundary is much more stable. Similarly, comparing the remaining Continuous Odd schemes (Equation (B.35) for the Odd1 Gridpoint, Equations (B.43) and (B.44) for the Odd2 Midpoint, and Equation (B.52) for the Odd2 Gridpoint), it is clear that the denominator is potentially small ($b \approx r_1 - 2r_{-1}(\frac{1}{2})$ for the Odd1 Gridpoint, $b \approx r_0 - 2r_{-1}(\frac{1}{2})$ for the Odd2 Midpoint, $b \approx r_1 - 2r_{-1}(\frac{1}{2})$ for the Odd2 Gridpoint). But it is uncertain which one is best, or even if any of them are practical. Luckily the condition number answers this question and reveals that the Odd1 methods are much more stable than the Odd2 methods, which are ill-conditioned and impractical for use. It also reveals that the Odd1 Midpoint folding for the left hand boundary is fairly stable and might be practical for use with a different folding scheme at the right hand boundary. This idea is explored in Section 5.3.1.

The same results hold true for the Continuous Periodic schemes. Comparing Equation (4.50) for the CP1 Midpoint method and Equation (B.62) for the CP1 Gridpoint approach, hints that the Gridpoint method is more stable than the Midpoint method because its denominator is larger than the Midpoint denominator; i.e., $r_1^{grid} - r_{-1}^{grid} > r_0^{mid} - r_{-1}^{mid}$. The condition numbers corroborate this notion. But the main benefit in this case is how the stability associated with the left and right interval folding schemes affects the overall stability of the method when combined. Since the Continuous Periodic methods have their left and right folding schemes dependent on each other, it is difficult to determine the stability from their inversion formulas (see Equation (B.72)). But according to Table 5.3, the two CP2 methods are completely unstable, even though their individual condition numbers for their left and right halves are almost identical to those of CP1. Therefore the condition number is a quick and reliable means for measuring the stability of a method.

5.2.3 Basis vector shapes near boundaries

The third comparison criteria listed above deals with the shape of the basis functions; in particular, their shape at their boundaries. Since compression often involves thresholding transform coefficients of small magnitude, it is important to know the structure of the associated basis functions. As the celebrated Riemann-Lebesgue Lemma shows, the magnitude of the Fourier coefficients generally tend to decrease as the frequency increases. Thus it is the high frequency basis functions that are usually the first to be eliminated. If these basis functions have most of their energy near their boundaries, then thresholding will tend to increase the error near the boundaries of the reconstructed signal. This was exhibited in Figures 5.3 and 5.7.

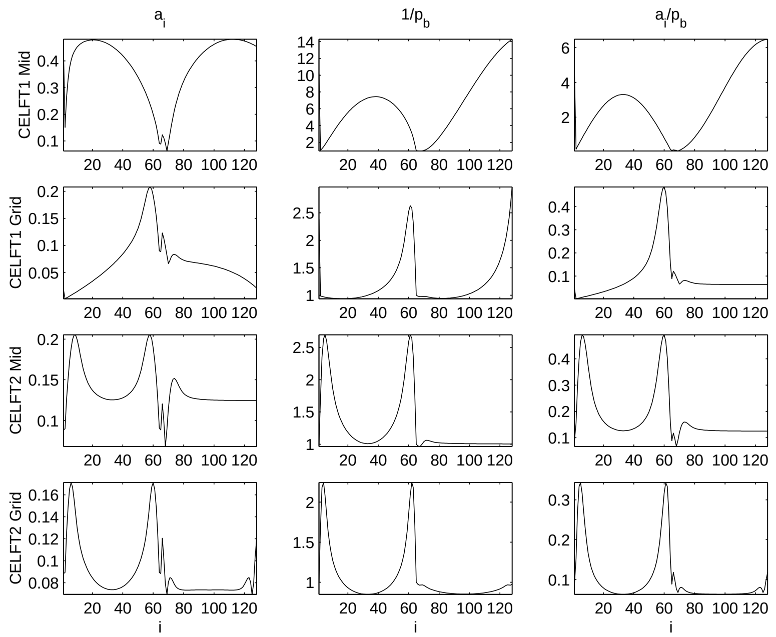

Figure 5.10: a_i and $\frac{1}{\tilde{p}_{\mathbf{b}_i}}$ and $\frac{a_i}{\tilde{p}_{\mathbf{b}_i}}$ for the four CELFT methods. Note that the DC component has been shifted to the center of each graph.

One of the items of interest is therefore the magnitude of the boundary values of each CBLFT basis vector, $\tilde{\mathbf{b}}_i$. So let $a_i = \frac{|\tilde{\mathbf{b}}_i(0)|+|\tilde{\mathbf{b}}_i(N)|}{2}$ be the average of the magnitudes of the boundary values of basis vector $\tilde{\mathbf{b}}_i$ of length $N+1$. In general, a method will exhibit less boundary effect due to compression if this number decreases as the frequency increases. Therefore, the second centralized moment of this vector, with DC component shifted to the center, should be small for methods which exhibit minimal boundary effect. This statistic is not quite accurate, though, when taken by itself. The reason is that all of the CBLFT coefficients are weighted by the values $\frac{1}{p_{\tilde{\mathbf{b}}_i}}$ as was shown in Equation (5.3). Therefore, variance of this vector should also be small for methods which minimize boundary effect. Figure 5.10 shows these values for the four CELFT versions. Notice that the CELFT1 Gridpoint values decrease when the frequency increases. This indicates that this method will probably have smaller reconstruction error near its boundaries than the other methods. Therefore, combining these two results into one statistic gives a means for rating a method's ability to reconstruct its boundary. So let E_{BASIS} be the variance of $\frac{a_i}{\tilde{p}_{\mathbf{b}_i}}$; i.e., $E_{BASIS} = \sum_{n=0}^{N}(n-\frac{N}{2})^2\frac{a_n}{\tilde{p}_{\mathbf{b}_n}}$. Table 5.4 lists these boundary effect values for all of the methods of the previous two charts.

Table 5.4: Boundary effect measure, E_{BASIS}, for various CBLFT methods when $N = 128$. Note that this statistic is independent of the input signal.

Method	E_{BASIS}	Method	E_{BASIS}
Even1 Mid	$6.30 \cdot 10^5$	CO2 Mid	$1.13 \cdot 10^{20}$
Even1 Grid	$8.50 \cdot 10^3$	CO2 Grid	$2.88 \cdot 10^{19}$
Even2 Mid	$3.29 \cdot 10^4$	CP1 Mid	$6.30 \cdot 10^5$
Even2 Grid	$1.90 \cdot 10^4$	CP1 Grid	$6.15 \cdot 10^5$
CO1 Mid	$6.30 \cdot 10^5$	CP2 Mid	$4.05 \cdot 10^{19}$
CO1 Grid	$2.08 \cdot 10^6$	CP2 Grid	$2.77 \cdot 10^{19}$

5.2.4 Summary of comparison criteria

The smaller that each of the three criteria is, the better the method. Thus, sorting the geometric mean of all three statistics results in a ranking of the best methods. These values are listed in Table 5.5. As can be seen, the CE1LFT

Table 5.5: Geometric mean of the three comparison criteria. Sorting these gives a ranking of the performance of each method.

Method	$(\ell^{0.1} * cond\# * E_{BASIS})^{\frac{1}{3}}$	Ranking
Even1 Mid	$1.66 \cdot 10^3$	4
Even1 Grid	$1.72 \cdot 10^2$	1
Even2 Mid	$2.59 \cdot 10^2$	3
Even2 Grid	$2.01 \cdot 10^2$	2
CO1 Mid	$4.25 \cdot 10^3$	7
CO1 Grid	$9.98 \cdot 10^3$	8
CO2 Mid	$5.79 \cdot 10^{12}$	12
CO2 Grid	$4.41 \cdot 10^{12}$	9
CP1 Mid	$4.17 \cdot 10^3$	6
CP1 Grid	$3.89 \cdot 10^3$	5
CP2 Mid	$4.54 \cdot 10^{12}$	10
CP2 Grid	$3.74 \cdot 10^{12}$	11

Gridpoint ranks the best of all of the methods. In general, the CELFT rate better than all of the other methods, followed closely by the CPLFT and then the COLFT.

5.3 Various Improvements

Since not all approaches give rise to methods which have stable left and right hand inversion formulas, yet there exist good left and right hand inversion formulas independently, it makes sense to combine the best of both worlds. This gives rise to new *hybrid* schemes.

5.3.1 Hybrid Schemes

One example uses the Even2 Midpoint extension at the right hand side, and the Continuous Odd2 Midpoint folding for the left hand side (CO2M-CE2MLFT). The results are shown in Figure 5.11. As can be seen, it periodizes the signal, adding C^1 continuity, while still maintaining stable reconstruction. Using a taper function or steeper rising cutoff function improves the results even more.

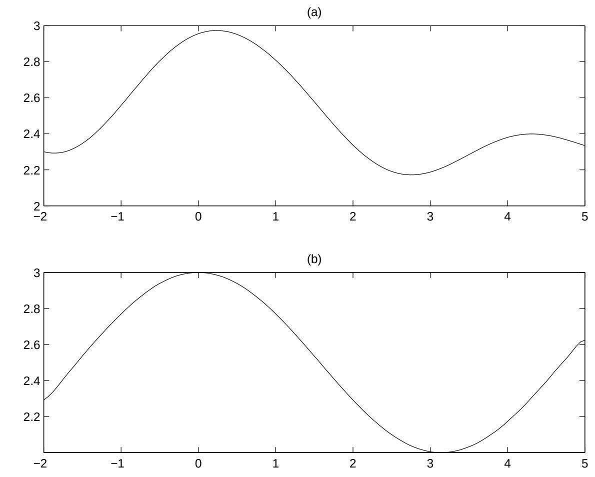

Figure 5.11: (a) Hybrid periodization (Continuous Odd2 Midpoint for the left hand side, and Even2 Midpoint for the right hand side), (b) 4:1 compression using this method.

Still another method involves eliminating the odd parity side altogether, and folding both edges of the subspace with even polarity. This type of boundary condition is associated with DCT-II. An example of the method is shown in Figure 5.12 where a Continuous Odd2 Midpoint method is employed at each boundary (CO2MLCT2). This method performs so well that 20:1 compression is used. Comparing it to the standard DCT-II compression hints that this new method will outperform JPEG-DCT. Another advantage about this approach is the fact that no information is mixed within the subspace because the periodic unfolding operator is not applied; hence no extra bumps are amplified in frequency. In

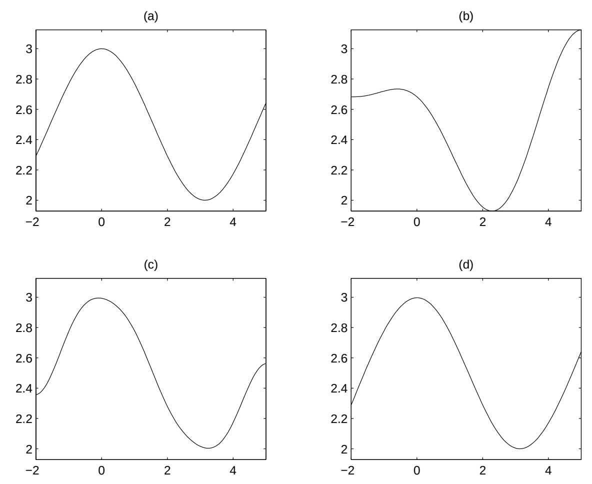

Figure 5.12: (a) Original signal, (b) Signal folded with the Continuous Odd2 Midpoint for the both sides, (c) 20:1 compression of original signal using DCT-II without any folding, (d) 20:1 compression of folded signal using DCT-II.

addition, looking at the window over the bases associated with this, it is clear that the basis vectors will not become unstable near the subspace boundaries. As a matter of fact, reconstruction errors will actually be suppressed. Figure 5.13 shows some sample basis vectors. Notice that both sides of each basis vector taper to zero. This makes for good frequency localization properties, especially when this method is cast in a best basis setting. As a comparison to the previous methods, Table 5.6 shows some the comparison criteria values for these two new schemes. Comparing the values to those of Table 5.5 shows that the CO2M-CE2MLFT method ranks fourth, while the CO2MLCT2 scheme ranks number 1.

Table 5.6: Comparison criteria for a couple of hybrid methods.

Method	$\ell^{0.1}$	cond #	E_{BASIS}	$(\ell^{0.1} * cond\# * E_{BASIS})^{\frac{1}{3}}$
CO2M-CE2MLFT	71.60	31.80	$8.43 \cdot 10^3$	$2.68 \cdot 10^2$
CO2MLCT2	47.36	24.44	$1.69 \cdot 10^1$	$2.69 \cdot 10^1$

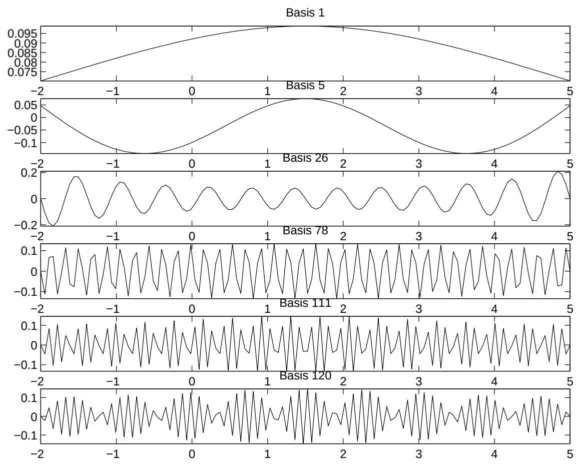

Figure 5.13: Sample bases for DCT-II style folding using the Continuous Odd2 Midpoint for both sides.

5.3.2 An Application To Brushlets

In addition to all of these methods, they can also be used to segment the frequency domain, giving rise to many new versions of the brushlet transform. Since they are operating on the frequency side with a sharp window, then this results in overlapping bases of infinite support. Compare the results of using the CO2MLCT2 brushlet (CO2MBT2) versus the standard brushlet shown in Figure 5.14.

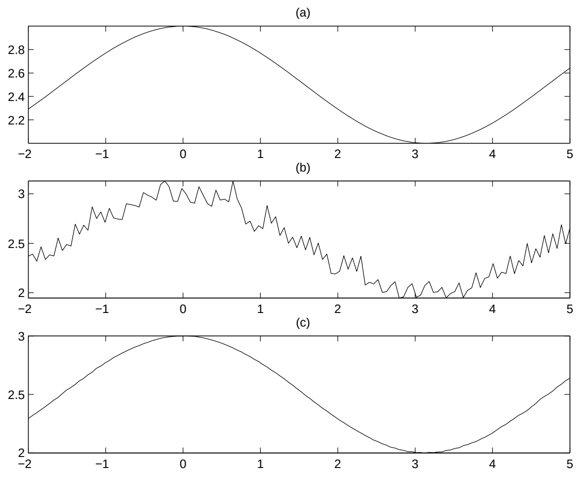

Figure 5.14: 4:1 compression using (b) the BT, and (c) the CO2MBT2.

One interesting result is shown in Figure 5.15. As can be seen, when using CO2MLBT2, the basis vector has most of its energy centered around one location. This is in direct contrast with the BT bass vectors which have their energy centered around two bumps. This improved time-frequency localization property of the CO2MBT2 translates into the improved signal representation.

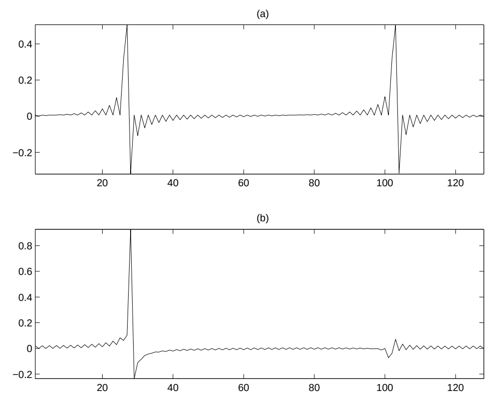

Figure 5.15: A sample basis vector for (a) the BT, and (b) CO2MBT2.

5.3.3 Storing Boundary Values

Another possible improvement is the retention of the exact boundary values for each subspace; that is, storing x_0 and/or x_N for use in the inversion formulas. Since many of the inversion formulas suffer from division by small numbers, the original boundary values for each subspace could be saved and coded in a lossless manner. Then, when reconstruction is needed, these exact values would greatly minimize the edge effect. As an example, consider storing only the right hand boundary value for the CO2 Midpoint scheme. The results are shown in Table 5.7. If this method is employed in a best basis setting, only the rightmost boundary value would need to be stored since using it to restore the rightmost interval would supply the adjacent interval with its rightmost value. This could be repeated for each interval working from right to left until the signal is restored. One drawback, though, is that reconstruction errors in any given interval could propagate to the adjacent interval on its left in a recursive manner.

Table 5.7: Condition number of the folding matrices for the left half interval (Left), right half interval (Right), and both sides together (Both). Notice that the condition number for the right half is drastically reduced by storing the original boundary value of the signal for use in reconstruction. This also reduces the overall condition number of the matrix (BOTH).

CO2 Midpoint	Left	Right	Both
Discard Right Boundary	24.44	$1.16 \cdot 10^{17}$	$3.55 \cdot 10^{17}$
Store Right Boundary	24.44	27.90	28.66

5.4 Comparison with Other Basis Dictionaries

With all of these methods now defined, it remains to add them to a best-basis algorithm for comparison with some of the more popular compression techniques such as LCT, LFT, WPT, and BT. Figure 5.16 and Table 5.8 illustrate the results. All of the methods shown in the left column are ones that operate by segmenting the spatial domain, whereas all of the methods in the right column operate by segmenting the frequency domain. Also, the six plots in the upper half of the figure illustrate results when using some of the well known methods such as JPEG-DCT, LTT, WP and BT, while the bottom six plots were all made by employing different versions of the CBLTT. For the wavelet packet transform, D02 (the Haar-Walsh wavelet packet utilizing Daubechies' 2-tap quadrature mirror filter) and C06 (6-tap coiflet with 2 vanishing moments) were used for comparison. Some things to note are that all three Continuous Boundary Brushlet Transform (CBBT) methods produced lower reconstruction errors than the BT, LCT and LFT. And all three CBLTT methods outperformed all but C06. But most importantly is the fact that the CO2MLCT2 outperformed every method including

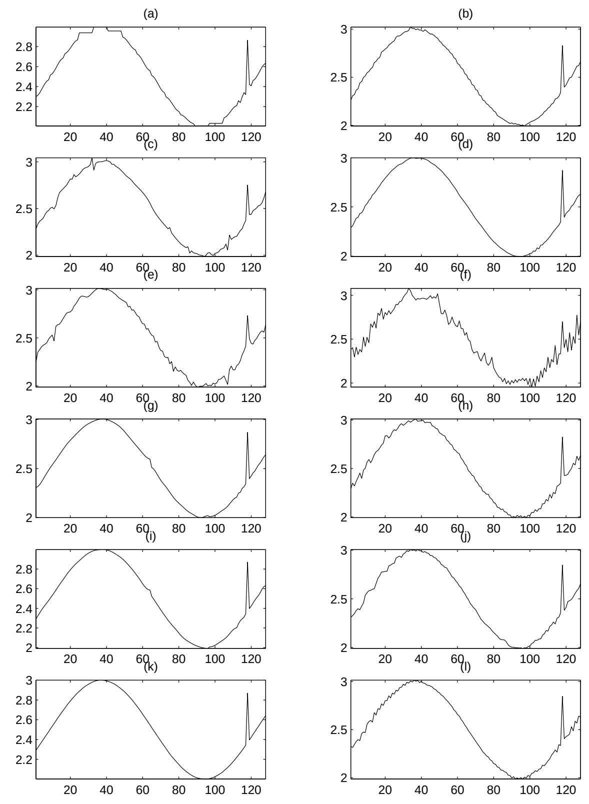

Figure 5.16: 4:1 compression using various methods: (a) JPEG-DCT, (b) D02, (c) LCT, (d) C06, (e) LFT, (f) BT, (g) CE1GLFT, (h) CE1GBT, (i) CO2M-CE2MLFT, (j) CO2M-CE2MBT, (k) CO2MLCT2, (l) CO2MBT2. Multiple folding was employed in all cases where applicable.

C06. As a matter of fact, the reconstruction error for C06 was 24 times higher than the CO2MLCT2.

But this test is not the fairest comparison. Since the original signal was based on smooth trigonometric functions, it is not surprising that methods whose basis functions are trigonometric functions performed so well, some even outperforming the wavelet packet transforms. D02 is better suited for representing piecewise constant functions, and C06 is better suited for noisy signals with a lot of sharp edges. Therefore, the following test provides a better means for comparison.

The above tests were repeated using a more complicated signal, the scanline

Table 5.8: Relative ℓ^2 error of the compressed signals shown in Figure 5.16 (4:1 compression ratio).

Method	Rel ℓ^2 error	Method	Rel ℓ^2 error
JPEG-DCT	$2.67 \cdot 10^{-5}$	D02	$1.60 \cdot 10^{-5}$
LCT	$7.95 \cdot 10^{-5}$	C06	$3.80 \cdot 10^{-6}$
LFT	$1.23 \cdot 10^{-4}$	BT	$5.52 \cdot 10^{-4}$
CE1GLFT	$8.36 \cdot 10^{-6}$	CE1GBT	$3.94 \cdot 10^{-5}$
CO2M-CE2MLFT	$4.13 \cdot 10^{-6}$	CO2M-CE2MBT	$2.59 \cdot 10^{-5}$
CO2MLCT2	$1.58 \cdot 10^{-7}$	CO2MBT2	$3.11 \cdot 10^{-5}$

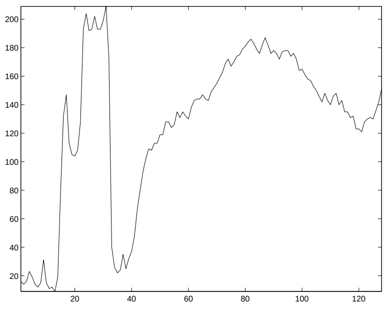

Figure 5.17: Sample signal taken from Lenna.

from Lenna shown in Figure 5.17, and also a higher level of compression. The results are shown in Figure 5.18 where 10:1 compression was used for all of the methods. Table 5.9 shows the corresponding relative ℓ^2 error associated with their reconstruction. In this case, all three CBLTT methods outperform all of the other methods.

A side note is that these periodization routines could also be used to treat the global signal prior to applying a wavelet transform. Since wavelet transforms, like the Fourier transform, suffer from edge effect problems for aperiodic signals, the addition of a Continuous Boundary periodization step could improve wavelet performance quite a bit.

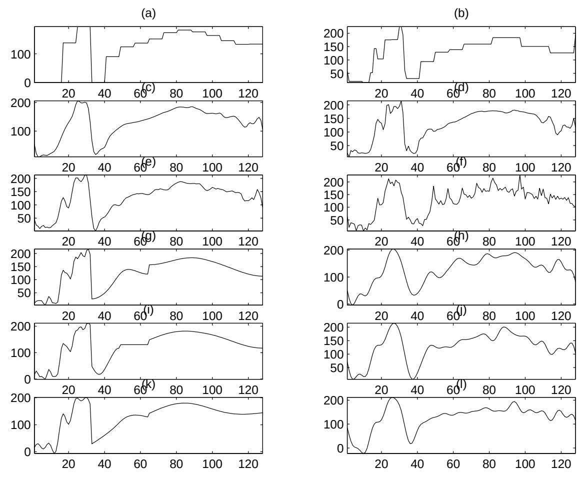

Figure 5.18: 10:1 compression using various methods: (a) JPEG-DCT, (b) D02, (c) LCT, (d) C06, (e) LFT, (f) BT, (g) CE1GLFT, (h) CE1GBT, (i) CO2M-CE2MLFT, (j) CO2M-CE2MLFT, (k) CO2MLCT2, (l) BCO2MLCT2 (DCT2 version). Multiple folding was employed in all cases where applicable.

5.4.1 Segmentation

One of the reasons that CBLTT performs well is that it does a good job of isolating structure into homogeneous regions. For example, looking at Figure 5.19, compare the partition patterns for a few of the methods for the test signal used in Figure 5.16. Notice how CBLTT does a nice job of isolating the sharp bump. This property greatly enhances the performance of the method. To better un-

Table 5.9: Relative ℓ^2 error of the compressed signals shown in Figure 5.18 (10:1 compression ratio).

Method	Rel ℓ^2 error	Method	Rel ℓ^2 error
JPEG	$2.20 \cdot 10^{-2}$	D02	$9.12 \cdot 10^{-3}$
LCT	$8.99 \cdot 10^{-3}$	C06	$8.72 \cdot 10^{-3}$
LFT	$8.70 \cdot 10^{-3}$	BT	$2.83 \cdot 10^{-2}$
CE1GLFT	$5.39 \cdot 10^{-3}$	CE1GBT	$2.20 \cdot 10^{-2}$
CO2M-CE2MLFT	$4.48 \cdot 10^{-3}$	CO2M-CE2MBT	$1.80 \cdot 10^{-2}$
CO2MLCT2	$4.17 \cdot 10^{-3}$	CO2MBT2	$2.18 \cdot 10^{-2}$

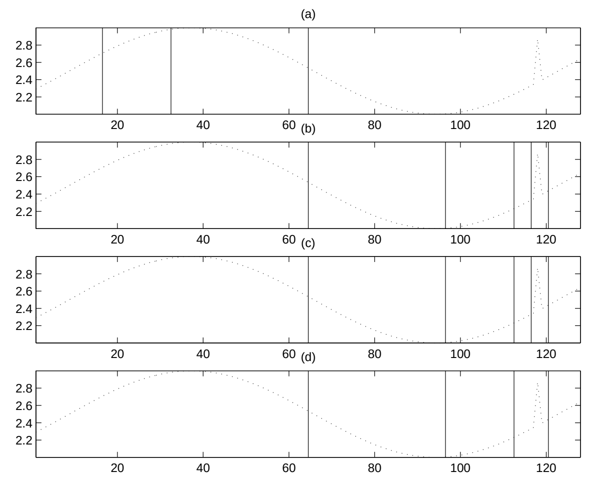

Figure 5.19: Basis partition patterns when using (a) LFT, (b) CE1GLFT, (c) CO2M-CE2MLFT, and (d) CO2MLCT2.

derstand the segmentation differences between LFT and CBLFT, consider the simple case shown in Figures 5.20 and 5.21 where x is a standard basis vector; i.e., $x = 0$ everywhere except at one location where it is equal to 1

$$x_n = \begin{cases} 1 & \text{if } n = k, \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

for $0 \leq n < M$. For illustrative purposes, CPLFT is used, although any CBLFT variation could be employed. In addition, complex-valued versions of both methods were chosen in order to lighten the notational burden and make the analysis less cumbersome.

Proposition 5.4.1. *If x is defined according to Equation* (5.15)*, and analyzed with the complex-valued LFT using ℓ^1 and full folding, then the bottom level of the decomposition will always be chosen,* regardless of the choice of k. *If, on the other hand, x is analyzed using the complex-valued CPLFT with ℓ^1 and full folding, then the smallest interval containing the bump will always be chosen, while the largest intervals will be chosen for the regions which have zero energy.*

See Appendix B.5 for the proof. This leads to very natural partition patterns when using the CPLFT. Figure 5.22 illustrates these results

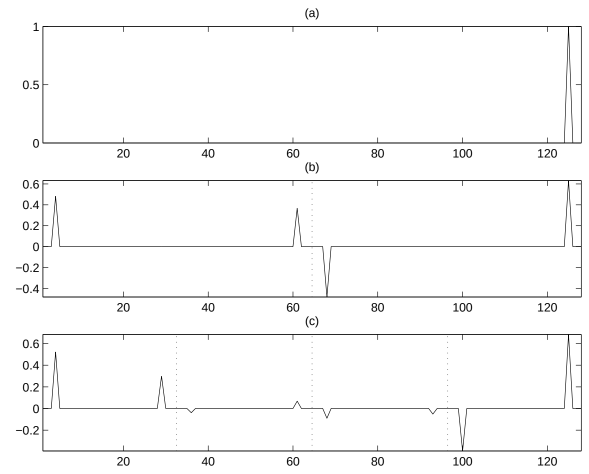

Figure 5.20: Hierarchical tree decomposition by LFT periodization. (a) Original signal, (b) Level 1 periodization, (c) Level 2 periodization.

In addition, CPLFT tends choose partition patterns in a more stable manner than LFT. For example, as was seen in Equation (B.99), CPLFT chooses the smallest interval surrounding the bump, regardless of the value of $p \in (0, 1]$. Hence, the partition pattern is always the same for this signal. But when using LFT, as $p \to 0$, the sparsity estimate for each parent subspace becomes less than its children. This causes LFT to choose larger subspaces as p decreases (see Figure 5.23).

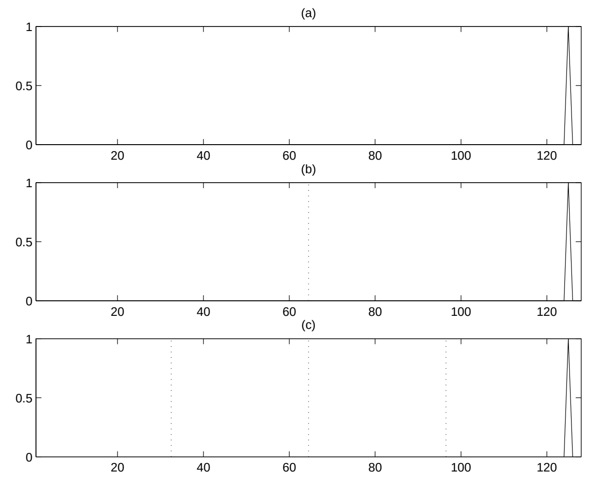

Figure 5.21: Hierarchical tree decomposition by CPLFT periodization. (a) Original signal, (b) Level 1 periodization, (c) Level 2 periodization.

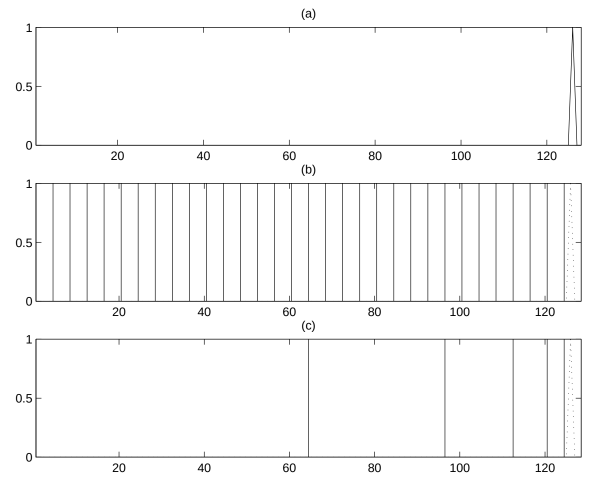

Figure 5.22: (a) Original signal, (b) LFT partition pattern, (c) CPLFT partition pattern.

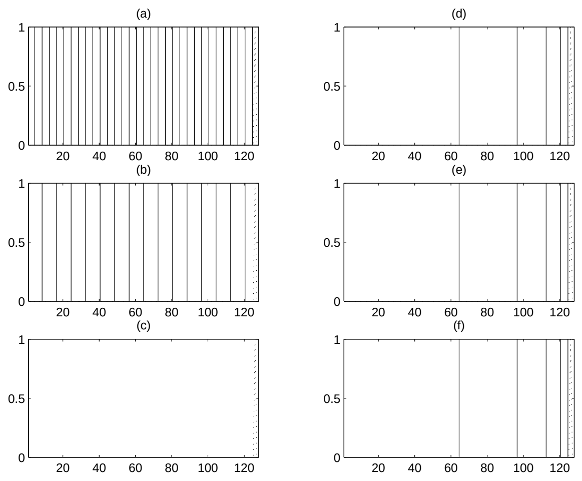

Figure 5.23: Best basis partition patterns using (a) LFT with $p = 1$ (b) LFT with $p = 0.1$, (c) LFT with $p = 0.01$, (a) CPLFT with $p = 1$ (b) CPLFT with $p = 0.1$, (c) CPLFT with $p = 0.01$

Chapter 6

CBLTT In Two Dimensions

All of these ideas can be extended to higher dimensions in the following manner. The non-isometric folding and unfolding procedures are applied as a tensor product; that is, they are first applied to the rows and then to the columns. Following this, the periodized signal is then scaled to preserve the isometry. For

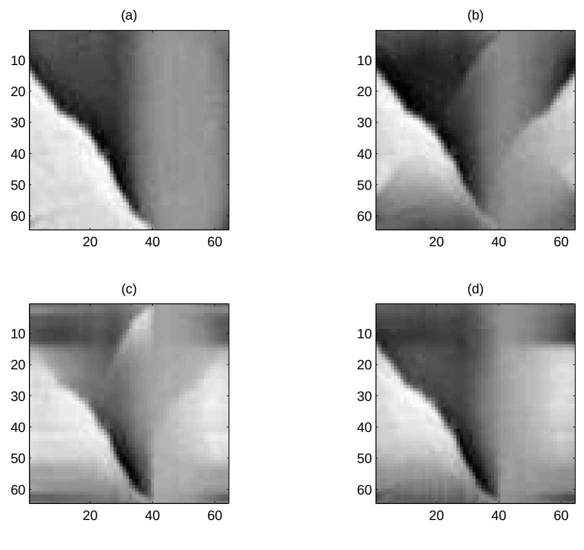

Figure 6.1: Two dimensional periodization. (a) Original image, (b) CE1M, (c) CO1M, (d) CP1M.

the even and continuous periodic extensions, it is straight forward to apply the folding operators as can be seen in Figure 6.1. Notice how the continuous periodic operations nicely periodize the subspace without mixing too much information or introducing any extra structure in frequency (Figure 6.2).

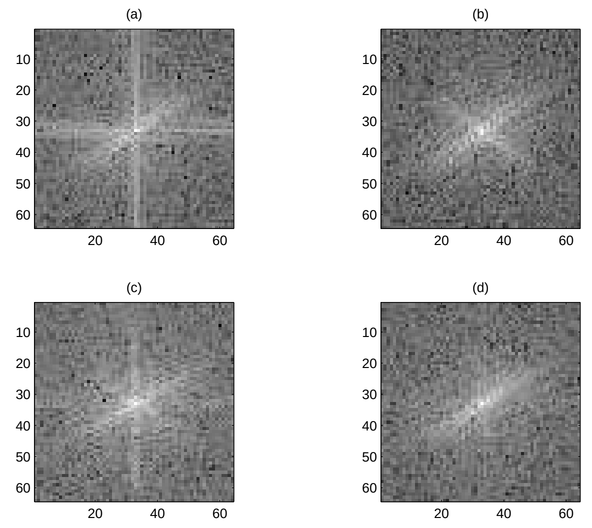

Figure 6.2: Fourier transform of the images in Figure 6.1: (a) Original image, (b) CE1M, (c) CO1M, (d) CP1M. Notice that CP1M reduces the edge effect, but doesn't introduce extra structure in frequency.

For the continuous odd periodization, though, artificial edges are introduced into the signal. As is illustrated in Figure 6.1(c) and Figure 6.2(c), vertical and horizontal stripes appear in the folded function because of the lack of continuity between the rows or columns of the extended parts. To be more specific, consider the following example. Assume the following two signals represent adjacent rows in an image

$$x_n = \begin{cases} \frac{1}{2} & \text{if } n = 0, 1, \\ -\frac{1}{2} & \text{if } 1 < n < M, \end{cases} \tag{6.1}$$

and $y_n = -x_n$ for $0 \leq n < M$. These signals, along with their continuous odd extensions, are plotted in Figure 6.3 for $M = 16$. Even though both signals are similar in value for each point in the interval $0 \leq n < M$, the distance between their extensions is sometimes much greater; i.e., $\|x-y\|_{\ell_\infty} = 1$ for $0 \leq n < 16$, and $\|x-y\|_{\ell_\infty} = 3$ for $-8 \leq n < 24$. This larger discrepancy between the two signal's extensions is folded back into the intervals, increasing the distance between the two folded signals; i.e., $\|\tilde{x} - \tilde{y}\|_{\ell_\infty} \approx 1.6$ for $0 \leq n < 16$ (see Figure 6.4). When viewed as two rows in an image, this procedure has the effect of producing an artificial horizontal edge in the image. Since the 2D Fourier Transform is applied as a tensor product of the 1D Fourier Transform, then the artificial edge will produce a sinc function in the Fourier domain.

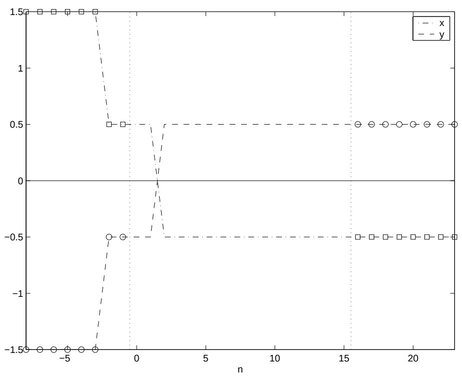

Figure 6.3: Two signals and their Continuous Odd extensions (CO1M). Notice that the difference between the two signal's extensions is larger than the distance between the two signals.

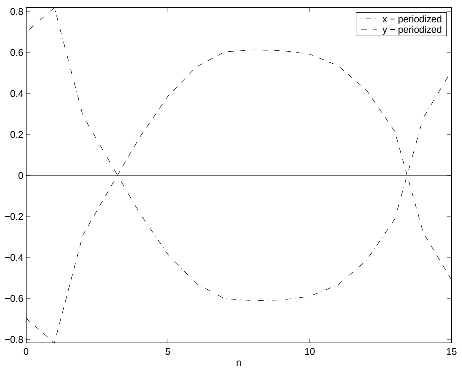

Figure 6.4: Results of periodizing the two signals shown in Figure 6.3. Notice that at certain locations the distance between the two periodized signals has increased over the distance between the original signals.

One possible way to decrease this artifact, is to use a trick introduced in the previous chapter. It is the idea of using steeper rising cutoff functions as well as a taper function to suppress and smooth the extension (see Section 4.4). The results are shown in Figure 6.5. As can be seen, the taper function decreases

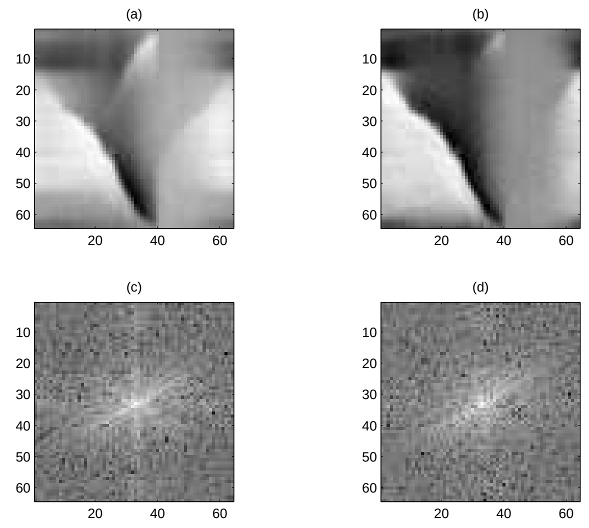

Figure 6.5: CO1 Midpoint periodization. (a) using r_1 without taper function, (b) using $r_3$with a taper function. Plots of their corresponding spectrum are shown in (c) and (d).

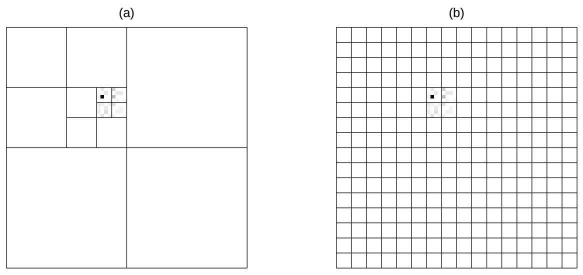

Figure 6.6: Partition pattern of a two-dimensional spike image using (a) CPLFT, and (b) LFT.

the striped effect. Also, the use of a taper function improves the reconstruction properties by decreasing the condition number as was seen in the last chapter.

Because of this, all of the properties exhibited in the one-dimensional case are also extended to two-dimensions. For example, if a two-dimensional spike image is partitioned using CBLFT and LFT, results similar to those of the last chapter for the one-dimensional case are observed (see Figure 6.6). In addition, the partition pattern of the CBLFT is more stable than the LCT and the LFT

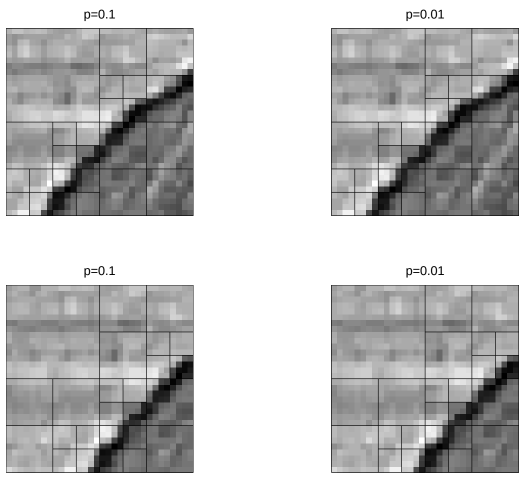

Figure 6.7: Comparing the left two plots to the right two plots shows the robustness and stability of the partition pattern under two different sparsity measures ($\ell^{0.1}$ versus $\ell^{0.01}$) using CBLFT. Comparing the top two plots to the bottom two plots shows the robust and stable partition pattern under a shift using CBLFT.

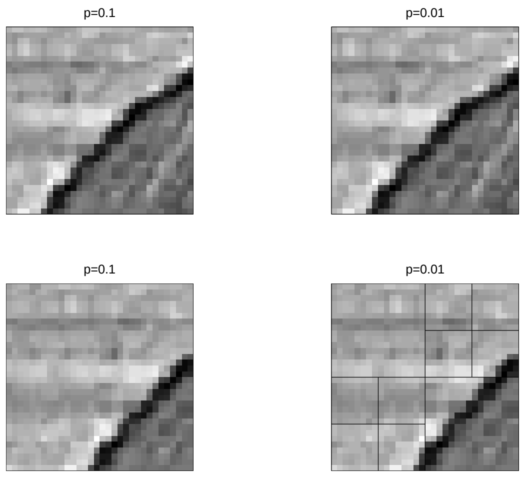

Figure 6.8: This is the same test as shown in Figure 6.7 but with LFT.

as can be seen in Figures 6.7 and 6.8. As the image is shifted, and as various values of ℓ^p are applied, the partitioning is more robust for CBLFT. Similarly, as Figures 6.9 and 6.10 illustrate, the CBLFT isolates edges better than the LCT/LST or the LFT as the image is rotated. All of these properties translate into improved compression which is illustrated next.

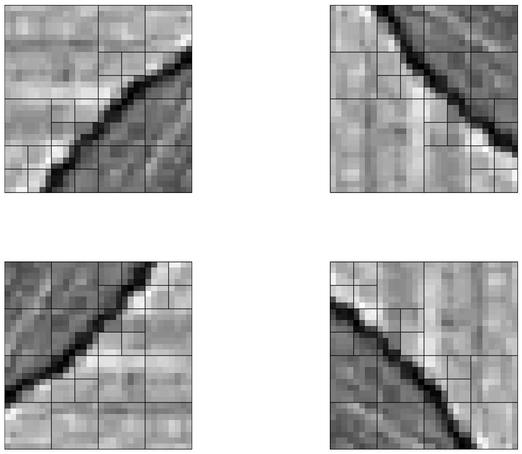
Figure 6.9: Robust partition patterns under a rotation of the image using CBLFT.

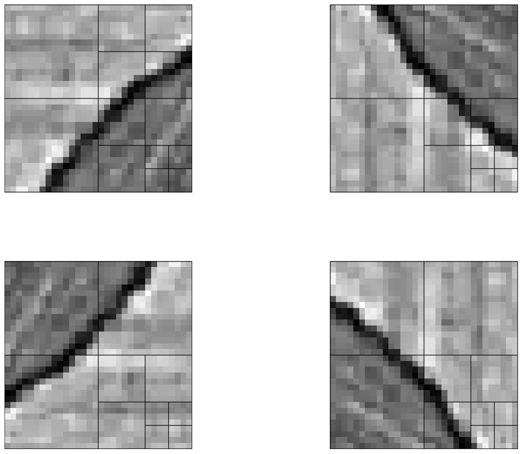
Figure 6.10: This is the same test as shown in Figure 6.9 but with LCT.

6.1 Image Compression

When it comes to image compression, a method's quality cannot always be ascertained by simply calculating the difference between a reconstructed image and its original. Quantifying error using a measure such as the ℓ^2-norm is less pertinent when analyzing images since the human visual system is more acute to certain types of error[1] ([38],pp.67-76). As an example, consider the two images shown in

[1]This is also true for 1-D audio signals, where the human audio system is more sensitive to certain types of error such as artificial signal discontinuities.

Figures 6.11 and 6.12.

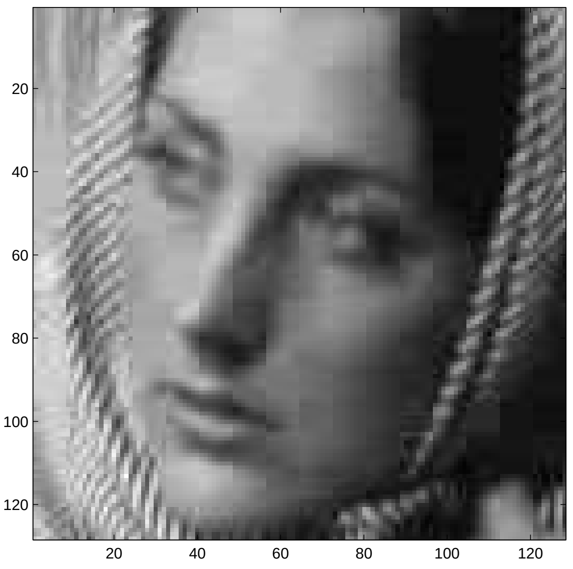

Figure 6.11: Although this image has smaller ℓ^2-error than the following image, aligned discontinuities in the error produce a blocking effect which is less pleasing to the human observer.

Each was compressed using a different method at the rate of 20:1. The ℓ^2-error associated with the first one is smaller than the second, indicating that the first one is a closer representation of the original image than the second. But even though the overall error is less for the first image, the error does not vary in a continuous fashion like it does in the second image. These discontinuities happen to align themselves in horizontal and vertical bands which are easily detected by the human eye. This unpleasant artifact is not present in the second image and the second image is therefore classified as the better reconstruction. Because of this subjective nature of rating image compression algorithms, the following tests were performed to gauge the performance of the some of the variations of the CBLTT algorithm.

The test consisted of compressing three different types of images using twelve different compression algorithms: six CBLTT variations were tested against six

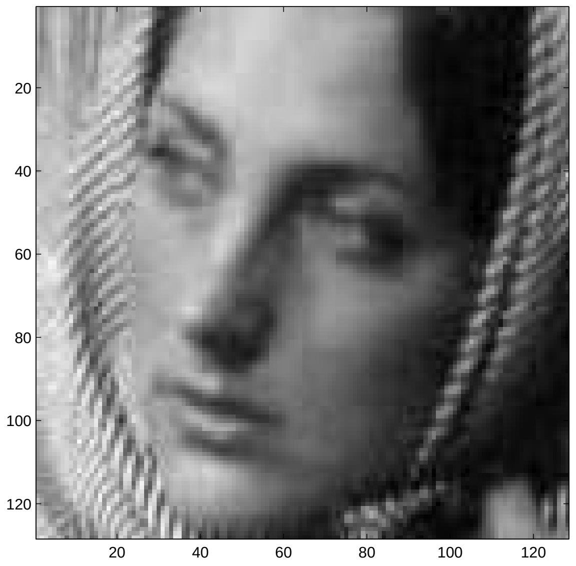

Figure 6.12: Although this and the previous image were compressed at the same rate (20:1), the lack of blocking artifact present in this image is more pleasing to the human observer than Figure 6.11.

well known methods. The three images were chosen to represent three different classes of images: line drawing, textural, and photo. Twenty five people were asked to rank the twelve images from each category separately, by sequentially eliminating the worst image from the set. They were allowed to view the original image at all times for comparison. The order in which they eliminated the images was recorded, and the tabulated results are presented in Tables 6.1, 6.2 and 6.3. Also included in each table is the method's name, an average ranking value, and the overall ranking of each method. As an example, take the eighth row of Table 6.1 which corresponds to the Continuous Even1 Gridpoint Brushlet Transform (CE1GBT). According to the chart, 21 people felt that this algorithm produced the highest quality compressed image and should therefore be ranked number 1. Four people felt that there were two other images of better quality; they therefore ranked this method third. The second column from the right contains a weighted average which was calculated by awarding the best method a value of 12, the second best a value of 11, methods chosen third a value of 10,

Table 6.1: Subjective rankings for 20:1 compression of the surfer image C.1

Method\Rank	1	2	3	4	5	6	7	8	9	10	11	12	Mean	Rank
JPEG-DCT				4			5	16					5.84	7
D02							12	8	5				5.28	8
LCT					10	15							7.40	6
C06				18	3		4						8.40	4
LFT			4		11	10							7.92	5
BT								1	16	8			3.72	9
CE1GLFT											25		2.00	11
CE1GBT	21		4										11.68	1
CO2M-CE2MLFT							4		4	17			3.64	10
CO2M-CE2MBT		25											11.00	2
CO2MLCT2												25	1.00	12
CO2MBT2	4		17	3	1								10.12	3

etc. Methods ranked last were given a value of 1. So the CE1GBT weighted mean is $\frac{21*12+4*10}{25} = 11.68$. Repeating this for each other row gives an overall score, which when ordered was used to rank each method against the others. This overall ranking is shown in the final column of each table.

For the compression, full folding was employed for every applicable method, and the steepness of the rising cutoff function was set to $m = 5$ for all of the CBLTT methods and it was set to 1 for the others. These conditions were chosen to achieve a high level of performance for each method; for example, fixed folding, or a steeper rising cutoff function, produced less pleasing results for LCT, LFT and BT. Conversely, the CBLTT methods perform better with steeper rising cutoff functions as was shown in the last chapter. For all cases, a compression rate of 20:1 was used. The full set of test images can be seen in Appendix C.

As can be seen in each table, a definite trend is easily observed. Clearly, all of the methods which exhibit a blocking effect (JPEG-DCT, D02, CE1GLFT, CO2M-CE2MLFT, CO2MLCT2) were evaluated poorly as was predicted. But

Table 6.2: Subjective rankings for 20:1 compression of the tile image C.14

Method\Rank	1	2	3	4	5	6	7	8	9	10	11	12	Mean	Rank
JPEG-DCT										3	1	21	1.60	12
D02							3	19	2	1			4.96	8
LCT						3	22						6.12	7
C06				10		12		3					7.56	6
LFT			4		16	5							8.12	5
BT								3	18		4		3.80	9
CE1GLFT										11	11	3	2.32	11
CE1GBT	22		3										11.68	1
CO2M-CE2MLFT									5	10	9	1	2.76	10
CO2M-CE2MBT	3	16	6										10.96	2
CO2MLCT2				11	9	5							8.24	4
CO2MBT2		9	12	4									10.20	3

all three of the Continuous Boundary Brushlet Transform (CBBT) methods performed well, consistently out ranking all of the other methods for all three classes of images. It should be noted that it may be possible to find particular images which cause the rankings to change. For instance, in the surfer image, the dominant lines are oriented along the 45 degree angle. This orientation is less conducive to certain transforms such as C06 which lack orientation capabilities. If the dominant lines were closer to 20 degrees, for example, the results from C06 might show improvement. But overall, the fact that the three CBBT methods performed so well on three different classes of images which were comprised of many different textures and patterns at various orientations and scales, hints at their superior performance.

There are three primary reasons why the CBBT methods perform so well. First is the fact that any energy from blocking effects which occur in the frequency domain is distributed evenly throughout the entire space domain. Since the human visual system is less acute to this type of error, then these methods produce results which are less disturbing to a human observer. Second is the

Table 6.3: Subjective rankings for 20:1 compression of the barbara image C.27

Method\Rank	1	2	3	4	5	6	7	8	9	10	11	12	Mean	Rank
JPEG-DCT							4	21					5.16	8
D02									25				4.00	9
LCT			2	3	10	10							7.88	6
C06	14					7	4						9.92	3
LFT			4	7	9	5							8.40	5
BT					1	3	17	4					6.04	7
CE1GLFT										13	12		2.52	10
CE1GBT	11	5	9										11.08	1
CO2M-CE2MLFT										12	13		2.48	11
CO2M-CE2MBT		19	6										10.76	2
CO2MLCT2												25	1.00	12
CO2MBT2		1	4	15	5								9.04	4

orientation capabilities associated with the CBBT. And third is the fact that the CBBT segments the frequency domain in a very natural manner. This means that the scale of the CBBT basis functions will closely match the scale of the underlying structure in the space domain.

Combining the results from all three charts gives overall means and rankings which are displayed in Table 6.4. As is shown, the three CBBT methods rank the highest.

Table 6.4: Overall ranking of the 12 methods for the three images.

Method	Mean	Rank
CE1GBT	11.48	1
CO2M-CE2MBT	10.91	2
CO2MBT2	9.79	3
C06	8.63	4
LFT	8.15	5
LCT	7.13	6
D02	4.75	7
BT	4.52	8
JPEG-DCT	4.20	9
CO2MLCT2	3.41	10
CO2M-CE2MLFT	2.96	11
CE1GLFT	2.28	12

Chapter 7

Conclusion

The Continuous Boundary Local Trigonometric Transform was shown to be a powerful tool for both signal analysis and signal synthesis. Many variations of this method were proposed and analyzed, and in the end the CBLTT was shown to be as good, in some cases better, than many of the conventional tools currently available. In particular, the LCT, LFT, BT, JPEG-DCT and WP were used for comparison since they all possess the ability to concentrate the signal energy in the space and frequency domain. Their primary drawback lies in the fact that they lack an effective way to periodize an initial signal. Without any information outside of the original signal, they are unable to both treat the boundary, as well as stably recover the original signal from the transformed coefficients. In addition, it was shown that as signals are decomposed into the hierarchical tree structure common to many of these approaches, instability of the folding process on or near a subspace boundary can cause increased edge effect for that subspace. Since children subspaces inherit some of their boundaries from their parents, the edge effect can translates recursively to lower levels of the decomposition. This often results in inefficient representation, improper segmentation and incorrect analysis of the signal. Consequently, signal compression and synthesis are negatively affected.

The CBLTT overcomes all of this by forcing continuity at the boundaries of the original signal, as well as at the boundaries of each subspace, by creating artificial extensions which are then used to periodize each interval. Many variations of this method were proposed and analyzed, and it was found that the CBLTT's ability to minimize the mixing of information between subspaces, as well as *within* subpspaces for some versions such as the CPLFT, while still reducing edge effect, caused improved analysis capabilities over the standard methods. It was shown that the stability of the algorithm under various sparsity measures, as well as the robustness under shifts and rotations, indicates that CBLTT is well suited for certain applications such as texture segmentation.

In addition, the CBLTT was found to have a stable inverse transform, and also possess the ability to minimize reconstruction edge effects due to quantization or thresholding errors. All of these properties result in improved compression. To be more specific, for one dimensional signals, three versions of CBLTT outperformed many of the other methods at varying levels of compression. They were the CE1GLFT which is based on the use of even boundary extensions and gridpoint folding operations, the CO2M-CE2MLFT which combines both even and continuous odd extensions along with midpoint folding operations, and the CO2MLCT2 which uses continuous odd extensions along with the DCT-II transform. If these three CBLTT versions are applied to the frequency domain of a signal, brushlet versions of the CBLTT result. When it came to image compression, these three CBBTs consistently outperformed all of the other methods on a wide variety of image types: line drawings, textures, and photos.

Overall, the CBLTT was found to excel at both analysis and synthesis and therefore shows great potential for use in a wide variety of signal processing applications. With all of the numerous variations of the CBLTT, it has the flexibility to be customized for specific purposes; for example, the CPLFT can be used for image segmentation, and the CE1GBT can be used for image compression. As added improvements and new versions of the algorithm are continually discovered, the CBLTT will hopefully become a common signal processing tool with a wide variety of uses.

Chapter 8

Further Research

One of the primary future goals is to devise methods which satisfy all three of the ranking constraints defined in Section 5.2. The main problem is that methods created to satisfy one constraint, may not satisfy another. The following ideas represent only a small sample of the possible approaches to circumventing these problems.

8.1 Increasing Stability of Existing Methods

Take the CO2LFT, both midpoint and gridpoint folding, for example. They were devised to increase the level of continuity at the boundary, thus increasing the sparsity of the transformed coefficients. Thus they satisfy the first of the ranking conditions. But their condition numbers are so large that they cannot be inverted in a stable manner for practical use. So one idea is to try to increase the stability of the method, without destroying the continuity at the boundary. One possible solution is to use more signal values when defining an extension. Under the current CO2LFT construction, the extension is made by reflecting the signal about the points x_0 and x_N. This resulted in the formulas of Equations B.36 and B.37 for midpoint folding, and Equations B.45 and B.46 for gridpoint folding.

In other words, each extension point is dependent on two signal values. Instead, an extension could be made by reflecting the signal about $x_{-\frac{1}{2}}$ and $x_{N+\frac{1}{2}}$. This results in the following formulas

$$\begin{aligned} x_{-n-1} &= x_0 - (x_n - x_0) - (x_1 - x_0) \\ &= 3x_0 - x_n - x_1 \end{aligned}$$

and

$$\begin{aligned} x_{N+n+1} &= x_N + (x_N - x_{N-n}) + (x_N - x_{N-1}) \\ &= 3x_N - x_{N-n} - x_{N-1}. \end{aligned}$$

Hence, the values of the extension are now dependent on three signal values rather than just two. This translates into a more stable inversion formula. Using this extension results in the following formulas for midpoint folding

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n-1} x_{-n-1} \\ &= r_n x_n + r_{-n-1}(3x_0 - x_1 - x_n) \end{aligned} \tag{8.1}$$

and

$$\begin{aligned} \tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1}(3x_N - x_{N-1} - x_{N-n}). \end{aligned} \tag{8.2}$$

For gridpoint folding, the formulas are

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= r_n x_n + r_{-n}(3x_0 - x_1 - x_{n-1}) \end{aligned} \tag{8.3}$$

and

$$\begin{aligned} \tilde{x}_{N-n} &= r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2} \\ &= r_{n+1} x_{N-n} - r_{-n-1}(3x_N - x_{N-1} - x_{N-n-1}). \end{aligned} \tag{8.4}$$

In matrix notation they become

$$\begin{bmatrix} r_0+2r_{-1} & r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 3r_{-2} & r_1+2r_{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 3r_{-3} & -r_{-3} & r_2-r_{-3} & 0 & 0 & 0 & 0 & 0 \\ 3r_{-4} & -r_{-4} & 0 & r_3-r_{-4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_3+r_{-4} & 0 & r_{-4} & -3r_{-4} \\ 0 & 0 & 0 & 0 & 0 & r_2+r_{-3} & r_{-3} & -3r_{-3} \\ 0 & 0 & 0 & 0 & 0 & 0 & r_1+2r_{-2} & -3r_{-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{-1} & r_0-2r_{-1} \end{bmatrix} x=\tilde{x} \qquad (8.5)$$

for midpoint folding, and

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-1} & r_1-r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-2} & -2r_{-2} & r_2 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-3} & -r_{-3} & -r_{-3} & r_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{-3} & r_3 & r_{-3} & -3r_{-3} \\ 0 & 0 & 0 & 0 & 0 & r_{-2} & r_2+r_{-2} & -3r_{-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 2r_{-1} & r_1-3r_{-1} \end{bmatrix} x = \tilde{x}. \qquad (8.6)$$

for gridpoint folding. When applied to the same test signal of Section 5.2, the sparsity values remain low, but the condition numbers are drastically reduced. Table 8.1 shows these values. Although the condition numbers are still too large to make these methods practical, it illustrates the basic approach for finding new schemes. It may be possible to attain additional stability by using different

Table 8.1: Sparsity and condition numbers of the folding matrices for the left half interval (Left), right half interval (Right), and both sides together (Both).

CO3	$\ell^{0.1}$	Left	Right	Both
Midpoint	47.59	406.51	$6.12 \cdot 10^7$	$6.13 \cdot 10^7$
Gridpoint	48.39	4810.50	$2.82 \cdot 10^8$	$2.82 \cdot 10^8$

signal values to define each extension point, but this is yet to be determined. Of course, the number of signal values used to define each extension point should be limited, since using most of the signal to compensate for boundary effects would be counterproductive. Hence, there is a trade-off. It should be noted that this approach could also be used to improve the condition number for the CP2LFT.

8.2 Adding C^1 Continuity to the CPLFT

A different approach to increasing continuity at the boundaries, and subsequently increasing sparsity is to exploit the results of Section 4.5.1 which showed that the CPLFT simply adds a scalar multiple of the function $h(t)$ to the input signal $x(t)$ in order to periodize it. Now, the function $h(t)$ makes the signal continuous when viewed as a periodic signal, but it does not make the derivative continuous. So the idea is to try to find a variation of $h(t)$ that adds C^1 continuity to the boundary of the signal when viewed in a periodic manner. Figure 8.1 illustrates the idea. Once this variant of $h(t)$ is determined, the periodization process shown in Equations (4.58) and (4.59) can be inverted to determine the corresponding extension, $g(t)$, and consequently the resultant CPLFT equations which are used in practice.

To better illustrate the idea, assume that $h(t)$ is the result of using a continuous odd periodization operator. An example is shown in Figure 8.1(e) which was found by subtracting the original signal from the periodized signal; i.e., $h(t) = W^*\mathbf{1}\tilde{U}_{CO}x - x$ where $\tilde{U}_{CO}$ is the continuous odd folding operator. Now, inverting the process shown in Equations (4.58) and (4.59) to recover $g(t)$ entails applying periodic folding, W, followed by the inverse of the following folding routine

$$\begin{aligned} Ug(t) &= \begin{cases} r\left(\frac{t}{\epsilon}\right) g(t) + r\left(\frac{-t}{\epsilon}\right) g(-t) & \text{if } 0 < t < \epsilon, \\ r\left(\frac{1-t}{\epsilon}\right) g(t) - r\left(\frac{t-1}{\epsilon}\right) g(2-t) & \text{if } 1-\epsilon < t < 1, \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} r\left(\frac{-t}{\epsilon}\right) g(-t) & \text{if } 0 < t < \epsilon, \\ -r\left(\frac{t-1}{\epsilon}\right) g(2-t) & \text{if } 1-\epsilon < t < 1, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

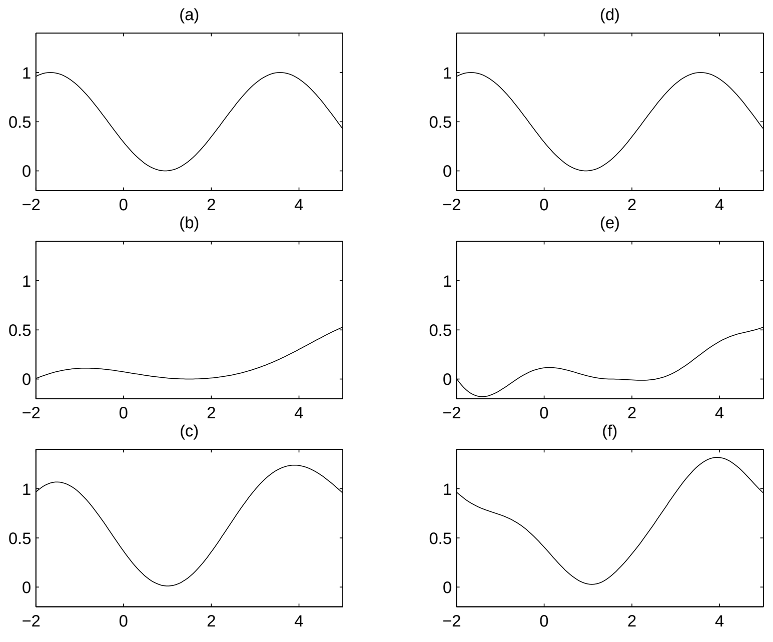

Figure 8.1: Adding the function shown in (b) to the original signal (a) results in the C^0-periodic signal shown in (c). On the other hand, adding the function shown in (e) to the original signal (d) results in the C^1-periodic signal shown in (f).

This will define the function $g(t)$ (Figure 8.2(c)), whose extensions can then be added to the periodic extension of the original signal (Figure 8.2(b)) to reveal the proper extensions to be used (Figure 8.2(a)). From these extensions, the underlying formulas for C^1-periodization can be found. It should be noted that the resulting extensions shown in Figure 8.2(a) match those of the continuous odd extension. The reason for this is because continuous odd extensions were originally used to define $h(t)$ for this example. In practice, different $h(t)$ functions should be used.

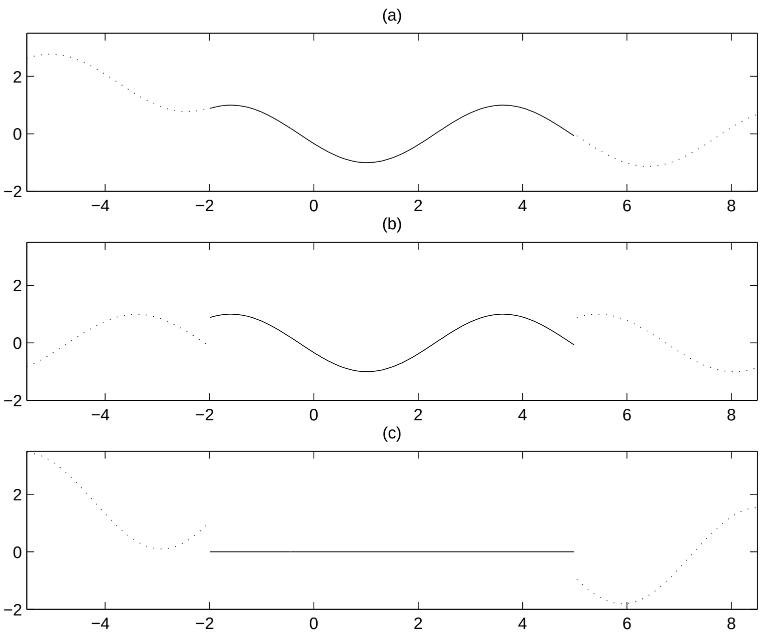

Figure 8.2: (a) A variation of the continuous periodic extension split into the sum of (b) a periodic extension, and (c) a piecewise function.

8.3 Improving the Thresholding Scheme and the Best Basis Criteria

Another idea which shows great potential is that of incorporating the $\tilde{p}_{\mathbf{b}}$ values, defined in Section 5.2, with a new thresholding scheme. For example, given a set of transformed CBLFT coefficients, the current approach for compression is simply to threshold the smallest values. But because of the nonuniform weights, $\frac{1}{p_{\mathbf{x}}\tilde{p}_{\mathbf{b}}}$, applied to these coefficients, throwing away a small transform coefficient may actually have a more severe effect than throwing away a larger one. In addition, since one of the primary goals of these methods is to minimize edge effect, then it makes sense to threshold those coefficients which will have minimal effect on the edges. Thus, if $\mathbf{c}$ is the vector of CBLFT coefficients, and $\tilde{\mathbf{b}}_i$ are the CBLFT basis functions, then thresholding the values $\tilde{c}_i = \frac{c_i}{p_{\mathbf{x}}\tilde{p}_{\mathbf{b}_i}}$ will reduce error. Figure 8.3 shows a comparison of the two thresholding approaches.

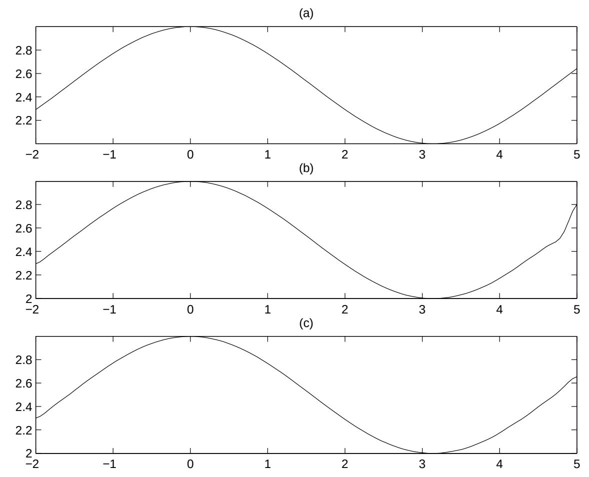

Figure 8.3: (a) Original signal, (b) 4:1 compression when thresholding c_i, (c) 4:1 compression when thresholding $\tilde{c}_i$.

Although this approach works well when applied to the entire signal, it cannot be used to threshold coefficients chosen with a best basis algorithm. It also cannot be used to choose the coefficients in a best basis algorithm. The reason is that it is not an isometric operation. Since the energy is not preserved between a parent subspace and its children, comparing them becomes a bit tricky. But the significant improvement in performance begs further attention.

Appendix A

Abbreviations

This appendix lists some of the Continuous Boundary abbreviations and definitions which are used throughout the thesis.

A.1 Continuous Boundary Abbreviations

Abbreviation	**Definition**
BLFTRC	The Brushlet Transform based on the Real-valued Cosine polarity Local Fourier Transform.
BLFTRMC	The Brushlet Transform based on the Real-valued Cosine polarity Local Fourier Transform with Multiple folding.
BT	The Brushlet Transform.
C06	6-tap coiflet with 2 vanishing moments.
CBBT	The Continuous Boundary Brushlet Transform.
CBLFT	The Continuous Boundary Local Fourier Transform.
CBLTT	The Continuous Boundary Local Trigonometric Transform.
CE1G	The Continuous Even1 Gridpoint folding scheme.
CE1GLFT	The Continuous Even1 Gridpoint Local Fourier Transform.
CE1M	The Continuous Even1 Midpoint folding scheme.
CE1MLFT	The Continuous Even1 Midpoint Local Fourier Transform.
CE2G	The Continuous Even2 Gridpoint folding scheme.

Abbreviation	**Definition**
CE2GLFT	The Continuous Even2 Gridpoint Local Fourier Transform.
CE2M	The Continuous Even2 Midpoint folding scheme.
CE2MLFT	The Continuous Even2 Midpoint Local Fourier Transform.
COLFT	The Continuous Odd Local Fourier Transform.
CO1G	The Continuous Odd1 Gridpoint folding scheme.
CO1GLFT	The Continuous Odd1 Gridpoint Local Fourier Transform.
CO1M	The Continuous Odd1 Midpoint folding scheme.
CO1MLFT	The Continuous Odd1 Midpoint Local Fourier Transform.
CO2G	The Continuous Odd2 Gridpoint folding scheme.
CO2GLFT	The Continuous Odd2 Gridpoint Local Fourier Transform.
CO2M	The Continuous Odd2 Midpoint folding scheme.
CO2M-CE2MBT	The Continuous Odd2 Midpoint folding scheme used for the even polarity boundary, and the Continuous Even2 Midpoint folding scheme used for the odd polarity boundary Brushlet Transform.
CO2M-CE2MLFT	The Continuous Odd2 Midpoint folding scheme used for the even polarity boundary, and the Continuous Even2 Midpoint folding scheme used for the odd polarity boundary Local Fourier Transform.
CO2MBT2	The Continuous Odd2 Midpoint Brushlet Transform. based on DCT-II.
CO2MLCT2	The Continuous Odd2 Midpoint Local Cosine Transform based on DCT-II.
CO2MLFT	The Continuous Odd2 Midpoint Local Fourier Transform.
COLFT	The Continuous Odd Local Fourier Transform.
CP1G	The Continuous Periodic1 Gridpoint folding scheme.
CP1GLFT	The Continuous Periodic1 Gridpoint Local Fourier Transform.
CP1M	The Continuous Periodic1 Midpoint folding scheme.
CP1MLFT	The Continuous Periodic1 Midpoint Local Fourier Transform.
CP2G	The Continuous Periodic2 Gridpoint folding scheme.

Abbreviation	**Definition**
CP2GLFT	The Continuous Periodic2 Gridpoint Local Fourier Transform.
CP2M	The Continuous Periodic2 Midpoint folding scheme.
CP2MLFT	The Continuous Periodic2 Midpoint Local Fourier Transform.
CPLFT	The Continuous Periodic Local Fourier Transform.
DCT	The Discrete Cosine Transform.
DCT-II	The Discrete Cosine Transform (type II boundary conditions).
DCT-IV	The Discrete Cosine Transform (type IV boundary conditions).
DCT2-II	The 2D Discrete Cosine Transform (type II boundary conditions).
DCT2-IV	The 2D Discrete Cosine Transform (type IV boundary conditions).
DCT/DST$\rightarrow$ LCT/LST	Real-valued brushlet transform based on the global Discrete Cosine (Sine) Transform followed by the Local Cosine (Sine) Transform.
DST	The Discrete Sine Transform.
DST-II	The Discrete Sine Transform (type II boundary conditions).
DST-IV	The Discrete Sine Transform (type IV boundary conditions).
DST2-II	The 2D Discrete Sine Transform (type II boundary conditions).
DST2-IV	The 2D Discrete Sine Transform (type IV boundary conditions).
DFT	The Discrete Fourier Transform.
D02	The Haar-Walsh Wavelet packet transform utilizing Daubechies' 2-tap quadrature mirror filter.
FFT	The Fast Fourier Transform.
FFTR	The Real-valued Fast Fourier Transform.
FFTR$\rightarrow$ LCT/LST	Real-valued brushlet transform based on the global Real-valued Fast Fourier Transform followed by the Local Cosine (Sine) Transform.
DST	The Discrete Sine Transform.

Abbreviation	**Definition**
FFTR→ LFTR	Real-valued brushlet transform based on the global Real-valued Fast Fourier Transform followed by the real-valued Local Fourier Transform.
JPEG	The Joint Photographic Experts Group transform.
JPEG2000	The Joint Photographic Experts Group transform based on Wavelet technology.
JPEG-DCT	The Joint Photographic Experts Group transform based on the Discrete Cosine Transform (same as JPEG).
LCT	The Local Cosine Transform.
LCTM	The Local Cosine Transform using multiple folding.
LFT	The standard complex-valued Local Fourier Transform.
LFTR	The Real-valued Local Fourier Transform
LST	The Local Sine Transform.
LTT	The Local Trigonometric Transform.
WP	Wavelet Packets.
WT	The Wavelet Transform.

Appendix B

Supplemental Formulas, Definitions and Proofs

This chapter contains supplemental definitions, analysis and proofs, as well as other supporting material which was referenced within the thesis. The inclusion here will hopefully aid in the understanding of key elements of the CBLTT.

B.1 The 2D Discrete Cosine/Sine Transform

A key ingredient of the 2D LTT transform is the 2D Discrete Cosine Transform type-IV (DCT2-IV) and the Discrete Sine Transform type-IV (DST2-IV). These transforms are closely related to the standard DCT2-II and DST2-II in that they are orthogonal transform and have fast FFT based algorithms associated with them, but they differ primarily in the symmetry of the transform basis functions at the boundaries of their intervals. In the derivation of the DCT (DST), the basis functions are found by numerically solving the differential equation $-u^{''} = \lambda u$ on $[0, \pi]$ using boundary conditions that produce cosines (sines). In particular, $u^{'}(0) = 0$ ($u^{'}(\pi) = 0$) along with Neumann boundary conditions, $u^{'}(\pi) = 0$ ($u^{'}(0) = 0$), or Dirichlet boundary conditions, $u(\pi) = 0$ ($u(\pi) = 0$). Since the discrete solution can be approximated in two primary ways, either centered at midpoints or at gridpoints, four cosine (sine) transforms arise; actually there are more choices if separate centering is allowed at each boundary. But if midpoint centering is employed, then the Dirichlet boundary conditions give rise to the DCT-II (DST-II) basis, whereas the Neumann boundary conditions give rise to the DCT-IV (DST-IV) basis [68]. It should be noted that, in practice, the 2D transforms are applied as a tensor product of their corresponding 1D version; i.e., the 1D version is first applied to the rows, and then to the columns. To be precise, the 2D versions of the DCT2-II (DST2-II) and DCT2-IV (DST2-IV) are provided next.

The DCT2-II of a function $x_{m,n}$ for $m = (0, \ldots, M$ and $n = 0, \ldots, N$ is defined as [2]

$$\begin{aligned} C^{II}_{j,k} &= \frac{2}{N+1} c_j c_k \sum_{m=0}^{M} \sum_{n=0}^{N} x_{m,n} \cos \frac{(m+\frac{1}{2})j\pi}{M+1} \cos \frac{(n+\frac{1}{2})k\pi}{N+1}, \\ & \text{for } j = 0, \ldots, M, \text{ and } k = 0, \ldots, N, \end{aligned} \tag{B.1}$$

where

$$c_k = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } k = 0, \\ 1, & \text{if } k = 1, \ldots, N, \\ 0 & \text{otherwise.} \end{cases} \tag{B.2}$$

Its inversion is given by

$$\begin{aligned} x_{m,n} &= \frac{2}{N+1} \sum_{j=0}^{M} \sum_{k=0}^{N} c_j c_k C^{II}_{j,k} \cos \frac{(m+\frac{1}{2})j\pi}{M+1} \cos \frac{(n+\frac{1}{2})k\pi}{N+1}, \\ & \text{for } m = 0, \ldots, M, \text{ and } n = 0, \ldots, N. \end{aligned} \tag{B.3}$$

Similarly, the DST2-II is defined as [72]

$$\begin{aligned} S^{II}_{j,k} &= \frac{2}{N+1} c_{j+1} c_{k+1} \sum_{m=0}^{M} \sum_{n=0}^{N} x_{m,n} \sin \frac{(m+\frac{1}{2})(j+1)\pi}{M+1} \sin \frac{(n+\frac{1}{2})(k+1)\pi}{N+1}, \\ & \text{for } j = 0, \ldots, M, \text{ and } k = 0, \ldots, N, \end{aligned} \tag{B.4}$$

and its inversion is

$$x_{m,n} = \frac{2}{N+1}\sum_{j=0}^{M}\sum_{k=0}^{N} c_{j+1}c_{k+1}S^{II}_{j,k}\sin\frac{(m+\frac{1}{2})(j+1)\pi}{M+1}\sin\frac{(n+\frac{1}{2})(k+1)\pi}{N+1},$$
$$\text{for } m = 0,\dots,M, \text{ and } n = 0,\dots,N. \quad \text{(B.5)}$$

Now, replacing j with $j+\frac{1}{2}$ and k with $k+\frac{1}{2}$ in the arguments of the cosines on the right hand side of Equations (B.1) and (B.3) yields the formulas for DCT2-IV. Now, replacing $j+1$ with $j+\frac{1}{2}$ and $k+1$ with $k+\frac{1}{2}$ in the arguments of the sines on the right hand side of Equations (B.4) and (B.5) yields the formulas for DST2-IV. They are

$$C^{IV}_{j,k} = \frac{2}{N+1}\sum_{m=0}^{M}\sum_{n=0}^{N} x_{m,n}\cos\frac{(m+\frac{1}{2})(j+\frac{1}{2})\pi}{M+1}\cos\frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1},$$
$$\text{for } j = 0,\dots,M, \text{ and } k = 0,\dots,N \quad \text{(B.6)}$$

which has inversion given by

$$x_{m,n} = \frac{2}{N+1}\sum_{j=0}^{M}\sum_{k=0}^{N} C^{IV}_{j,k}\cos\frac{(m+\frac{1}{2})(j+\frac{1}{2})\pi}{M+1}\cos\frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1},$$
$$\text{for } m = 0,\dots,M, \text{ and } n = 0,\dots,N, \quad \text{(B.7)}$$

and

$$S^{IV}_{j,k} = \frac{2}{N+1}\sum_{m=0}^{M}\sum_{n=0}^{N} x_{n}\sin\frac{(m+\frac{1}{2})(j+\frac{1}{2})\pi}{M+1}\sin\frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1},$$
$$\text{for } j = 0,\dots,M, \text{ and } k = 0,\dots,N, \quad \text{(B.8)}$$

with the following inverse

$$x_{m,n} = \frac{2}{N+1}\sum_{j=0}^{M}\sum_{k=0}^{N} S^{II}_{j,k}\sin\frac{(m+\frac{1}{2})(j+\frac{1}{2})\pi}{M+1}\sin\frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{N+1},$$
$$\text{for } m = 0,\dots,M, \text{ and } n = 0,\dots,N. \quad \text{(B.9)}$$

B.2 Real-Valued FFT in Two Dimensions

The relationship between complex-valued 2D FFT coefficients and real valued coefficients is derived below.

$$\begin{aligned}
\sqrt{MN}x_{k,l} &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\hat{x}_{m,n}e^{2\pi i\left(\frac{mk}{M}+\frac{nl}{N}\right)} \\
&= \sum_{m=0}^{M-1} e^{2\pi i\frac{mk}{M}}\left[\hat{x}_{m,0}+\sum_{n=1}^{\frac{N}{2}-1}\hat{x}_{m,n}e^{2\pi i\frac{nl}{N}}+(-1)^l\hat{x}_{m,\frac{N}{2}}+\sum_{n=\frac{N}{2}+1}^{N-1}\hat{x}_{m,n}e^{2\pi i\frac{nl}{N}}\right] \\
&= \sum_{m=0}^{M-1}\hat{x}_{m,0}e^{2\pi i\frac{mk}{M}}+\sum_{n=1}^{\frac{N}{2}-1}e^{2\pi i\frac{nl}{N}}\sum_{m=0}^{M-1}\hat{x}_{m,n}e^{2\pi i\frac{mk}{M}}+(-1)^l\sum_{m=0}^{M-1}\hat{x}_{m,\frac{N}{2}}e^{2\pi i\frac{mk}{M}} \\
&\quad+\sum_{n=\frac{N}{2}+1}^{N-1}e^{2\pi i\frac{nl}{N}}\sum_{m=0}^{M-1}\hat{x}_{m,n}e^{2\pi i\frac{mk}{M}} \\
&= \hat{x}_{0,0}+2\sum_{m=1}^{\frac{M}{2}-1}\operatorname{Re}(\hat{x}_{m,0})\cos\left(2\pi\frac{mk}{M}\right)+(-1)^k\hat{x}_{\frac{M}{2},0} \\
&\quad-2\sum_{m=\frac{M}{2}+1}^{M-1}\operatorname{Im}(\hat{x}_{m,0})\sin\left(2\pi\frac{mk}{M}\right)+\sum_{n=1}^{\frac{N}{2}-1}\hat{x}_{0,n}e^{2\pi i\frac{nl}{N}}
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{n=1}^{\frac{N}{2}-1} e^{2\pi i\frac{nl}{N}} \sum_{m=1}^{\frac{M}{2}-1} \hat{x}_{m,n} e^{2\pi i\frac{mk}{M}} + (-1)^k \sum_{n=1}^{\frac{N}{2}-1} \hat{x}_{\frac{M}{2},n} e^{2\pi i\frac{nl}{N}} \\
&+\sum_{n=1}^{\frac{N}{2}-1} e^{2\pi i\frac{nl}{N}} \sum_{m=\frac{M}{2}+1}^{M-1} \hat{x}_{m,n} e^{2\pi i\frac{mk}{M}} + (-1)^l \hat{x}_{0,\frac{N}{2}} \\
&+2(-1)^l \sum_{m=1}^{\frac{M}{2}-1} \mathrm{Re}(\hat{x}_{m,\frac{N}{2}}) \cos\left(2\pi\frac{mk}{M}\right) + (-1)^{k+l}\hat{x}_{\frac{M}{2},\frac{N}{2}} \\
&-2(-1)^l \sum_{m=\frac{M}{2}+1}^{M-1} \mathrm{Im}(\hat{x}_{m,\frac{N}{2}}) \sin\left(2\pi\frac{mk}{M}\right) + \sum_{n=\frac{N}{2}+1}^{N-1} \hat{x}_{0,n} e^{2\pi i\frac{nl}{N}} \\
&+\sum_{n=\frac{N}{2}+1}^{N-1} e^{2\pi i\frac{nl}{N}} \sum_{m=1}^{\frac{M}{2}-1} \hat{x}_{m,n} e^{2\pi i\frac{mk}{M}} + (-1)^k \sum_{n=\frac{N}{2}+1}^{N-1} \hat{x}_{\frac{M}{2},n} e^{2\pi i\frac{nl}{N}} \\
&+\sum_{n=\frac{N}{2}+1}^{N-1} e^{2\pi i\frac{nl}{N}} \sum_{m=\frac{M}{2}+1}^{M-1} \hat{x}_{m,n} e^{2\pi i\frac{mk}{M}} \\
= \;& \hat{x}_{0,0} + (-1)^k \hat{x}_{\frac{M}{2},0} + (-1)^l \hat{x}_{0,\frac{N}{2}} + (-1)^{k+l}\hat{x}_{\frac{M}{2},\frac{N}{2}} \\
&+2\sum_{m=1}^{\frac{M}{2}-1} \mathrm{Re}(\hat{x}_{m,0}) \cos\left(2\pi\frac{mk}{M}\right) - 2\sum_{m=\frac{M}{2}+1}^{M-1} \mathrm{Im}(\hat{x}_{m,0}) \sin\left(2\pi\frac{mk}{M}\right) \\
&+2(-1)^l \sum_{m=1}^{\frac{M}{2}-1} \mathrm{Re}(\hat{x}_{m,\frac{N}{2}}) \cos\left(2\pi\frac{mk}{M}\right) - 2(-1)^l \sum_{m=\frac{M}{2}+1}^{M-1} \mathrm{Im}(\hat{x}_{m,\frac{N}{2}}) \sin\left(2\pi\frac{mk}{M}\right) \\
&+2\sum_{n=1}^{\frac{N}{2}-1} \mathrm{Re}(\hat{x}_{0,n}) \cos\left(2\pi\frac{nl}{N}\right) - 2\sum_{n=\frac{N}{2}+1}^{N-1} \mathrm{Im}(\hat{x}_{0,n}) + \sin\left(2\pi\frac{nl}{N}\right) \\
&+2(-1)^k \sum_{n=1}^{\frac{N}{2}-1} \mathrm{Re}(\hat{x}_{\frac{M}{2},n}) \cos\left(2\pi\frac{nl}{N}\right) - 2(-1)^k \sum_{n=\frac{N}{2}+1}^{N-1} \mathrm{Im}(\hat{x}_{\frac{M}{2},n}) \sin\left(2\pi\frac{nl}{N}\right) \\
&+2\sum_{m=1}^{\frac{M}{2}-1}\sum_{n=1}^{\frac{N}{2}-1} \mathrm{Re}(\hat{x}_{m,n}) \left[\cos\left(2\pi\frac{mk}{M}\right)\cos\left(2\pi\frac{nl}{N}\right) - \sin\left(2\pi\frac{mk}{M}\right)\sin\left(2\pi\frac{nl}{N}\right)\right]
\end{aligned}
$$

$$+2\sum_{m=1}^{\frac{M}{2}-1}\sum_{n=\frac{N}{2}+1}^{N-1}\mathrm{Re}(\hat{x}_{m,n})\left[\cos\left(2\pi\frac{mk}{M}\right)\cos\left(2\pi\frac{nl}{N}\right)-\sin\left(2\pi\frac{mk}{M}\right)\sin\left(2\pi\frac{nl}{N}\right)\right]$$

$$+2\sum_{m=\frac{M}{2}+1}^{M-1}\sum_{n=1}^{\frac{N}{2}-1}\mathrm{Im}(\hat{x}_{m,n})\left[\cos\left(2\pi\frac{mk}{M}\right)\sin\left(2\pi\frac{nl}{N}\right)+\sin\left(2\pi\frac{mk}{M}\right)\cos\left(2\pi\frac{nl}{N}\right)\right]$$

$$+2\sum_{m=\frac{M}{2}+1}^{M-1}\sum_{n=\frac{N}{2}+1}^{N-1}\mathrm{Im}(\hat{x}_{m,n})\left[\cos\left(2\pi\frac{mk}{M}\right)\sin\left(2\pi\frac{nl}{N}\right)+\sin\left(2\pi\frac{mk}{M}\right)\cos\left(2\pi\frac{nl}{N}\right)\right]$$

$$= \sum_{m=0}^{\frac{M}{2}}\sum_{n=0}^{\frac{N}{2}}\underbrace{\sqrt{2}\mathrm{Re}(\hat{x}_{m,n})}_{\dagger}\cdot\overbrace{\sqrt{2}\cos\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger}(\star)$$

$$+\sum_{m=1}^{\frac{M}{2}-1}\sum_{n=\frac{N}{2}+1}^{N-1}\underbrace{\sqrt{2}\mathrm{Re}(\hat{x}_{m,n})}_{\dagger}\cdot\overbrace{\sqrt{2}\cos\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger}$$

$$-\sum_{m=\frac{M}{2}+1}^{M-1}\sum_{n=0}^{\frac{N}{2}}\underbrace{\sqrt{2}\mathrm{Im}(\hat{x}_{m,n})}_{\dagger}\cdot\overbrace{\sqrt{2}\sin\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger}$$

$$-\sum_{m=\frac{M}{2}}^{M}\sum_{n=\frac{N}{2}+1}^{N-1}\underbrace{\sqrt{2}\mathrm{Im}(\hat{x}_{m,n})}_{\dagger}\cdot\overbrace{\sqrt{2}\sin\left[2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)\right]}^{\ddagger}(\star\star)$$

It should be noted that every $\sqrt{2}$ in $(\star)$ should be replaced by a 1 when (m,n) equals $(0,0)$, $(\frac{M}{2},0)$, $(0,\frac{N}{2})$, or $(\frac{M}{2},\frac{N}{2})$. Also, when $m=M$ in $(\star\star)$, $m=0$ should be used instead.

In this form, it is easy to see the relationship between the complex and real FFT coefficients. The real-valued coefficients denoted by †, and their corresponding basis function is marked with a ‡. This form also gives rise to a natural ordering of the coefficient table that is similar to the ordering of the complex FFT coefficients (Figure 2.12 and Figure 2.13).

B.3 Fourier Coefficient Rate of Decay for the CPLFT

As defined in section 4.5.1, the addition of a scalar multiple of $h(t) = T_I g(t)$ to a function has the effect of periodizing that function. The manner in which it achieves this can be illustrated by simply taking the Fourier transform of $h(t)$. For illustrative simplicity, let $r(t) = r_{[0]}(t) = \sin\left[\frac{\pi}{4}(1+t)\right]$ and $x(t) = t^n$. Then if $F_b(x)$, $b \in \mathbb{Z}$, is the b^{th} Fourier coefficient of a function x, $F_b[T_I x(t)] = F_b[f(t)] + \Delta x F_b[h(t)]$ where

$$\begin{aligned} F_0[h(t)] &= -2\epsilon \int_0^1 r(t) r(-t) dt \\ &= -2\epsilon \int_0^1 \cos\left(\frac{\pi}{2} t\right) dt \\ &= -\frac{4\epsilon}{\pi}. \end{aligned} \tag{B.10}$$

and

$$
\begin{aligned}
F_b[h(t)] &= \int_0^1 h(t)e^{-2\pi ibt}dt \\
&= \int_0^\epsilon r\left(\frac{-t}{\epsilon}\right)\left[r\left(\frac{-t}{\epsilon}\right) - r\left(\frac{t}{\epsilon}\right)\right]e^{-2\pi ibt}dt \\
&\quad - \int_{1-\epsilon}^1 r\left(\frac{t-1}{\epsilon}\right)\left[r\left(\frac{1-t}{\epsilon}\right) - r\left(\frac{t-1}{\epsilon}\right)\right]e^{-2\pi ibt}dt \\
&= \epsilon\int_0^1 r(-t)[r(-t)-r(t)]e^{-2\pi ib\epsilon t}dt + \epsilon\int_1^0 r(-t)[r(t)+r(-t)]e^{-2\pi ib(1-\epsilon t)}dt \\
&= \epsilon\int_0^1 r(-t)[r(-t)-r(t)]e^{-2\pi ib\epsilon t}dt - \epsilon\int_0^1 r(-t)[r(t)+r(-t)]e^{2\pi ib\epsilon t}dt \\
&= \epsilon\int_0^1 r^2(-t)\left(e^{-2\pi ib\epsilon t} - e^{2\pi ib\epsilon t}\right)dt - \epsilon\int_0^1 r(-t)r(t)\left(e^{-2\pi ib\epsilon t} + e^{2\pi ib\epsilon t}\right)dt \\
&= -2\epsilon\int_0^1 r(-t)r(t)\cos(2\pi ib\epsilon t)dt - 2i\epsilon\int_0^1 r^2(-t)\sin(2\pi ib\epsilon t)dt \\
&= -2\epsilon\int_0^1 \sin\left[\frac{\pi}{4}(1-t)\right]\sin\left[\frac{\pi}{4}(1+t)\right]\cos(2\pi ib\epsilon t)dt \\
&\quad - 2i\epsilon\int_0^1 \sin^2\left[\frac{\pi}{4}(1-t)\right]\sin(2\pi ib\epsilon t)dt \\
&= -\epsilon\int_0^1 \cos\left(\frac{\pi}{2}t\right)\cos(2\pi ib\epsilon t)dt - i\epsilon\int_0^1 \left[1-\sin\left(\frac{\pi}{2}t\right)\right]\sin(2\pi ib\epsilon t)dt \\
&= -\frac{\epsilon}{2}\int_0^1 \cos\left[\pi\left(\frac{1}{2}-2b\epsilon\right)t\right] + \cos\left[\pi\left(\frac{1}{2}-2b\epsilon\right)t\right]dt \\
&\quad - \frac{i\epsilon}{2}\int_0^1 2\sin(2\pi b\epsilon t) - \cos\left[\pi\left(\frac{1}{2}-2b\epsilon\right)t\right] + \cos\left[\pi\left(\frac{1}{2}-2b\epsilon\right)t\right]dt \\
&= -\frac{\epsilon}{2}\left\{\frac{\sin\left[\pi\left(\frac{1}{2}-2b\epsilon\right)\right]}{\pi\left(\frac{1}{2}-2b\epsilon\right)} + \frac{\sin\left[\pi\left(\frac{1}{2}+2b\epsilon\right)\right]}{\pi\left(\frac{1}{2}+2b\epsilon\right)}\right\} \\
&\quad + \frac{i\epsilon}{2}\left\{\frac{2\cos\left(2\pi b\epsilon\right)}{2\pi b\epsilon} + \frac{\sin\left[\pi\left(\frac{1}{2}-2b\epsilon\right)\right]}{\pi\left(\frac{1}{2}-2b\epsilon\right)} - \frac{\sin\left[\pi\left(\frac{1}{2}+2b\epsilon\right)\right]}{\pi\left(\frac{1}{2}+2b\epsilon\right)}\right\} \\
&= \frac{\epsilon\cos(2\pi b\epsilon)}{\pi}\left(\frac{1}{4b\epsilon-1} - \frac{1}{4b\epsilon+1}\right) \\
&\quad + i\left[\frac{\cos(2\pi b\epsilon)-1}{2\pi b} - \frac{\epsilon\cos(2\pi b\epsilon)}{\pi}\left(\frac{1}{4b\epsilon-1} + \frac{1}{4b\epsilon+1}\right)\right] \\
&= \frac{2\epsilon\cos(2\pi b\epsilon)}{\pi(16b^2\epsilon^2-1)} - \frac{i}{2\pi b}\left[1 + \frac{\cos(2\pi b\epsilon)}{b(16b^2\epsilon^2-1)}\right]. \qquad \text{(B.11)}
\end{aligned}
$$

B.4 Alternate Versions of the CBLTT

This section lists a few of the alternate versions of the CBLTT that have been derived and analyzed the earlier chapters. They constitute only a small sample of all of the possible extensions that can be used. Only the formulas for folding and unfolding are derived here. Equations (5.1) and (5.2) must be used in conjunction with them in order to achieve the isometry.

B.4.1 The Even2 Midpoint Extension

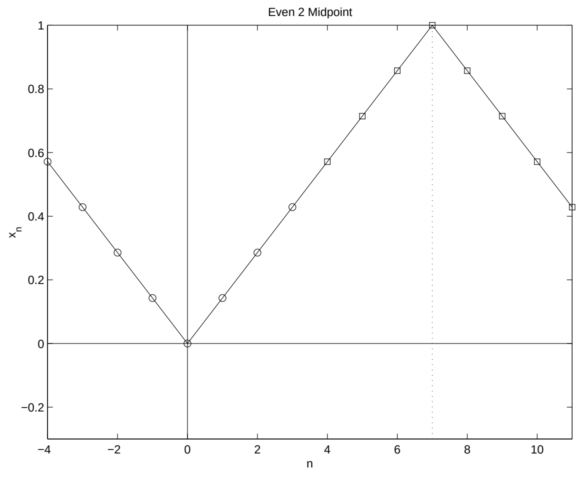

Figure B.1: Example of the Even2 Midpoint extension with $N = 7$.

Using Figure B.1, the values for the extension to the left of x_0 are given by

$$x_{-n} = x_n \tag{B.12}$$

and the values of the extension to the right of x_N are

$$x_{N+n} = x_{N-n}. \tag{B.13}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n-1} x_{-n-1} \\ &= r_n x_n + r_{-n-1} x_{n+1}\end{aligned} \tag{B.14}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1} x_{N-n-1}. \end{aligned} \tag{B.15}$$

Solving for x in Equations (B.14) and (B.15) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - r_{-n-1} x_{n+1}}{r_n} \tag{B.16}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1} x_{N-n-1}}{r_n}. \tag{B.17}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} r_0 & r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_1 & r_{-2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_2 & r_{-3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_3 & r_{-4} & 0 & 0 & 0 \\ 0 & 0 & 0 & -r_{-4} & r_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -r_{-3} & r_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -r_{-2} & r_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -r_{-1} & r_0 \end{bmatrix} x = \tilde{x}. \tag{B.18}$$

Now this approach yields implicit inversion formulas when full folding is employed. And as with all of these methods, the above matrix should not be inverted, but rather an explicit equation should be derived for computational efficiency. This can be achieved by noticing that the middle two equations in Equation (B.18) have two unknowns. So simply solving for the middle two values $x_{\frac{N-1}{2}}$ and $x_{\frac{N+1}{2}}$ in these two equations yields the following explicit formulas

$$x_{\frac{N-1}{2}} = r_{\frac{N-1}{2}} \tilde{x}_{\frac{N-1}{2}} - r_{-\frac{N+1}{2}} \tilde{x}_{\frac{N+1}{2}} \tag{B.19}$$

and

$$x_{\frac{N+1}{2}} = r_{\frac{N-1}{2}} \tilde{x}_{\frac{N+1}{2}} + r_{-\frac{N+1}{2}} \tilde{x}_{\frac{N-1}{2}} \tag{B.20}$$

Once these values are computed, they can simply be used in Equations (B.16) and (B.17) in a recursive manner.

B.4.2 The Even2 Gridpoint Extension

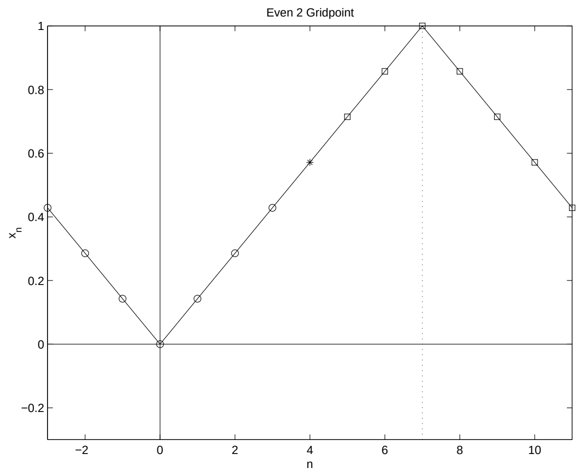

Figure B.2: Example of the Even2 Gridpoint extension with $N = 7$.

As can be seen in Figure B.2, the values for the extension to the left of x_0 are given by

$$x_{-n} \quad = \quad x_n \tag{B.21}$$

and the values of the extension to the right of x_N are

$$x_{N+n} \quad = \quad x_{N-n}. \tag{B.22}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= (r_n + r_{-n-1}) x_n \end{aligned} \tag{B.23}$$

and at the right hand edge as

$$\begin{aligned} \tilde{x}_{N-n} &= r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2} \\ &= r_{n+1} x_{N-n} - r_{-n-1} x_{N-n-2}. \end{aligned} \tag{B.24}$$

Solving for x in Equations (B.23) and (B.24) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n}{r_n + r_{-n}} \tag{B.25}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1}x_{N-n-2}}{r_{n+1}}. \tag{B.26}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_1 + r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_2 + r_{-2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_3 + r_{-3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -r_{-3} & 0 & r_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -r_{-2} & 0 & r_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -r_{-1} & 0 & r_1 \end{bmatrix} x = \tilde{x}. \tag{B.27}$$

Since full folding yields implicit inversion formulas, an explicit equation should be derived for computational efficiency. This can be achieved by noticing that the left hand side can be inverted. So as long as this side is unfolded first, then the right hand side is defined and Equation (B.26) can be explicitly solved in a recursive manner.

B.4.3 The Continuous Odd1 Gridpoint Extension

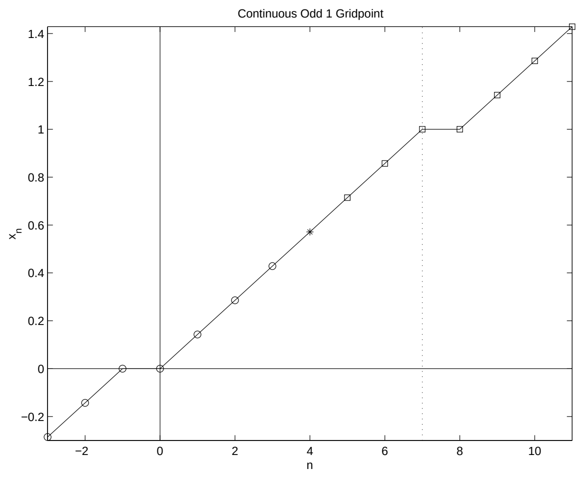

Figure B.3: Example of the Continuous Odd1 Gridpoint extension with $N = 7$.

As Figure B.3 shows, the values for the extension to the left of x_0 are given by

$$x_{-n-1} = 2x_0 - x_n \tag{B.28}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} = 2x_N - x_{N-n}. \tag{B.29}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= r_n x_n + r_{-n}(2x_0 - x_{n-1})\end{aligned} \tag{B.30}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2} \\ &= r_{n+1} x_{N-n} - r_{-n-1}(2x_N - x_{N-n-1}).\end{aligned} \tag{B.31}$$

Solving for x in Equations (B.30) and (B.31) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - 2r_{-n}x_0 + r_{-n}x_{n-1}}{r_n} \tag{B.32}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + 2r_{-n-1}x_N - r_{-n-1}x_{N-n-1}}{r_{n+1}}. \tag{B.33}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-1} & r_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-2} & -r_{-2} & r_2 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-3} & 0 & -r_{-3} & r_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{-3} & r_3 & 0 & -2r_{-3} \\ 0 & 0 & 0 & 0 & 0 & r_{-2} & r_2 & -2r_{-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{-1} & r_1 - 2r_{-1} \end{bmatrix} x = \tilde{x}. \tag{B.34}$$

The inversion formulas are implicit for the right hand side. But by simply applying Gaussian Elimination, an explicit formula for x_N is found to be

$$\begin{aligned} a &= \tilde{x}_N - \tfrac{r_{-1}}{r_2}\left(\tilde{x}_{N-1} - \tfrac{r_{-2}}{r_3}\left(\ldots\left(\tilde{x}_{N-R+3} - \tfrac{r_{-R+2}}{r_{R-1}}\left(\tilde{x}_{N-R+2} - r_{-R+1}x_{N-R+1}\right)\right)\ldots\right)\right) \\ b &= r_1 - 2r_{-1}\left(1 - \tfrac{r_{-2}}{r_2}\left(1 - \tfrac{r_{-3}}{r_3}\left(\ldots\left(1 - \tfrac{r_{-R+2}}{r_{R-2}}\left(1 - \tfrac{r_{-R+1}}{r_{R-1}}\right)\right)\ldots\right)\right)\right) \\ x_N &= \tfrac{a}{b}. \end{aligned} \tag{B.35}$$

With this defined, it can be used in Equation (B.33) in a recursive manner to give an explicit solution.

B.4.4 The Continuous Odd2 Midpoint Extension

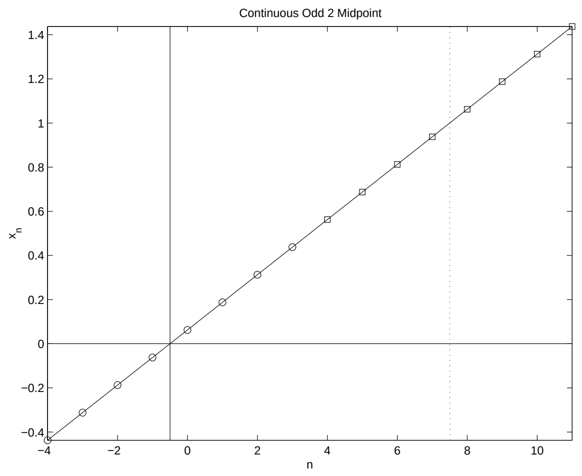

Figure B.4: Example of the Continuous Odd2 Midpoint extension with $N = 7$.

According to Figure B.4, the values for the extension to the left of x_0 are given by

$$x_{-n} = 2x_0 - x_n \tag{B.36}$$

and the values of the extension to the right of x_N are

$$x_{N+n} = 2x_N - x_{N-n}. \tag{B.37}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n-1} x_{-n-1} \\ &= r_n x_n + r_{-n}(2x_0 - x_{n+1}) \qquad \text{(B.38)}\end{aligned}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1}(2x_N - x_{N-n-1}). \qquad \text{(B.39)}\end{aligned}$$

Solving for x in Equations (B.38) and (B.39) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - r_{-n-1}(2x_0 - x_{n+1})}{r_n} \qquad \text{(B.40)}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1}(2x_N - x_{N-n-1})}{r_n}. \qquad \text{(B.41)}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} r_0 + 2r_{-1} & -r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-2} & r_1 & -r_{-2} & 0 & 0 & 0 & 0 & 0 \\ 2r_{-3} & 0 & r_2 & -r_{-3} & 0 & 0 & 0 & 0 \\ 2r_{-4} & 0 & 0 & r_3 & -r_{-4} & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{-4} & r_3 & 0 & 0 & -2r_{-4} \\ 0 & 0 & 0 & 0 & r_{-3} & r_2 & 0 & -2r_{-3} \\ 0 & 0 & 0 & 0 & 0 & r_{-2} & r_1 & -2r_{-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{-1} & r_0 - 2r_{-1} \end{bmatrix} x = \tilde{x}. \qquad \text{(B.42)}$$

Applying Gaussian Elimination to this matrix yields an explicit formula for x_0

$$\begin{aligned} a &= \tilde{x}_0 + \tfrac{r_{-1}}{r_1}\left(\tilde{x}_a + \tfrac{r_{-2}}{r_2}\left(\ldots\left(\tilde{x}_{R-2} + \tfrac{r_{-R+1}}{r_{R-1}}\left(\tilde{x}_{R-1} + r_{-R}x_R\right)\right)\ldots\right)\right) \\ b &= r_0 + 2r_{-1}\left(1 + \tfrac{r_{-2}}{r_1}\left(1 + \tfrac{r_{-3}}{r_2}\left(\ldots\left(1 + \tfrac{r_{-R+1}}{r_{R-2}}\left(1 + \tfrac{r_{-R}}{r_{R-1}}\right)\right)\ldots\right)\right)\right) \\ x_0 &= \tfrac{a}{b}. \qquad \text{(B.43)}\end{aligned}$$

and for x_N

$$\begin{aligned} a &= \tilde{x}_N - \tfrac{r_{-1}}{r_1}\left(\tilde{x}_{N-1} - \tfrac{r_{-2}}{r_2}\left(\ldots\left(\tilde{x}_{N-R+2} - \tfrac{r_{-R+1}}{r_{R-1}}\left(\tilde{x}_{N-R+1} - r_{-R}x_{N-R}\right)\right)\ldots\right)\right) \\ b &= r_0 - 2r_{-1}\left(1 - \tfrac{r_{-2}}{r_1}\left(1 - \tfrac{r_{-3}}{r_2}\left(\ldots\left(1 - \tfrac{r_{-R+1}}{r_{R-2}}\left(1 - \tfrac{r_{-R}}{r_{R-1}}\right)\right)\ldots\right)\right)\right) \\ x_N &= \tfrac{a}{b}. \end{aligned} \tag{B.44}$$

These can then be used recursively in Equation (B.41) to give an explicit solution.

B.4.5 The Continuous Odd2 Gridpoint Extension

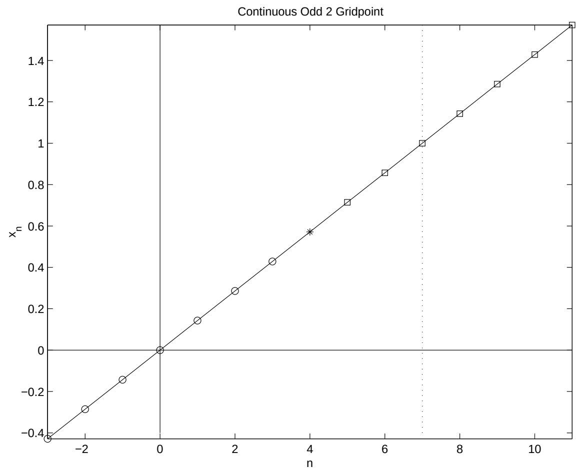

Figure B.5: Example of the Continuous Odd2 Gridpoint extension with $N = 7$.

From Figure B.5, the values for the extension to the left of x_0 are given by

$$x_{-n} = 2x_0 - x_n \tag{B.45}$$

and the values of the extension to the right of x_N are

$$x_{N+n} = 2x_N - x_{N-n}. \tag{B.46}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= (r_n - r_{-n})x_n + 2r_{-n}x_0 \end{aligned} \tag{B.47}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_{n+1}x_{N-n} - r_{-n-1}x_{N+n+2} \\ &= r_{n+1}x_{N-n} - r_{-n-1}(2x_N - x_{N-n-2}). \end{aligned} \tag{B.48}$$

Solving for x in Equations (B.47) and (B.48) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - 2r_{-n}x_0}{r_n - r_{-n}} \tag{B.49}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + 2r_{-n-1}x_N - r_{-n-1}x_{N-n-2}}{r_{n+1}}. \tag{B.50}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-1} & r_1 - r_{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 2r_{-2} & 0 & r_2 - r_{-2} & 0 & 0 & 0 & 0 & 0 \\ 2r_{-3} & 0 & 0 & r_3 - r_{-3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{-3} & 0 & r_3 & 0 & -2r_{-3} \\ 0 & 0 & 0 & 0 & r_{-2} & 0 & r_2 & -2r_{-2} \\ 0 & 0 & 0 & 0 & 0 & r_{-1} & 0 & r_1 - 2r_{-1} \end{bmatrix} x = \tilde{x}. \tag{B.51}$$

For computational efficiency, applying Gaussian Elimination to the right hand side yields But by simply applying Gaussian Elimination, an explicit formula for x_N is found to be

$$\begin{aligned} a &= \tilde{x}_N - \tfrac{r_{-1}}{r_3}\left(\tilde{x}_{N-1} - \tfrac{r_{-3}}{r_5}\left(\ldots\left(\tilde{x}_{N-R+4} - \tfrac{r_{-R+3}}{r_{R-1}}\left(\tilde{x}_{N-R+2} - r_{-R+1}x_{N-R}\right)\right)\ldots\right)\right) \\ b &= r_1 - 2r_{-1}\left(1 - \tfrac{r_{-3}}{r_3}\left(1 - \tfrac{r_{-5}}{r_5}\left(\ldots\left(1 - \tfrac{r_{-R+3}}{r_{R-3}}\left(1 - \tfrac{r_{-R+1}}{r_{R-1}}\right)\right)\ldots\right)\right)\right) \\ x_N &= \tfrac{a}{b}. \end{aligned} \tag{B.52}$$

Using this in Equation (B.50) in a recursive manner gives an explicit solution.

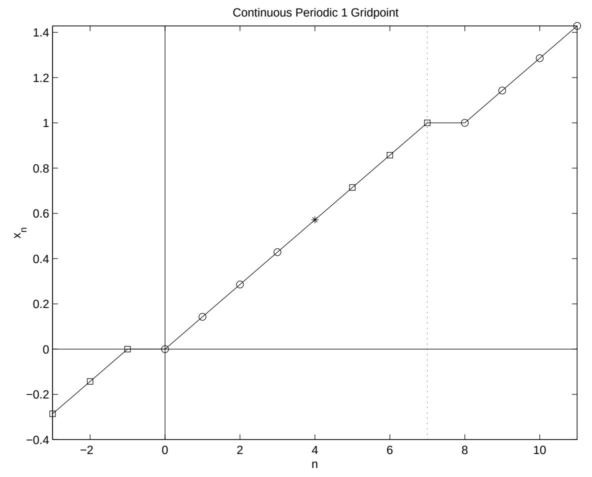

Figure B.6: Example of the Continuous Periodic1 Gridpoint extension with $N = 7$.

B.4.6 The Continuous Periodic1 Gridpoint Extension

If a signal is extended according to Figure B.6, then the values of the extension to the left of x_0 are given by

$$x_{-n-1} = x_{N-n} + x_0 - x_N \tag{B.53}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} = x_n + x_N - x_0. \tag{B.54}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= r_n x_n + r_{-n}(x_{N-n+1} + x_0 - x_N) \end{aligned} \tag{B.55}$$

and at the right hand edge as

$$\begin{aligned} \tilde{x}_{N-n} &= r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2} \\ &= r_{n+1} x_{N-n} - r_{-n-1}(x_{n+1} + x_N - x_0). \end{aligned} \tag{B.56}$$

Solving for x in Equations (B.55) and (B.56) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - r_{-n}(x_{N-n+1} + x_0 - x_N)}{r_n} \tag{B.57}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1}(x_{n+1} + x_N - x_0)}{r_{n+1}}. \tag{B.58}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-1} & r_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-2} & 0 & r_2 & 0 & 0 & 0 & r_{-2} & -r_{-2} \\ r_{-3} & 0 & 0 & r_3 & 0 & r_{-3} & 0 & -r_{-3} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ r_{-3} & 0 & 0 & -r_{-3} & 0 & r_3 & 0 & -r_{-3} \\ r_{-2} & 0 & -r_{-2} & 0 & 0 & 0 & r_2 & -r_{-2} \\ r_{-1} & -r_{-1} & 0 & 0 & 0 & 0 & 0 & r_1 - r_{-1} \end{bmatrix} x = \tilde{x}. \tag{B.59}$$

In order to invert this, first note that it is easy to recover x_0, x_1 and x_N

$$x_0 = \frac{\tilde{x}_0}{2r_0}, \tag{B.60}$$

$$\begin{aligned} x_1 &= \frac{\tilde{x}_1 - r_{-1}x_0}{r_1} \\ &= \frac{2r_0\tilde{x}_1 - r_{-1}\tilde{x}_0}{2r_0r_1} \end{aligned} \tag{B.61}$$

and

$$\begin{aligned} x_N &= \frac{\tilde{x}_N - r_{-1}x_0 + r_{-1}x_1}{r_1 - r_{-1}} \\ &= \frac{2r_0r_1\tilde{x}_n - r_{-1}[(r_1 + r_{-1})\tilde{x}_0 - 2r_0\tilde{x}_1}{2r_0(r_1 - r_{-1})}. \end{aligned} \tag{B.62}$$

Using these results, and exploiting the symmetry in Equation (B.59), generalized inversion formulas for $x_2, \ldots, x_{N-1}$ are found by simultaneously solving a system of equations derived from rows n and $N-n+1$ for $n = 2, \ldots, R$ in Equation (4.48).

These two equations have two unknowns, and so the unique solution is found to be

$$x_n = r_n\tilde{x}_n - r_{-n}\tilde{x}_{N-n+1} - r_{-n}[r_n - r_{-n}][x_0 - x_N)] \tag{B.63}$$

and

$$x_{N-n+1} = r_{-n}\tilde{x}_n + r_{-n}\tilde{x}_{N-n+1} - r_{-n}[r_n + r_{-n}][x_0 - x_N)]. \tag{B.64}$$

B.4.7 The Continuous Periodic2 Midpoint Extension

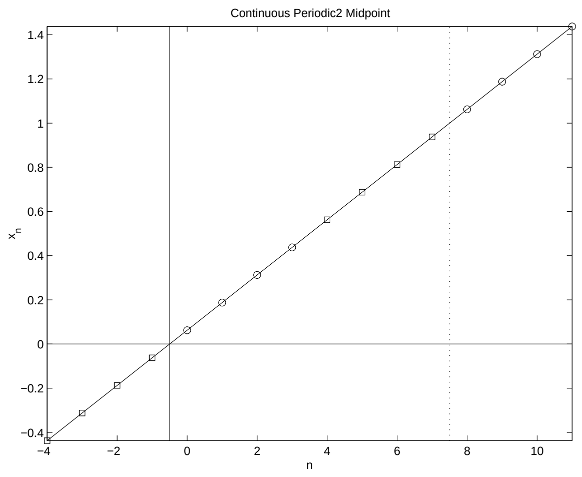

Figure B.7: Example of the Continuous Periodic2 Midpoint extension with $N = 7$.

As Figure B.7 illustrates, the values for the extension to the left of x_0 are given by

$$x_{-n-1} \quad = \quad x_{N-n-1} + x_0 - x_N \tag{B.65}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} \quad = \quad x_{n+1} + x_N - x_0. \tag{B.66}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned}\tilde{x}_n &= r_n x_n + r_{-n-1}x_{-n-1} \\ &= r_n x_n + r_{-n-1}(x_{N-n+1} + x_0 - x_N)\end{aligned} \tag{B.67}$$

and at the right hand edge as

$$\begin{aligned}\tilde{x}_{N-n} &= r_n x_{N-n} - r_{-n-1} x_{N+n+1} \\ &= r_n x_{N-n} - r_{-n-1}(x_{n+1} + x_N - x_0). \end{aligned} \quad \text{(B.68)}$$

Solving for x in Equations (B.67) and (B.68) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - r_{-n-1}(x_{N-n+1} + x_0 - x_N)}{r_n} \quad \text{(B.69)}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1}(x_{n+1} + x_N - x_0)}{r_n}. \quad \text{(B.70)}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} r_{-1}+r_0 & 0 & 0 & 0 & 0 & 0 & r_{-1} & -r_{-1} \\ r_{-2} & r_1 & 0 & 0 & 0 & r_{-2} & 0 & -r_{-2} \\ r_{-3} & 0 & r_2 & 0 & r_{-3} & 0 & 0 & -r_{-3} \\ r_{-4} & 0 & 0 & r_3+r_{-4} & 0 & 0 & 0 & -r_{-4} \\ r_{-4} & 0 & 0 & 0 & r_3-r_{-4} & 0 & 0 & -r_{-4} \\ r_{-3} & 0 & 0 & -r_{-3} & 0 & r_2 & 0 & -r_{-3} \\ r_{-2} & 0 & -r_{-2} & 0 & 0 & 0 & r_1 & -r_{-2} \\ r_{-1} & -r_{-1} & 0 & 0 & 0 & 0 & 0 & r_0-r_{-1} \end{bmatrix} x = \tilde{x}. \quad \text{(B.71)}$$

This matrix can be inverted via gaussian elimination in the following manner. For a general version of this matrix of size N, multiply the first row by $-\frac{r_1}{r_{-1}}$ and add it to row $N-1$ to eliminate the r_1 term in row $N-1$. Then multiply this modified row by $\frac{r_2}{r_{-2}}$ and add it to row 3 in order to eliminate the r_2 term from row 3. Repeat this process in the following order for rows $5 \to N-3 \to 7 \to N-5 \to \ldots \to \frac{N}{2}+1$. This will cause all entries in row $\frac{N}{2}+1$, except for the first and last entry, to be eliminated. Now, repeat this process again, but start with row N and proceed according to the following order, $N \to 2 \to N-2 \to 4 \to N-4 \to \ldots \to \frac{N}{2}$. This will eliminate all entries in row $\frac{N}{2}$ except for the first and last. In other words, if R is the radius of the rising cutoff function, then rows $\frac{N}{2}$ and $\frac{N}{2}+1$ become

$$\begin{bmatrix} a & 0 & \dots & 0 & b \\ c & 0 & \dots & 0 & d \end{bmatrix} \tag{B.72}$$

where

$$\begin{aligned}
a &= \frac{r_{R-1}+r_{-R}}{r_{-R+1}}\left(-\frac{r_{R-2}}{r_{-R+2}}\left(\frac{r_{R-3}}{r_{-R+3}}\left(\dots\left(-\frac{r_2}{r_{-2}}\left(r_1+r_{-2}\right)+r_{-3}\right)\right.\right.\right. \\
&\quad \left.\left.\left.\dots\right)+r_{-R+2}\right)+r_{-R+1}\right)+r_{-R} \\
b &= \frac{r_{R-1}-r_{-R}}{r_{-R+1}}\left(-\frac{r_{R-2}}{r_{-R+2}}\left(\frac{r_{R-3}}{r_{-R+3}}\left(\dots\left(-\frac{r_2}{r_{-2}}\left(-r_1\frac{r_{-1}-r_0}{r_{-1}}-r_{-2}\right)-r_{-3}\right)\right.\right.\right. \\
&\quad \left.\left.\left.\dots\right)-r_{-R+2}\right)-r_{-R+1}\right)-r_{-R} \\
c &= -\frac{r_{R-1}+r_{-R}}{r_{-R+1}}\left(\frac{r_{R-2}}{r_{-R+2}}\left(-\frac{r_{R-3}}{r_{-R+3}}\left(\dots\left(\frac{r_2}{r_{-2}}\left(-r_1\frac{r_{-1}+r_0}{r_{-1}}+r_{-2}\right)+r_{-3}\right)\right.\right.\right. \\
&\quad \left.\left.\left.\dots\right)+r_{-R+2}\right)+r_{-R+1}\right)+r_{-R} \\
d &= -\frac{r_{R-1}-r_{-R}}{r_{-R+1}}\left(\frac{r_{R-2}}{r_{-R+2}}\left(-\frac{r_{R-3}}{r_{-R+3}}\left(\dots\left(\frac{r_2}{r_{-2}}\left(r_1-r_{-2}\right)-r_{-3}\right)\right.\right.\right. \\
&\quad \left.\left.\left.\dots\right)-r_{-R+2}\right)-r_{-R+1}\right)-r_{-R}.
\end{aligned}$$

Thus, solving this system of two equations in two unknowns will yield x_0 and x_N which can be used to recover the remainder of the signal. Unfortunately, a problem arises. Upon inspection of $a, b, c,$ and d, it is clear that the first term in each number (the term which is the product of $\frac{r_{i-1}}{r_{-i}}$) will dominate for large N. Since $\frac{r_i}{r_{-i-1}} > 1$ for all $0 \le i < R$, then this term grows exponentially with N. For example, for $N = 128$, a, b, c and d are all greater than $1.79 \cdot 10^{308}$, the largest number representable using IEEE double precision arithmetic. So overflow destroys the determinant calculation and causes this system to become singular. Even without this problem, though, it is clear that for large N, $a \approx -c$ and $b \approx -d$ causing the determinant to equal 0; i.e., $ad - bc \to 0$ as $N \to \infty$ causing the matrix to become singular.

Since the condition number of this matrix is so large (see Table 5.3), the method is not practical for use. An interesting side note, though, is that the folding procedure for each half interval is not singular when taken alone. Therefore it might be possible to use one of them in a hybrid scheme, in combination with another method.

B.4.8 The Continuous Periodic2 Gridpoint Extension

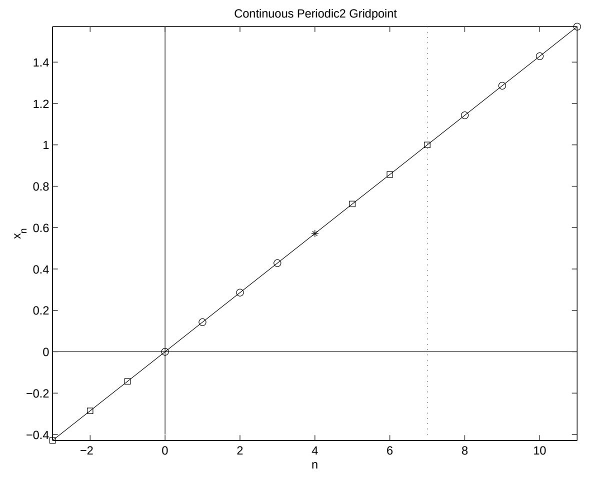

Figure B.8: Example of the Continuous Periodic2 Gridpoint extension with $N = 7$.

As can be seen in Figure B.8, the values for the extension to the left of x_0 are given by

$$x_{-n-1} = x_{N-n-1} + x_0 - x_N \tag{B.73}$$

and the values of the extension to the right of x_N are

$$x_{N+n+1} = x_{n+1} + x_N - x_0. \tag{B.74}$$

Thus folding at the left hand edge is defined as

$$\begin{aligned} \tilde{x}_n &= r_n x_n + r_{-n} x_{-n} \\ &= r_n x_n + r_{-n}(x_{N-n} + x_0 - x_N) \end{aligned} \tag{B.75}$$

and at the right hand edge as

$$\begin{aligned} \tilde{x}_{N-n} &= r_{n+1} x_{N-n} - r_{-n-1} x_{N+n+2} \\ &= r_{n+1} x_{N-n} - r_{-n-1}(x_{n+2} + x_N - x_0). \end{aligned} \tag{B.76}$$

Solving for x in Equations (B.75) and (B.76) yields the following formula for inverse folding at the left hand side

$$x_n = \frac{\tilde{x}_n - r_{-n}(x_{N-n} + x_0 - x_N)}{r_n} \tag{B.77}$$

and at the right hand side

$$x_{N-n} = \frac{\tilde{x}_{N-n} + r_{-n-1}(x_{n+2} + x_N - x_0)}{r_{n+1}}. \tag{B.78}$$

Graphically, this can be represented as a matrix with the following structure

$$\begin{bmatrix} 2r_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{-1} & r_1 & 0 & 0 & 0 & 0 & r_{-1} & -r_{-1} \\ r_{-2} & 0 & r_2 & 0 & 0 & r_{-2} & 0 & -r_{-2} \\ r_{-3} & 0 & 0 & r_3 & r_{-3} & 0 & 0 & -r_{-3} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ r_{-3} & 0 & 0 & 0 & -r_{-3} & r_3 & 0 & -r_{-3} \\ r_{-2} & 0 & 0 & -r_{-2} & 0 & 0 & r_2 & -r_{-2} \\ r_{-1} & 0 & -r_{-1} & 0 & 0 & 0 & 0 & r_1 - r_{-1} \end{bmatrix} x = \tilde{x}. \tag{B.79}$$

As calculated in Section 5.2, the condition number is too large to make this method usable. Hence, the explicit inversion formula is not derived.

B.5 Segmentation of a Standard Basis Function Using CPLFT and LFT

To better understand the segmentation differences between LFT and CBLFT, the following simple case was presented in Section 5.4.1. Segmenting a standard basis function

$$x_n = \begin{cases} 1 & \text{if } n = k, \\ 0 & \text{otherwise} \end{cases} \tag{B.80}$$

for $0 \leq n < M$, using the complex-valued LFT and the complex valued CPLFT results in the partition patterns shown in Figures 5.20 and 5.21. For illustrative purposes, CPLFT was used, although any CBLFT variation could be employed.

In addition, complex-valued versions of both methods were chosen in order to lighten the notational burden and simplify the analysis. The following proposition and subsequent analysis illustrate and quantify the differences between the two methods.

Proposition B.5.1. *(i) If x is defined according to Equation* (B.80)*, and analyzed with the complex-valued LFT using ℓ^1 and full folding, then the bottom level of the decomposition will always be chosen,* regardless of the choice of k. *(ii) If, on the other hand, x is analyzed using the complex-valued CPLFT with ℓ^1 and full folding, then the smallest interval containing the bump will always be chosen, while the largest intervals will be chosen for the regions which have zero energy.*

Proof. (i) For LFT, the coefficients for level 0 are found by periodizing level 0 and then taking a FFT. That is

$$\begin{aligned}\hat{x}_m^{(0,0)} &= \frac{1}{\sqrt{M}}\sum_{n=0}^{M-1} x_n e^{-2\pi i \frac{mn}{M}} \\ &= \frac{1}{\sqrt{M}} e^{-2\pi i \frac{mk}{M}}. \end{aligned} \tag{B.81}$$

The superscript, (i,j), indicates the level of the decomposition, i, and the subspace index, j, when counting from the left; that is, $(2,3)$ is the rightmost subspace of the level 2 decomposition. From Equation (B.81), the sparsity estimate for level 0, using ℓ^1 for simplicity, is

$$\begin{aligned}\|\hat{x}_m^{(0,0)}\|_{\ell^p} &= \left(\sum_{m=0}^{M-1}\left|\frac{1}{\sqrt{M}}e^{-2\pi i\frac{mk}{M}}\right|^p\right)^{\frac{1}{p}} \\ &= \left[M\left(\frac{1}{\sqrt{M}}\right)^p\right]^{\frac{1}{p}} \\ &= M^{\frac{2-p}{2p}} \\ &= \sqrt{M} \text{ for } p = 1. \end{aligned} \tag{B.82}$$

For level 1, folding yields

$$Ux_n = \begin{cases} r_- & \text{if } n = k, \\ r_+ & \text{if } n = M - k - 1, \\ 0 & \text{otherwise} \end{cases} \tag{B.83}$$

where $r_- = r_{-k}$ and $r_+ = r_{k-1}$ are used for simplicity. It should be noted that full folding is employed so that the rising cutoff function for this level has a radius of $\frac{M}{4}$. Periodized unfolding gives

$$Tx_n = \begin{cases} r_+r_- & \text{if } n = k, \\ r_-^2 & \text{if } n = \frac{M}{2} - k - 1, \\ -r_+r_- & \text{if } n = \frac{M}{2} + k, \\ r_+^2 & \text{if } n = M - k - 1. \end{cases} \tag{B.84}$$

Again, for simplicity, let $a = r_+r_-$, $b = r_-^2$, $c = -r_+r_-$ and $d = r_+^2$. Then on $0 \leq n < \frac{M}{2}$

$$\begin{aligned} \hat{x}_m^{(1,0)} &= \sqrt{\frac{2}{M}} \sum_{n=0}^{M-1} x_n e^{-2\pi i \frac{mn}{M}} \\ &= \sqrt{\frac{2}{M}} \left[a e^{-2\pi i \frac{2mk}{M}} + b e^{-2\pi i \frac{2m\left(\frac{M}{2}-k-1\right)}{M}} \right] \\ &= \sqrt{\frac{2}{M}} \left(\left\{ a \cos\left(-4\pi \frac{mk}{M}\right) + b \cos\left[-4\pi \frac{m\left(\frac{M}{2}-k-1\right)}{M}\right] \right\} \right. \\ &\quad \left. + i \left\{ a \sin\left(-4\pi \frac{mk}{M}\right) + b \sin\left[-4\pi \frac{m\left(\frac{M}{2}-k-1\right)}{M}\right] \right\} \right) \end{aligned} \tag{B.85}$$

and

$$\begin{aligned}
\left|\hat{x}_m^{(1,0)}\right|^2 &= \frac{2}{M}\left\{a^2\cos^2\left(-4\pi\frac{mk}{M}\right)+a^2\sin^2\left(-4\pi\frac{mk}{M}\right)\right. \\
&\quad + 2ab\cos\left(-4\pi\frac{mk}{M}\right)\cos\left[-4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right] \\
&\quad + 2ab\sin\left(-4\pi\frac{mk}{M}\right)\sin\left[-4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right] \\
&\quad \left.+b^2\cos\left[-4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right]+b^2\sin\left[-4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right]\right\} \\
&= \frac{2}{M}\left(a^2+b^2+2ab\left\{\cos\left(-4\pi\frac{mk}{M}\right)\cos\left[-4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right]\right.\right. \\
&\quad \left.\left.+\sin\left(-4\pi\frac{mk}{M}\right)\sin\left[-4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right]\right\}\right) \\
&= \frac{2}{M}\left\{a^2+b^2+2ab\cos\left[-4\pi+4\pi\frac{m\left(\frac{M}{2}-k-1\right)}{M}\right]\right\} \\
&= \frac{2}{M}\left\{a^2+b^2+2ab\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]\right\} \\
&= \frac{2r_-^2}{M}\left\{1+2r_+r_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]\right\} \qquad \text{(B.86)}
\end{aligned}$$

Repeating this for $\frac{M}{2} \leq n < M$ yields

$$\left|\hat{x}_m^{(1,1)}\right|^2 = \frac{2r_+^2}{M}\left\{1-2r_+r_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]\right\} \qquad \text{(B.87)}$$

so that the sparsity estimate is given by

$$\begin{aligned}
\|\hat{x}_m\|_{\ell^1} &= \sum_{m=0}^{\frac{M}{2}-1} \left(\sqrt{\frac{2r_-^2}{M} \left\{ 1 + 2r_+r_- \cos\left[-2\pi \frac{2m(2k-1)}{M} \right] \right\}} \right. \\
&\quad \left. + \sqrt{\frac{2r_+^2}{M} \left\{ 1 - 2r_+r_- \cos\left[-2\pi \frac{2m(2k-1)}{M} \right] \right\}} \right) \\
&= \sqrt{\frac{2}{M}}(r_+ + r_-) \sum_{m=0}^{\frac{M}{2}-1} \sqrt{1 + 2r_+r_- \cos\left[-2\pi \frac{2m(2k-1)}{M} \right]} \\
&\leq \sqrt{\frac{M}{2}}(r_+ + r_-) \\
&\leq \sqrt{M}. \qquad \text{(B.88)}
\end{aligned}$$

Recall from Equation (B.82) that the sparsity estimate for level 0 is $\sqrt{M}$; hence, in a best basis algorithm, level 1 will always be chosen over level 0.

This can be repeated for level 2 to give similar results. That is, folding the left half of level 2 yields

$$U_{\text{level }2}T_{\text{level }1}x_n = \begin{cases} R_+a + R_-b & \text{if } n = k, \\ R_+b - R_-a & \text{if } n = \frac{M}{2} - k - 1, \\ 0 & \text{otherwise} \end{cases} \qquad \text{(B.89)}$$

where $R_- = R_{-k}$ and $R_+ = R_{k-1}$ are values for the level 2 rising cutoff function with radius of $\frac{M}{8}$. Periodized unfolding gives

$$T_{\text{level }2}T_{\text{level }1}x_n = \begin{cases} R_+^2a + R_+R_-b & \text{if } n = k, \\ R_+R_-a + R_-^2 & \text{if } n = \frac{M}{4} - k - 1, \\ R_-^2 - R_+R_-b & \text{if } n = \frac{M}{4} + k, \\ R_+^2b - R_+R_-a & \text{if } n = \frac{M}{2} - k - 1. \end{cases} \qquad \text{(B.90)}$$

Using this, the sparsity estimate for the left half of level 2 is

$$\begin{aligned}
\|\hat{x}_m^{(2,0\&1)}\|_{\ell^1} &= \sum_{m=0}^{\frac{M}{4}-1} \left(\sqrt{\frac{4(R_+a+R_-b)^2}{M} \left\{ 1 + 2R_+R_- \cos\left[-2\pi \frac{4m(2k-1)}{M} \right] \right\}} \right. \\
&\quad \left. + \sqrt{\frac{4(R_+b-R_-a)^2}{M} \left\{ 1 - 2R_+R_- \cos\left[-2\pi \frac{4m(2k-1)}{M} \right] \right\}} \right) \\
&= \frac{2}{\sqrt{M}} [R_+(a+b) - R_-(a-b)] \\
&\quad \cdot \sum_{m=0}^{\frac{M}{4}-1} \sqrt{1 + 2R_+R_- \cos\left[-2\pi \frac{4m(2k-1)}{M} \right]} \\
&= \frac{r_-[R_+(r_+ + r_-) - R_-(r_+ - r_-)]}{\sqrt{M}} \\
&\quad \cdot \sum_{m=0}^{\frac{M}{2}-1} \sqrt{1 + 2R_+R_- \cos\left[-2\pi \frac{2m(2k-1)}{M} \right]} \qquad \text{(B.91)}
\end{aligned}$$

To get an upper bound on Equation (B.91), first bound $R_+(r_+ + r_-) - R_-(r_+ - r_-)$. To do this, let

$$\begin{aligned}
\tilde{R}_{[m]}(t) &= R_{[m]}(t) \left[r_{[m]}(t) + r_{[m]}(-t) \right] - R_{[m]}(-t) \left[r_{[m]}(t) - r_{[m]}(-t) \right] \\
&= R_{[m]}(t) \left[R_{[m]}\left(\frac{t}{2} \right) + R_{[m]}\left(-\frac{t}{2} \right) \right] \\
&\quad - R_{[m]}(-t) \left[R_{[m]}\left(\frac{t}{2} \right) - R_{[m]}\left(-\frac{t}{2} \right) \right]. \qquad \text{(B.92)}
\end{aligned}$$

for $-1 \leq t \leq 1$. Note that as $m \to \infty$, $\tilde{R}_{[m]} \searrow 1$; that is, $\tilde{M}$ will be maximized when $m = 0$. Letting $R(t) = R_{[m]}(t)$ results in the following

$$\begin{aligned}
\tilde{R}'(t) &= R'(t)\left[R\left(\frac{t}{2}\right) + R\left(-\frac{t}{2}\right)\right] + \frac{R(t)}{2}\left[R'\left(\frac{t}{2}\right) - R'\left(-\frac{t}{2}\right)\right] \\
&\quad + R'(-t)\left[R\left(\frac{t}{2}\right) - R\left(-\frac{t}{2}\right)\right] - \frac{R(-t)}{2}\left[R'\left(\frac{t}{2}\right) + R'\left(-\frac{t}{2}\right)\right] \\
&= \frac{\pi\sqrt{2}}{8}\left[\cos\left(\frac{\pi}{4}t\right) - \sin\left(\frac{\pi}{4}t\right)\right]\left\{\frac{\sqrt{2}}{2}\left[\cos\left(\frac{\pi}{8}t\right) + \sin\left(\frac{\pi}{8}t\right)\right]\right. \\
&\quad \left.+\frac{\sqrt{2}}{2}\left[\cos\left(\frac{\pi}{8}t\right) - \sin\left(\frac{\pi}{8}t\right)\right]\right\} + \frac{\sqrt{2}}{4}\left[\cos\left(\frac{\pi}{4}t\right) + \sin\left(\frac{\pi}{4}t\right)\right] \\
&\quad \cdot\left\{\frac{\pi\sqrt{2}}{8}\left[\cos\left(\frac{\pi}{8}t\right) - \sin\left(\frac{\pi}{8}t\right)\right] - \frac{\pi\sqrt{2}}{8}\left[\cos\left(\frac{\pi}{8}t\right) + \sin\left(\frac{\pi}{8}t\right)\right]\right\} \\
&\quad + \frac{\pi\sqrt{2}}{8}\left[\cos\left(\frac{\pi}{4}t\right) + \sin\left(\frac{\pi}{4}t\right)\right]\left\{\frac{\sqrt{2}}{2}\left[\cos\left(\frac{\pi}{8}t\right) + \sin\left(\frac{\pi}{8}t\right)\right]\right. \\
&\quad \left.-\frac{\sqrt{2}}{2}\left[\cos\left(\frac{\pi}{8}t\right) - \sin\left(\frac{\pi}{8}t\right)\right]\right\} - \frac{\sqrt{2}}{4}\left[\cos\left(\frac{\pi}{4}t\right) - \sin\left(\frac{\pi}{4}t\right)\right] \\
&\quad \cdot\left\{\frac{\pi\sqrt{2}}{8}\left[\cos\left(\frac{\pi}{8}t\right) - \sin\left(\frac{\pi}{8}t\right)\right] + \frac{\pi\sqrt{2}}{8}\left[\cos\left(\frac{\pi}{8}t\right) + \sin\left(\frac{\pi}{8}t\right)\right]\right\} \\
&= \frac{\pi}{4}\cos\left(\frac{\pi}{8}t\right)\left[\cos\left(\frac{\pi}{4}t\right) - \sin\left(\frac{\pi}{4}t\right)\right] - \frac{\pi}{8}\sin\left(\frac{\pi}{8}t\right)\left[\cos\left(\frac{\pi}{4}t\right) + \sin\left(\frac{\pi}{4}t\right)\right] \\
&\quad + \frac{\pi}{4}\sin\left(\frac{\pi}{8}t\right)\left[\cos\left(\frac{\pi}{4}t\right) + \sin\left(\frac{\pi}{4}t\right)\right] - \frac{\pi}{8}\cos\left(\frac{\pi}{8}t\right)\left[\cos\left(\frac{\pi}{4}t\right) - \sin\left(\frac{\pi}{4}t\right)\right] \\
&= \frac{\pi}{8}\left\{\cos\left(\frac{\pi}{8}t\right)\left[\cos\left(\frac{\pi}{4}t\right) - \sin\left(\frac{\pi}{4}t\right)\right] + \sin\left(\frac{\pi}{8}t\right)\left[\cos\left(\frac{\pi}{4}t\right) + \sin\left(\frac{\pi}{4}t\right)\right]\right\} \\
&= \frac{\pi}{8}\left[\cos\left(\frac{\pi}{8}t\right) - \sin\left(\frac{\pi}{8}t\right)\right]. \qquad \text{(B.93)}
\end{aligned}$$

Thus, $\tilde{R}'(t) = 0$ when $t = \frac{8}{\pi}\tan^{-1}(1) = \ldots, -14, -6, 2, 10, \ldots$. Therefore, on $-1 \le t \le 1$, the max occurs at the boundary $t = 1$; i.e., $\tilde{R}(t) \le \tilde{R}(1) = \sqrt{2}\cos\left(\frac{\pi}{8}\right)$. Plugging this into Equation (B.94) yields

$$\|\hat{x}_m^{(2,0\&1)}\|_{\ell^1} \le r_-\sqrt{\frac{2}{m}}\cos\frac{\pi}{8}\sum_{m=0}^{\frac{M}{2}-1}\sqrt{1 + 2R_+R_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]} \qquad \text{(B.94)}$$

Using Equation (B.87), the sparsity estimate for the left half of level 1 is found to be

$$\|\hat{x}_m^{(1,0)}\|_{\ell^1} \geq r_-\sqrt{\frac{2}{m}}\sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2r_+r_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]}. \tag{B.95}$$

Comparing these two values reveals that they almost identical except for the $\cos\frac{\pi}{8}$ term and the radius of the rising cutoff function. The summation term will be increase as R_+R_- decreases. And since $R_+R_- \leq r_+r_-$, then

$$\sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2R_+R_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]} \geq \sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2r_+r_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]}. \tag{B.96}$$

This inequality becomes even larger if the rising cutoff function becomes less steep; i.e., if $m = 0$ for $R(t) = r_{[m]}(t)$. So, fixing $m = 0$, the left hand side of Equation (B.96) will become maximized when $R(t)R(-t) = 0$; that is, when $t = 1$. Similarly, the right hand side of Equation (B.96) will become minimized when $t = 1$. But because of the $\cos\left(\frac{\pi}{8}\right)$ term, the following question arises

$$\begin{aligned}
&\cos\left(\frac{\pi}{8}\right)\sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2R_+R_-\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]}\\
&\leq \cos\left(\frac{\pi}{8}\right)\sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2R(1)R(-1)\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]}\\
&= \cos\left(\frac{\pi}{8}\right)\frac{M}{2}\\
&\leq ? \sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2R\left(\frac{1}{2}\right)R\left(-\frac{1}{2}\right)\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]} \qquad \text{(B.97)}\\
&\leq \sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2r_-r_+\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]}.
\end{aligned}$$

To verify Inequality (B.97), rearrange the terms to get

$$\begin{aligned}
\cos\left(\frac{\pi}{8}\right) &\leq? \frac{\sum_{m=0}^{\frac{M}{2}-1}\sqrt{1+2R\left(\frac{1}{2}\right)R\left(-\frac{1}{2}\right)\cos\left[-2\pi\frac{2m(2k-1)}{M}\right]}}{\frac{M}{2}}\\
&= \int_0^1\sqrt{1+\frac{\sqrt{2}}{2}\cos(2\pi t)dt}.
\end{aligned} \tag{B.98}$$

Since, $\cos\left(\frac{\pi}{8}\right) \approx 0.92$ and $\int_0^1 \sqrt{1 + \frac{\sqrt{2}}{2}\cos(2\pi t)}dt \approx 0.96$, then the above inequality is always true. What this means, is that the left two level 2 subspaces will always be chosen over the left half of level 1 *regardless of the choice of* k in the original signal x. Repeating this analysis in a recursive manner results in the bottom subspaces always being chosen. This is shown in Figure 5.22(b).

(ii) In contrast to LFT, is the partition pattern chosen by CPLFT. As was shown in Equation (B.82), the sparsity estimate for level 0 is $\sqrt{M}$. Since all of the subspaces in the CPLFT decomposition are independent of each other, the sparsity estimate for every subspace on each level is 0 except for the interval containing the bump. Thus, the sparsity estimate for level 1 is given by

$$\begin{aligned} \|\hat{x}_m\|_{\ell^p} &= \left(\sum_{m=0}^{\frac{M}{2}-1} \left| \frac{1}{\sqrt{\frac{M}{2}}} e^{-2\pi i \frac{mk}{\frac{M}{2}}} \right|^p \right)^{\frac{1}{p}} \\ &= \left[\frac{M}{2} \left(\sqrt{\frac{2}{M}} \right)^p \right]^{\frac{1}{p}} \\ &= \left(\frac{M}{2} \right)^{\frac{2-p}{2p}} \\ &= \sqrt{\frac{M}{2}} \text{ for } p = 1. \end{aligned} \tag{B.99}$$

The sparsity estimate for level L is $\left(\frac{M}{L+1}\right)^{\frac{2-p}{2p}}$. Therefore the smallest interval containing the bump will always be chosen, while the largest intervals will be chosen for the regions which are zero. This leads to a very natural partition pattern shown in Figure 5.22(c). □

Appendix C

Image Set Used For Chapter 6 Algorithm Comparison

The following images were used to make the subjective evaluation of the compression quality in tables 6.1, 6.2, and 6.3. There are three original images representing 4 different classes of images: the line drawing of a surfer shown in Figure C.1, the textured image a sandstone seen in Figure C.14 and the digital photograph of Barbara shown in Figure C.27. Each image was compressed at a rate of 20-to-1 using twelve different algorithms. As was mentioned in section 4.2, Compression was achieved by simple thresholding; that is, only the most energetic 5 percent of the transformed coefficients were retained for these experiments. The method employed is listed in the title above each image. Full folding was employed for every applicable method, and the steepness of the rising cutoff function was 5 for all of the CBLTT methods and 1 for the others.

C.1 The Surfer Image Set

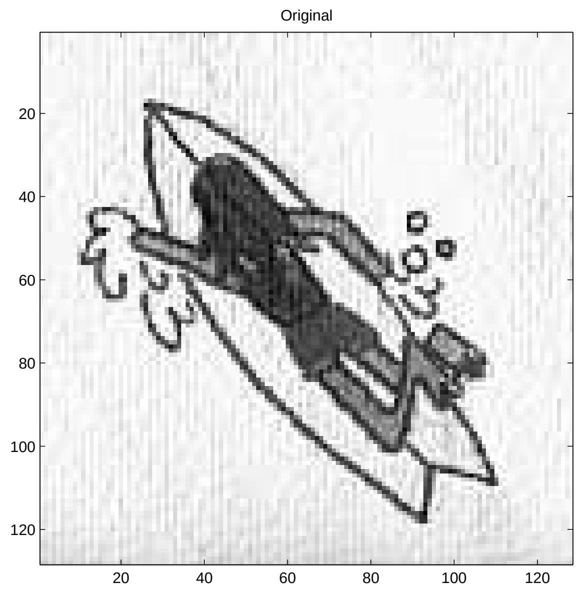

Figure C.1: Original surfer image.

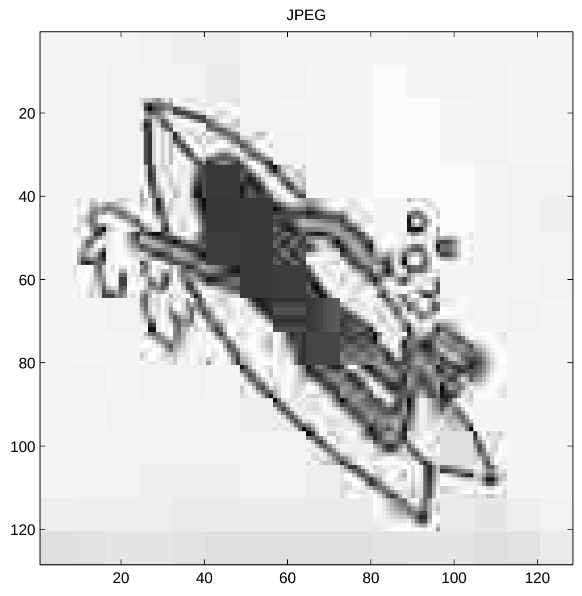

Figure C.2: 20:1 compression of Figure C.1 using JPEG-DCT.

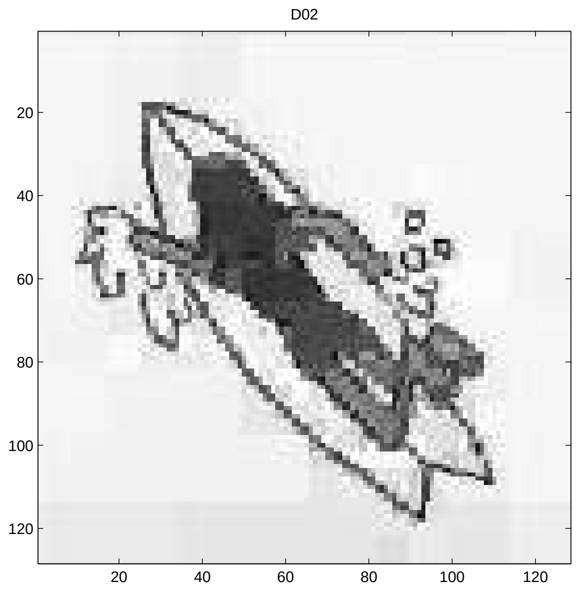

Figure C.3: 20:1 compression of Figure C.1 using D02.

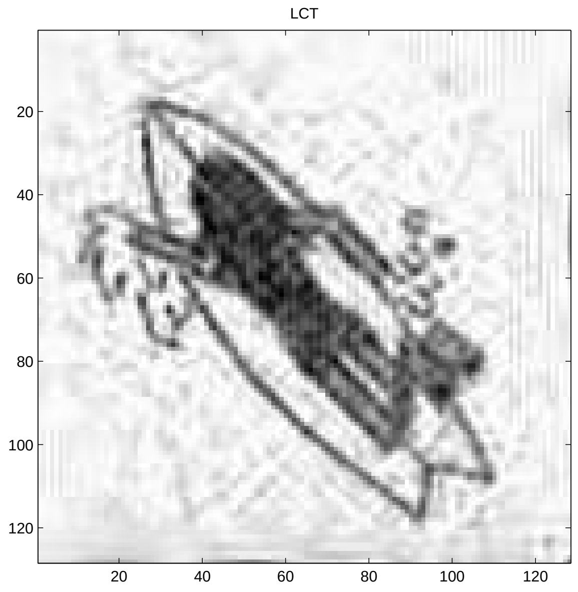

Figure C.4: 20:1 compression of Figure C.1 using LCT.

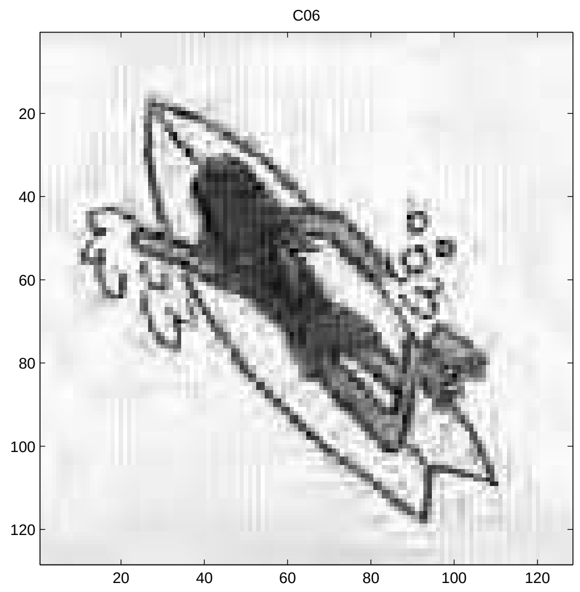

Figure C.5: 20:1 compression of Figure C.1 using C06.

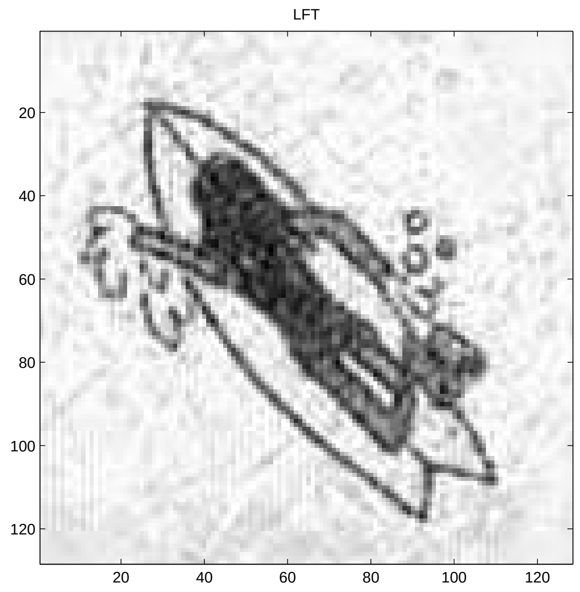

Figure C.6: 20:1 compression of Figure C.1 using LFT.

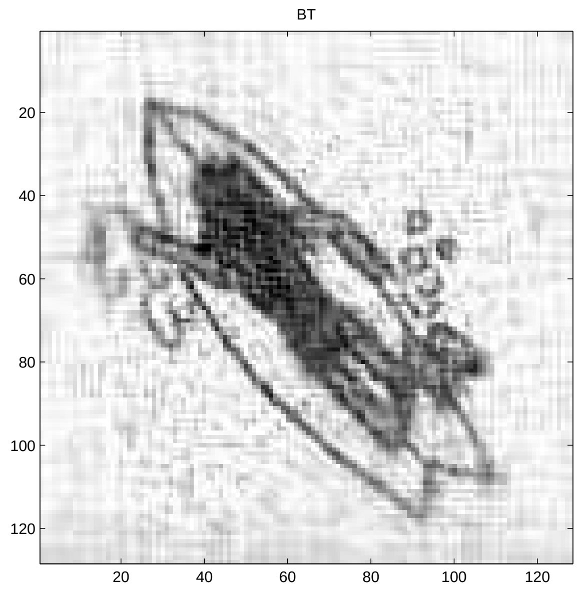

Figure C.7: 20:1 compression of Figure C.1 using BT.

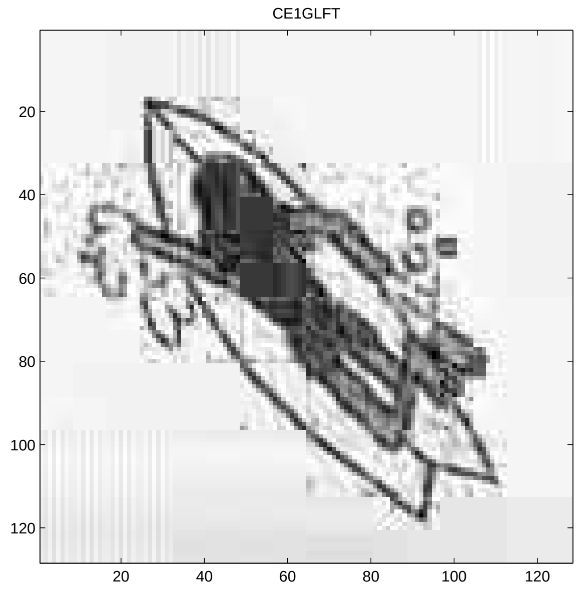

Figure C.8: 20:1 compression of Figure C.1 using CE1GLFT.

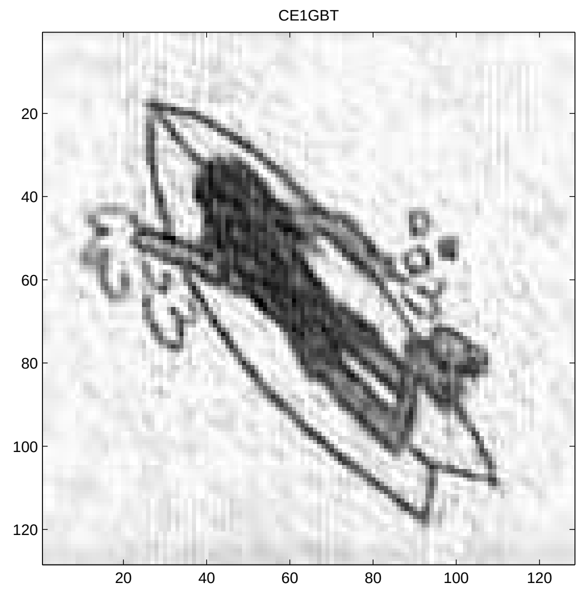

Figure C.9: 20:1 compression of Figure C.1 using CE1GBT.

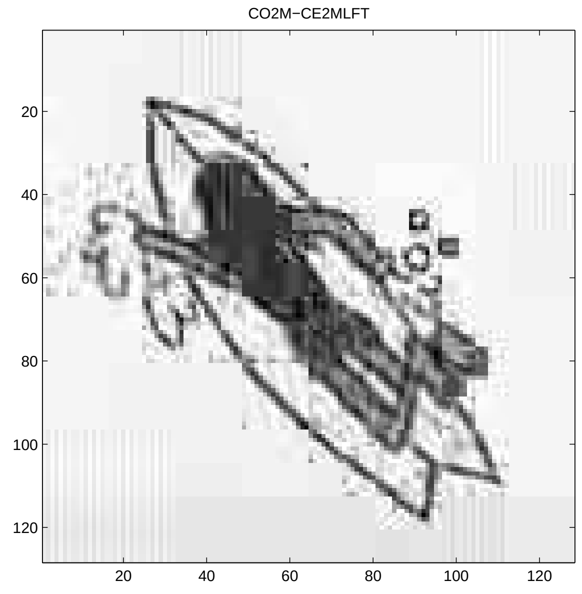

Figure C.10: 20:1 compression of Figure C.1 using CO2M-CE2MLFT.

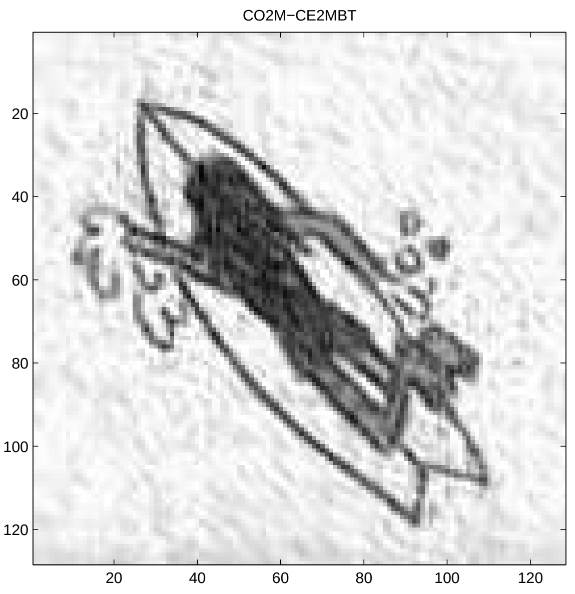

Figure C.11: 20:1 compression of Figure C.1 using CO2M-CE2MBT.

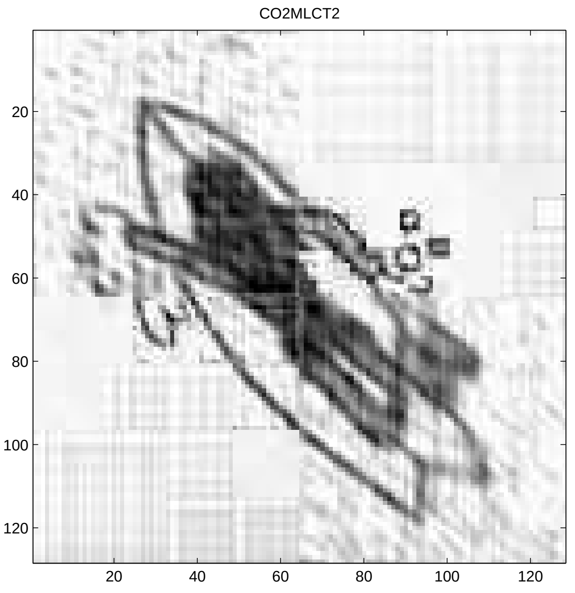

Figure C.12: 20:1 compression of Figure C.1 using CO2MLCT2.

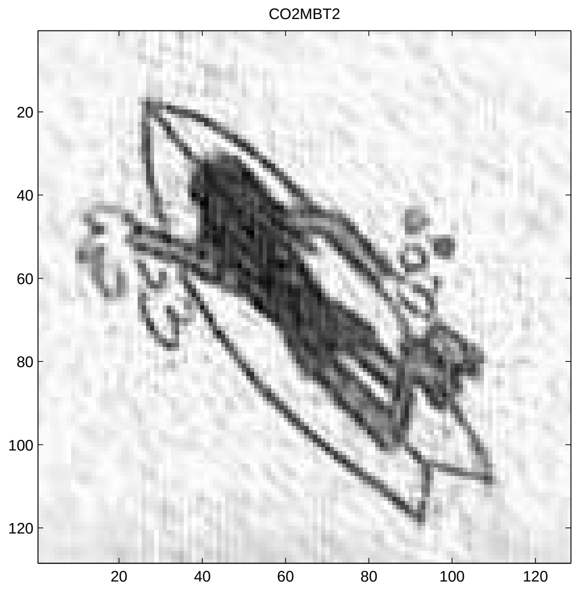

Figure C.13: 20:1 compression of Figure C.1 using CO2MBT2.

C.2 The Tile Image Set

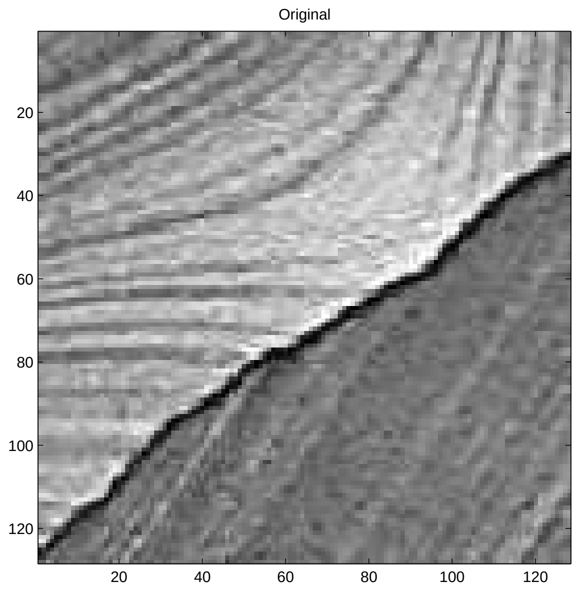

Figure C.14: Original tile image.

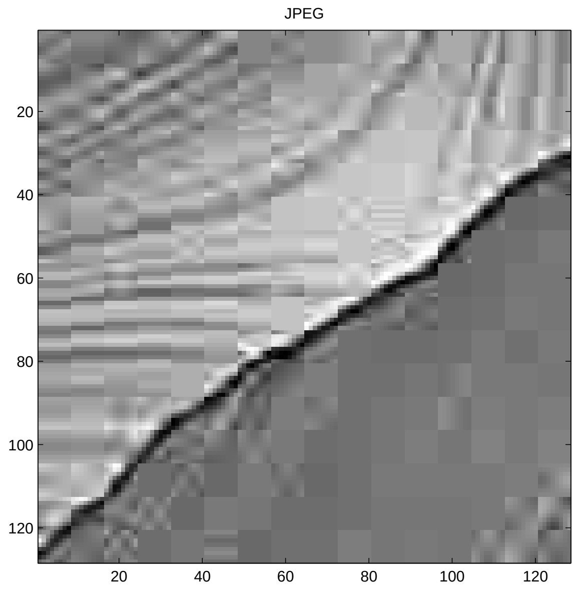

Figure C.15: 20:1 compression of Figure C.14 using JPEG-DCT.

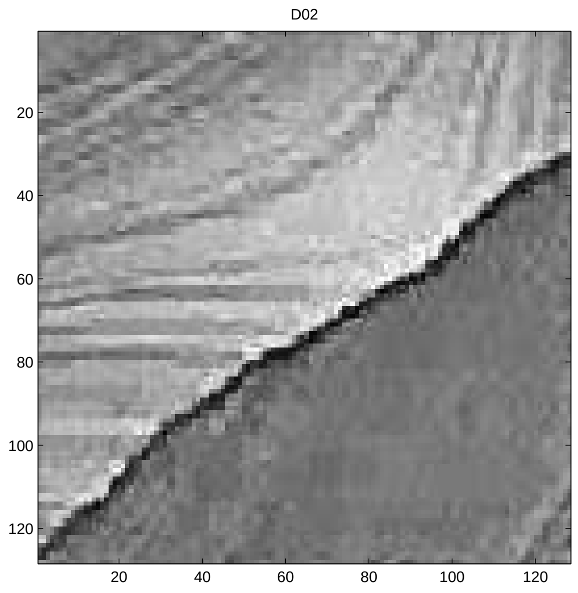

Figure C.16: 20:1 compression of Figure C.14 using D02.

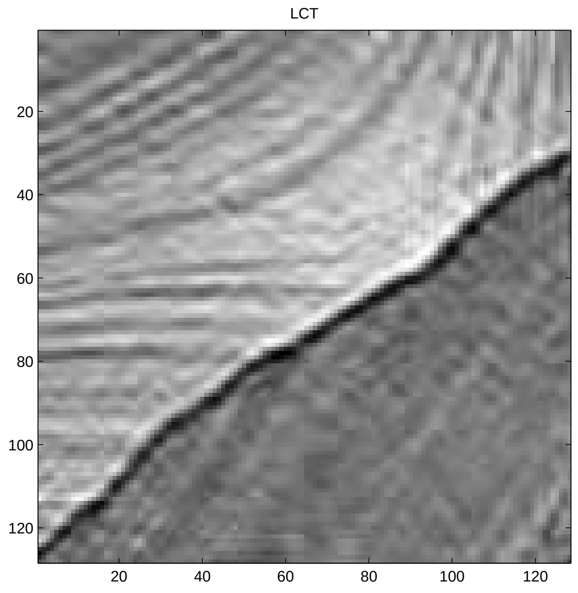

Figure C.17: 20:1 compression of Figure C.14 using LCT.

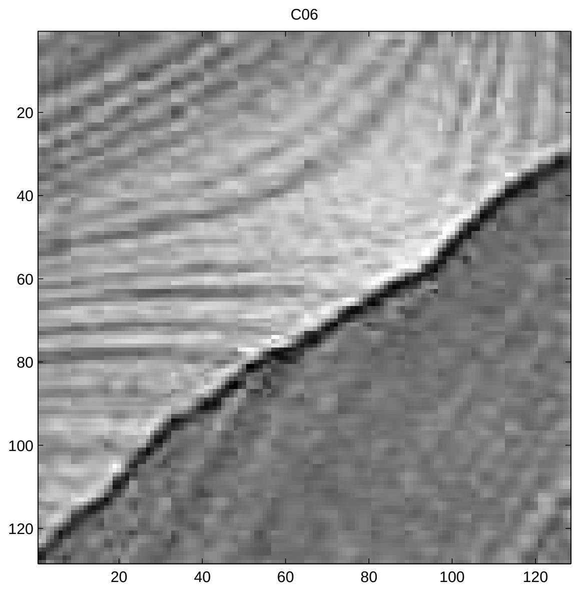

Figure C.18: 20:1 compression of Figure C.14 using C06.

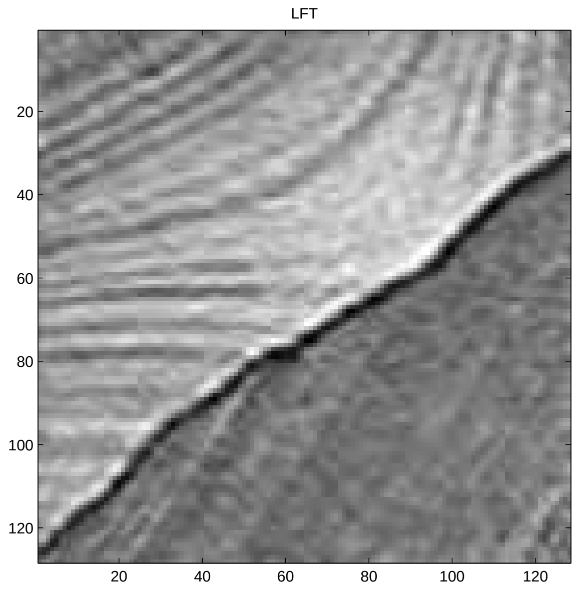

Figure C.19: 20:1 compression of Figure C.14 using LFT.

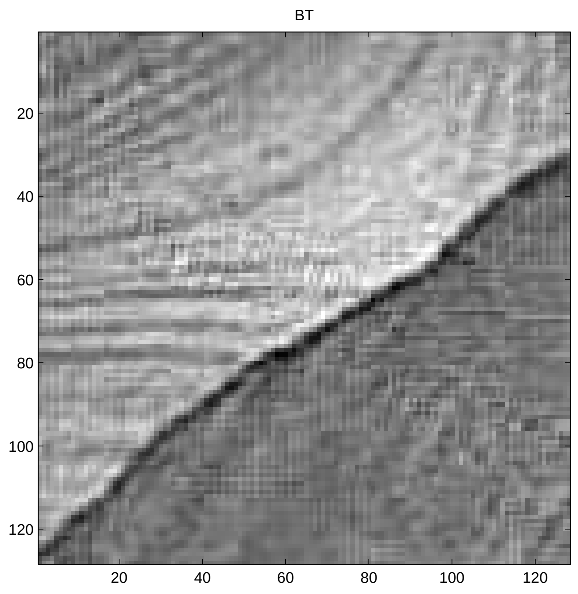

Figure C.20: 20:1 compression of Figure C.14 using BT.

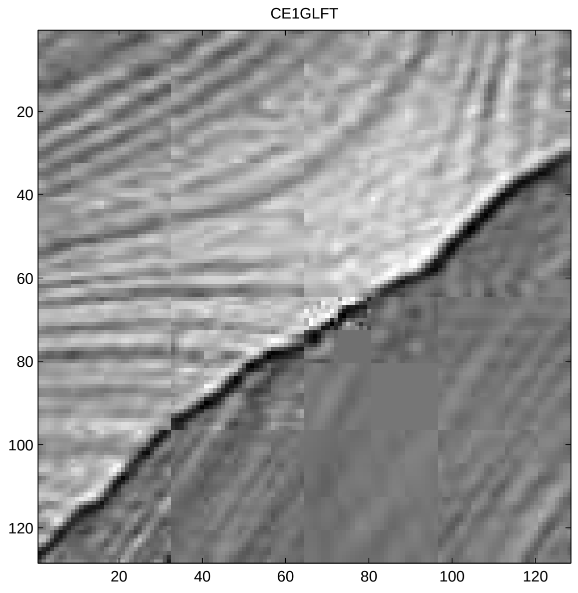

Figure C.21: 20:1 compression of Figure C.14 using CE1GLFT.

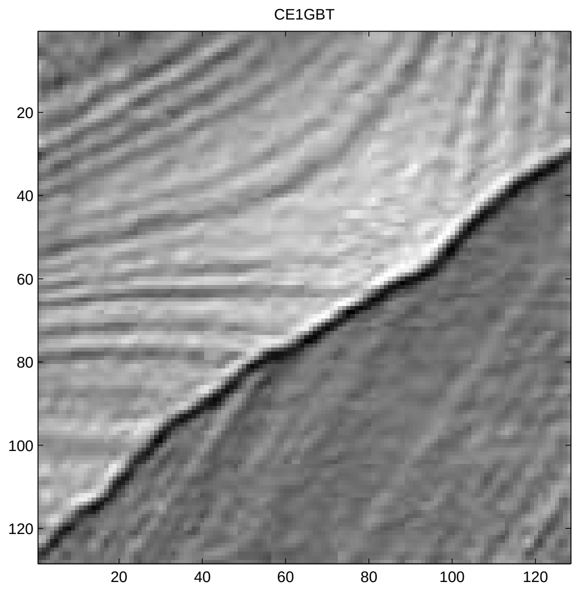

Figure C.22: 20:1 compression of Figure C.14 using CE1GBT.

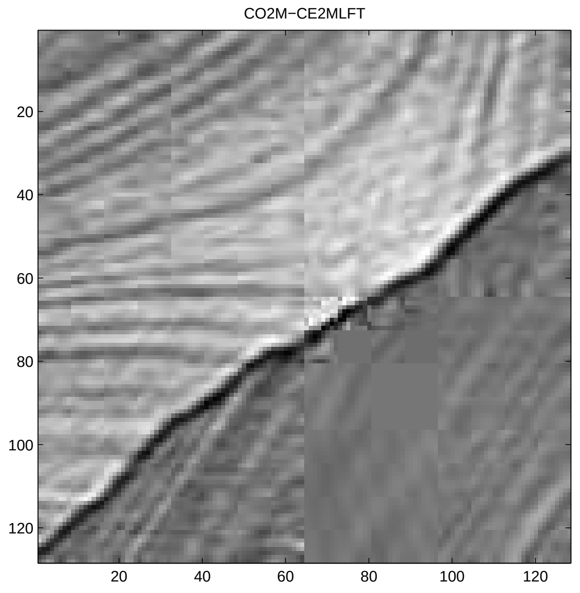

Figure C.23: 20:1 compression of Figure C.14 using CO2M-CE2MLFT.

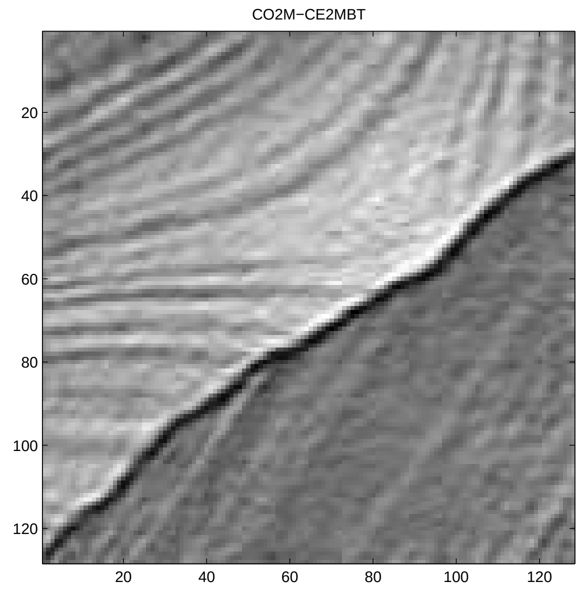

Figure C.24: 20:1 compression of Figure C.14 using CO2M-CE2MBT.

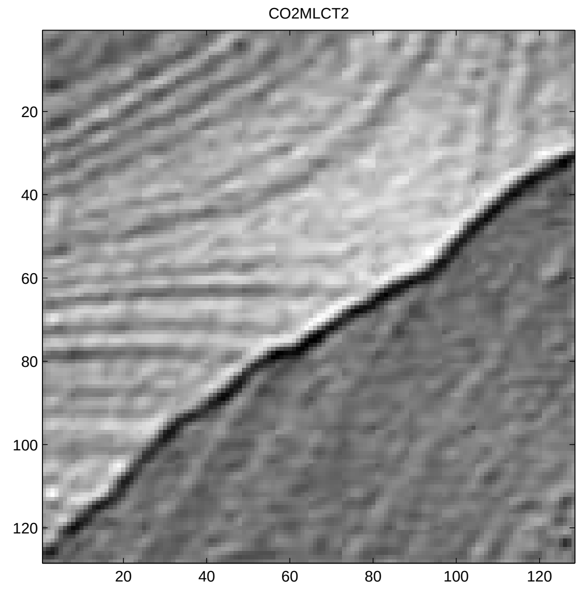

Figure C.25: 20:1 compression of Figure C.14 using CO2MLCT2.

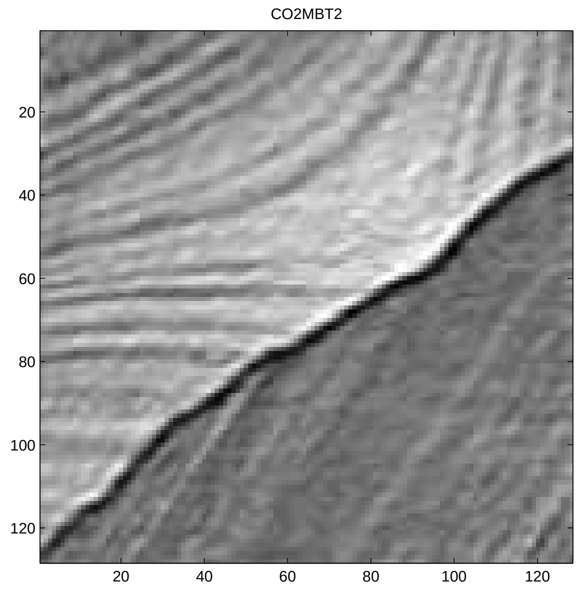

Figure C.26: 20:1 compression of Figure C.14 using CO2MBT2.

C.3 The Barbara Image Set

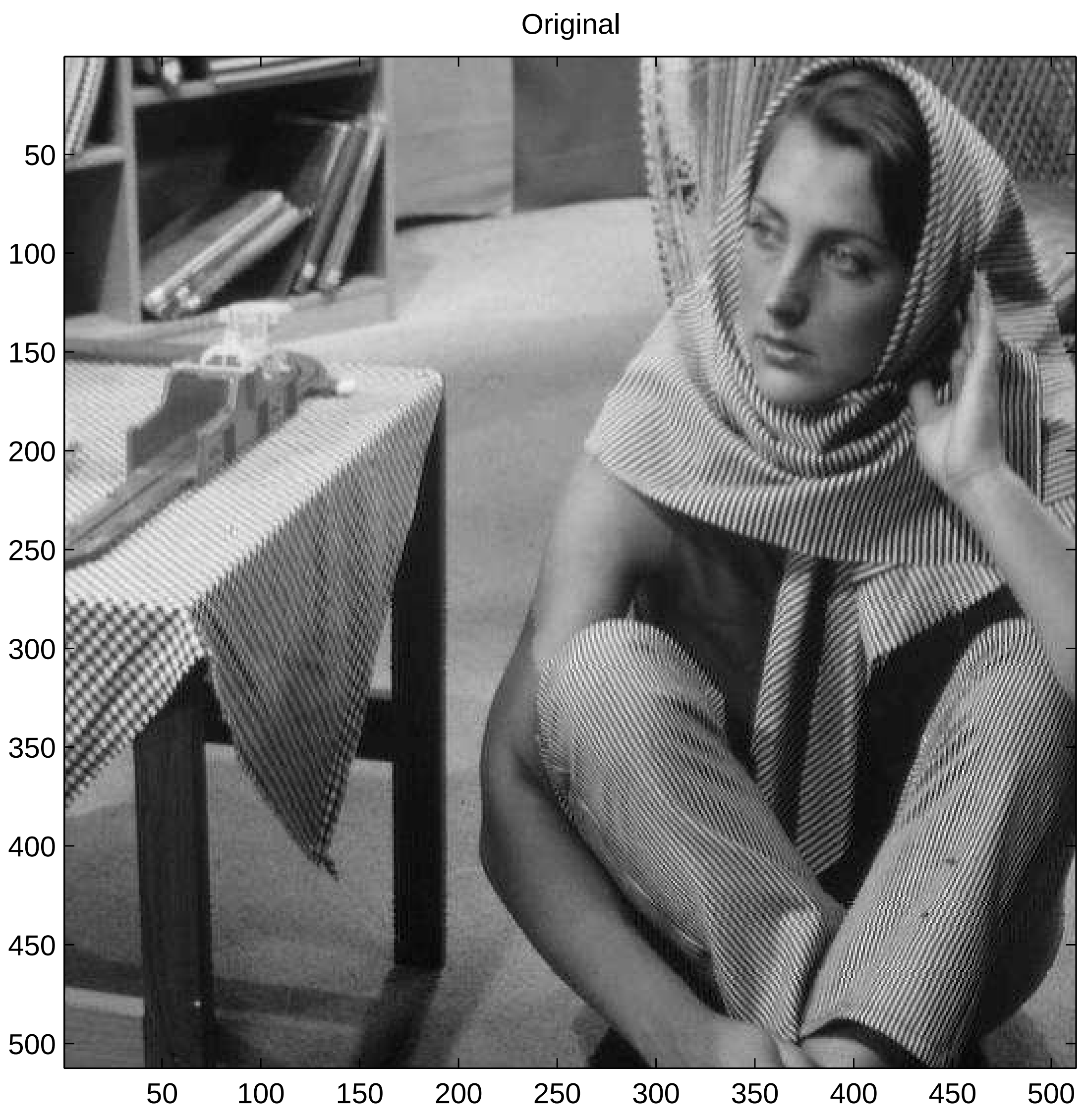

Figure C.27: Original barbara image.

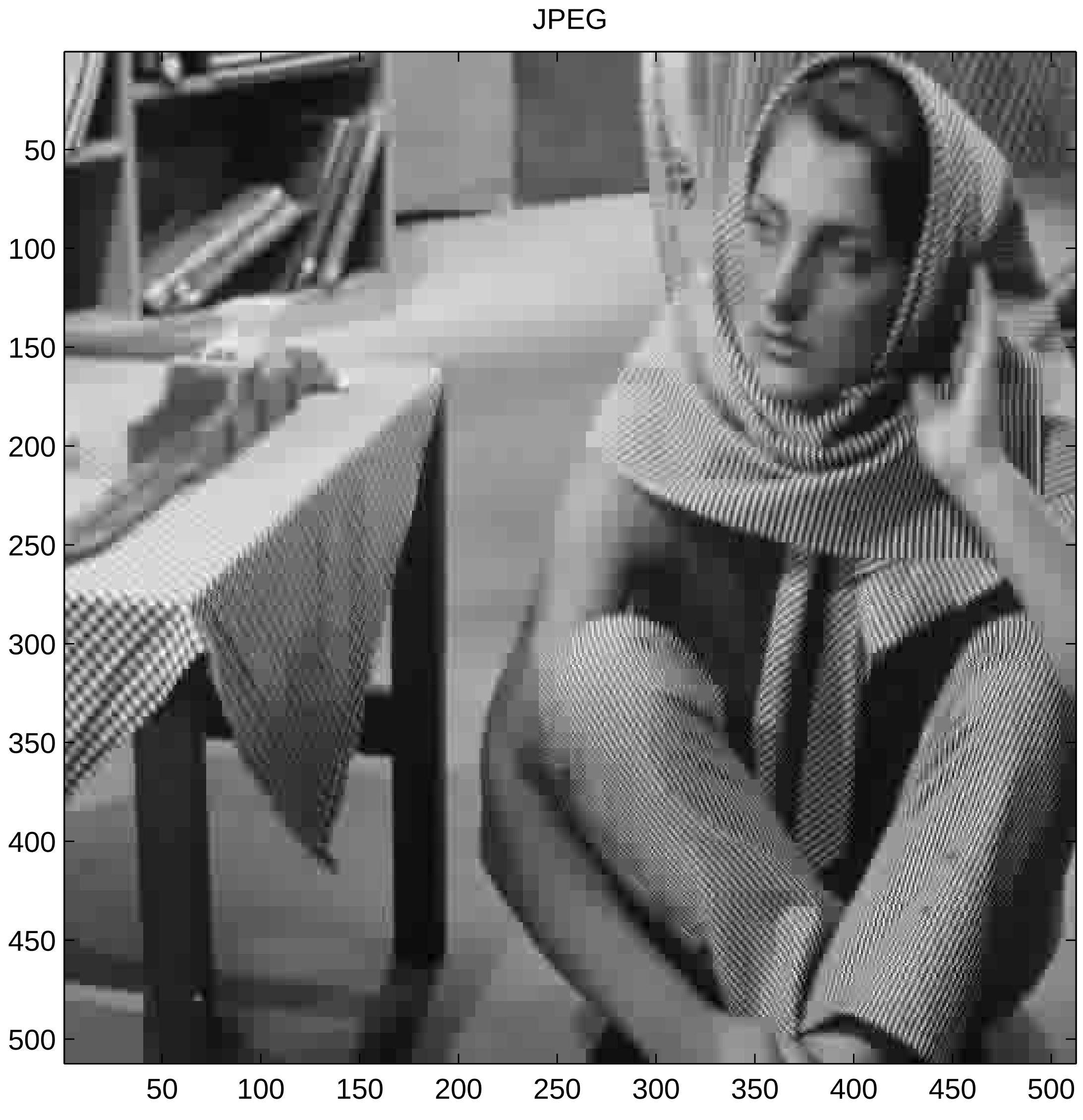

Figure C.28: 20:1 compression of Figure C.27 using JPEG-DCT.

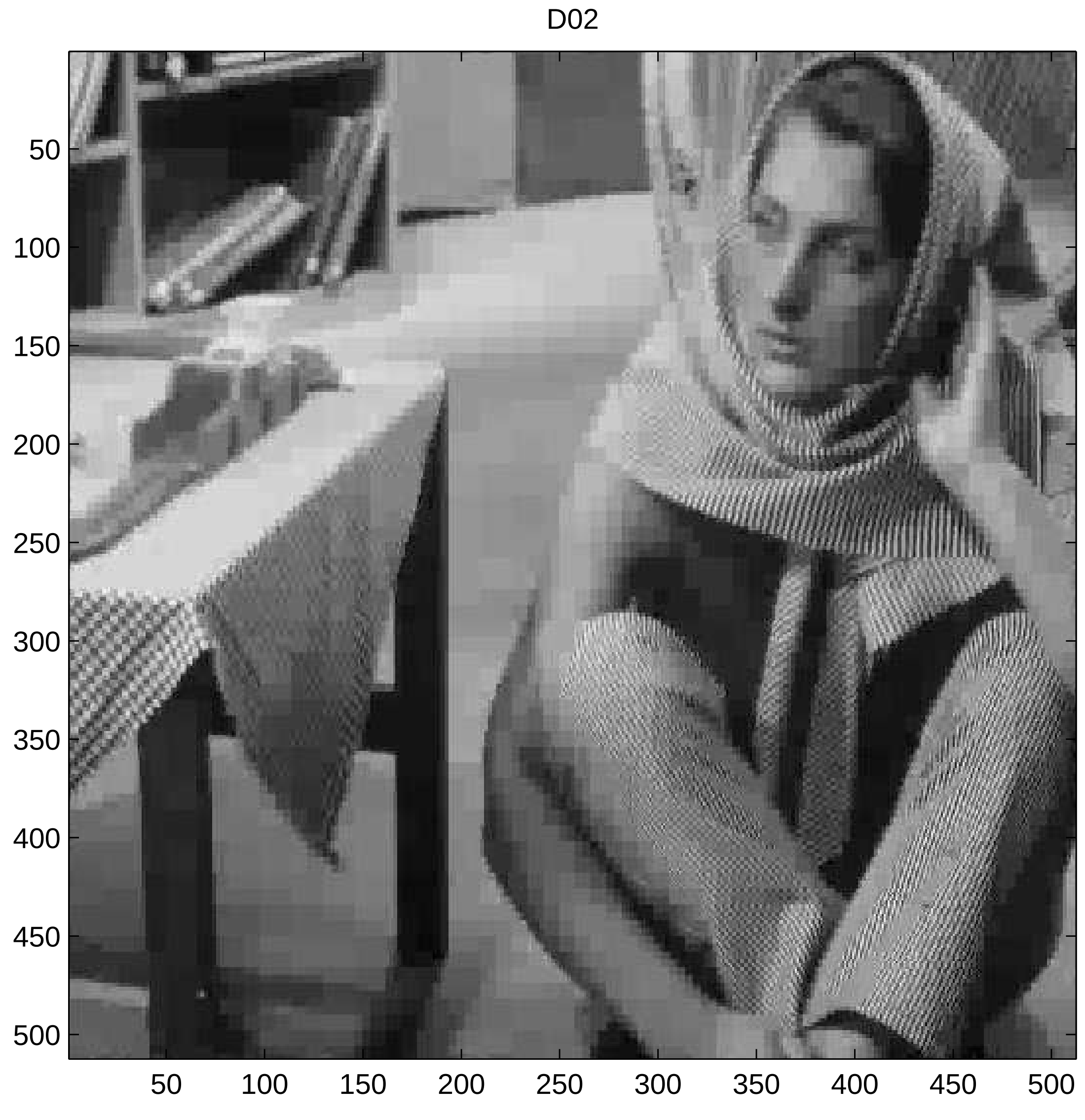

Figure C.29: 20:1 compression of Figure C.27 using D02.

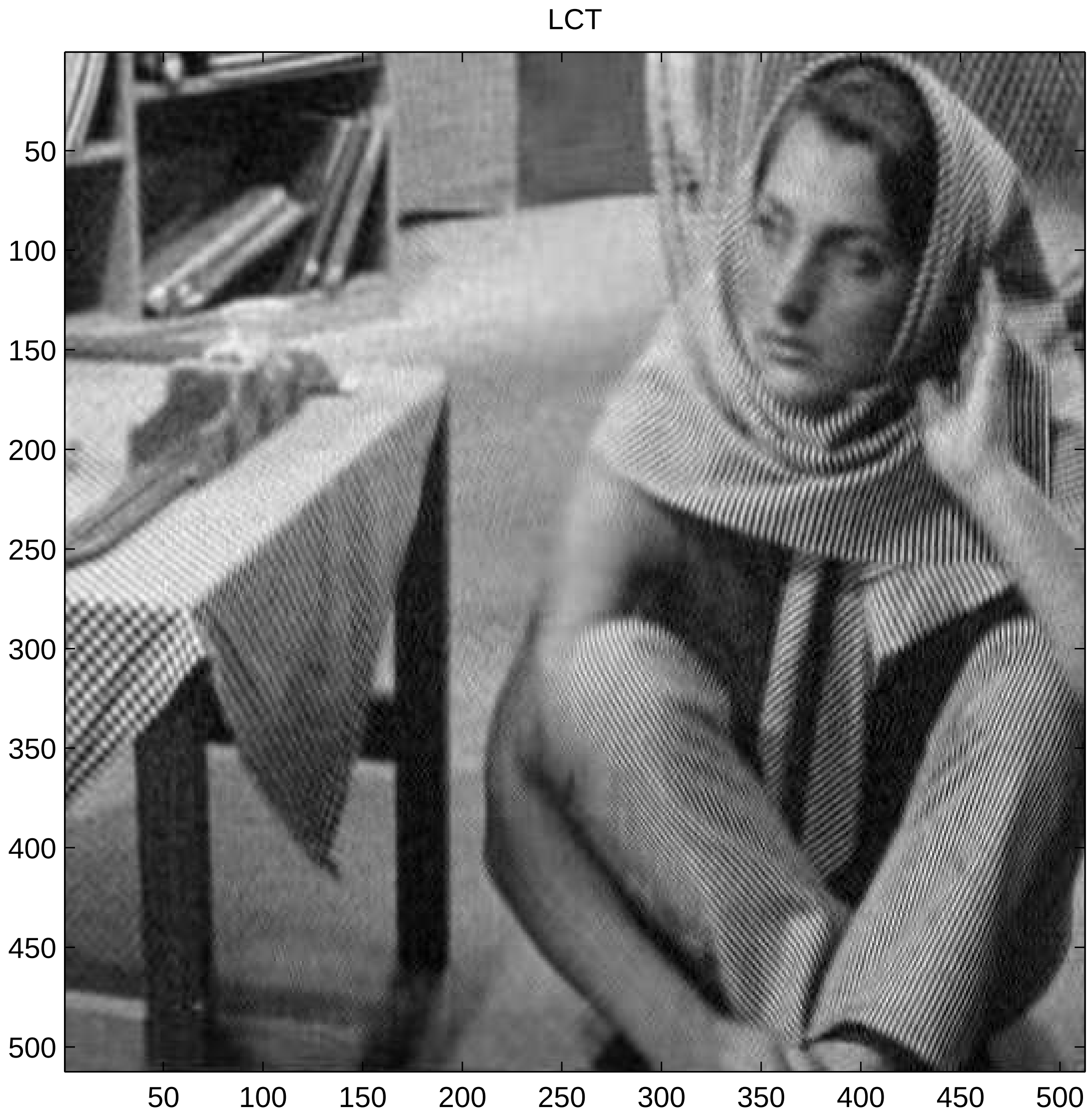

Figure C.30: 20:1 compression of Figure C.27 using LCT.

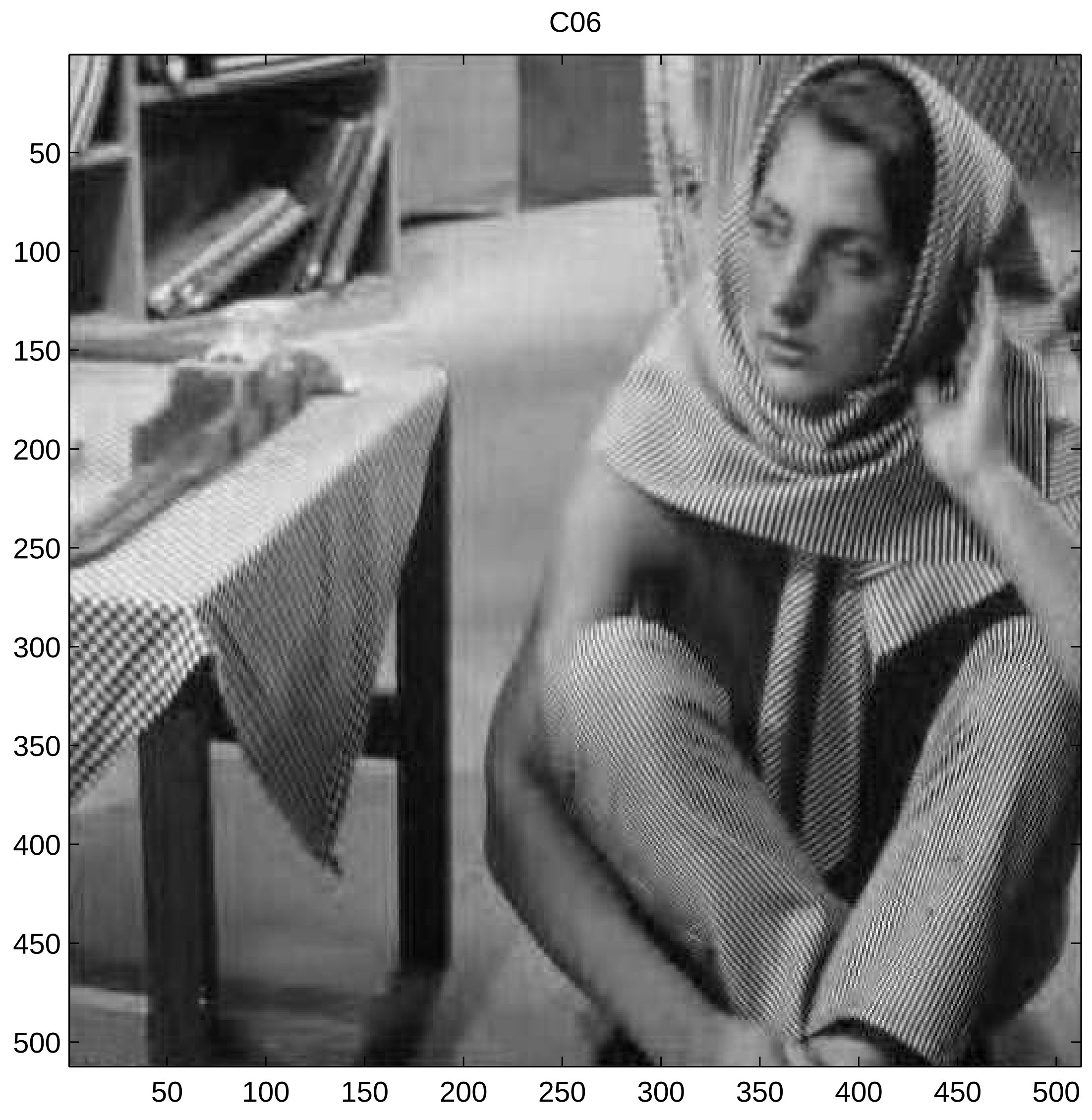

Figure C.31: 20:1 compression of Figure C.27 using C06.

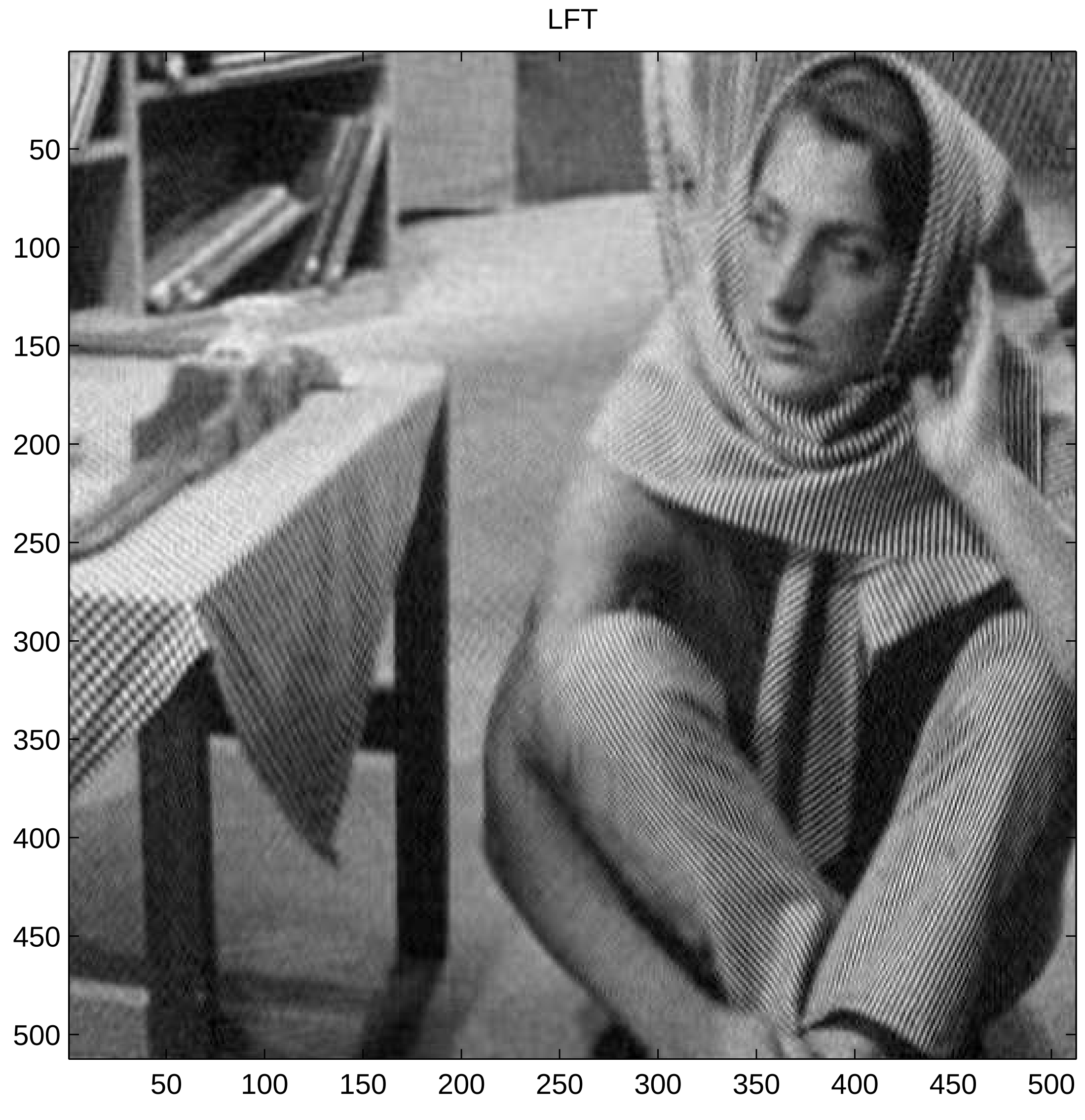

Figure C.32: 20:1 compression of Figure C.27 using LFT.

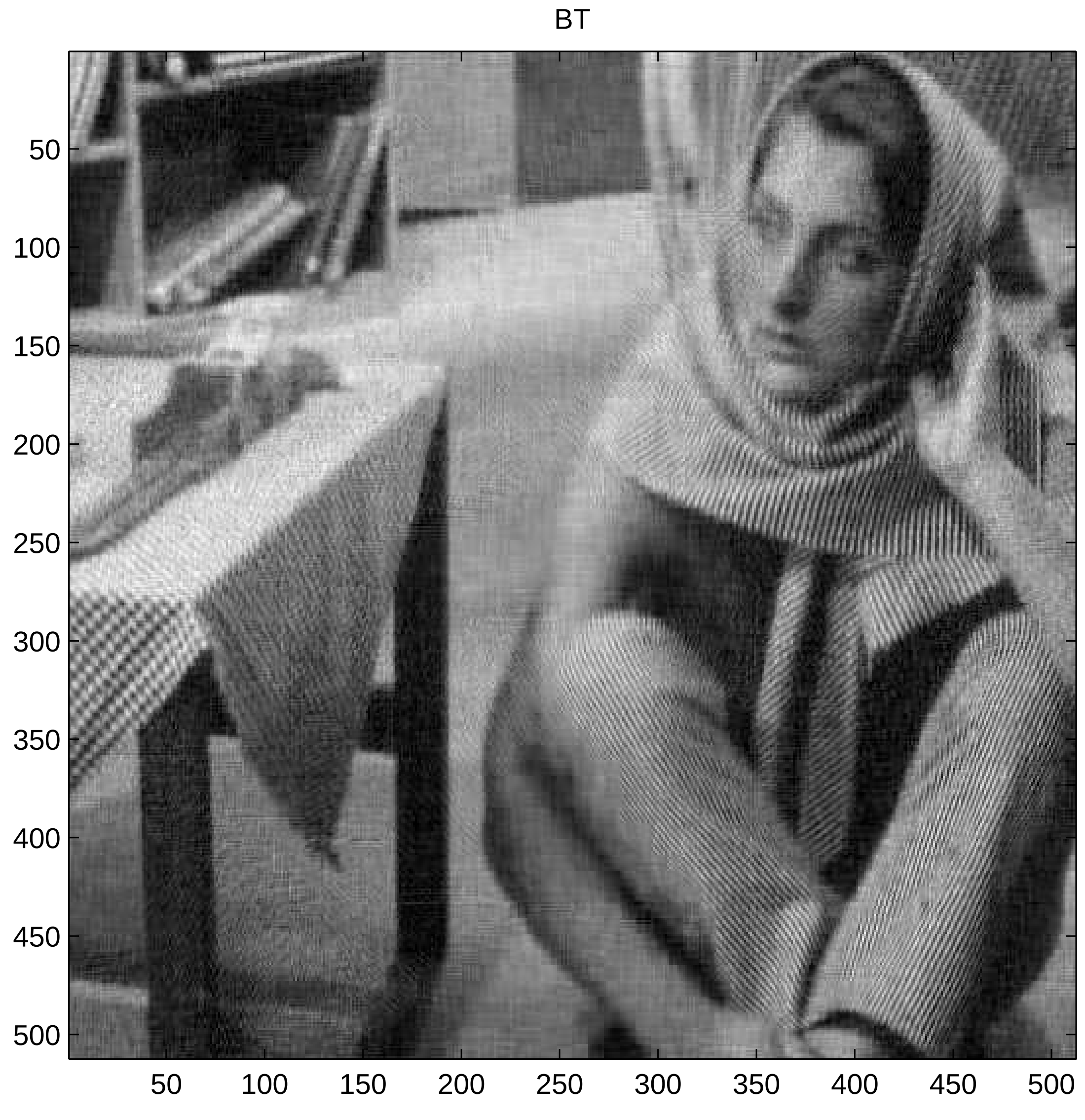

Figure C.33: 20:1 compression of Figure C.27 using BT.

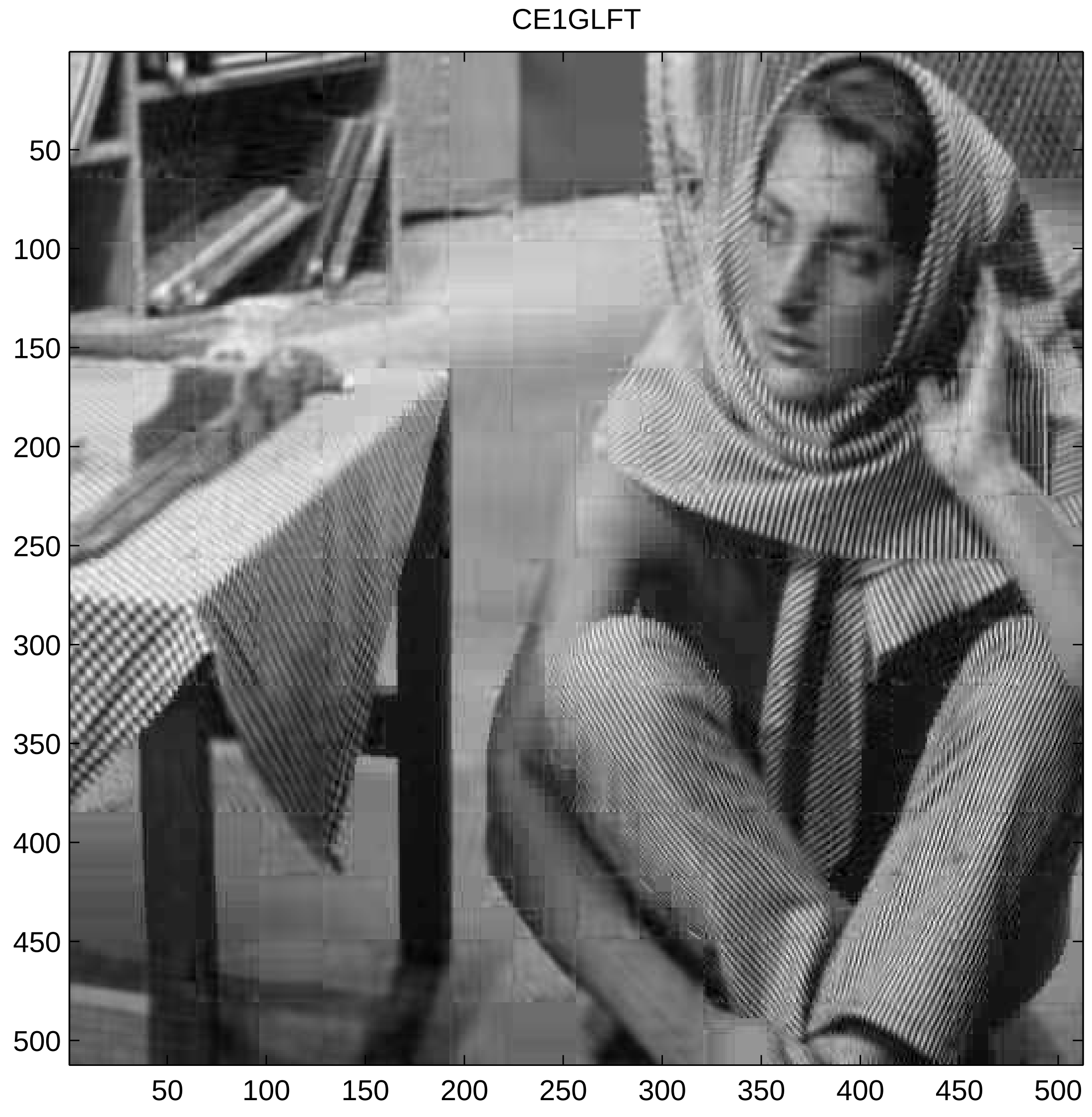

Figure C.34: 20:1 compression of Figure C.27 using CE1GLFT.

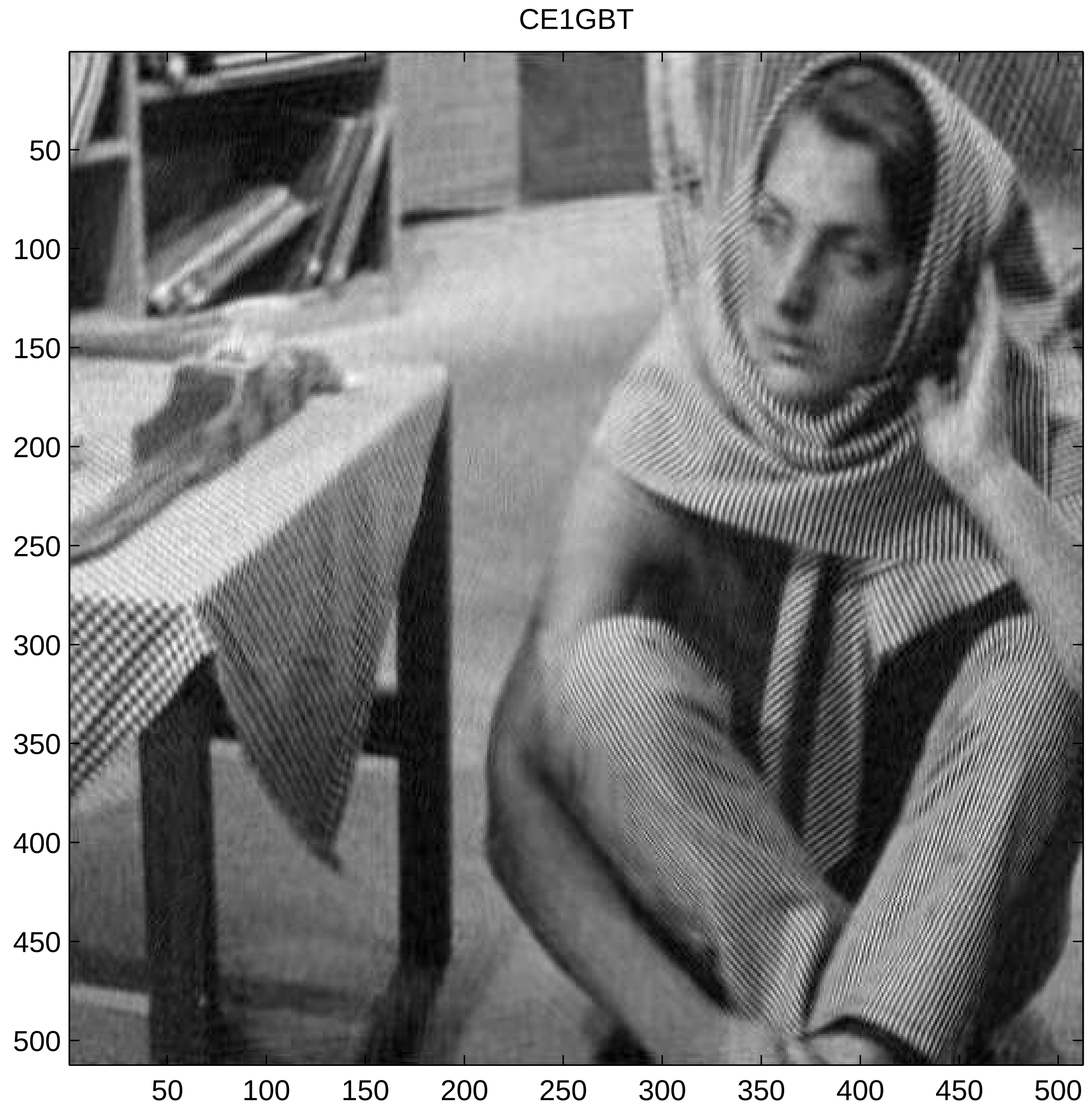

Figure C.35: 20:1 compression of Figure C.27 using CE1GBT.

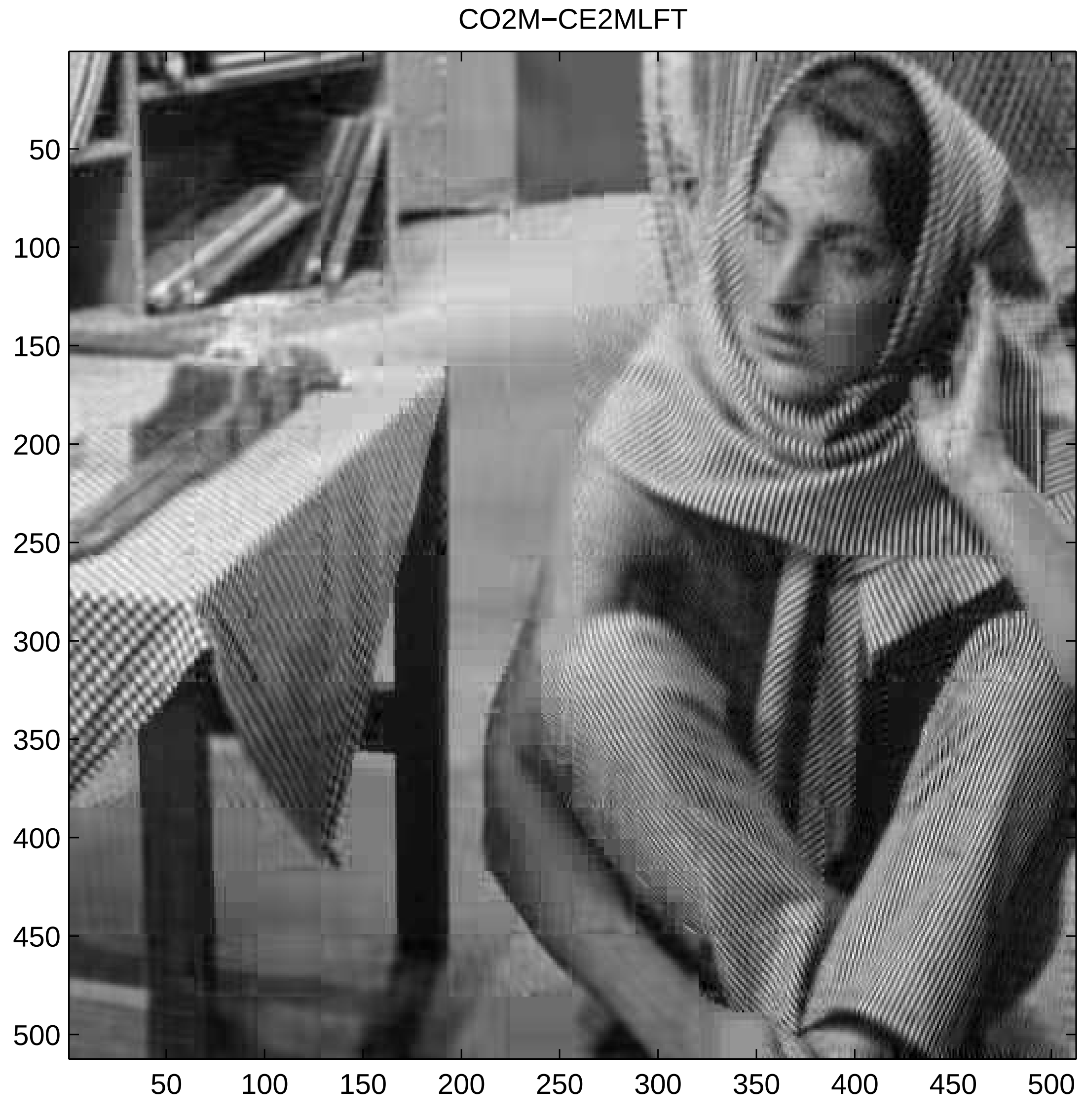

Figure C.36: 20:1 compression of Figure C.27 using CO2M-CE2MLFT.

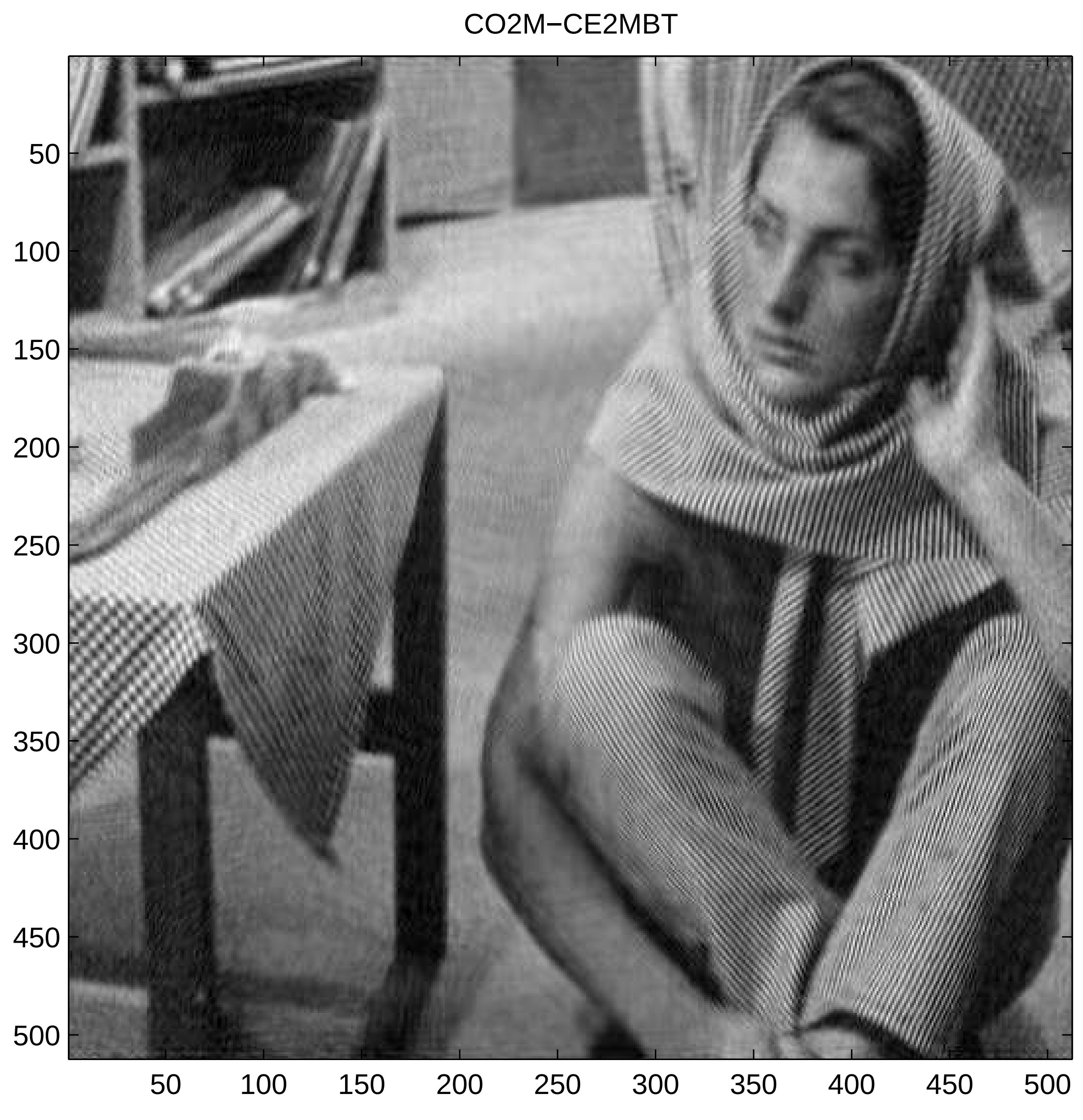

Figure C.37: 20:1 compression of Figure C.27 using CO2M-CE2MBT.

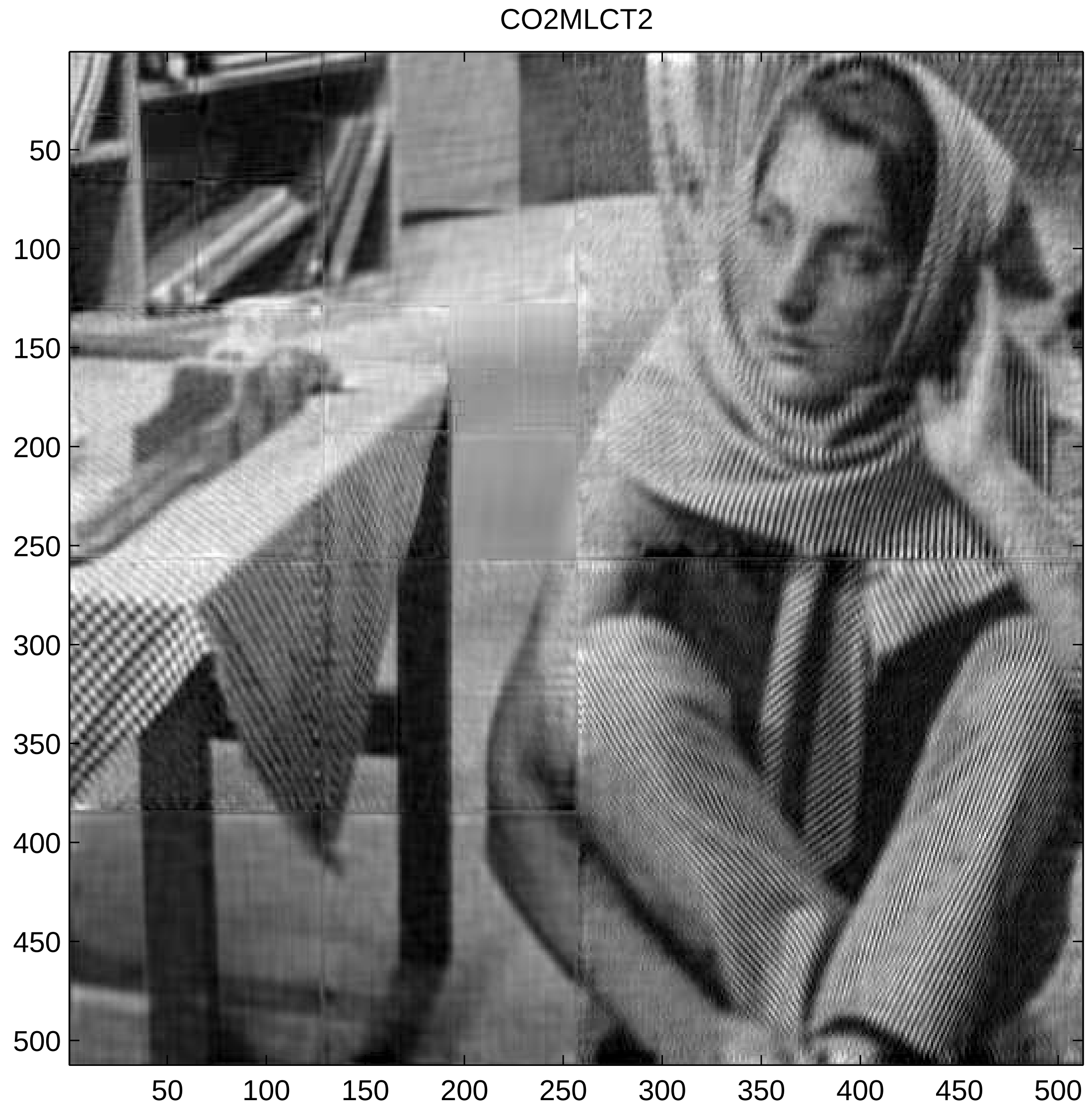

Figure C.38: 20:1 compression of Figure C.27 using CO2MLCT2.

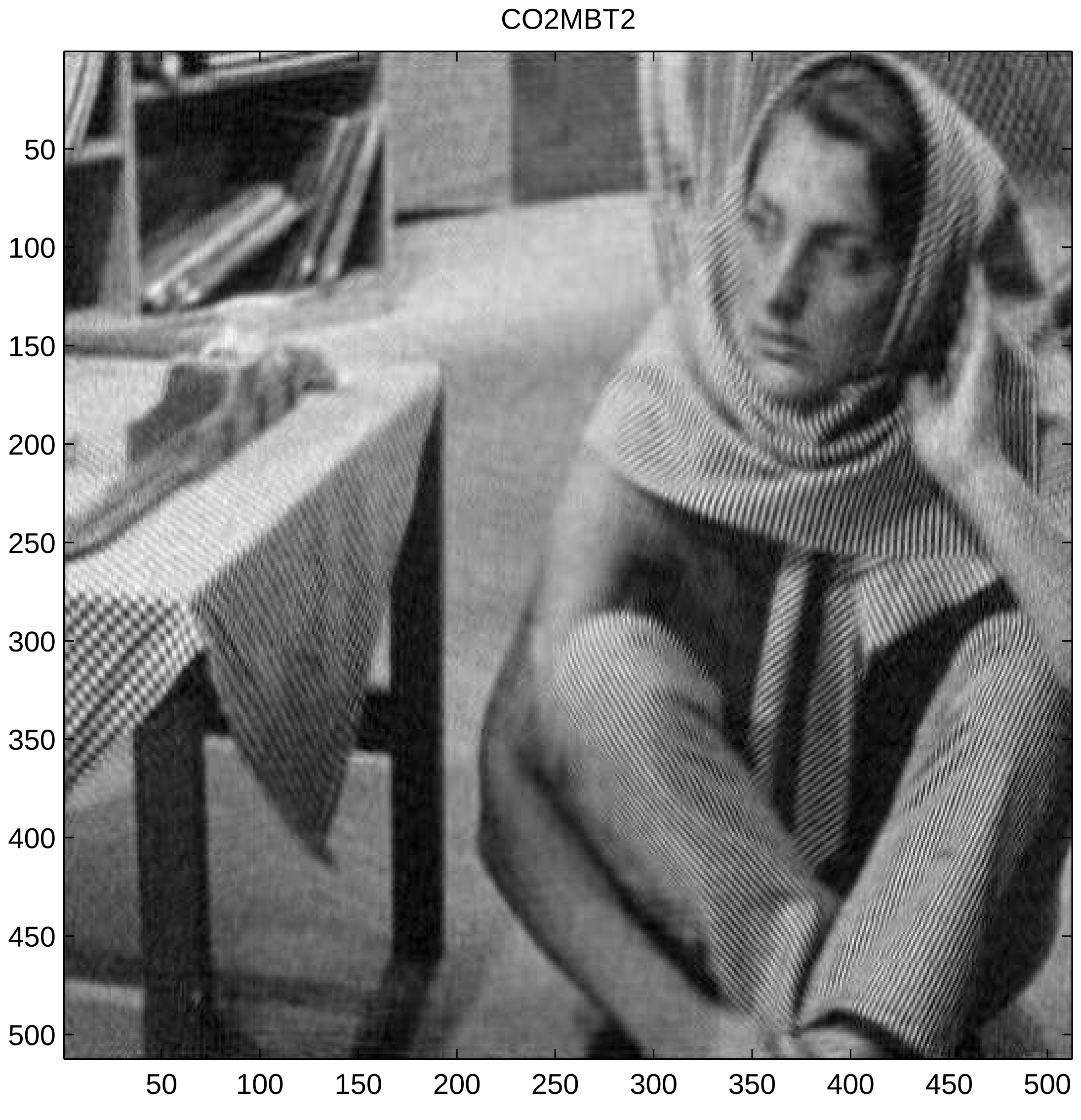

Figure C.39: 20:1 compression of Figure C.27 using CO2MBT2.

Bibliography

[1] Aharoni, G. and Averbuch, A. and Coifman, R. and Israeli, M. (1993). Local Cosine Transform - A Method for the Reduction of the Blocking Effect in JPEG. *J. of Mathematical Imaging and Vision, Special Issue of Wavelets*, **3**, 7-38.

[2] Ahmed, N. and Natarjan, T. and Rao, K. "Discrete cosine transform", *IEEE Trans. Comput.*, vol. C-23, 1974, 90-93.

[3] Auscher, P. and Weiss, G. and Wickerhauser, M. Local sine and cosine bases of Coifman and Meyer and the construction of smooth wavelets, *Wavelets: A Tutorial in Theory and Applications*, C. K. Chui, ed., 237-256, Academic Press, 1992.

[4] Averbuch, A. (1998). Edge/Image Enhancement. *Notes de Cours du Èmile Borel*, UMS **839** CNRS-UPMC, 1-29.

[5] Battle, G. *Wavelets & Renormalization.* World Scientific Pub Co, 1999.

[6] Bell, A. and Sejnowski, T. The 'Independent Components' of Natural Scenes are Edge Filters. Submitted to *Vision Research.*

[7] Bénichou, B. and Saito, N. "Sparsity versus statistical independence in adaptive signal representations: A case study of the spike process".

[8] Briggs, W. and Henson, V. *The DFT.* SIAM, Philadelphia, 1995.

[9] Brodatz, P. (1966) “Textures: A Photographic Album for Artists and Designers”, Dover Publ.

[10] Cover, T. and Thomas, J. (1985) “Elements of Information Theory”, John Wiley & Sons, Inc.

[11] Chan, Y. *Wavelet Basics.* Kluwer Academic Publishers, 1995.

[12] Coifman, R. and Donoho, D. (1998). Translation-Invariant De-Noising *Tech. Rep.*, Yale University and Stanford University, 1-26.

[13] Coifman, R. and Wickerhauser, M. (1992). Entropy-based algorithms for best basis selection *IEEE Trans. Inform. Theory*, Vol. **38**, 713-719.

[14] Daubechies, I. (1992). *Ten lectures on wavelets.* SIAM, Philadelphia, PA.

[15] Day, M. (1940). The spaces L^p with $0 < \mathrm{p} < 1$. *Bull. Amer. Math. Soc.*, Vol. **46**, 816-823.

[16] Deng, B. and Jawerth, B. and Peters, G. and Sweldens, W. (1993). Wavelet probing for compression based segmentation. *Mathematical Imaging*, Vol. **2034**, 266-276.

[17] DeValois, R. L. and DeValois, K. K. (1991) “Spatial Vision”, Oxford Psychology Series.

[18] Donoho, D. (1994). On Minimum Entropy segmentation. *Technical Report 450*, Department of Statistics, Stanford University.

[19] Donoho, D. (1998). Sparse components of images and optimal atomic decompositions. *Tech. Rep.*, Department of Statistics, Stanford University.

[20] Donoho, D. (1993). Nonlinear Wavelet Methods for Recovery of Signals, Densities, and Spectra from Indirect and Noisy Data. *American Mathematical Society*, Vol. **47**, 173-205.

[21] Donoho, D. (1998). Data Compression and Harmonic Analysis. *IEEE Transactions of Information Theory*, Vol. **44**, No. **6**, 2435-2476.

[22] Donoho, D. L. and Candès, E. J. (2000). Curvelets - A Surprisingly Effective Nonadaptive Representation For Objects With Edges. *Saint-Malo Proceedings*, Vanderbilt University Press, 1-10.

[23] Donoho, D. and Vetterli, M. and DeVore, R. and Daubechies, I. (1998). Data compression and harmonic analysis. *IEEE Trans. Inform. Theory*, **44**(6), 2435-2476.

[24] Fang, X. and Séré, E. (1994). Adapted multiple folding local trigonometric transforms and wavelet packets. *Appl. Comput. Harmonic Anal.* Vol. **1**, 169-179.

[25] Field, D. (1984). What is the goal of sensory coding? *Neural Computation*, Vol. **16**, 559-601.

[26] Field, D. (1987). Relations between the statistics of natural images and the response properties of cortical cells *J. Opt. Soc. Am. A*, Vol. **4**, No. **12**, 2379-2394.

[27] Field, D. and Hayes, A. and Hess, R. (1993). Contour Integration by the Human Visual System: Evidence for a Local "Association Field". *Vision Res.*, Vol. **33**, No. **2**, 173-193.

[28] Gilbert, C. (1992). Horizontal Integration and Cortical Dynamics *Neuron*, Vol. **9**, 1-13.

[29] Guler, S. and Derin, H. (1991). Boundary Detection in Textured Images Using Constrained Graduated Non-Convexity Method. *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, Vol. **1607**, 266-275.

[30] Hall, P. (1987). On Kullback-Leibler Loss and Density Estimation. *The Annals of Statistics*, Vol. **15**, No. **4**, 1491-1519.

[31] Hall, P. (1990). Akaike's information criterion and Kullback-Leibler Loss for histogram density estimation. *Probability Theory and Related Fields*, 449-467.

[32] Hall, P. and Morton, S.C. (1993). On the Estimation of Entropy. *Ann. Inst. Statist. Math.* , Vol. **45**, No. **1**, 69-88.

[33] Harris, F. (1978). On the Use of Windows for harmonic Analysis with the Discrete Fourier Transform. *Proceedings of the IEEE*, Vol. **66**, No. **1**, 51-84.

[34] van Hateren, J. and van der Schaaf, A. (1998). Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex. *Proc. R. Soc. Lond.*, Vol. **B**, No. **265**, 359-366.

[35] Heeger, D. and Bergen, J. Pyramid-Based Texture Analysis/Synthesis *Tech. Rep.*, Stanford University and SRI Davis Sarnoff Research Center, 1-8.

[36] Hirschman, I. (1956). A Note on Entropy, 152-156.

[37] Huang, J. and Mumford, D. (1999). Statistics of Natural Images and Models. *Tech. Rep.*, Brown University.

[38] Hubel, D. *Eye, Brain and Vision.* USA: Scientific American Library Series, 1988.

[39] Jain, A. and Ranganath, S. (1981). Extrapolation Algorithms for Discrete Signals with Applications in Spectral Estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing.*, Vol. **ASSP-29**, No. **4**, 830-845.

[40] Jain, A. *Fundamentals of Digital Image Processing.* Prentice-Hall, Inc., 1989.

[41] Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loève Procedure for the Characterization of Human Faces *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. **12**, No. **1**, 103-108.

[42] Kobayashi, T. and Shockey, D. Correlation of fracture surface topography with fatigue load spectrum *Proceedings of the 3rd National Turbine Engine High Cycle Fatigue Conference*, San Antonio, TX, February 2-5, 1998.

[43] Leclerc, Y. and Zucker, S. (1987). The local structure of image discontinuities in one dimension. *IEEE Trans. Pat. Anal. Mach. Intel.*, Vol. **PAMI-9**, No. **3**, 341-355.

[44] Leclerc, Y. (1989). Constructing simple stable descriptions for image partitioning. *Int. J. Comp. Vis.*, Vol. **3**, 73-102.

[45] Lewicki, M. and Olshausen, B. (1999). A probabilistic framework for the adaptation and comparison of image codes. Submitted to *J. Opt. Soc. Am. A*.

[46] Lin, Y. and Chang, T. and Kuo, C. (1993). Texture segmentation using wavelet packets. *Mathematical Imaging*, Vol. **2034**, 277-287.

[47] Mallat, S. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, 1998.

[48] Mallat, S. (1989). Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics Speech and Signal Processing*, **37**(12): 2091-2110.

[49] Malvar, H. (1990). Lapped transforms for efficient transform/subband coding. *IEEE Trans. Acoust. Sign. Speech Process.*, **38**: 969-978.

[50] Malvar, H. and Staelin, D. (1989). The LOT: Transform coding without blocking effects. *IEEE Transactions on Acoustics Speech and Signal Processing*, **37**: 553-559.

[51] Matviyenko, G. (1996). Optimized Local Trigonometric Bases. *Applied and Computational Harmonic Analysis*, **3**(24): 301-323.

[52] Meyer, F. (2000). Fast Compression of Seismic Data with Local Trigonometric Bases. *EDICS Category: IP 1.1 Coding*, Department of Electrical Engineering, University of Colorado at Boulder.

[53] Meyer, F. and Coifman, R. (1997). Brushlets: a tool for directional image analysis and image compression *Appl. Comput. Harmonic Anal.*, Vol. **4**, 147-187.

[54] Olshausen, B. and Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, Vol. **381**, 607-609.

[55] Olshausen, B. and Field, D. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, Vol. **37**, 3327-3338.

[56] Olshausen, B. and Field, D. (1996). Natural image statistics and efficient coding? *Network: Computation in Neural Systems*, Vol. **7**, 333-339.

[57] Papoulis, A. (1975) A New Algorithm in Spectral Analysis and Band-Limited Extrapolation. *IEEE Transactions on Circuits and Systems*, Vol. **CAS-22**. No. **9**, 735-742.

[58] Percival, D. B. and Walden, A. T. (1993). *Spectral Analysis For Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge University Press, 331-374.

[59] Porat, M. and Zeevi, Y. (1989). Localized Texture Processing in Vision: Analysis and Synthesis in the Gaborian Space *IEEE Trans. Biomed. Eng.*, Vol. **36**, No. **1**, 115-129.

[60] Saito, N. (2001). Image Approximation and Modeling via Least Statistically-Dependent Bases, *Pattern Recognition*, Vol. **34**, 1765-1784.

[61] Saito, N. (1999). The Local Fourier Dictionary: A Natural Tool for Data Analysis. In *Wavelet Applications in Signal and Image Processing VI*, A.F.

Laine, M.A. Unser, and A. Aldroubi, eds., Proc. SPIE 3813, 610-624, Invited paper.

[62] Saito, N. (1994). Simultaneous Noise Suppression and Signal Compression using a Library of Orthonormal Bases and the Minimum Description Length Criterion. *Wavelets in Geophysics*, 299-324.

[63] Saito, N. and Larson, B. and Bénichou, B. (2000). Sparsity vs. statistical independence from a best-basis view point *Wavelet Applications in Signal and Image Processing VIII*, A. Aldroubi, A.F. Laine, and M. A. Unser, eds., Proc. SPIE 4119, pp.474-486, 2000. Invited paper.

[64] Simoncelli, E. Statistical Models for Images: Compression, Restoration and Synthesis. *31st Asilomar Conference on Signals Systems, and Computers*, Pacific Grove, CA, November 2-5, 1997.

[65] Simoncelli, E. and Portilla, J. (1998) "Texture Characterization via Joint Statistics of Wavelet Coefficient Magnitudes", Fifth International Conference on Image Processing, Chicago, IL, IEEE Signal Processing Society, October 4-7 1998.

[66] Slepian, D. (1983)*Some Comments On Fourier Analysis, Uncertainty and Modeling*. SIAM Review, Vol. **25**. No. **3**, 379-393.

[67] Slepian, D. and Pollak, H. (1960). Prolate Spheroidal Wave Functions, Fourier Analysis and Uncertainty - I. *The Bell System Technical Journal.*, Vol., 43-63.

[68] Strang, G. (1999)*The Discrete Cosine Transform*. SIAM Review, Vol. **41**. No. **1**, 135-147.

[69] Villemoes, L. (1999). Adapted bases of time-frequency local cosines. Submitted to *Adapted and Computational Harmonic Analysis*.

[70] Weaver, H. *Applications of Discrete and Continuous Fourier Analysis.* John Wiley & Sons, Inc., New York, Library of Congress Cataloging in Publication Data, 1993.

[71] Webber, Chris. (2001) Predictions Of The Spontaneous Symmetry-breaking Theory For Visual Code Completeness And Spatial Scaling In Single-cell Learning Rules, *Neural Computation*, 13(5), 1023-1043.

[72] Wickerhauser, M. *Adapted Wavelet Analysis from Theory to Software.* A. K. Peters, Ltd., 1994.

[73] Wilson, R. (1987). Finite Prolate Spheroidal Sequences and Their Applications I: Generation and Properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, Vol. **PAMI-9**, No. **6**, 787-795.

[74] Wilson, R. and Spann, M. (1988). Finite Prolate Spheroidal Sequences and Their Applications II: Image Feature Description and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, Vol. **10**, No. **2**, 193-203.

[75] Xiao, H. and Rokhlin, V. and Yarvin, N. (2000). Prolate Spheroidal Wave Functions, Quadrature, and Interpolation. *Research Report* YALEU/DCS/ RR-1199, 1-59.

[76] Zhu, S. and Lee, T. and Yuille, A. (1994) Region Competition, *Tech. Rep.*, Harvard Robotics Laboratory, No. **94-10**, 1-36.