Spring 1997

Adaptive multiresolution visualization of large multidimensional multivariate scientific datasets

Pak Chung Wong

University of New Hampshire, Durham

ADAPTIVE MULTIRESOLUTION VISUALIZATION OF LARGE MULTIDIMENSIONAL MULTIVARIATE SCIENTIFIC DATASETS

BY

Pak Chung Wong

M.S., Western Michigan University (1986)
B.S., The Ohio State University (1984)

DISSERTATION

Submitted to the University of New Hampshire
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in

Computer Science

May 1997

UMI Number: 9730847

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

This dissertation has been examined and approved.

Dissertation director, R. Daniel Bergeron
Professor and Chair of Computer Science

Ted M. Sparr
Professor of Computer Science

Lee Zia

Lee L. Zia
Associate Professor of Mathematics

Philip J. Hatcher
Associate Professor of Computer Science

Raymond Greenlaw
Associate Professor of Computer Science

10 May 1997
Date

Dedication

To my parents, my brother, my sister, Patricia, Maria, and Christina.

Acknowledgments

This thesis would not have been possible without the help of many others, whom I would like to acknowledge here.

Professor R. Daniel Bergeron is a gifted teacher, advisor, scholar and raconteur. He has been the one constant source of guidance, wisdom and research interaction in my academic career. I am proud to be his Ph.D. student.

Professor Ted M. Sparr has been a continuing source of inspiration. His influence shapes my studies in large database management, and goes much deeper.

Professor Lee L. Zia has my gratitude for showing me the wonderful world of wavelets. He profoundly influenced how the direction of my research is defined today.

I consider myself fortunate to have Professor Ray Greenlaw and Professor Phil Hatcher in my depth exam committee as well as the thesis committee. Their critiques and insights have greatly improved the quality of my thesis.

I am grateful for the National Science Foundation Research Assistantship (under grant IRI-9117153), which funded my studies for most of my graduate career. I am also thankful for the generosity of the Department of Computer Science, who supplemented the assistantship at various points.

Many people made my graduate years at UNH pleasant and interesting, and they deserve credit as well. Thanks to Linda Andrews, Gina Ross, Andrew Evans, and everyone else in the department office for their enthusiastic support. I have had several fine officemates to make my academic experience more fun than I had imagined. In chronological order: David Kao, Jubin Dave, K. Suresh Kumar, Oktay Ariska, Andrew Crabb, Kudiarasu Nalliannan, Matt Plumlee, and Marc Czapcsynski.

I am grateful to my parents, brother and sister, for their endless support and advice, without which I am certain that I would not have come so far.

Finally, I want to specially thank my wife, Patricia, for her steady love and support. She has shown extraordinary patience in caring for our two much delighted daughters, Maria and Christina, during the course of researching and writing this thesis.

Contents

List of Tables

List of Figures

ABSTRACT

ADAPTIVE MULTIRESOLUTION VISUALIZATION OF LARGE MULTIDIMENSIONAL MULTIVARIATE SCIENTIFIC DATASETS

by

Pak Chung Wong
University of New Hampshire, May, 1997

The sizes of today's scientific datasets range from megabytes to terabytes, making it impossible to directly browse the raw datasets visually. This presents significant challenges for visualization scientists who are interested in supporting these datasets. In this thesis, we present an adaptive data representation model which can be utilized with many of the commonly employed visualization techniques when dealing with large amounts of data. Our hierarchical design also alleviates the long standing visualization problem due to limited display space. The idea is based on using compactly supported orthogonal wavelets and additional downsizing techniques to generate a hierarchy of fine to coarse approximations of a very large dataset for visualization.

An adaptive data hierarchy, which contains authentic multiresolution approximations and the corresponding error, has many advantages over the original data. First, it allows scientists to visualize the overall structure of a dataset by browsing its coarse approximations. Second, the fine approximations of the hierarchy provide local details of the interesting data subsets. Third, the error of the data representation can provide the scientist with information about the authenticity of the data approximation. Finally, in a client-server network environment, a coarse representation can increase the efficiency of a visualization process by quickly giving users a rough idea of the dataset before they decide whether to continue the transmission or to abort it. For datasets which require long rendering time, an authentic approximation of a very large dataset can speed up the visualization process greatly.

Variations on the main wavelet-based multiresolution hierarchy described in this thesis also lead to other multiresolution representation mechanisms. For example, we investigate the uses of norm projections and principal components to build multiresolution data hierarchies of large multivariate datasets. This leads to the development of a more flexible dual multiresolution visualization environment for large data exploration.

We present the results of experimental studies of our adaptive multiresolution representation using wavelets. Utilizing a multiresolution data hierarchy, we illustrate that information access from a dataset with tens of millions of data values can be achieved in real time. Based on these results, we propose procedures to assist in generating a multiresolution hierarchy of a large dataset. For example, the findings indicate that an ordinary computed tomography volume dataset can be represented effectively for some tasks by an adaptive data hierarchy with less than 1.5% of its original size.

Chapter 1

Introduction

As a tool for applying computers to science, visualization offers a way to see the unseen. As a technology, Visualization in Scientific Computing promises radical improvements in the human/computer interface and may make human-in-the-loop problems approachable.

– NSF Executive Summary on Visualization in Scientific Computing,
Computer Graphics, Vol. 21, No. 6, Nov 1987, p. vii

The role of visualization in scientific research is well documented [MDB87, REE$^+$94, Gal95, NMH97]. Our research focuses on one of the visualization sub-fields, known as *multidimensional multivariate (mdmv) visualization* [WB97a]. Mdmv visualization research is rich in concepts and practical applications. Previous research, unfortunately, suffers from the general shortcomings of all scientific visualization research: it cannot handle a massive volume of data.

It is often the case that scientists are primarily interested in analyzing only small subsets of a larger dataset at the highest resolution obtained. The sheer quantity of data, however, makes it infeasible for them to explore the data in its highest resolution. Very large scientific data needs to be reduced to a meaningful size to make it useful to the scientist. An important requirement, therefore, is to efficiently survey the full dataset at a lower resolution and then be able to identify and analyze interesting subsets of the data in greater detail.

The primary goal of our research is to develop an adaptive data representation that allows rapid analysis of vast amounts of scientific data in a progressive refinement environment. The goal has led to the development of a hierarchy of data processing stages, each of which is a coarse approximation of the data at the previous level of the hierarchy.

1.1 Hierarchical Multiresolution Data Access

A key point of hierarchical multiresolution data access is that data analysis must ultimately be done at the highest available spatial and temporal resolution, yet there is far too much data for all of it to be analyzed at this level. Therefore, the analysis process often requires a number of intermediate stages which contain smaller coarser representations of the data. Figure 1-1 depicts an adaptive zooming hierarchy of one dimensional scientific data. In this example, two intermediate stages are created in the hierarchy, and the second stage quickly shrinks to a relatively small size compared to the original. The irregular zooming ranges in intermediate stages allow scientists to emphasize important features and de-emphasize the less interesting areas.

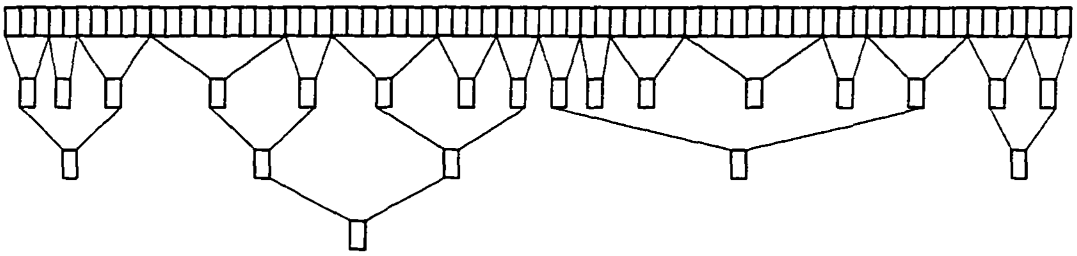


Figure 1-1: A zooming hierarchy of a one dimensional data stream.

1.2 Definition of Multiresolution Hierarchy

Our goal is to define operations or functions that accurately reduce the size of a very large segment of data into multiple sub-levels. For example, by continuously applying a pairwise average function to one dimensional data, we generate a multiresolution hierarchy with a compression ratio (i.e., the size of output divided by the size of input) of 0.5 at each level. The same idea can be applied to higher dimensional data. In each resolution, we take averages of each item's closest neighbors. In Figure 1-2 one dimensional data of four items is zoomed down to two by taking the average of every two items. In two dimensional data, four items are averaged in each resolution. Similarly, eight items are combined during each step in decomposing three dimensional data. Note that with n-dimensional data, each average operation uses 2^n data items. This implies a decomposition ratio

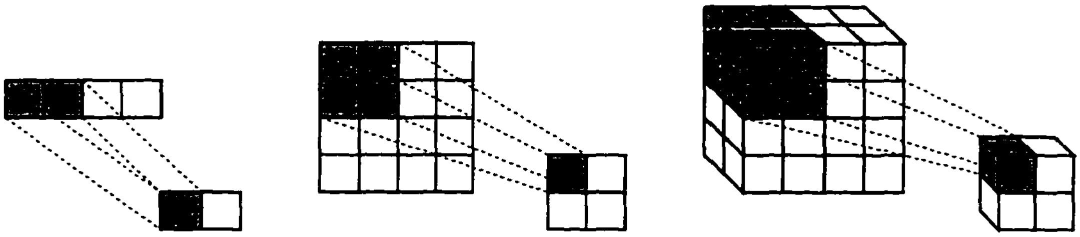


Figure 1-2: A pairwise averaging of one, two, and three dimensional scientific data.

of 2^{-n}.

In our multiresolution representation, new approximations are generated during the zooming process. Each of the sub-levels contains most, if not all, of the major characteristics of its next higher resolution. Although this new lower resolution representation may be useful for improving the performance of analysis, we are actually increasing total memory requirements by generating it. We improve memory utilization only if the new representation allows us to discard the higher resolution representation. If we need to revisit the higher resolution data later, we must be able to either recreate it or restore it. This dilemma occurs at every level of resolution. Fortunately, wavelets provide us with opportunities for building multiresolution hierarchies such that the intermediate levels of the hierarchy can be reconstructed from lower levels.

1.3 Hierarchical Representation Using Wavelets

The wavelet transform is a mathematical tool that can be utilized to extract or encode information. Given a one dimensional dataset with n items, an application of an orthogonal wavelet transform generates $\frac{n}{2}$ coefficients of *approximations*, and $\frac{n}{2}$ coefficients of *details*, as shown in Figure 1-3. The operation can be applied iteratively to the approximation part to get increasingly coarse zooming data. The number of data values of each level is reduced by 2^{-n} where n is the number of dimensions.

Wavelet transforms are invertible. The magic lies in the wavelet details of each resolution. If both the approximations and details of any one stage are available, it is possible to have a *lossless* reconstruction of the approximation part of the next higher resolution. For example, if both the approximations and details of stage $i + 3$ are available, the approximation of stage $i + 2$ can be

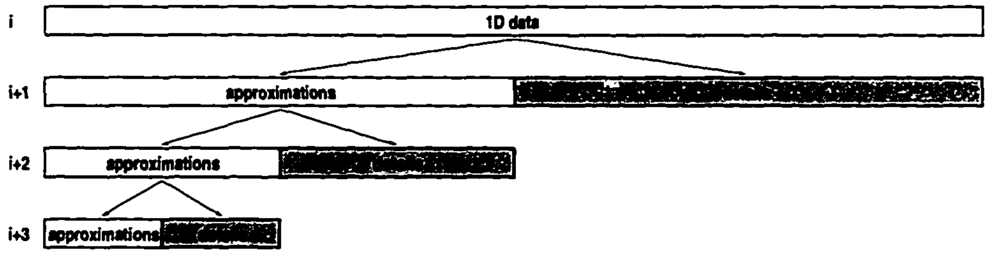


Figure 1-3: Wavelet transforms on one dimensional data stream.

reconstructed without loss. In fact, the whole hierarchy can be rebuilt in this manner provided that the details are available in all resolutions.

The pairwise average operation discussed in the Section 1.2 is, in fact, the approximation part of a *Haar* wavelet [Dau92]. In wavelet terminology, this pairwise average operation is only part of the decomposition of a wavelet transform. The simple pairwise average operation does not by itself allow reconstruction. The detail part of the Haar wavelet is computed from multiples of pairwise differences of the input dataset.

1.4 Overview of Contents

We begin in Chapter 2 with a detailed discussion of compactly supported orthogonal wavelets. In Chapter 3, we present the basic concepts of multidimensional multivariate visualization, and some of the most commonly used visualization techniques.

In Chapter 4, we introduce the concepts and design approaches of an adaptive multiresolution hierarchy. We then show how we can benefit from using wavelets to generate a multiresolution data representation of a very large dataset. Chapter 5 discusses the issues of data authenticity using wavelets. Chapter 6 presents a system prototype which illustrates the importance of tracking the error data.

In Chapters 7 and 8 we describe how these techniques can be used to support very large data visualization, and in Chapter 9, we show how a data hierarchy representation can be generated by norm projections.

Chapter 10 turns to a new perspective of data hierarchy based on multidimensional scaiing.

We describe the process of metric scaling and compare it to other visualization techniques. We investigate how the results can be used to enhance previously described visualization techniques.

Chapter 11 presents the experimental results of our study on multiresolution data hierarchy. Real life high dimensional datasets are used to demonstrate the performance of our design. Lastly Chapter 12 contains some concluding thoughts about the techniques presented in this thesis, and lists some problems that remain unresolved.

In addition to some basic knowledge of vector spaces, we assume some background in Hilbert spaces and eigenvectors, particularly in Chapters 2 and 10. We refer the reader unfamiliar with these topics to a basic text on linear spaces, such as [DM90]. Most of the work in this thesis originally appeared in a sequence of eleven papers: [WB97b, WB97d, WB96a, WCB96, WB96b, WB95a, WB94, WB97a, WB97c, WB95b, WB93].

Chapter 2

A Child's Garden of Wavelet Transforms

Nothing to do with sea or anything else. Over and over it vanishes with the wave.
– Shinkichi Takahashi

In this chapter, we establish some facts about scaling functions and wavelets that are used later in the thesis. See Appendix A for definitions of symbols used and Appendix B for mathematical definitions on vector spaces, normed linear spaces, Hilbert spaces, and other definitions needed for a good understanding of wavelets. The properties and characteristics of compactly supported orthogonal wavelets are further discussed in Chapter 5.

2.1 Dilation and Translation

Wavelet transforms involve two primary operations applied to time/space functions: *dilation* by 2

$$f(x) \rightarrow f(2x),$$

and *translation*

$$f(x) \rightarrow f(x+c)$$

where c is a constant $\in \mathbb{N}$. In the following discussion, the term *function* is often interchanged with the term *vector* (see Definition 2 in Appendix B).

2.2 Haar Basis as an Example

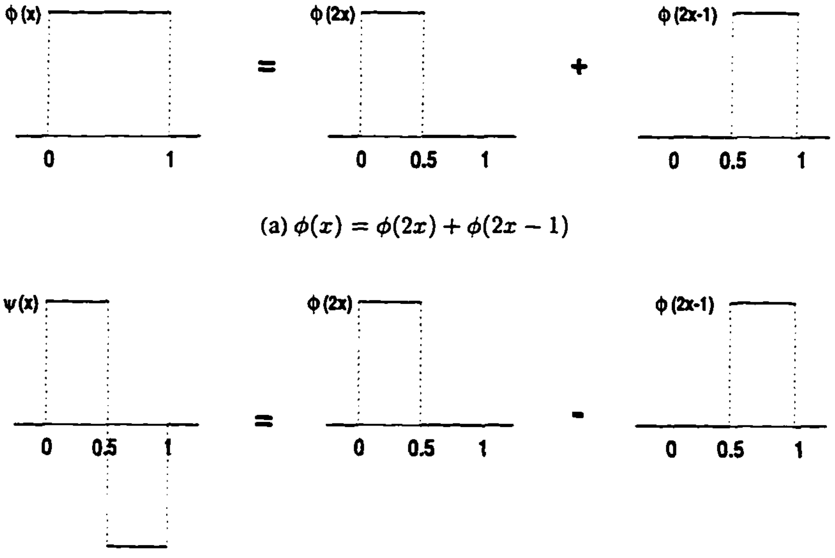
Before we move on to the discussion of general wavelets, we introduce a simple example to illustrate some of the basic concepts. Here we consider a function ϕ where

$$\phi(x) = \begin{cases} 1, & 0 \leq x < 1 \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

and a function ψ where

$$\psi(x) = \begin{cases} 1, & 0 \leq x < \frac{1}{2} \\ -1, & \frac{1}{2} \leq x < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

The function ϕ is the *scaling* function and ψ is the *mother wavelet* of the *Haar* wavelet transform [Dau92, Dau93]. Both can be constructed by a linear combination of dilations and translations of ϕ, as depicted in Figure 2-1. The function $\psi(x)$ shown in Figure 2-1 is piecewise constant over one-half of the unit interval. Finer resolution wavelets can be generated by a combination of dilations and translations of $\psi(x)$. In Figure 2-2, the function $\psi(2x)$, which is piecewise constant over



(a) $\phi(x) = \phi(2x) + \phi(2x-1)$

(b) $\psi(x) = \phi(2x) - \phi(2x-1)$

Figure 2-1: The constructions of a) ϕ and b) ψ.

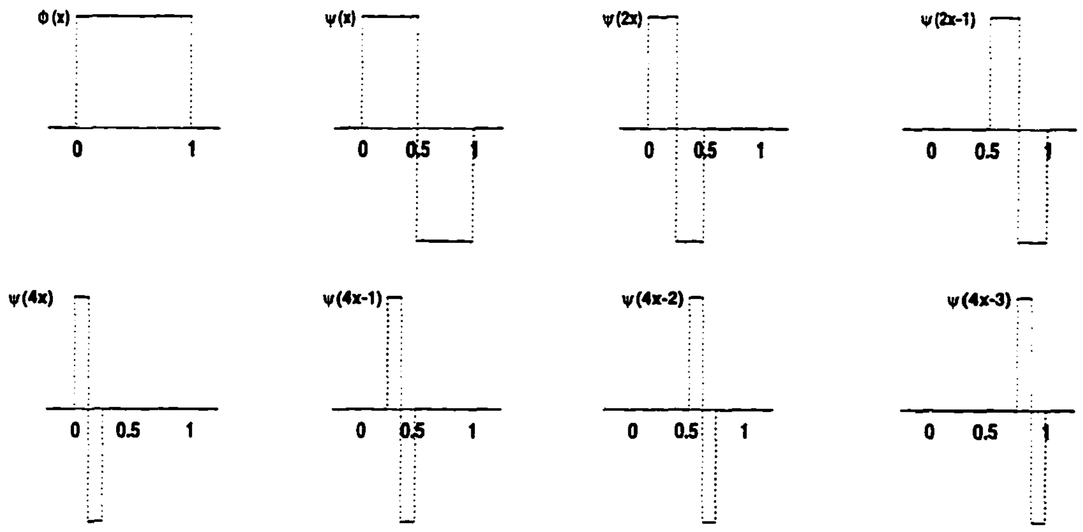


Figure 2-2: Left to right, top to bottom: $\phi(x)$, $\psi(x)$, $\psi(2x)$, $\psi(2x-1)$, $\psi(4x)$, $\psi(4x-1)$, $\psi(4x-2)$, and $\psi(4x-3)$ of a Haar wavelet.

one-fourth of the unit interval, is generated by a dilation of $\psi(x)$. The function $\psi(2x-1)$ is generated by a translation of $\psi(2x)$. Similarly, finer resolution wavelets $\psi(4x)$, $\psi(4x-1)$, $\psi(4x-2)$, and $\psi(4x-3)$, which are piecewise constant over one-eighth of the unit interval, are generated in the same fashion. These functions can be represented by vectors, each entry of which represents one-eighth of the unit interval as follows:

$$\phi(x) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \psi(x) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}, \psi(2x) = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \psi(2x-1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix},$$

$$\psi(4x) = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \psi(4x-1) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \psi(4x-2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \psi(4x-3) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}.$$

The next step is to show how to represent an equally spaced function $f(x)$ on the unit interval with the functions ϕ and ψ of different resolutions. Suppose we have an equally spaced step function $f(x)$ on the unit interval defined by

$$f = \begin{bmatrix} 12 \\ 6 \\ 3 \\ -1 \\ 5 \\ -1 \\ 1 \\ -1 \end{bmatrix}. \tag{2.3}$$

The entries in $f(x)$ represent the value of the function over one-eighth of the unit interval. Our goal is to find constants $\{\alpha_i, 0 \leq i \leq 7\}$ such that

$$\begin{aligned} f &= \alpha_0\phi(x) + \alpha_1\psi(x) + \alpha_2\psi(2x) + \alpha_3\psi(2x-1) + \alpha_4\psi(4x) + \alpha_5\psi(4x-1) \\ &\quad + \alpha_6\psi(4x-2) + \alpha_7\psi(4x-3). \end{aligned}$$

The process starts with the calculations of the *averages* (given by $\frac{x_1+x_2}{2}$) and one half of the *differences* (given by $\frac{x_1-x_2}{2}$) between consecutive pairs of entries of the function f. In Figure 2-3, function f_1, which is piecewise constant over one-fourth of the unit interval, is generated by applying average operations to consecutive pairs of entries of f. The function f_1 is also known as the

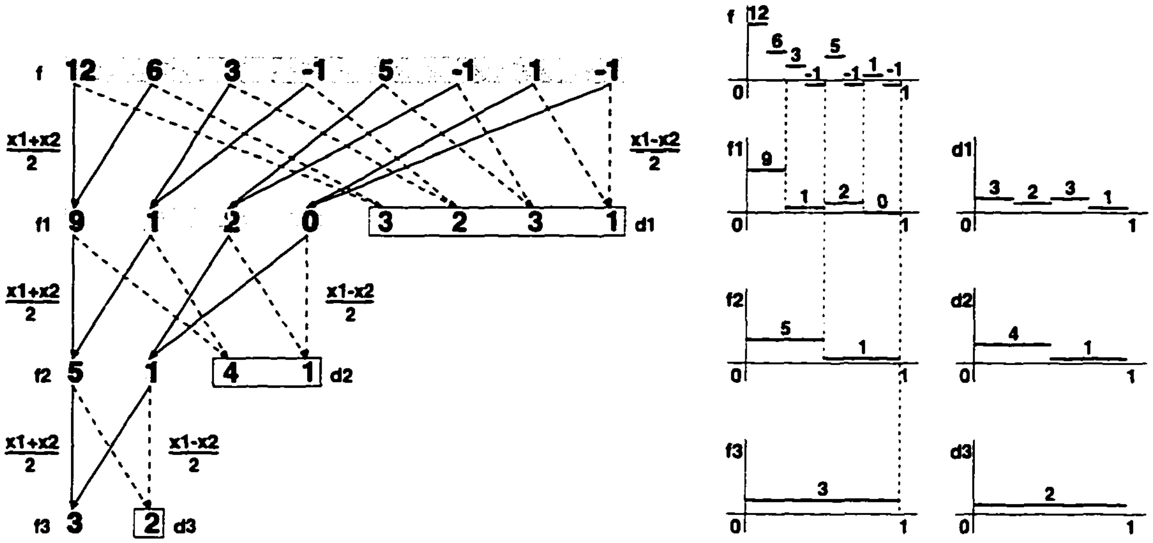


Figure 2-3: Multiresolution Haar decomposition of a piecewise constant function over the unit interval.

wavelet approximation of f. The differences between consecutive pairs of entries of f, which are represented by the function d_1 in Figure 2-3, are called *wavelet coefficients* of the decomposition. They are also referred to as the *wavelet details* of the decomposition. A lossless reconstruction of f is possible if both f_1 and d_1 are available.

By applying another decomposition to the approximation function f_1, we have two new functions, f_2 and d_2, which represent the wavelet approximation and the details of the decomposition of f_1. To complete the Haar transform for this function, another decomposition is applied to the approximation function f_2. The resultant functions, f_3 and d_3, are now piecewise constant over the unit interval. The only entry of function f_3 is indeed the average of all the entries in f.

This decomposition procedure is sometimes called a pyramidal algorithm [Mal89]. The functions f, f_1, f_2, and f_3, which are depicted in shaded rectangles in Figure 2-3, represent a multiresolution approximation hierarchy of the function f, with f being the finest and f_3 the coarsest.

The entries in functions f_3, d_3, d_2, and d_1 (i.e., $\{3, 2, 4, 1, 3, 2, 3, 1\}$) are indeed the constants

$\{\alpha_i, 0 \le i \le 7\}$ described in Equation 2.4. So the decomposition of the function $f(x)$ is given by

$$f = \begin{bmatrix} 12 \\ 6 \\ 3 \\ -1 \\ 5 \\ -1 \\ 1 \\ -1 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} + 4 \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} +$$

$$3 \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} . \qquad (2.4)$$

2.3 Multiresolution Transform and Scaling Function

In this section, we present the definition of a *multiresolution approximation* of $L^2(\mathbb{R})$. The definition and properties of the scaling function, ϕ, which was introduced in Section 2.2, are discussed. The implementation of a hierarchical multiresolution transform is also described.

2.3.1 Multiresolution Approximation of $L^2(\mathbb{R})$

We briefly describe the vector space, $V_{2^j} \subset L^2(\mathbb{R})$ for $j \in \mathbb{Z}$, which we called a multiresolution approximation of $L^2(\mathbb{R})$. The vector space V_{2^j} can be interpreted as the set of all approximations at the resolution 2^j of functions in $L^2(\mathbb{R})$. Any vector space that satisfies the properties described below is a multiresolution approximation of $L^2(\mathbb{R})$. In our discussion, we often use the term 2^j to describe an arbitrary resolution. This is because in an orthogonal wavelet, the size of the function $f(x)$ is always an integer power of two. Before we move on, we need to define a new approximation operator and its projection characteristics.

Let A_{2^j} be a linear operator which approximates a function $f(x) \in L^2(\mathbb{R})$. We call the approximation a *projection* since

$$A^2_{2^j} f(x) = A_{2^j}(A_{2^j} f(x)) = A_{2^j} f(x). \tag{2.5}$$

The operator, A_{2^j}, is a projection operator on a particular vector space $V_{2^j} \subset L^2(\mathbb{R})$. This space, V_{2^j}, has the following properties:

1. For functions $f(x), g(x) \in V_{2^j} \subset L^2(\mathbb{R})$, $\|g(x) - f(x)\| \geq \|A_{2^j} f(x) - f(x)\|$. This means that other function $g(x) \in V_{2^j}$ has a larger difference with function $f(x)$ than $A_{2^j} f$. In other words, A_{2^j} is the best approximation to f among all functions in V_{2^j}. This is shown in Figure 2-4. The shadow area is the span (see Definition 5 in Appendix B) of approximations which defines V_{2^j}. The difference $f(x) - A_{2^j} f(x)$ is indicated by the dotted vector. By definition, A_{2^j} is an orthogonal projection of $f(x)$ onto vector space V_{2^j}.

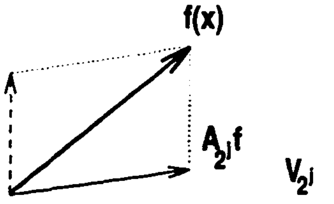


Figure 2-4: Projection of $f(x)$ onto space V_{2^j}.

2. The approximated function $A_{2^{j+1}} f$ contains all the information to compute $A_{2^j} f$. This can be explained visually with a simple example based on the Haar functions. For simplicity, let $j = 0$. Let $f(x) \in V_1$, and $g(x), h(x) \in V_2$. Since function $f(x)$ is piecewise constant on the unit interval, it can be represented by functions such as $g(x)$ which is piecewise constant on a half unit interval, as depicted in Figure 2-5. This is interpreted as

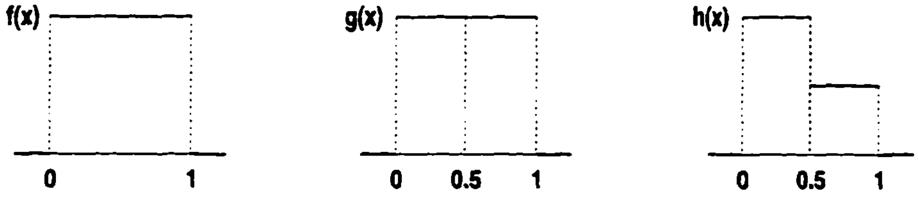


Figure 2-5: Given $f(x) \in V_1$, and $g(x), h(x) \in V_2$, then $V_1 \subset V_2$ but $V_2 \not\subset V_1$.

$$V_{2^j} \subset V_{2^{j+1}}$$

for $j \in \mathbb{Z}$. However, function $h(x) \in V_2$ cannot be represented by $f(x) \in V_1$. Thus

$$V_{2^{j+1}} \not\subset V_{2^j}.$$

3. Since the projection operator, A_{2^j}, is used in all resolutions, the approximated function space can be derived from one resolution to another according to their resolution values. This can be written as

$$f(x) \in V_{2^j} \Leftrightarrow f(2x) \in V_{2^{j+1}}$$

for $j \in \mathbb{Z}$. Again we use a box function with $j = 0$ to show the idea. In Figure 2-6, $f(x) \in V_1$, and $g(x)$ is a dilated $f(x)$, i.e., $g(x) = f(2x)$. And $g(x)$ is piecewise constant on half unit intervals, i.e., $g(x) \in V_2$.

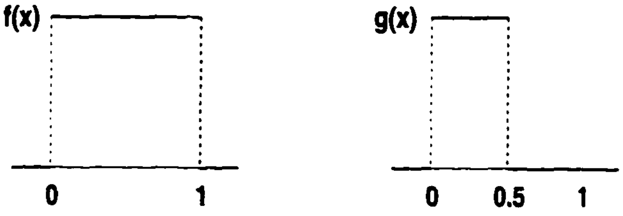


Figure 2-6: Given $f(x) \in V_1$, and $g(x) \in V_2$, then $f(x) \subset V_1 \leftrightarrow g(x) \subset V_2$.

4. When $f(x)$ is translated by $2^{-j}n$ for $n \in \mathbb{Z}$, $A_{2^j}f(x)$ is translated by the same amount. That is to say,

$$A_{2^j}(f(x - 2^{-j}n)) = (A_{2^j}f)(x - 2^{-j}n) \text{for } n \in \mathbb{Z}.$$

5. Since $A_{2^j}f(x)$ is obtained by a projection from $f(x)$, by definition, $A_{2^j}f(x) \subset f(x)$. As j decreases, the approximated function contains less and less information. On the other hand, as j increases, the approximated function converges to $f(x)$ eventually. By Definition 29 in Appendix B, these can be described as

$$\lim_{j \to +\infty} V_{2^j} = \bigcup_{j \in \mathbf{Z}} V_{2^j}$$

which is *dense* (see Definition 11 in Appendix B) in $L^2(\mathbb{R})$, and

$$\lim_{j \to -\infty} V_{2^j} = \bigcap_{j \in \mathbf{Z}} V_{2^j} = \{0\}.$$

Theorem 1 For $j \in \mathbb{Z}$, let V_{2^j} be a multiresolution approximation of $L^2(\mathbb{R})$. There exists a unique scaling function $\phi(x) \in L^2(\mathbb{R})$ such that if we set $\phi_{2^j}(x) = 2^j\phi(2^j x)$ for $j \in \mathbb{Z}$, then $\sqrt{2^{-j}}\ \phi_{2^j}(x - \sqrt{2^{-j}}\ n)$, is an *orthonormal basis* (see Definition 28 in Appendix B) of V_{2^j}.

The proof of this theorem can be found in [Mal89]. The Fourier Transform of $\phi(x)$ shows that most energy is concentrated on $[-\pi/2, \pi/2]$ [Dau92]. In signal processing terms, it has the shape of a low-pass filter. A low-pass filter filters out the high frequency component, and passes the low frequency range without attenuation.

From Definition 28 in Appendix B, for all $f(x) \in L^2(\mathbb{R})$, the orthogonal projection of $f(x)$ onto V_{2^j} can be represented as:

$$\begin{aligned} A_{2^j}f(x) &= \sum_{n=-\infty}^{+\infty} \langle f(u),\ (\sqrt{2^{-j}}\ \phi_{2^j}(u - 2^{-j}n))\rangle\ (\sqrt{2^{-j}}\ \phi_{2^j}(x - 2^{-j}n)) \\ &= 2^{-j} \sum_{n=-\infty}^{+\infty} \langle f(u),\ \phi_{2^j}(u - 2^{-j}n)\rangle\ \phi_{2^j}(x - 2^{-j}n) \\ &= 2^{-j} \sum_{n=-\infty}^{+\infty} (A^d_{2^j}f)\ \phi_{2^j}(x - 2^{-j}n) \qquad (2.6) \end{aligned}$$

where

$$A^d_{2^j}f = \langle f(u),\ \phi_{2^j}(u - 2^{-j}n)\rangle. \qquad (2.7)$$

We call the set of inner products, $A^d_{2^j}f$, the *discrete approximation* at the resolution 2^j. The infinite sum results from the infinite number of vectors in the orthonormal basis in V_{2^j}. If we consider a finite orthonormal basis with vectors x_1 and x_2, Equation 2.7 can be depicted in Figure 2-7. In an

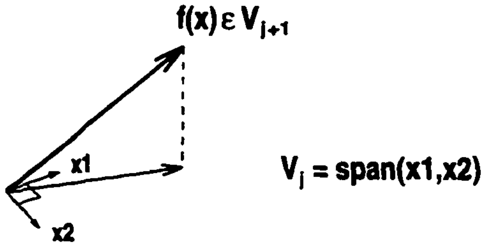


Figure 2-7: Discrete approximation of $f(x)$ at the resolution 2^j..

infinite dimensional space, the shaded region represents $span(\sqrt{2^{-j}}\ \phi_{2^j}(x - \sqrt{2^{-j}}\ n))$. But our finite example only shows $span(x_1,\ x_2)$.

2.3.2 Implementation of a Multiresolution Transform

We describe an iterative algorithm to calculate the approximations. For $n \in \mathbb{Z}$, $\phi_{2^j}(x - 2^{-j}n) \in V_{2^j} \subset V_{2^{j+1}}$. $\phi_{2^j}(x - 2^{-j}n)$ can be expanded in $V_{2^{j+1}}$ as

$$\begin{aligned}
\phi_{2^j}(x - 2^j n) &= \sum_{k=-\infty}^{+\infty} \langle \phi_{2^j}(u - 2^{-j}n), \sqrt{2^{-j-1}}\phi_{2^{j+1}}(u - 2^{-j-1}k)\rangle\ \sqrt{2^{-j-1}}\phi_{2^{j+1}}(x - 2^{-j-1}k) \\
&= 2^{-j-1} \sum_{k=-\infty}^{+\infty} \langle \phi_{2^j}(u - 2^{-j}n),\ \phi_{2^{j+1}}(u - 2^{-j-1}k)\rangle\ \phi_{2^{j+1}}(x - 2^{-j-1}k). \qquad (2.8)
\end{aligned}$$

By dilating the scaling functions and changing variables, the inner product integral can be written as:

$$\begin{aligned}
& 2^{-j-1}\ \langle \phi_{2^j}(u - 2^{-j}n),\ \phi_{2^{j+1}}(u - 2^{-j-1}k)\rangle \\
= {}& 2^{-j-1} \int_{-\infty}^{+\infty} \phi_{2^j}(u - 2^{-j}n)\ \phi_{2^{j+1}}(u - 2^{-j-1}k)\ du \\
= {}& 2^{-j-1} \int_{-\infty}^{+\infty} 2^j\phi(2^j(u - 2^{-j}n))\ 2^{j+1}\phi(2^{j+1}(u - 2^{-j-1}k))\ du \\
= {}& 2^j \int_{-\infty}^{+\infty} \phi(2^j u - n)\ \phi(2^{j+1}u - k)\ du \\
= {}& 2^j \int_{-\infty}^{+\infty} \phi(2^{-1}w)\ \phi(2^{j+1}(2^{-j}(2^{-1}w + n)) - k)\ 2^{-j-1}\ dw \\
= {}& 2^j \int_{-\infty}^{+\infty} \phi(2^{-1}w)\ \phi(2(2^{-1}w + n) - k)\ 2^{-j-1}\ dw \\
= {}& \int_{-\infty}^{+\infty} 2^{-1}\phi(2^{-1}w)\ \phi(w + 2n - k)\ dw \\
= {}& \int_{-\infty}^{+\infty} \phi_{2^{-1}}(w)\ \phi(w - (k - 2n))\ dw
\end{aligned}$$

$$
\begin{aligned}
&= \int_{-\infty}^{+\infty} \phi_{2^{-1}}(u)\, \phi(u-(k-2n))\, du \\
&= \langle \phi_{2^{-1}}(u),\, \phi(u-(k-2n))\rangle. \qquad (2.9)
\end{aligned}
$$

Substituting Equation 2.9 into Equation 2.8, we get

$$\phi_{2^j}(x-2^j n) = \sum_{k=-\infty}^{+\infty} \langle \phi_{2^{-1}}(u),\, \phi(u-(k-2n))\rangle\, \phi_{2^{j+1}}(x-2^{-j-1}k). \qquad (2.10)$$

Substituting Equation 2.10 into Equation 2.7, we have

$$
\begin{aligned}
A^d_{2^j} f &= \langle f(u),\, \phi_{2^j}(u-2^{-j}n)\rangle \\
&= \left\langle f(u), \left(\sum_{k=-\infty}^{+\infty} \langle \phi_{2^{-1}}(u), \phi(u-(k-2n))\rangle\, \phi_{2^{j+1}}(u-2^{-j-1}k) \right) \right\rangle \\
&= \left(\sum_{k=-\infty}^{+\infty} \langle \phi_{2^{-1}}(u), \phi(u-(k-2n))\rangle \right) \langle f(u),\, \phi_{2^{j+1}}(u-2^{-j-1}k)\rangle. \qquad (2.11)
\end{aligned}
$$

Let H be the discrete filter with impulse response $h(n)$ such that for $n \in \mathbb{Z}$,

$$h(n) = \langle \phi_{2^{-1}}(u),\, \phi(u-n)\rangle, \qquad (2.12)$$

and let $\tilde{H}$ be the symmetric filter such that $\tilde{h}(n) = h(-n)$. By substituting Equation 2.12 into Equation 2.11, we have

$$A^d_{2^j} f = \langle f(u),\, \phi_{2^j}(u-2^{-j}n)\rangle = \sum_{k=-\infty}^{+\infty} \tilde{h}(2n-k)\, \langle f(u),\, \phi_{2^{j+1}}(u-2^{-j-1}k)\rangle. \qquad (2.13)$$

Mallat [Mal89] describes Equation 2.13 with a tree hierarchy. The approximation, $A^d_{2^j} f$, can be calculated by convolving $A^d_{2^{j+1}} f$ with $\tilde{H}$ and keeping every other sample of output. This is illustrated in Figure 2-8 which can be explained by the Haar example in Section 2.2. The convolution of

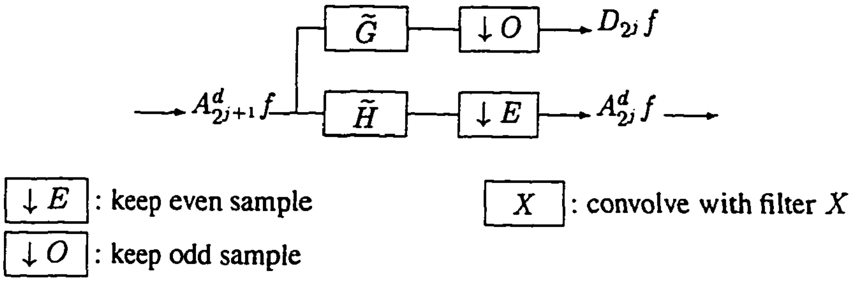

$\tilde{G}$
$\downarrow O$
$D_{2^j} f$
$A^d_{2^{j+1}} f$
$\tilde{H}$
$\downarrow E$
$A^d_{2^j} f$
$\downarrow E$: keep even sample
$\downarrow O$: keep odd sample
X : convolve with filter X

Figure 2-8: $A^d_{2^{j+1}} f$ is decomposed into a coarser approximation $A^d_{2^j} f$ and detail $D_{2^j} f$.

$\tilde{H}$ with function f of Equation 2.3 is given by

$$A^d_{2^2} f = \tilde{H}\, f = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & & & & & \\ & & \frac{1}{2} & \frac{1}{2} & & & & \\ & & & & \frac{1}{2} & \frac{1}{2} & & \\ & & & & & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 12 \\ 6 \\ 3 \\ -1 \\ 5 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix}.$$

By repeating the same process as described in Figure 2-8, we have

$$A^d_{2^1} f = \tilde{H}\, A^d_{2^2} f = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & \\ & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \tag{2.14}$$

and

$$A^d_{2^0} f = \tilde{H}\, A^d_{2^1} f = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \end{bmatrix}, \tag{2.15}$$

which is the last and the only high frequency (α_0 in Equation 2.4) left in the vector.

2.4 The Wavelet Representation

2.4.1 The Detail Signal

The second part of a wavelet transform involves the extraction of the difference, known as the *detail*, between $A_{2^{j+1}} f(x)$ and $A_{2^j} f(x)$. By applying Theorem 5 and Definition 29 in Appendix B, the detail at resolution 2^j, O_{2^j}, is given by the orthogonal projection of $f(x)$ onto the *orthogonal complement* of V_{2^j}, i.e.,

$$O_{2^j} = (V_{2^j})^{\perp},$$

and

$$O_{2^j} \oplus V_{2^j} = V_{2^{j+1}}. \tag{2.16}$$

To calculate the projection on O_{2^j}, we need to define its orthonormal basis.

Theorem 2 For $j \in \mathbb{Z}$, let $(V_{2^j})_{j\in\mathbb{Z}}$ be a multiresolution vector space sequence, $\phi(x)$ the scaling function, and H the corresponding conjugate filter. Let $\Psi(x)$ be a function whose Fourier transform is given by $\hat{\psi}(\omega) = G(\frac{\omega}{2})\hat{\phi}(\frac{\omega}{2})$ with $G(\omega) = e^{-i\omega}\overline{H(\omega+\pi)}$ where $\hat{\phi}$ is the Fourier Transform of ϕ. Let $\psi_{2^j}(x) = 2^j\psi(2^jx)$ denote the dilation of $\psi(x)$ by 2^j. Then $(\sqrt{2^{-j}}\psi_{2^j}(x - 2^jn))_{n\in\mathbb{Z}}$ is an orthonormal basis of O_{2^j} and $(\sqrt{2^{-j}}\psi_{2^j}(x - 2^jn))_{n\in\mathbb{Z}^2}$ is an orthonormal basis of $L^2(\mathbb{R})$. $\psi(x)$ is called an orthogonal wavelet.

The proof of this theorem can be found in [Mal89]. According to Definition 28 in Appendix B and Theorem 2, the orthogonal projection onto space O_{2^j}, $P_{O_{2^j}}f(x)$, can be represented as:

$$\begin{aligned} P_{O_{2^j}}f(x) &= \sum_{n=-\infty}^{+\infty} \langle f(u),\ (\sqrt{2^{-j}}\,\psi_{2^j}(u-2^{-j}n))\rangle\ (\sqrt{2^{-j}}\,\psi_{2^j}(x-2^{-j}n)) \\ &= 2^{-j}\sum_{n=-\infty}^{+\infty} \langle f(u),\ \psi_{2^j}(u-2^{-j}n)\rangle\ (\psi_{2^j}(x-2^{-j}n)). \end{aligned} \tag{2.17}$$

We call the set of inner products,

$$D_{2^j}f = \langle f(u),\ \psi_{2^j}(u-2^{-j}n)\rangle,$$

the *discrete detail* at resolution 2^j. It contains the *difference* of information between $A^d_{2^{j+1}}f$ and $A^d_{2^j}f$.

2.4.2 Implementation of an Orthogonal Wavelet

The implementation of an orthogonal wavelet is very similar to the previous discussion of a multiresolution transformation. For $n \in \mathbb{Z}$, $\psi_{2^j}(x - 2^{-j}n) \in O_{2^j} \subset V_{2^{j+1}}$. So $\psi_{2^j}(x - 2^{-j}n)$ can be expanded in $V_{2^{j+1}}$ as

$$\begin{aligned} \psi_{2^j}(x-2^jn) &= \sum_{k=-\infty}^{\infty} \langle \psi_{2^j}(u-2^{-j}n),\ \sqrt{2^{-j-1}}\phi_{2^{j+1}}(u-2^{-j-1}k)\rangle\ \sqrt{2^{-j-1}}\phi_{2^{j+1}}(x-2^{-j-1}k) \\ &= 2^{-j-1}\sum_{k=-\infty}^{\infty} \langle \psi_{2^j}(u-2^{-j}n),\ \phi_{2^{j+1}}(u-2^{-j-1}k)\rangle\ \phi_{2^{j+1}}(x-2^{-j-1}k). \end{aligned} \tag{2.18}$$

Much like Equations 2.8 to 2.11, we have

$$\begin{aligned} D_{2^j}f &= \langle f(u),\ \psi_{2^j}(u-2^{-j}n)\rangle \\ &= \sum_{k=-\infty}^{+\infty} \langle \psi_{2^{-1}}(u), \phi(u-(k-2n))\rangle\ \langle f(u),\ \phi_{2^{j+1}}(u-2^{-j-1}k)\rangle. \end{aligned} \tag{2.19}$$

Let G be the discrete filter with impulse response

$$g(n) = \langle \psi_{2^{-1}}(u),\ \phi(u-n)\rangle, \tag{2.20}$$

and let $\tilde{G}$ be the symmetric filter such that $\tilde{g}(n) = g(-n)$. By substituting Equation 2.20 into Equation 2.19, we have

$$\langle f(u),\ \psi_{2^j}(u-2^{-j}n)\rangle = \sum_{k=-\infty}^{+\infty} \tilde{g}(2n-k)\ \langle f(u),\ \phi_{2^{j+1}}(u-2^{-j-1}k)\rangle.$$

This process can be described by the tree hierarchy depicted in Figure 2-8 of the previous section. The detail $D_{2^j}f$ can be obtained by convolving $A^d_{2^{j+1}}f$ with $\tilde{G}$ and retaining every other sample of the output. A more detailed explanation of the implementation can be found in [Mal89].

Once again, we use the Haar example in Section 2.2 to explain the process. The convolution of f in Equation 2.3 with $\tilde{G}$ is given by

$$D_{2^2}f = \tilde{G}\ f = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & & & & & & \\ & & \frac{1}{2} & -\frac{1}{2} & & & & \\ & & & & \frac{1}{2} & -\frac{1}{2} & & \\ & & & & & & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 12 \\ 6 \\ 3 \\ -1 \\ 5 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

By repeating the same process, we have

$$D_{2^1}f = \tilde{G}\ A^d_{2^2}f = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & & \\ & & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix},$$

and

$$D_{2^0} f = \tilde{G}\, A^d_{2^1} f = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \end{bmatrix}. \tag{2.21}$$

This completes a Haar wavelet decomposition.

G is the mirror filter of H. While H is a low-pass filter, G is a high-pass filter [Mal89]. $A^d_{2^j} f$ gives the approximation of function f, while $D_{2^j} f$ gives a measure of irregularity of the information between consecutive resolutions. Whenever the difference between $A^d_{2^{j+1}} f$ and $A^d_{2^j} f$ are significant, $D_{2^j} f$ has high values.

2.4.3 Signal Reconstruction

We now show that the original vector f can be reconstructed by reversing the previous process of decomposition. By Equation 2.16, Theorem 1, and Theorem 2, the function $\phi_{2^{j+1}}(x - 2^{-j-1}n)$ can be decomposed into

$$\begin{aligned}
&\phi_{2^{j+1}}(x - 2^{-j-1}n) \\
= &\sum_{k=-\infty}^{+\infty} \langle \phi_{2^j}(u - 2^{-j}k),\ \sqrt{2^{-j}}\phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle\ \sqrt{2^{-j}}\phi_{2^j}(x - 2^{-j}k)\ + \\
&\sum_{k=-\infty}^{+\infty} \langle \psi_{2^j}(u - 2^{-j}k),\ \sqrt{2^{-j}}\phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle\ \sqrt{2^{-j}}\psi_{2^j}(x - 2^{-j}k) \\
= &\ 2^{-j} \sum_{k=-\infty}^{+\infty} \langle \phi_{2^j}(u - 2^{-j}k),\ \phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle\ \phi_{2^j}(x - 2^{-j}k)\ + \\
&\ 2^{-j} \sum_{k=-\infty}^{+\infty} \langle \psi_{2^j}(u - 2^{-j}k),\ \phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle\ \psi_{2^j}(x - 2^{-j}k).
\end{aligned} \tag{2.22}$$

If we take the inner product of $f(x)$ with both sides of Equation 2.22, we have

$$\begin{aligned}
&\langle f(u),\ \phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle \\
= &\ 2^{-j} \sum_{k=-\infty}^{+\infty} \langle \phi_{2^j}(u - 2^{-j}k),\ \phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle\ \langle f(u),\ \phi_{2^j}(x - 2^{-j}k) \rangle\ + \\
&\ 2^{-j} \sum_{k=-\infty}^{+\infty} \langle \psi_{2^j}(u - 2^{-j}k),\ \phi_{2^{j+1}}(u - 2^{-j-1}n) \rangle\ \langle f(u),\ \psi_{2^j}(x - 2^{-j}k) \rangle \\
= &\ 2 \sum_{k=-\infty}^{+\infty} \langle \phi_{2^{-1}}(u),\ \phi(u - (k - 2n)) \rangle\ \langle f(u),\ \phi_{2^j}(x - 2^{-j}k) \rangle\ + \\
&\ 2 \sum_{k=-\infty}^{+\infty} \langle \psi_{2^{-1}}(u),\ \phi(u - (k - 2n)) \rangle\ \langle f(u),\ \psi_{2^j}(x - 2^{-j}k) \rangle.
\end{aligned} \tag{2.23}$$

Substituting Equation 2.12 and Equation 2.20 into Equation 2.23 yields

$$\begin{aligned}&\langle f(u),\ \phi_{2^{j+1}}(x-2^{-j-1}n)\rangle\\ =\ &2\sum_{k=-\infty}^{+\infty} h(n-2k)\,\langle f(u),\ \phi_{2^j}(x-2^{-j}k)\rangle + 2\sum_{k=-\infty}^{+\infty} g(n-2k)\,\langle f(u),\ \phi_{2^j}(x-2^{-j}k)\rangle.\end{aligned}$$

Mallat [Mal89] uses another tree hierarchy to explain the data reconstruction process. Approximation $A^d_{2^{j+1}}f$ is reconstructed by putting one zero between each sample of $A^d_{2^j}f$ and $D_{2^j}f$, and then convolving the resulting data with filter H and G, as shown in Figure 2-9.

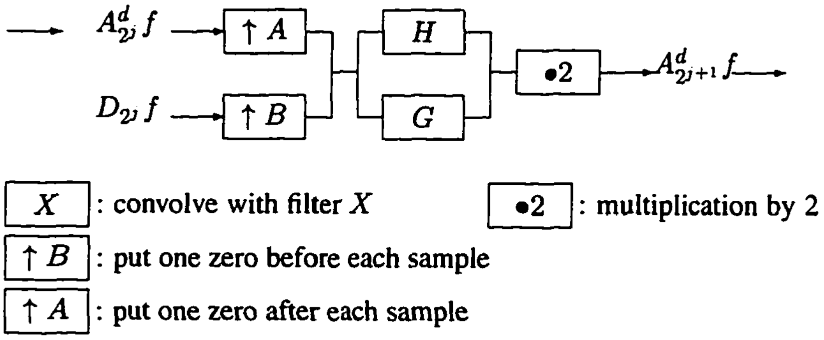

$A^d_{2^j}f$
$D_{2^j}f$
↑A
↑B
H
G
•2
$A^d_{2^{j+1}}f$
X : convolve with filter X
•2 : multiplication by 2
↑B : put one zero before each sample
↑A : put one zero after each sample

Figure 2-9: Reconstruction of $A^d_{2^{j+1}}f$ from a coarser resolution $A^d_{2^j}f$ and detail $D_{2^i}f$.

For example, we have $A^2_{2^0}f = [3]$ from Equation 2.15, and $D_{2^0}f = [2]$ from Equation 2.21. After inserting zeros, [3] becomes $\begin{bmatrix}3\\0\end{bmatrix}$ and [2] becomes $\begin{bmatrix}0\\2\end{bmatrix}$. They are then put together as $\begin{bmatrix}3\\2\end{bmatrix}$ before the reconstruction filter is applied, i.e.,

$$A^d_{2^1}f = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & -\frac{1}{2}\end{bmatrix}\begin{bmatrix}3\\2\end{bmatrix} = \begin{bmatrix}\frac{5}{2}\\ \frac{1}{2}\end{bmatrix}.$$

After the $\boxed{\bullet 2}$ operation, we have

$$A^d_{2^2}f = \begin{bmatrix}5\\1\end{bmatrix},$$

as shown in Equation 2.14.

2.5 Multidimensional Orthogonal Wavelet Representation

We extend our discussion of wavelet representation from one dimensional to multidimensional functions. Mallat gives the first implementation of an orthogonal wavelet representation of a two dimensional function $f(x,y) \in L^2(\mathbb{R}^2)$ in [Mal89]. It is proved that each two dimensional vector space $V^2_{2^j}$ can be decomposed as a tensor product of two identical subspaces of $L^2(\mathbb{R})$,

$$V^2_{2^j} = V^1_{2^j} \otimes V^1_{2^j}, \tag{2.24}$$

and the scaling function $\Phi(x,y)$ can be described as

$$\Phi(x,y) = \phi(x)\ \phi(y),$$

where $\phi(x)$ is the one dimensional scaling function of the multiresolution approximation $V^1_{2^j}$. Let $\Phi_{2^j}(x,y) = 2^{2^j}\Phi(2^j x, 2^j y)$. The orthogonal basis of V_{2^j} can be written as

$$2^{-j}\ \Phi_{2^j}(x - 2^{-j}n,\ y - 2^{-j}m)$$

for $(n,m) \in \mathbb{Z}$. The approximation of function $f(x,y)$ at resolution 2^j, $A^2_{2^j}f(x,y)$, can then be described by the inner products

$$A^2_{2^j}f(x,y) = \langle f(x,y),\ \phi_{2^j}(x - 2^{-j}n)\ \phi_{2^j}(y - 2^{-j}n)\rangle.$$

Mallat [Mal89] states that by extending Theorem 1, we can build an orthonormal basis of O_{2^j} by scaling and translating three wavelet functions $\Psi^1(x,y)$, $\Psi^2(x,y)$, and $\Psi^3(x,y)$.

Theorem 3 For $j \in \mathbb{Z}$, let V_{2^j} be a separable multiresolution approximation of $L^2(\mathbb{R}^2)$. Let $\Phi(x,y) = \phi(x)\ \phi(y)$ be the associated two dimensional scaling function. Let $\psi(x)$ be the one dimensional wavelet associated with the scaling function $\phi(x)$. Then, the three wavelets

$$\begin{aligned} \Psi^1(x,y) &= \phi(x)\ \psi(y), \\ \Psi^2(x,y) &= \psi(x)\ \phi(y), \\ \Psi^3(x,y) &= \psi(x)\ \psi(y). \end{aligned} \tag{2.25}$$

are such that

$$(2^{-j}\Psi^1_{2^j}(x-2^{-j}n,\ y-2^{-j}m), 2^{-j}\Psi^2_{2^j}(x-2^{-j}n,\ y-2^{-j}m), 2^{-j}\Psi^3_{2^j}(x-2^{-j}n,\ y-2^{-j}m))_{(n,m)\in Z^2}$$

is an orthonormal basis of O_{2^j} and

$$(2^{-j}\Psi^1_{2^j}(x-2^{-j}n,\ y-2^{-j}m), 2^{-j}\Psi^2_{2^j}(x-2^{-j}n,\ y-2^{-j}m), 2^{-j}\Psi^3_{2^j}(x-2^{-j}n,\ y-2^{-j}m))_{(n,m)\in Z^3}$$

is an orthonormal basis of $L^2(\mathbb{R}^2)$.

The proof of this theorem can be found in [Mal89]. The difference of information between $A^2_{2^{j+1}}f$ and $A^2_{2^j}f$ is given by three details

$$\begin{aligned} D^1_{2^j}f &= (\langle f(x,y),\ \Psi^1_{2^j}(x-2^{-j}n,\ y-2^{-j}m)\rangle), \\ D^2_{2^j}f &= (\langle f(x,y),\ \Psi^2_{2^j}(x-2^{-j}n,\ y-2^{-j}m)\rangle), \\ D^3_{2^j}f &= (\langle f(x,y),\ \Psi^3_{2^j}(x-2^{-j}n,\ y-2^{-j}m)\rangle), \end{aligned} \tag{2.26}$$

for $(n,m) \in \mathbb{Z}^2$.

An example of a two dimensional wavelet transform is depicted in Figure 2-10. The function

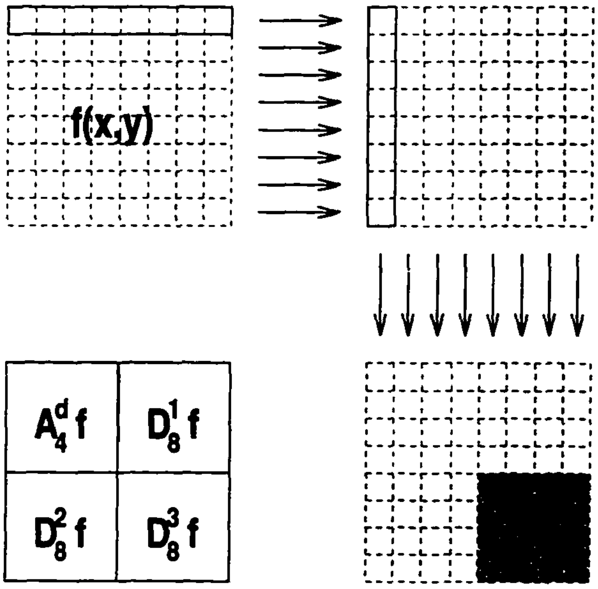


Figure 2-10: Wavelet transform on two dimensional function$f(x,y)$.

$f(x,y)$ in the upper left corner is a discrete 8×8 array. Since the three wavelet ($\Psi^1(x,y)$, $\Psi^2(x,y)$, and $\Psi^3(x,y)$) are given by separable products of the functions ϕ and ψ, the approximations and details can be computed by separable filters. In this example, one dimensional wavelets are applied to each row of function $f(x,y)$, followed by an application of a one dimensional wavelet to the columns of the result. The resultant approximated functions and details are shown at the bottom half of Figure 2-10.

Muraki [Mur92, Mur93] further expands this two dimensional wavelet transform into three dimensions. Much like Equation 2.24, the vector space V_{2^j} is now decomposed as a tensor product of three identical subspaces of $L^2(\mathbb{R})$,

$$V_{2^j} = V_{2^j}^1 \otimes V_{2^j}^1 \otimes V_{2^j}^1,$$

and one can construct an orthonormal basis $L^2(\mathbb{R}^3)$ by the following seven wavelets:

$$\begin{aligned}
\Psi^1(x,y,z) &= \phi(x)\,\phi(y)\,\psi(z)\\
\Psi^2(x,y,z) &= \phi(x)\,\psi(y)\,\phi(z)\\
\Psi^3(x,y,z) &= \phi(x)\,\psi(y)\,\psi(z)\\
\Psi^4(x,y,z) &= \psi(x)\,\phi(y)\,\phi(z)\\
\Psi^5(x,y,z) &= \psi(x)\,\phi(y)\,\psi(z)\\
\Psi^6(x,y,z) &= \psi(x)\,\psi(y)\,\phi(z)\\
\Psi^7(x,y,z) &= \psi(x)\,\psi(y)\,\psi(z).
\end{aligned} \qquad (2.27)$$

In fact, this process can be expanded to any dimension [Dau92]. When the dimensionality grows, the number of equations gets larger, and each of the equations gets longer. It is computationally expensive to apply wavelets to high dimensional data.

Chapter 3

Multidimensional Multivariate Visualization

In this chapter, we describe the concepts and applications of the field of multidimensional multivariate (mdmv) visualization. We begin in Section 3.1 with a historical overview. We then introduce the terminology of mdmv in Section 3.2. Finally, Section 3.3 highlights some of the most commonly used mdmv visualization techniques.

3.1 Background

The last three decades of mdmv visualization development can be roughly characterized into four phases. The classic exploratory data analysis (EDA) book by Tukey [Tuk77], the 1987 NSF workshop on Visualization in Scientific Computing [MDB87], and the IEEE Visualization '91 conference [NR91] are the watersheds defining these phases. The first phase was primarily concerned with the graphical presentation of either one or two variate data. The second phase was dominated by Tukey's exploratory data analysis. Scientists started looking at graphical data with a different perspective. Although most of the graphics was still two dimensional, scientists were able to encode data with multiple parameters, i.e., multivariate, into meaningful two dimensional plots. The momentum of this work carried on through the next phase when NSF recognized the importance of mdmv data visualization. The involvement of computer scientists accelerated the growth of the research by computerizing many of the old ideas and developing many new ones. The mission was formally defined and many promising concepts were developed during the following few years. The final (current) phase is concerned with the elaboration and assessment of mdmv visualization techniques. It remains to be seen whether the existing mdmv visualization concepts can lead to better visualization of a problem and better understanding of the underlying science.

3.2 Terminology

Unfortunately, the mdmv literature suffers from ill-defined and inconsistent terminology. The term *dimensionality* is especially overloaded. Some people consider dimension as the number of independent variables in an algebraic equation [MGTS90]. Others take dimension as measurements of any sort (breadth, length, height, and thickness). Even the prefix *multi* is frequently interchanged with another prefix *hyper*. In statistics literatures, the prefix *multi* means two or more, indicating a natural breakpoint between one and two dimensions in probabilistic methods. For the breakpoint between three and four (or beyond), the prefix *hyper* is used [Cle93]. We use the prefix *multi* to refer to dimensionality of two or more.

Beddow [Bed92] points out the difference between multidimensional *objects* and multidimensional *data*. Multidimensional *objects* are *spatial* objects, and the goal is to understand their geometry. The most common forms are two dimensional images and three dimensional volumes. They can best be described as compact subsets of n dimensional Euclidean spaces $\mathbb{R}^n$. Multidimensional *data*, on the other hand, represents sparse sets of n dimensional points which represent multiple parameters. Mathematically these parameters can be classified into two categories: *dependent* and *independent* [KK93]. Some statisticians prefer the terms *factor* and *response* [Cle93]. A variable is said to be dependent if it is a function of another variable, the independent variable. The relationship of an independent variable x and a dependent variable y can best be described by the mathematical equation $y = f(x)$. We adopt the convention that the term multidimensional refers to the dimensionality of the independent variables, while the term multivariate refers to the dimensionality of the dependent variables [BCH+95]. This is the most popular way to describe the dimensionality of mdmv data sets in scientific visualization literature. For example, a three dimensional volume space in which both temperature and pressure are observed and recorded in various locations produces 3d2v data. Beddow [Bed92] argues that analytic methods used to explore n-dimensional Euclidean spaces $\mathbb{R}^n$ are not appropriate for general multivariate analysis. In mdmv visualization research, the emphasis shifts away from the strong mathematical definition of dependent and independent variates towards the broader definition of multiple variables or factors. This happens not only in mdmv scientific visualization research but also in statistical studies. The tools are different, but the goal is the same: to find the hidden relationships between the variables (also known as *fitting* in statistics).

In general, raw *scientific data* can be categorized into a hierarchy of data types. The most general and the lowest of the hierarchy is *nominal* data, whose values have no inherent ordering. For example, the names of the fifty states are nominal data. The next higher type of the hierarchy is *ordinal* data, whose values are ordered, but for which no meaningful distance metric exists. The seven rainbow colors (i.e., red, orange, ...) belong to this category. The highest of the hierarchy is *metric* data, which has a meaningful distance metric between any two values. Times, distances, and temperatures are examples. If we bin metric data into ranges, it becomes ordinal data. If we further remove the ordering constraints, the data is nominal.

The above 3d2v temperature/pressure example more or less implies that each 3 dimensional coordinate contains *simple* (i.e., neither a set nor an interval) and *atomic* (i.e., not composite) values of pressure and temperature. This is different from the case when we measure, for example, the chemical contents of a volume. Each coordinate now has a set (instead of a simple value) of composite data (e.g., chemical elements). The varying numbers of values of a variate plotted in any single dimensional point is known as the *density* of that coordinate.

3.3 Visualization Techniques

This section summarizes the general concepts of some of the most commonly used mdmv techniques including scatterplot matrix, HyperSlice, parallel coordinates, and brushing. These techniques are used in the examples and illustrations throughout the thesis.

3.3.1 Scatterplot Matrix

The traditional two dimensional point and line plots are among the most popular visualization techniques for data with a small number of variates. This technique can be enhanced by putting an array of plots into one display, so as to add another variate to the visual presentation.

One of the more popular statistics mdmv visualization techniques is the scatterplot matrix which presents multiple adjacent scatterplots. Each display panel in a scatterplot matrix is identified by its row and column numbers in the matrix. For example, the identity of the lower left panel of the matrix in Figure 3-1 is (1, 1), and the upper left panel is (3, 1). The empty diagonal panels denote the variable names. Panel (1, 2) is a scatterplot of parameter X against Y while panel (2, 1) is the reverse, i.e., Y versus X. In a scatterplot matrix, every variate is treated identically. The basic idea is to visually *link* features in one panel with features in others. The redundancy is designed to improve the effect of visual linking. The technique is further enhanced with the help of reference grids. The pattern can be detected in both horizontal and vertical directions. The concept of linking is also discussed in [BMMS91].

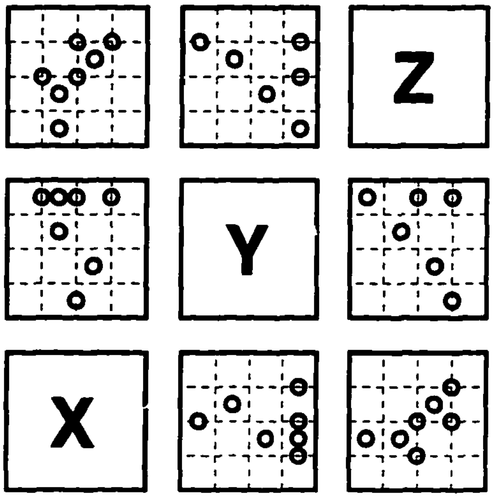


Figure 3-1: A scatterplot matrix display of data with three variates X, Y, and Z.

Despite the popularity of scatterplot matrices in mdmv visualization applications, nobody knows the identity of the original inventor [Cle93]. The technique was first presented in [CCKT83]. A variety of powerful tools using this kind of multi-panel display are presented in [Cle93].

3.3.2 HyperSlice

Like the scatterplot matrix, HyperSlice [vWvL93] has a matrix of panels, although each individual scatterplot is replaced with color or grey shaded graphics representing a scalar function of the variates. Furthermore, panels along the diagonal show the scalar function in terms of a single variate.

HyperSlice defines a focal point of interest $c = (c_1, c_2, \cdots, c_n)$ and a set of scalar widths w_k, where $k = 1, \cdots, n$. Only data within the range $R_k = [c_k - w_k/2, c_k + w_k/2]$ for each k are displayed in the panel matrix. Other data only appears if the user steers the focal point near it. Like the coordinate system used in the scatterplot matrix, a HyperSlice panel is identified by a horizontal and a vertical coordinate. For an off-diagonal panel (i, j) such that $i \neq j$, the color shows the value of the scalar function that results from fixing the values of all variates except i and j to the values of the focal point, while varying i and j over their ranges in R_i and R_j respectively. The (i, i) diagonal panel shows a graph of the scalar function versus one variate which changes over its range in R_i, with all other variate values fixed at c_k for $k \neq i$.

The most important improvement of HyperSlice over the traditional scatterplot matrix is the idea of interactively *navigating* in the data around a user defined focal point. The user changes the focal point by interacting with any of the panels, as shown in Figure 3-2. The user moves the mouse

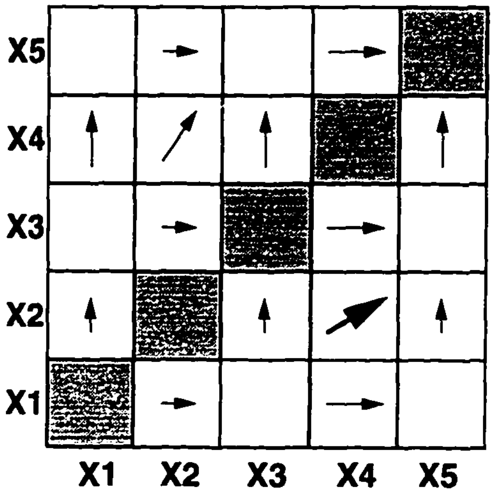


Figure 3-2: Navigate a five variate HyperSlice by dragging panel (2, 4).

into any panel and defines a direction by button down, move, and up. For example, the boldface arrow in panel (2, 4) (using the coordinate system described previously in Section 3.3.1) represents such an interaction. The direction of each arrow shows the motion of the focal point when the focal point is dragged in panel (2, 4). Notice that the length (magnitude) of the vertical arrows across the X_2 row, is the same as the vertical component of the arrow in (2, 4). Similarly, each horizontal arrow in column X_4 has the same length as the horizontal component of the arrow in panel (2, 4). Panels solely related to X_1, X_3, and X_5 move perpendicular to the image plane. Since the matrix is somewhat similar to an orthogonal matrix (along the grey diagonal panel), the motion in the upper left half is the mirror projection of the lower right.

Interactive data navigation is a welcome addition to direct manipulation graphics. Changing the focal point in one panel affects two variates which in turn results in simultaneous visual changes in displays of these variates with others. HyperSlice is an example of a successful elaboration which builds on another successful tool.

The basic ideas of HyperSlice can be extended to discrete datasets using data projection in a technique called *prosection* [STDS95, TSDS96]. Chapter 9 presents a dual multiresolution HyperSlice which allows a user to control the *physical data* resolution using orthogonal wavelets, as well as the *logical display* resolution using *norm* projections. The system provides a progressive refinement environment to support very large data visualization.

3.3.3 Parallel Coordinates

All techniques we have discussed so far are designed to do data analysis on multidimensional *data*. They are not really aimed at studying the *geometry* of multidimensional *objects*. Parallel coordinates [IRC87, ID87, ID90], on the other hand, can do both.

In a parallel coordinate system, the axes of a multidimensional space are defined as parallel vertical lines separated by a distance *d*. A *point* in Cartesian coordinates corresponds to a *polyline* in parallel coordinates. To avoid confusion, we use lower case letters for lines, and upper case letters for points in Cartesian spaces. In parallel coordinates, we use similar conventions except we put a bar superscript on all letters.

To see how a multidimensional object is represented in parallel coordinates, consider a simple two dimensional straight line

$$l : x_2 = mx_1 + b$$

where $m < \infty$. In Figure 3-3, the continuous straight line in Cartesian coordinates is sampled, and the values are plotted on corresponding axes in parallel coordinates. A collection of points, A, sampled from a straight line in Cartesian coordinates, corresponds to a set of lines $\overline{A}$ in parallel coordinates that intersect at the point $\bar{l}$,

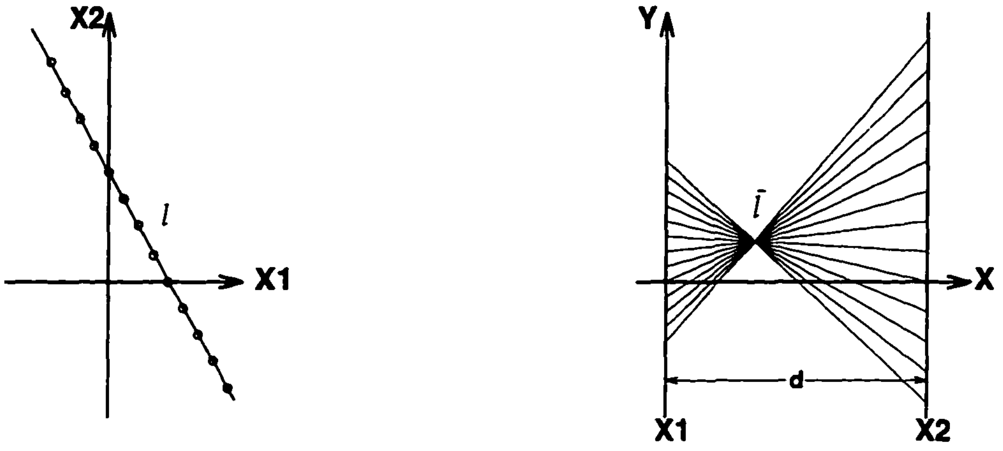


Figure 3-3: Left: Two dimensional Cartesian coordinates. Right: Parallel coordinates.

$$\bar{l} : \left(\frac{d}{1-m}, \frac{b}{1-m}\right)$$

for $m \neq 1$. For example, given a straight line $x_1/2 - x_2 + 1 = 0$, and $d = 5$, we get $m = 1/2$ and $b = 1$. When the sampled points are plotted in parallel coordinates, all lines intersect at $(\frac{5}{1-1/2}, \frac{1}{1-1/2}) = (10, 2)$, as shown in Figure 3-4. Notice that the location of the intersection point shows an important property of the data. In Figure 3-3 where two variates x_1 and x_2 are inversely proportional to each other, i.e., $x_1 \propto 1/x_2$, the intersection point is between the two parallel axes. In Figure 3-4, where variate x_1 is directly proportional to x_2, i.e. $x_1 \propto x_2$, the intersection point is located outside the two parallel axes.

Parallel coordinates allow humans to visualize three dimensional time series data as a single static display. A simple application is aircraft collision checking [ID90]. In Figure 3-5, the loca-

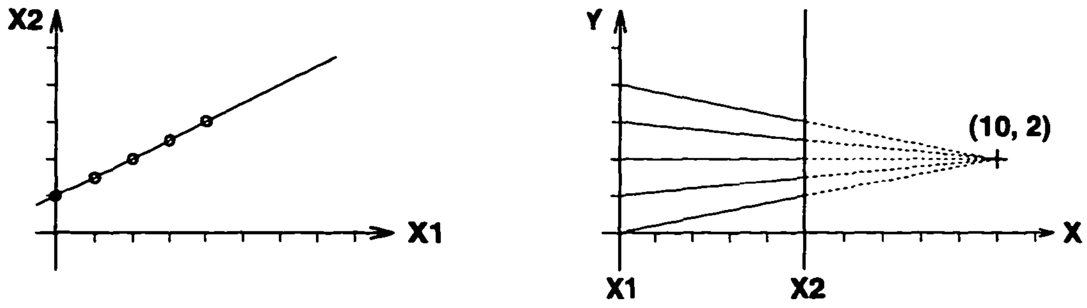


Figure 3-4: Left: The straight line, $x_1/2 - x_2 + 1 = 0$, is plotted in Cartesian coordinates. Right: the same line is plotted in parallel coordinates. The intersection point, $(10, 2)$, is located outside the two axes.

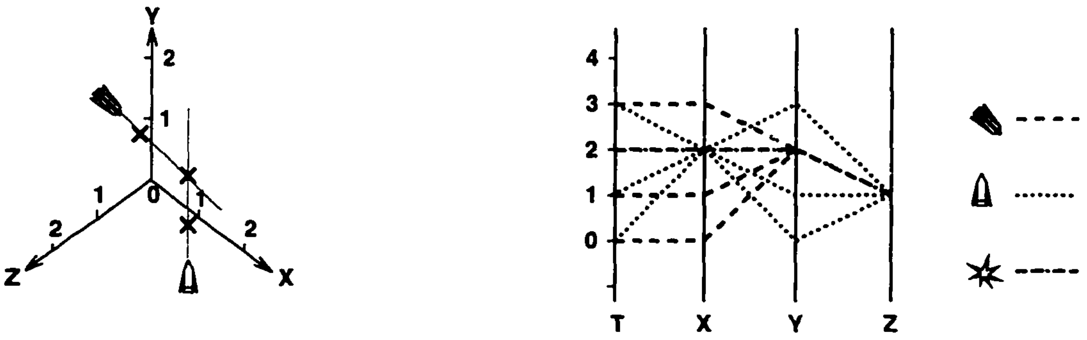


Figure 3-5: Left: Locations of two aircraft in three dimensional Cartesian coordinates at time 0. Right: The trajectory is plotted with time axis, T, in parallel coordinates.

tions of two aircraft are displayed in one frame using Cartesian coordinates and parallel coordinates showing all frames simultaneously. It is almost impossible to confirm a collision solely by judging the locations of two aircraft in any one single view of a three dimensional Cartesian plot. For example, suppose we have an isometric projection [FvDFH90] with three aircraft located at coordinates $(0,0,0)$, $(1,1,1)$, and $(2,2,2)$ at time t. They are all displayed at the same spot, yet no collision occurs. On the other hand, a parallel coordinate plot including time t, and the coordinates x, y, z is shown on the right hand side of Figure 3-5. Two aircraft collide if and only if they are in the same location at the same time. That means there will be a collision in location $(2,2,1)$ at $T = 2$. A four dimensional intersection can be detected by searching for any overlapping dashed lines. In our example, an overlap is detected at $(2,2,2,1)$ of the parallel coordinate plot.

To help avoid collision, *parallelograms* can be defined along with the trajectory of the aircraft. The number, size, and shape of the parallelograms are computed according to each plane's relative velocity and locations with respect to others, as shown in Figure 3-6. If the two aircraft are flying at

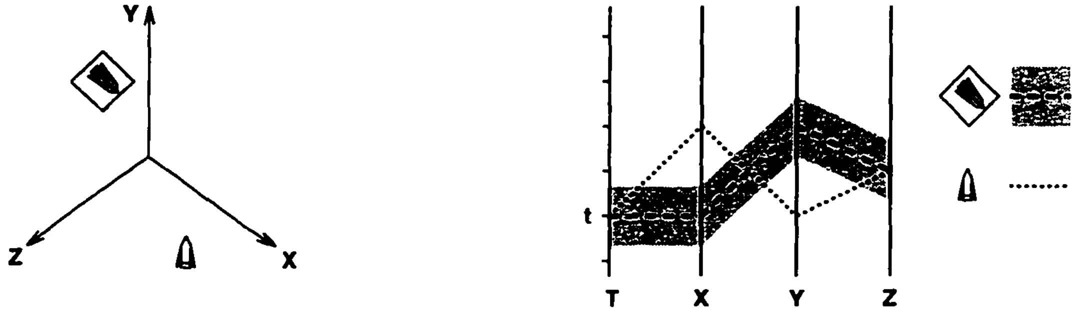


Figure 3-6: Left: Conflict parallelogram. Right: At time *t*, the graph shows the location of one aircraft is not entirely inside the grey area of the others, so no collision occurs.

the same velocity, the lower right hand aircraft must avoid any contact with the parallelogram of the upper left hand aircraft. In Figure 3-6, the safety zone is indicated in grey in the parallel coordinate plot. There is a conflict at any time if the plotting of location/time of one aircraft is entirely inside the grey area of another. Like our previous example, it is almost impossible to spot the conflict in a single three dimensional Cartesian plot.

Parallel coordinates can also be used to study correlations among variates in mdmv data analysis. By spotting the locations of the intersection points (see Figures 3-3 and 3-4), we can have a rough idea about the relationships between each pair of variates. This is one of the more promising applications of parallel coordinates in mdmv visualization. The problem with this technique is the limited space available for each parallel axis. The display can rapidly darken with even a modest amount of data.

3.3.4 Brushing

Brushing was first presented in [BC87] and is included as one of the many direct manipulation techniques in [Cle93]. Buja et al. [BMMS91] used the terms *focusing* and *linking* to describe various brushing techniques. Focusing involves data selection, dimension reduction, and data layout manipulation such as zooming. A sequence of focusing views are linked together so that the information of individual views can be integrated into a coherent image.

Cleveland [Cle93] describes two kinds of brushing for a scatterplot matrix: *labeling* and *enhanced linking*. *Labeling* involves an interactive brush (e.g., a mouse pointer) that causes information label(s) to pop-up for particular display item(s). In *enhanced linking*, the brush is an adjustable rectangle. It is used to cover a set of points in one of the panels. Figure 3-7 shows a rectangle brush in panel $(3, 2)$. Data inside the rectangle is displayed with a "+" instead of a "o." The same changes are applied to the corresponding data points in the other panels. By looking at different panels and comparing the vertical and horizontal extent of the brush, this enhanced linking technique provides a powerful direct manipulation tool for visual analysis. The effect of brushing is more intense in a dynamic interactive display. More applications can be found in [Cle93].

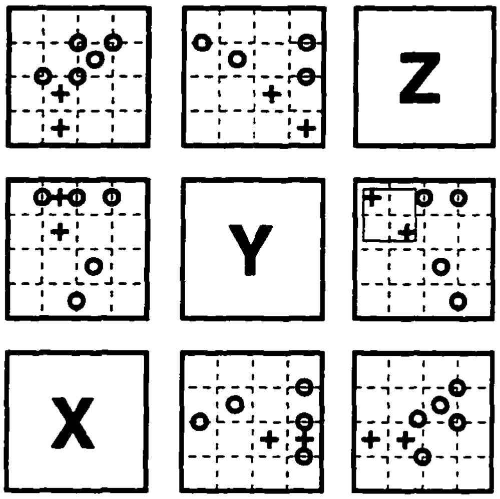


Figure 3-7: Enhanced brushing with the square brush located on panel (3,2).

3.3.5 XmdvTool

Cleveland begins and ends his book [Cle93] with the same quote, "Tools matter". The idea is that you have to pick the right technique to visualize mdmv data. The implication is that no technique alone is powerful and flexible enough to handle all mdmv scientific data. Ward [War94, MW95] integrates four popular static mdmv visualization techniques into a single analysis system, XmdvTool. The four techniques included are: scatterplot (panel matrix), dimension stacking (hierarchical display), star glyph (iconography), and parallel coordinates. The original brushing [BC87], which relies heavily on mouse clicking interaction, is modified into a more complicated tool with user-controlled parameters. A user can control the shape, size, boundary, position, motion, and orientation of the *n*-dimensional brush. Some of these parameters are customized while the others can be controlled with slider widgets. The original brushing was implemented on the scatterplot matrix only. Brushing in XmdvTool is implemented on all four options. One of the major differences compared to the original version is that the brush itself is also displayed with the brushed data as shown in Figure 3-8.

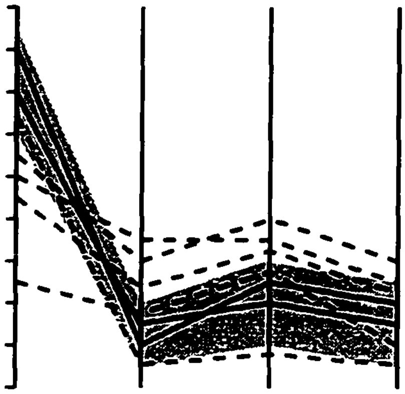


Figure 3-8: A high dimensional data brushing is applied to a four variate dataset. The grey area indicates the data brush.

Chapter 4

Issues of Multiresolution Data Representation

In Chapter 1 we briefly describe the concepts of a multiresolution data hierarchy. This chapter presents the design and implementation issues of an effective data hierarchy. In particular, we discuss important issues of building data hierarchies according to resources and requirements.

4.1 Multiresolution Visualization

We begin with an adaptive multiresolution data analysis example of a real-life scientific dataset. The dataset contains concentrations of seven chemical elements provided by the Greenland Ice Sheet Project Two (GISP2) [MMM$^+$93] from the Institute for the Study of Earth, Oceans and Space of the University of New Hampshire. The GISP2 project recovered a two mile long ice core from the central plateau of the Greenland ice sheet. The resultant record covers a period of 20,000 years, and provides a multivariate time-series record documenting climatic and atmospheric change and forcing.

The multivariate dataset used in this example covers the past 13,000 years. The data has seven variates. We extract one of them (8,192 records of calcium concentration) to illustrate a complete multiresolution hierarchy of one dimensional one variate data. The data is first decomposed by a Haar wavelet to form a fine to coarse hierarchy. A partial reconstruction is then presented to simulate a multiresolution analysis process. Some of the characteristics of this process are highlighted.

4.1.1 Wavelet Decomposition

In Figure 4-1, the graph at the top is the original data, and the rest are approximations at various resolutions. Polylines are used to plot the approximation data. Lower calcium concentrations indicate conditions similar to today's warm climate and higher concentrations indicate colder and glacial conditions. For example, the first peaks moving from left to right of these graphs are identified as the Younger Dryas Events, a 1,300 year period of cold and glacial conditions that ended about 11,000 years ago. The peaks on the right are the Last Ice Age which ended about 15,000 years ago.

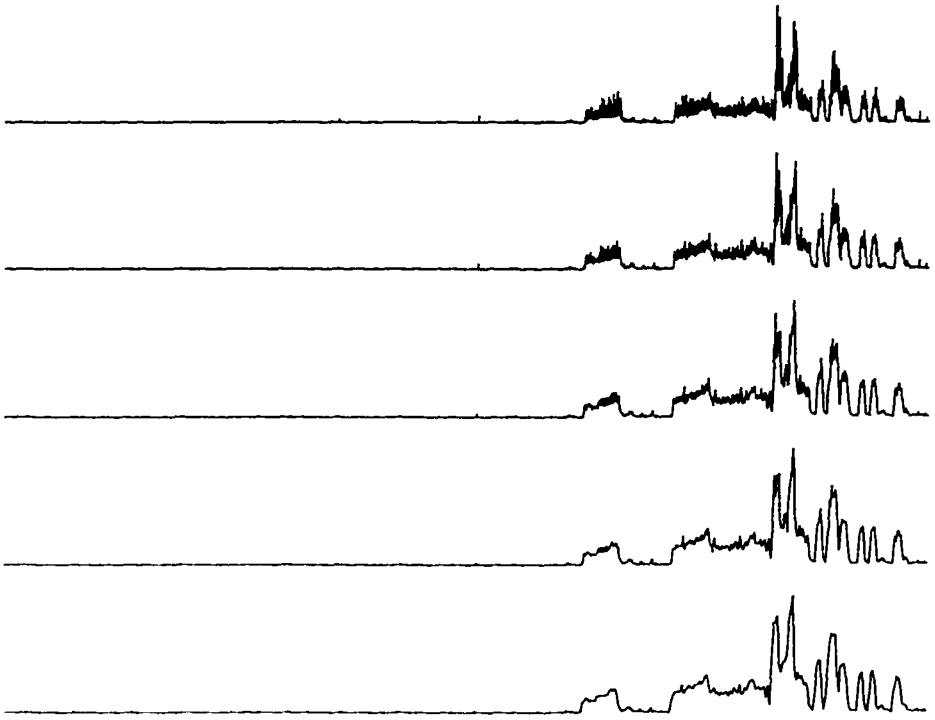


Figure 4-1: Wavelet decomposition of GISP2 calcium data. The top graph is the original data; the rest are approximations with 4,096, 2,048, 1,024 and 512 data values.

These five graphs illustrate a fine to coarse decompositions of 8,192 data items. The top graph was generated from the entire dataset. The rest of the graphs have 4,096, 2,048, 1,024, and 512 data values. While megabyte sized data is not considered very large data, this example shows the zooming capability of an orthogonal wavelet. Bear in mind that some of the fine details are invisible in the higher resolution graphs because the data has higher resolution than the laser printer output.

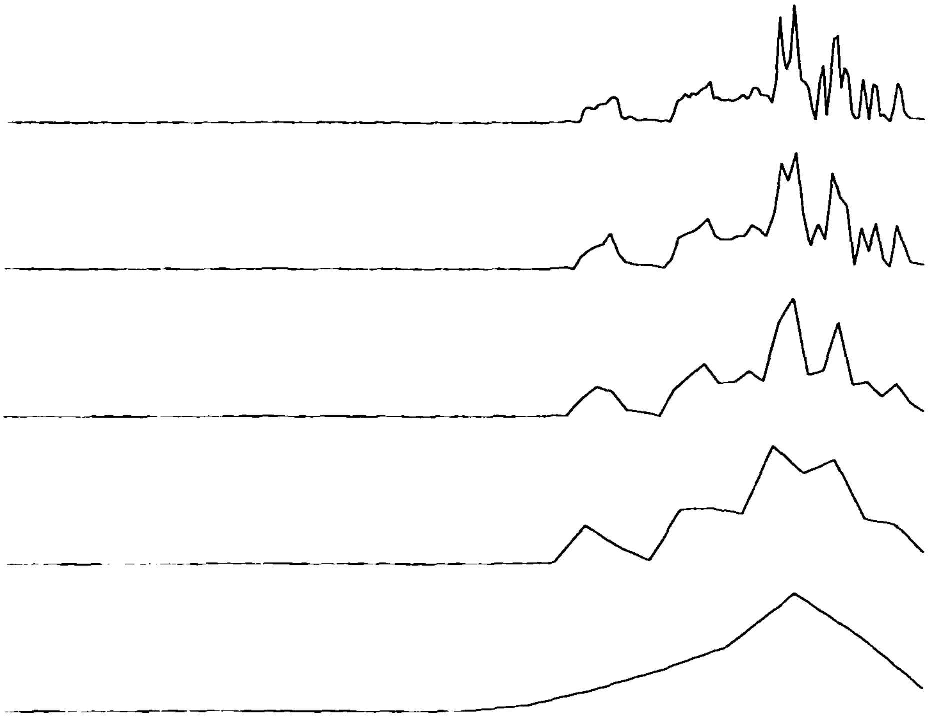


Figure 4-2: Wavelet decomposition of GISP2 calcium data with 256, 128, 64, 32, and 16 data values.

Figure 4-2 shows the next five resolutions with 256, 128, 64, 32, and 16 data values. After five resolutions of decomposition with 256 approximations left, the graph at the top of Figure 4-2 still retains an excellent approximation of the original data. Even in the bottom graph of Figure 4-2, we can still see the peak of the Last Ice Age with only 16 data values left.

4.1.2 Wavelet Reconstructions

Figure 4-3 depicts a hierarchical partial reconstruction of wavelet coefficients from the previous decomposition. In this example, we start with a very coarse resolution with only 32 coefficients. After the interesting area (the dotted rectangle on the right side) is identified, we study the marked data at a resolution that is three levels up in the hierarchy using 256 coefficients ($2^3 \times 32 = 2^3 \times 2^5 = 2^8 = 256$). The zooming to finer resolutions (i.e., more coefficients) of the marked data continues until it reaches a resolution defined by 4,096 coefficients. At this point we are positioned to study the true original data of the first major peak of the Young Dryas Events (the dotted rectangle on the

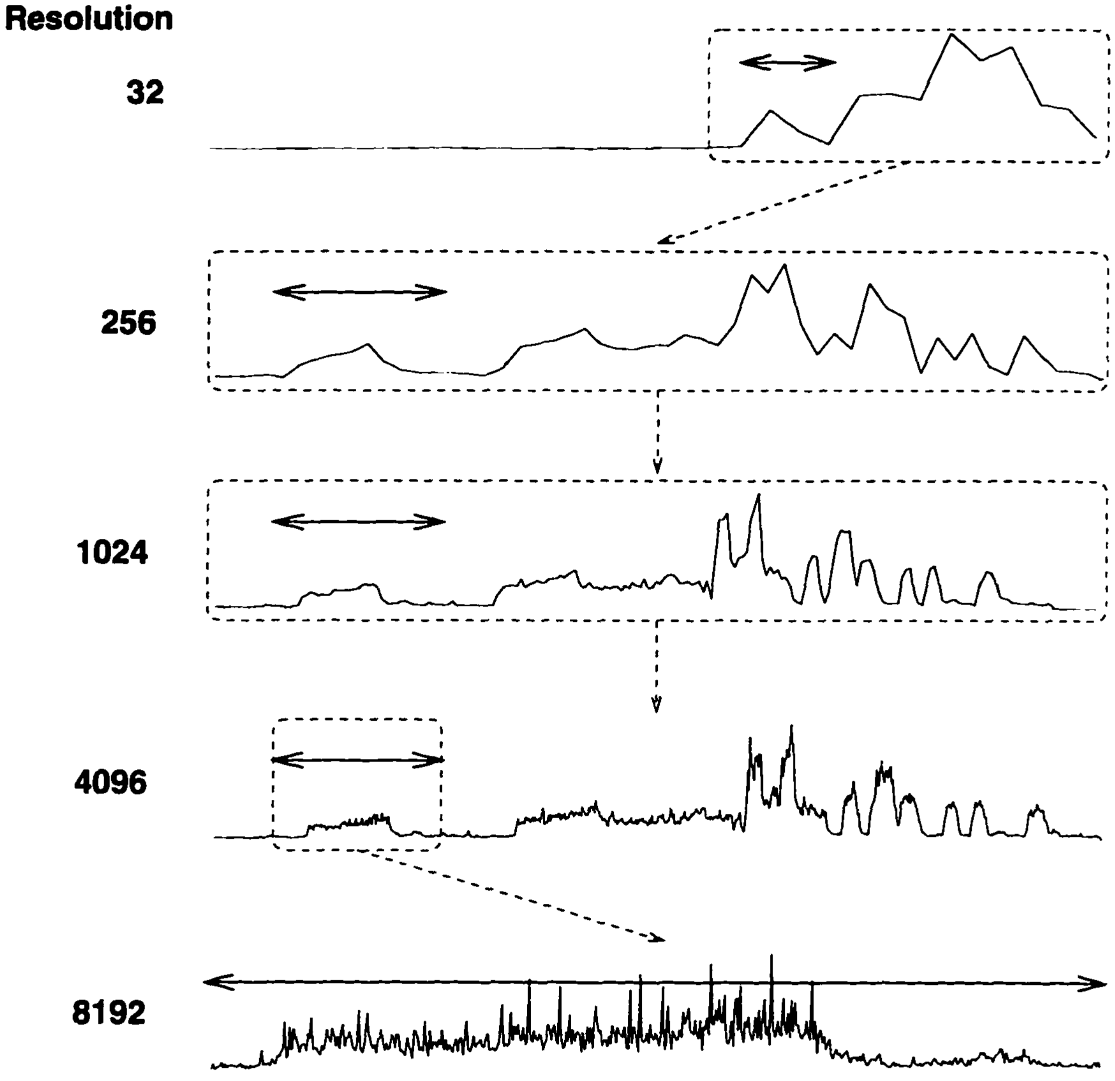


Figure 4-3: A partial reconstruction hierarchy. Dotted rectangles mark the zooming windows. Double-head arrows indicate the same time frame, which is the Younger Dryas, in every resolution.

left side). This subset of the original data is presented in the bottom graph.

One of the important points illustrated with this example is the power of multiresolution hierarchy generated by orthogonal wavelets. In this case, we identify the interesting spots from a very fast but somewhat less accurate display of 32 data items. Even though the original data is a relatively small dataset, the important idea is the number of resolutions in the hierarchy. Theoretically, for data with 8,192 items, a total of 10 resolutions can be generated with a compactly supported orthogonal wavelet. Terabyte sized data, however, can be scaled down to gigabyte sized data, also in the same number of resolutions with the same wavelet. Larger sized data can be considered as a longer version of this example.

Another point we wish to illustrate here is the ability to access any subset of the data in any resolution. Instead of saturating available resources on sending/receiving huge amounts of data, only a subset of data (small enough to be displayed on screen) is transported at a time.

4.2 Adaptive Resolution

As shown in Figure 1-3 of Chapter 1, the wavelet decomposition generates more data to build the multiresolution hierarchy. For one dimensional data with N data items, the first decomposition generates $N/2$ approximations and $N/2$ details. The remainder of the hierarchy (including approximations and details of every stage) occupies

$$\frac{N}{2}+\frac{N}{4}+\frac{N}{8}+\frac{N}{16}+\cdots \leq N$$

units of memory. Thus an upper bound on memory is $N/2+N/2+N = 2N$ units. In this example, we also assume the original input is destructable, otherwise another N memory units are required to keep it. This scenario supports the fastest data access, but also requires the largest amount of memory.

We can still achieve lossless reconstruction with only N memory units at the expense of additional computation. In this design, we only store the details of each stage except the lowest resolution, for which we store both the approximation and the detail. The maximum memory required is N units. Since none of the approximations are kept, extra calculations are required to reconstruct them sequentially (from coarse to fine.) An effective design of a hierarchical representation is a combination of data truncation, intermediate stage elimination, and quiescent state elimination.

4.2.1 Data Truncation

One of the most important properties of a wavelet transform is that it produces highly localized coefficients in the space domain. Many small coefficients can be discarded with only minimal effect, yielding *lossy* representations at significantly reduced storage costs. We demonstrate test results of an extreme case in [WB94]. A reconstruction of three-dimensional isosurface data from only three percent of the wavelet coefficients has a correlation $\simeq 0.74$ compared to the original data. When we are dealing with terabyte sized data, data truncation may be the most important part of the hierarchical representation design.

A major issue for data truncation is the determination of a *threshold* value. One approach is to allow users to decide that the largest n wavelet coefficients should be retained. Each value must be examined for this purpose. Because today's scientific data almost always comes with a minimum amount of metadata such as minimum/maximum values, an alternative is to keep only those coefficients within certain magnitudes.

4.2.2 Intermediate Stage Elimination

It is also possible to only keep some intermediate stages of the hierarchy. Since a scientist may not need all intermediate stages for data analysis, some of them can be eliminated in order to save memory, as long as enough information is retained to rebuild the eliminated stages. For example, in Figure 1-3 of Chapter 1, stage $j + 2$ can be eliminated as its approximation can always be reconstructed on the fly from stage $j + 3$. In case the detail of $j + 2$ is needed, it can be decomposed from stage $j + 1$.

4.2.3 Quiescent State Elimination

We observed that during wavelet decompositions, there are times when a majority of data features stay intact within resolutions [WB95a]. We describe this phenomenon as the *quiescence* of a data hierarchy, which is defined as follows

> In a progressive refinement environment, *quiescence* is a state of inactivity in which most of the distinctiveness of the refining target stays.

Only the lowest resolution representation needs to be maintained from a set of resolutions that are part of a single quiescent state. More details and examples are presented in Chapters 5 and 11.

4.3 Authenticity Analysis

There are two major sources of information loss: loss due to wavelet space projection (i.e., decomposition) and that due to partial data eliminations as described in Section 4.2. To measure the quality of the data approximations, we define a common evaluation and representation of the accuracy of a coarse resolution based on the loss of information between the coarse resolution and the fine resolution representations. We call this the *authenticity* of the representation [WB95a].

The greatest advantage of this method is that it does not require any post-transformation computation. An alternative is to use conventional error measures such as the correlation and root-mean-square-error. A comparison between the data before and after the truncation process indicates the error of the approximation at that resolution.

4.4 Partitioned Wavelet Transforms

Analogous to any other matrix-oriented computation, efficient wavelet algorithms require that the target data reside in a random access memory. The problem is lack of space: we do not have terabyte sized memory. In such a case, the data has to be stored on some external medium, such as CD-ROM.

External storage matrix computation has been studied for decades, more with tapes than with disks as the external device. The Fast Fourier Transform (FFT), was studied as early as 1967 for cases in which the data did not fit in fast memory and had to be stored on a single disk. Using eight disk files, Singleton [Sin67] describes how to compute the FFT for an N element vector in (lg N) I/O passes. Brenner [Bre69] presents two external FFT algorithms. One makes $\Theta(N/M)$ I/O passes, for $N > M$; the other makes $\Theta(\lg\ M)$ I/O passes for $N \gg M$, where M is the memory size. Floyd [Flo72] shows that general permutations on external storage require $\Omega((N\ \lg\ B)/M)$ I/O passes, where B is the block size. The similarity in computation between FFT and wavelet transforms makes us believe that our design approach is a realistic one. We apply some of these theories in our model. The hard part is determining the best sequence of merges. Knuth [Knu73, Section 5.4.9] contains a study for disks. The following is an unjustly simplified example of what we called a partitioned wavelet transform.

We note that each wavelet decomposition of each resolution can be done in non-sequential fashion. Very large data can first be separated into smaller units before wavelet operations are applied. Memory size as well as data size constraints of an orthogonal wavelet have to be considered. We assume that the size of the transformed data is always an integer power of two. (Not all wavelets have this restriction [Dau92], but the algorithms can be generalized.) In Figure 4-4, one-dimensional

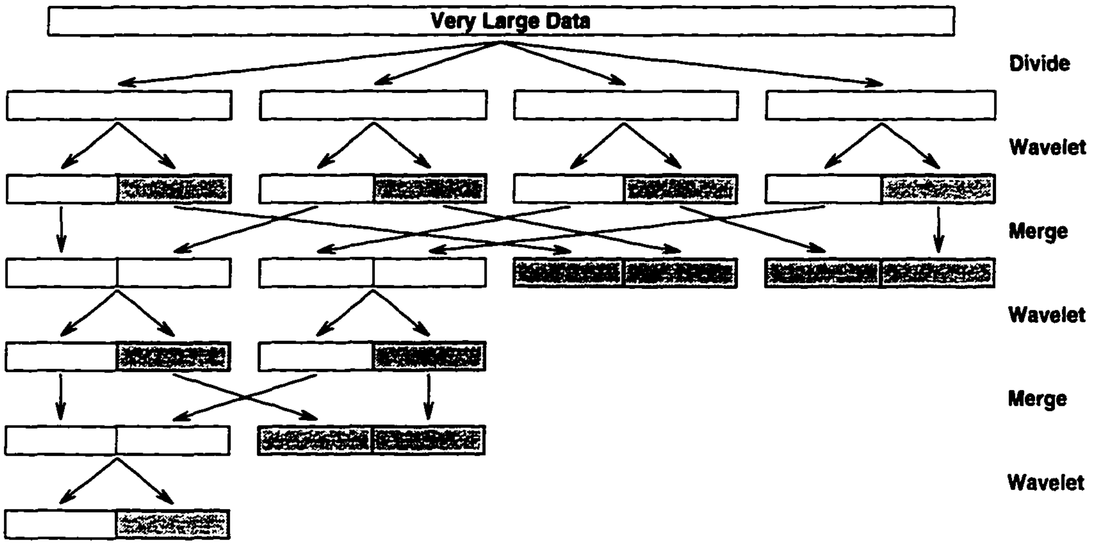


Figure 4-4: Partition wavelet transform.

very large data is divided into four units small enough to apply a wavelet transform efficiently. The approximations as well as the details of all wavelet applications are then merged in order. After the merge, we have two partitions of approximations and two partitions of details. More applications of wavelet decomposition followed by merges eventually complete the process.

As with other kinds of convolution processes, wavelet transforms have end-effects caused by the boundary conditions on the data vector. The wavelet filter uses $2p$ data values at resolution k to produce each data value at resolution $k + 1$. In order to avoid data shifting, most implementations use a centered filter. In other words, item j at resolution $k + 1$ is generated from items $2j - p$ to $2j + p - 1$. This creates a problem at both ends. For example, for D_4 ($p = 2$) the first data item must be generated from data items -2, -1, 0, which do not exist. We must eliminate these end-effects between partitions by supplying a few more data items at the end of each data partition to complete a regular matrix multiplication. Figure 4-5 shows our design. This example of partitioning a one-dimensional wavelet transform can also be applied to multidimensional data. The number of duplications depends on the number of vanishing moments of the wavelet. The size of the partitions depends on the size of the data, N, the block size, B, and the memory size, M.

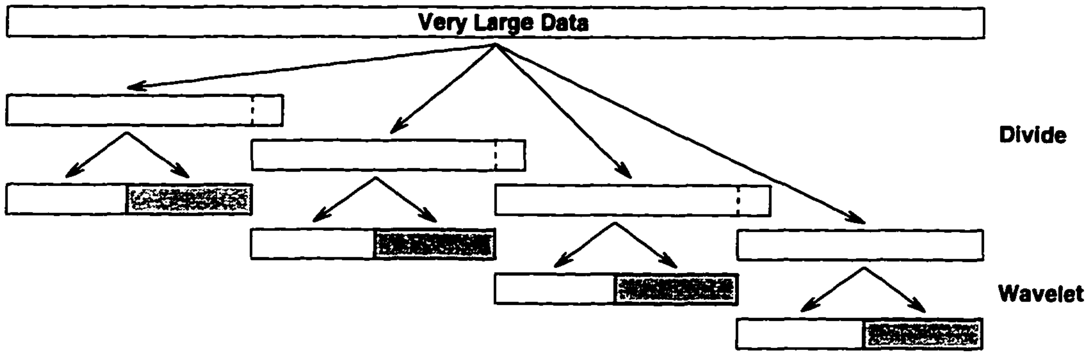


Figure 4-5: The extra data indicates the redundancy required to complete a regular matrix multiplication of each data partition.

Chapter 5

Authenticity Analysis

In the next few chapters, we turn from the high level outlines of the multiresolution data representation to the details of how a multiresolution hierarchy is built and used. In this chapter, we investigate the properties and characteristics of compactly supported orthogonal wavelets and their impact on the authenticity issues of one dimensional data hierarchies. We also establish measures to track error information due to data reductions. Finally, we present a real-life two-dimensional visualization example to show the advantages of wavelets over two of the very commonly used sampling functions. This chapters is based on [WB95a].

5.1 General Error Measure

For many applications a close approximation to the original can be tolerated. Often there are many detail coefficients of a wavelet transform that have a small enough magnitude that they can be ignored. The process is usually followed by a reverse transform operation, which produces a lossy reconstruction. This is by far the most popular application of wavelets.

A common way to measure the effectiveness of a lossy operation is by visual comparison based on feature classification by a human being on a small number of features such as the peak value and fundamental frequency. This *subjective* method assumes medium-sized data which can be displayed in a single picture.

Traditionally, applications which require higher error accuracy might use more *objective* (quantitative) measures such as the root-mean-square error, given by

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(f_i - f_i')^2}$$

where f_i and f'_i are the original and the reconstructed data respectively. Another function which describes the closeness of f_i and f'_i is the linear correlation given by

$$\frac{\sum_i (f_i - \overline{f})(f'_i - \overline{f'_i})}{\sqrt{\sum_i (f_i - \overline{f_i})^2 \sum_i (f'_i - \overline{f'_i})^2}}.$$

Unfortunately, these methods require both functions to be the same size. That means we have to reconstruct the data to the original size before we can figure out the error. Instead of computing an error measure based on comparing the approximation coefficients to the input data, we compute one based on the detail coefficients.

5.2 Wavelet Authenticity

We would like to have some mechanism for validating that a lower resolution representation of a dataset is an authentic approximation. Fortunately, by using a wavelet representation, the energy loss due to orthogonal projection can be obtained from the wavelet details of each resolution. Once we define a measure of the energy loss, we can use that measure for both analysis and visualization of the error. Defining an error measure that is consistent throughout the wavelet coefficient hierarchy is complicated by the change in coefficient scale and the different numbers of coefficients at each level of the hierarchy. Suppose w_{jk} is the wavelet detail at resolution j. Two of the commonly used error metrics are the L^1 and L^2 norms, in which the error due to projection from spaces V_{j+1} to V_j are given by

$$\sum_k |w_{jk}| \qquad \text{and} \qquad \sqrt{\sum_k w_{jk}^2}$$

respectively.

It is known that some wavelets lose more energy than others during decomposition. This brings us to the discussion of vanishing moments.

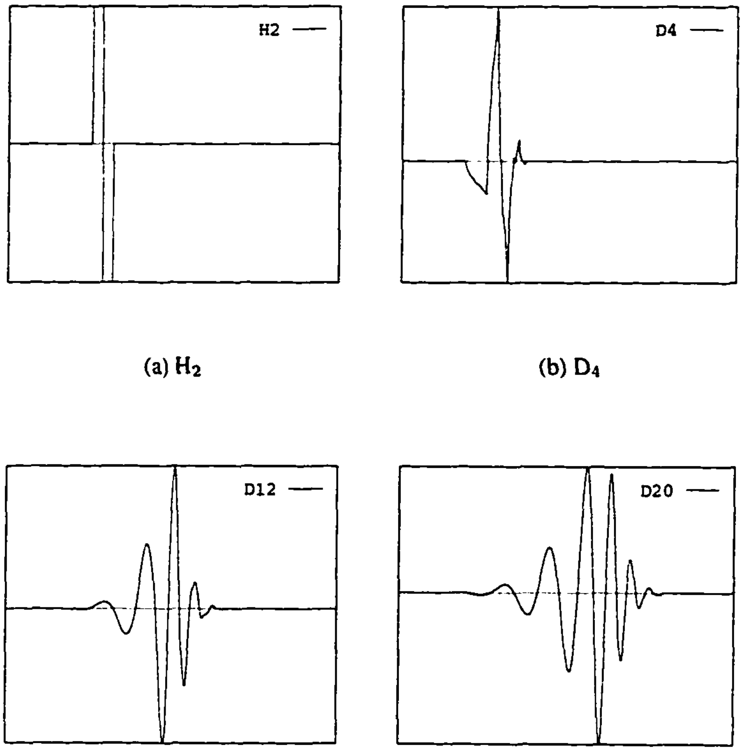
5.3 Vanishing Moments

Our discussion is restricted to compactly supported orthogonal wavelets. Others such as the Morlet wavelet and the Meyer wavelet have infinite support on the whole real line because they use sinusoids as the building blocks. They are C^{∞}.

The accuracy of a piecewise wavelet approximation can be characterized by the number of vanishing moments p of the wavelet. Strang [Str89] describes this as how well the polynomials 1, x, x^2, $\cdots$, x^{p-1} are reproduced by the approximation. For a wavelet ψ with p vanishing moments,

$$\int \psi(x)x^m dx = 0 \tag{5.1}$$

where $m = 0, \cdots, p-1$. In wavelet literature, the value of $2p$ is usually used as a subscript to identify a wavelet. This comes from the fact that the number of coefficients of a wavelet filter with p vanishing moments is equal to $2p$. Figure 5-1 shows H_2 (Haar wavelet with $p = 1$), D_4, D_{12}, and D_{20} (Daubechies wavelets with $p = 2$, 6, and 10.) It is also true that a compactly supported wavelet with p vanishing moments is p times continuously differentiable, i.e., C^p.



(a) H_2

(b) D_4

(c) $\mathrm{D}_1$2

(d) $\mathrm{D}_2$0

Figure 5-1: Four compactly supported orthogonal wavelets.

5.4 Function Pattern

The *pattern* of a function is loosely defined by its *shape* or *look* rather than its mathematical definition. Since most graphic displays are normalized in our application, the number of a function's *extreme values* (i.e., local maximum and minimum) and their *relative locations* are far more important than the absolute values. Following the same philosophy, we pay more attention to the *frequency* or the *period* of a periodic function than its *amplitude.*

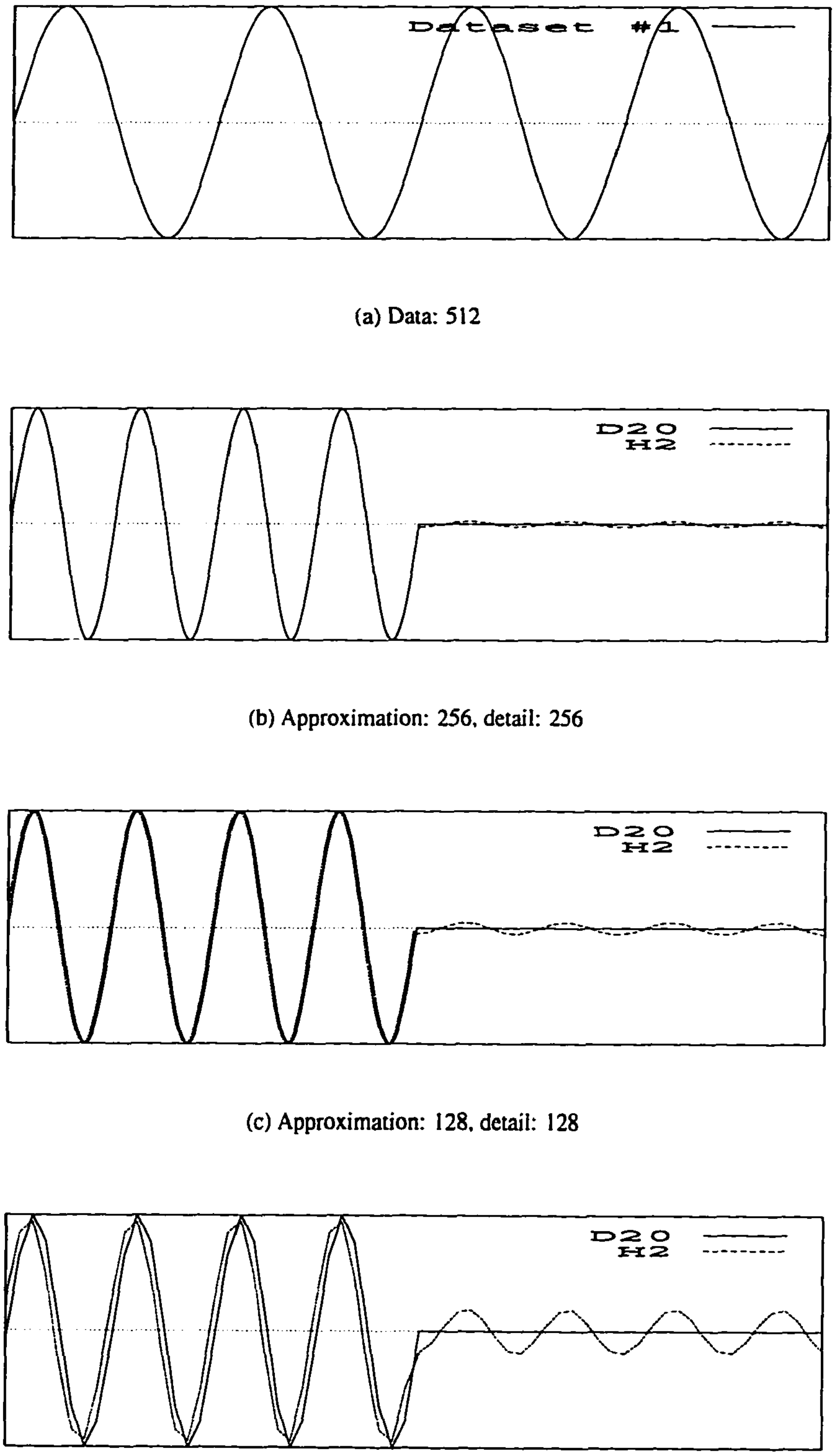
5.5 Approximation Characteristics

In this section, compactly supported orthogonal wavelets with $p = 1$ to 10 are applied to sinusoid-based test data with different patterns. We only plot the results generated by wavelets H_2 and D_{20}, which represent the lowest and the highest number of vanishing moments in our discussion. These examples are used to clarify the approximation characteristics of orthogonal wavelets. Note that in order to normalize the wavelets at every resolution, a normalizing factor $\frac{1}{\sqrt{2}}$ is introduced in the orthogonal matrix [Str89]. The same factor is applied to all other wavelet operations presented in this thesis.

5.5.1 Effect of Vanishing Moment on Merging of Data

A higher number of vanishing moments implies more coefficients in a wavelet filter matrix, which means that the implementation requires that more numbers be multiplied and added (merged) together during decomposition. However, the features in the data are not necessarily merged any faster. This is illustrated by a sinusoid with $2^9 = 512$ discrete samples shown in Figure 5-2a. Figure 5-2b depicts the first resolution of the decomposition. The left half is the approximation and the right is the detail. The tiny wavelet detail indicates good approximation for both wavelets.

Figure 5-2c shows the same dataset represented with $2^8 = 256$ wavelet coefficients. Both wavelets show strong approximations and weak details, with D_{20} having smaller details. After two more resolutions, the size of the wavelet coefficients is down to $2^6 = 64$, as shown in Figure 5-2d. H_2 continues to lose more energy that D_{20}. Both wavelets, however, manage to maintain the basic



(a) Data: 512

(b) Approximation: 256, detail: 256

(c) Approximation: 128, detail: 128

(d) Approximation: 32, detail: 32

Figure 5-2: a) Dataset #1 with 512 items. Wavelet coefficients with b) 512, c) 256, and d) 64 items.

pattern (four peaks) of the original function.

Although more numbers are multiplied with higher p values during decomposition, the merging process is dominated by the number of data values produced.

5.5.2 Energy Loss versus Resolution

Each wavelet decomposition introduces energy loss (error) into the approximation. By examining Figures 5-2b – 5-2d, we see that H_2 has significantly larger energy loss at each resolution. However, the amount of energy loss depends not only on the wavelet itself, but also on the *smoothness* of the data, which may change during each resolution. This is discussed in the following section.

5.5.3 Energy Loss versus Data Frequency

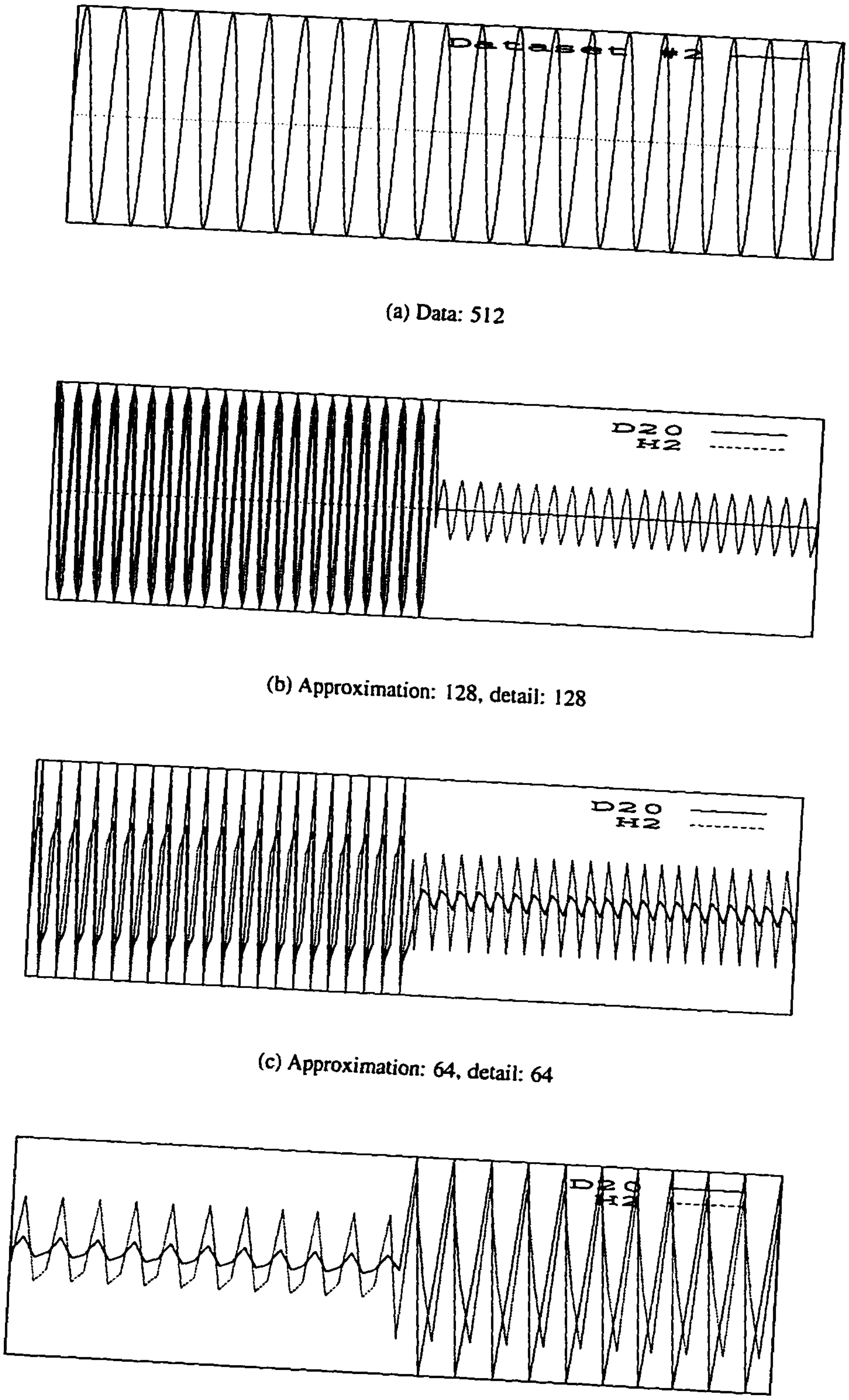
We use a higher frequency sinusoid to demonstrate the effects of energy loss versus data frequency. The discrete function has the same resolution ($2^9 = 512$) as the previous one. There are a total of 21 peaks as depicted in Figure 5-3a.

The rate of decay of waveleι decomposition is governed by the number of vanishing moments of the wavelet. In general, as illustrated in Figure 5-2, smooth functions like D_{20} approximate functions better than H_2. However, this is not always true.

Figure 5-3b shows the wavelet coefficients after two resolutions with 256 discrete items. Because of the smoothness of the data, the very small energy loss of D_{20} hardly shows up in the figure. Both H_2 and D_{20} retain all 21 peaks even though H_2 produces larger details. The next resolution is depicted in Figure 5-3c. It has 128 discrete wavelet coefficients: 64 approximations and 64 details. All 21 spikes stay in the approximation, which is rather non-smooth by now.

Disaster hits when the length of the dataset reaches $2^6 = 64$. The *Nyquist* limit (the lower bound to retain all the spikes) is passed. As a result, features merge and large details are created. For the first time in our examples, D_{20} has larger details than H_2 as shown in Figure 5-3d.

H_2 does not approximate functions accurately because it only has one vanishing moment. However, it does not tend rapidly to zero at finer levels, and the accuracy also depends on the smoothness of the function. When the function is smooth, more vanishing moments lead to smaller wavelet de-



(a) Data: 512

(b) Approximation: 128, detail: 128

(c) Approximation: 64, detail: 64

(d) Approximation: 32, detail: 32

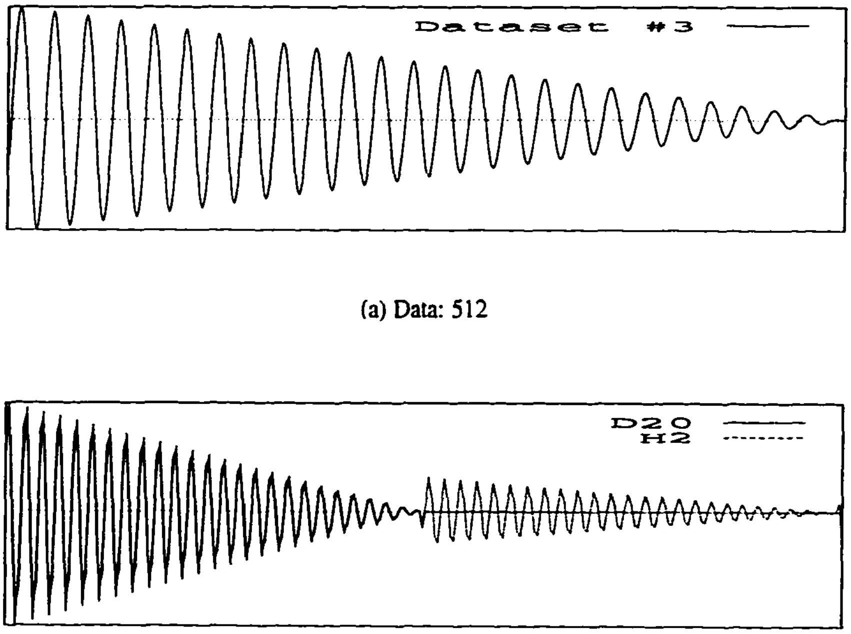
Figure 5-3: a) Dataset #2 with 512 items. Wavelet coefficients with b) 256, c) 128, and d) 64 items.

tails. On the other hand, more vanishing moments also lead to more large wavelet details when the function is non-smooth, so D_{20} is not always a better choice over H_2.

The phenomenon shown in Figures 5-3b – 5-3d is repeated for the rest of the resolutions. The first three resolutions of this example are marked by their inactivity or repose. Most of the major features of the approximations stay the same, i.e., the details are very small. That creates a state of *quiescence*. It also, however, establishes a *horizon* beyond which disaster lurks. In Figure 5-3c, half of the spikes in the approximation are gone, and the details (the energy loss) are actually much larger than the approximations.

5.5.4 Energy Loss versus Data Value

Energy loss of a wavelet decomposition depends on the data values. Higher data values imply more energy loss, and vice versa. This is illustrated by a sinusoid whose amplitude decreases continuously as depicted in Figure 5-4a.



(a) Data: 512

(b) Approximation: 128, detail: 128

Figure 5-4: a) Dataset #3 with 512 items. b) Wavelet coefficients with 256 items.

After two resolutions, the wavelet coefficients (with 256 discrete values) of dataset #3 is shown in Figure 5-4b. The values of the details increase as the data values increase.

5.5.5 Compression Pitfall

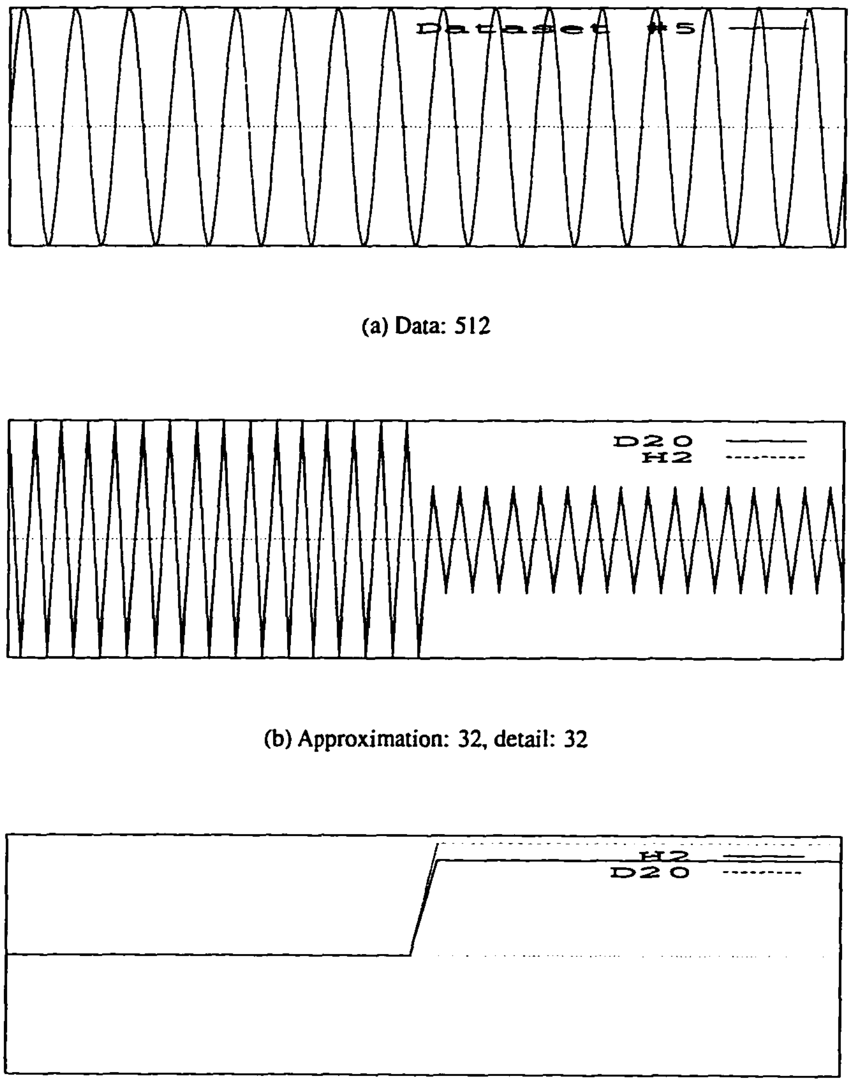
We would expect that the wavelet decomposition of a dataset (with floating point computations) will continue until the size of the approximation reaches $2p$. However, certain patterns, even with high energy content, can vanish suddenly and prematurely.

The sinusoid in Figure 5-5a is created with an integer value of cycles within the 512 discrete values. After three resolutions of decomposition, both H_2 and D_{20} shown in Figure 5-5b have perfect zigzag patterns. In fact, the pattern interleaves high-low discrete values with the same amount of energy but reversed directions. Then the energy vanishes suddenly as shown in Figure 5-5c when a high energy spike zeros out its neighbor, a low energy data item with exactly the same absolute value.

5.6 Error Tracking

We do not claim that our sinusoid-based examples simulate real life scientific data. They are presented because they show the special characteristics of orthogonal wavelets. Now we present results of non-sinusoid functions. Since wavelet transforms are well adapted to respond locally to rapid changes in function values [Dau92], orthogonal wavelets are often used as edge detectors [Mal89, MZ92]. Energy loss during wavelet decomposition usually implies edge (i.e., pattern) changes of the approximations. Small changes produce little wavelet details, which can largely be ignored. Large changes, however, produce significant wavelet details. These details are good indicators of the quality of the wavelet approximation of each resolution.

A function with 512 discrete values demonstrates the idea of using wavelet details to measure the authenticity of wavelet approximations, as described at the beginning of section 5.2. The function has seven special features including: 1) two discrete steps, 2) a portion of a sinusoid, 3) two steep slopes ($f(x) = x^3$ and its mirror image), 4) a sharp spike, 5) some fluctuating signals, 6) a large flat block, and 7) a small block. Each of them is identified with its feature number as depicted in



(a) Data: 512

(b) Approximation: 32, detail: 32

(c) Approximation: 16, detail: 16

Figure 5-5: a) Dataset #5 with 512 items. Wavelet coefficients with b) 64 and c) 32 items.

Figure 5-6a.

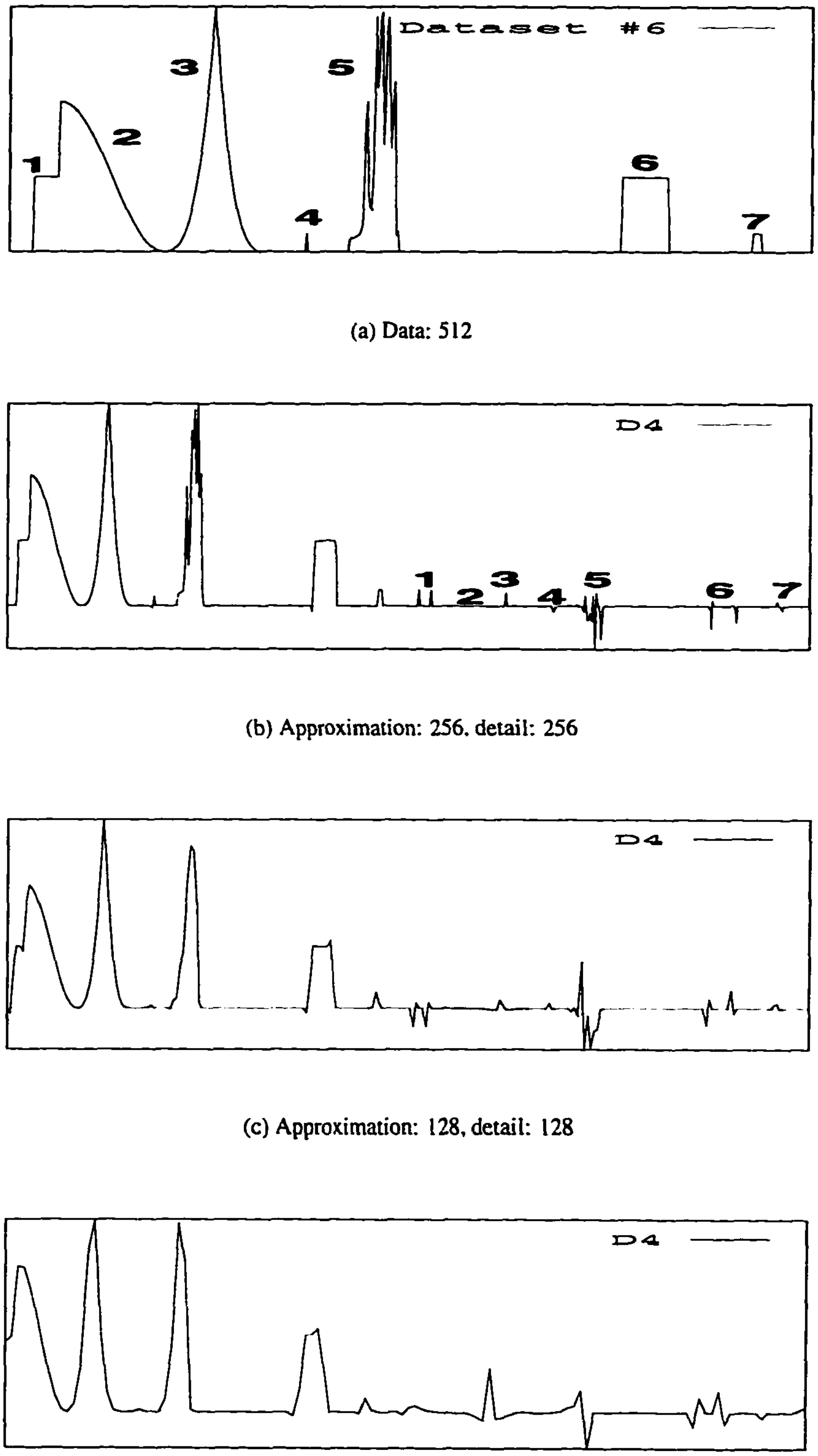
When D_4 is applied to this function, it produces 256 discrete approximations followed by 256 details, as shown in Figure 5-6b. In the graph, the wavelet details corresponding to the original features are marked with the same numbers. Starting from the left of the details, we see two sharp spikes which indicate the two sudden slope changes of feature 1. The energy loss is due to the fact that these two sharp edges are smoothed out. The details of feature 2 are almost invisible, reflecting the high approximation power of D_4 on sinusoids. The two steep slopes of feature 3 create only small spikes, which also indicate good approximations. The isolated spike of feature 4 is preserved with some loss. The fluctuating signals of feature 5 produce the largest amount of loss, indicating major pattern changes. Each of the following two blocks (features 6 and 7) produce two consecutive spikes, indicating the two sharp edges of the blocks. The next two resolutions are shown in Figures 5-6c and 5-6d.

The *maxima* and the *sums* of the absolute values of wavelet details of the first four resolutions are listed in Table 5.1. The *summation* of wavelet details shows the overall approximation quality, while the *maximum* indicates the largest local error. We notice from the graphs that D_4 keeps the function pattern intact. This is largely reflected by the correlated numbers in the table.

N	$max\|w_i\|$	$\sum \|w_i\|$
256	0.9124	6.2810
128	1.5074	10.7591
64	1.1257	9.1903
32	1.2708	8.8527

Table 5.1: A summary of D_4 on dataset #6.

These numbers, however, fail to show the energy loss of a particular feature such as the fluctuating signals, which produce the largest details. Table 5.2 is a summary of wavelet details based on the different features. In general, the fluctuating signals (feature 5) have the biggest pattern changes, which explains why they have the largest details. Feature 1 has one of the larger losses. The two steps are totally smoothed out after the second resolution.



(a) Data: 512

(b) Approximation: 256, detail: 256

(c) Approximation: 128, detail: 128

(d) Approximation: 64, detail: 64

Figure 5-6: a) Dataset #6 with 512 items. Wavelet coefficients with b) 512, c) 256, and d) 128 items.

N	$\sum_{feature} \lvert w_i \rvert$						
	1	2	3	4	5	6	7
256	0.708	0.014	0.477	0.177	**3.763**	0.966	0.177
128	1.557	0.081	0.754	0.225	**6.177**	1.641	0.325
64	~2.560		1.432	0.099	**3.790**	0.813	0.498
32	~7.177					~1.676	

Table 5.2: A summary (by feature) of D_4 on dataset #6.

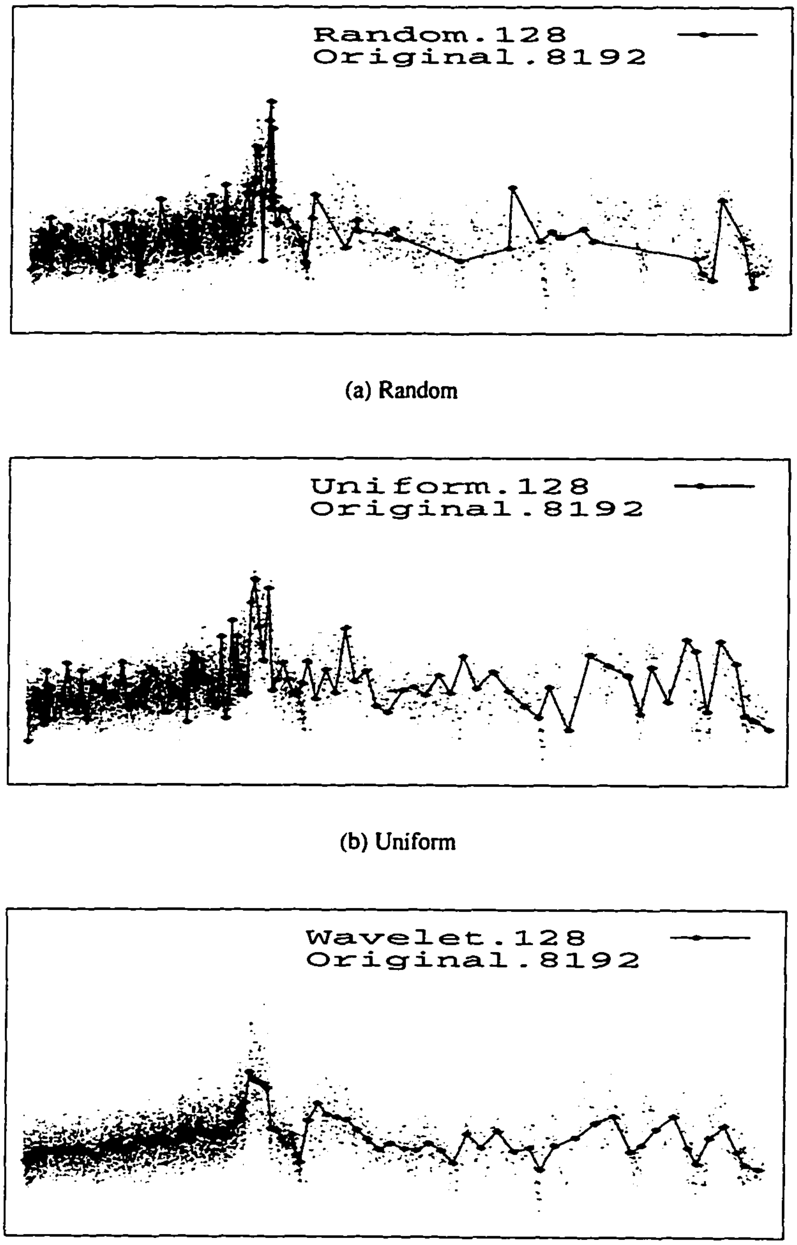
Table 5.3 lists the results of D_{20} applied to dataset #2 as presented in Section 5.5. The resolution printed in bold shows the occurrence of aliasing as well as the end of the first quiescence covering the previous three resolutions. A second quiescence starts right after this resolution.

N	$max\lvert w_i \rvert$	$\sum \lvert w_i \rvert$
256	0.3718	0.9792
128	0.4690	1.3124
64	1.2059	13.2433
32	**3.8953**	**72.6761**
16	2.1756	8.5699
8	0.7576	1.6806

Table 5.3: A summary of D_{20} on dataset #2.

5.7 The Advantages of Wavelets

A bivariate time series data set with 8192 data points is used to demonstrate the advantages of wavelets in supporting scientific visualization. The dataset is fed into a *random* sampling function, a *uniform* sampling function, and a Haar wavelet to generate three data hierarchies with the size of each resolution equal to $2^n, n \in \mathbb{N}$. In the random sampling function, data points are selected randomly for display according to the indices generated by the Unix function *lrand48()*. In pixel-based graphics, a common belief is that both the random and the uniform sampling functions are convenient but not very effective, particularly when the sampling rate is low [GW87]. In this example, we pick the resolution with only 128 (less than 1.6%) data points to study. The results of the three functions are shown in Figure 5-7. The backgrounds of these graphs are scatterplots of the



(a) Random

(b) Uniform

(c) Wavelet

Figure 5-7: The scatterplots in the background are the original data. The polylines in the foreground are generated by (a) a random sampling function, (b) a uniform sampling function, and (c) a Haar wavelet.

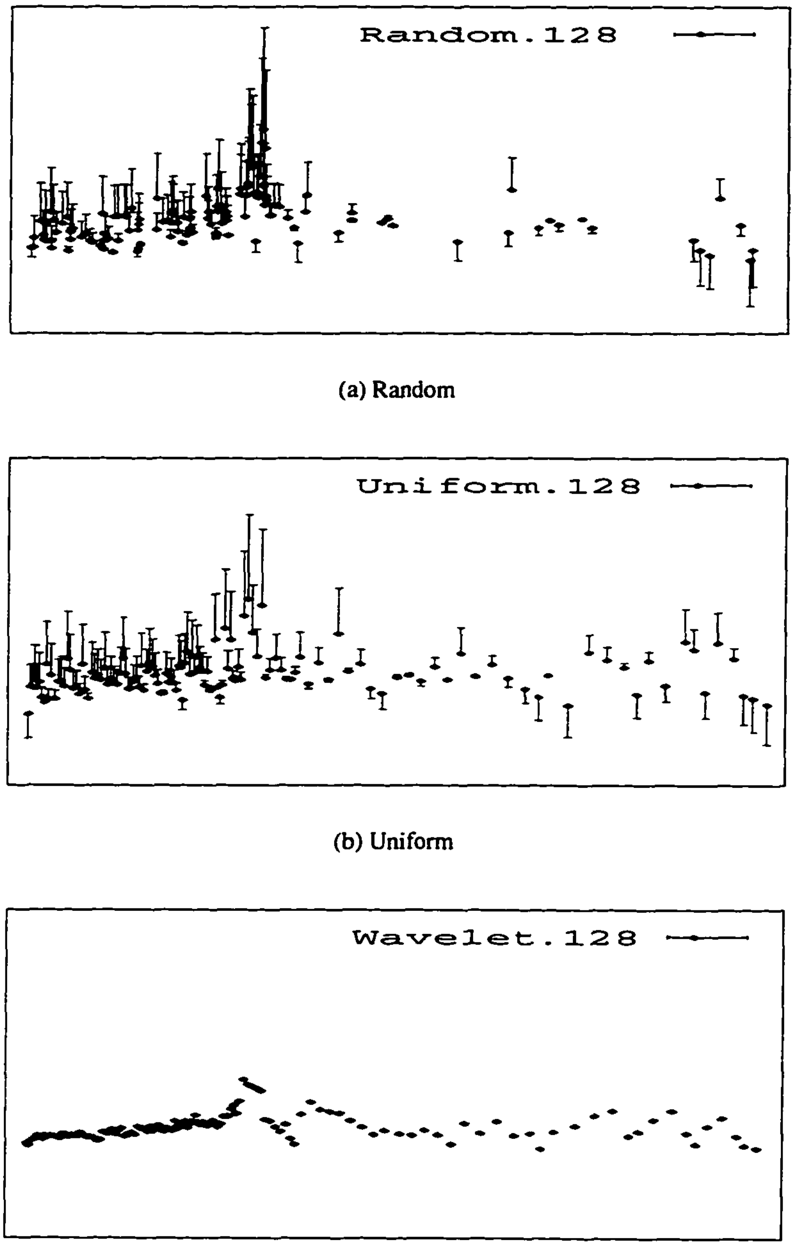
input data. Notice the data density is higher (as it is darker) on the left hand side of the graphs. The selected data is plotted on top of the scatterplots using *polylines*. Although all three functions show fairly good approximations at a very superficial level, there are a number of discrepancies among them (such as the magnitude and the spread of the data) that deserve further attention.

Each of the approximation points is required to represent well over 60 neighboring data points. We notice that the sampling rate of the random function (Figure 5-7a) is very irregular. Some of the steps are much larger than the others, even among consecutive points. These are very undesirable effects as our goal is to clear up a dense data plot. Any uneven sampling area may imply that further approximations are needed. Although this problem does not show up in the approximations generated by the uniform sampling function and the wavelet (Figures 5-7b and c), there are still visual differences (such as magnitude fluctuations) between them. We cannot decide the better approximation solely by judging the shapes of the polylines. As a result, we turn our attention to more objective measures.

Given the n-dimensional Euclidean space $\mathbb{R}^n$ containing m scatter points, x_{ij} where $i \in [1, n]$ and $j \in [1, m]$, the *center of mass*, $\overline{M}$, of the data is defined as

$$\overline{M} = \frac{1}{m} \sum_{i=1}^{m} \left(\prod_{j=1}^{n} x_{ij} \right) .$$

Statistically, this represents the first moment of the scatter points. The Euclidean distance between $\overline{M}$ and the corresponding approximation point is a measure of the quality of the approximation. Figure 5-8 re-plots the same approximation data described in Figure 5-7 using *error-bars*. The length of each bar reflects the difference between the approximation point and its corresponding $\overline{M}$. The problems of the random sampling and the uniform sampling (Figures 5-8a and b) show very clearly. Not only are the magnitudes of the differences very large, they also vary inconsistently. On the other hand, the error-bars of the wavelet approximation (Figure 5-8c) show no difference at all. In other words, the low frequency data points generated by the wavelet are consistent and very accurate approximations to represent their neighboring data values. For operations with high frequency requirements such as outlier detection, wavelets also provide the *wavelet details* to filter out the high frequency outliers.



(a) Random

(b) Uniform

(c) Wavelet

Figure 5-8: The approximations plotted with error-bars are generated from (a) a random sampling function, (b) a uniform sampling function, and (c) a Haar wavelet.

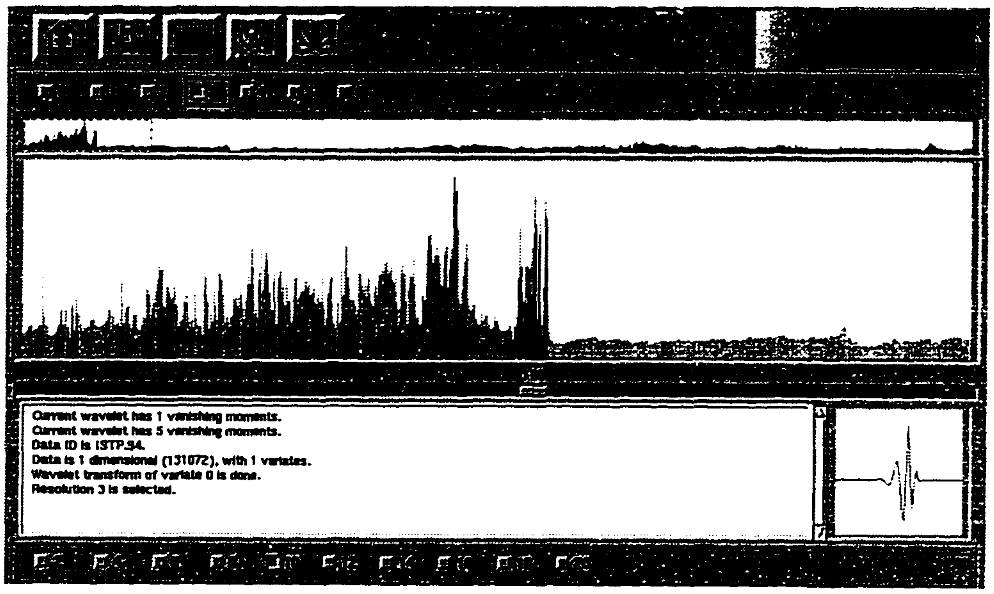
Chapter 6

Visualization of Error

We show how the error metrics described in Chapter 5 can be used to support multiresolution data exploration. A simple multiresolution visualization tool is presented to illustrate the importance of error information of a one-dimensional data hierarchy. The display of data along with the representation error is a critical component of effective interactive exploration of large data sets.

6.1 Visualizing One Dimensional Data

We describe a wavelet-based multiresolution visualization system and use it to display approximations of scientific data and its corresponding error. The system, as depicted in Figure 6-1, supports



```
Current wavelet has 1 vanishing moments.
Current wavelet has 5 vanishing moments.
Data ID is ISTP.94.
Data is 1 dimensional (131072), with 1 variates.
Wavelet transform of variate 0 is done.
Resolution 3 is selected.
```

Figure 6-1: The system displays a coarse approximation of a one-dimensional dataset with 131,072 items.

progressive refinement data analysis with resolution as fine as one data item per pixel.

6.1.1 User Interface Overview

The system front-end is divided into eight window panes. The first pane contains five command buttons, as shown in Figure 6-2. The first button is the *data bank* command, which initiates all the system I/O operations. The next one is the *wavelet* command button. It looks up the wavelet selection table and generates the multiresolution wavelet approximations accordingly. The middle button allows scientists to select colormaps. It is followed by a context-sensitive *help* command button, and the *exit* button.

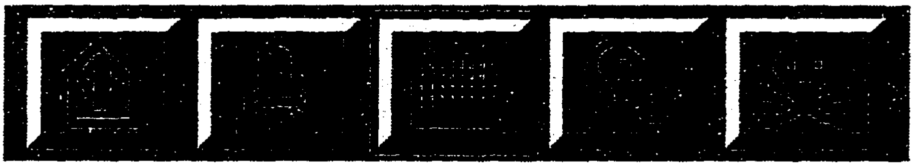


Figure 6-2: System command buttons.

The colormap window pane in the upper right of Figure 6-1 displays all the colors currently available for data mapping. Figure 6-3 depicts five pre-defined colormaps described by Levkowitz and Herman [LH92] for visualization. They are the rainbow scale, the heated-object scale, the magenta scale, the blue scale, and the linearized optimal color scale.

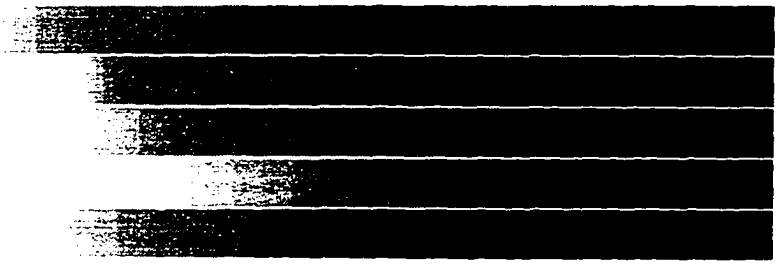


Figure 6-3: From top to bottom: the rainbow scale, the heated-object scale, the magenta scale, the blue scale, and the linearized optimal color scale. See also Color Plate 1 in Appendix D.

The resolution selection buttons pane is shown in Figure 6-4. In this example, the one-dimensional

Figure 6-4: Multiresolution buttons.

input data has a total of 131,072 items. The coarsest resolution supported by the system is 1,024. That means there are a total of 7 resolutions available, with resolution 0 being the coarsest.

Figure 6-5 depicts a portion of the data range selection pane. The background of the slider is the coarsest approximation of the original data. It serves as a rough guideline for navigating through very large scientific data. The rectangular shaped rubberband marks the range of data being selected. The position of the rubberband is controlled by the mouse. The box size depends on the display resolution and the size of the original data.

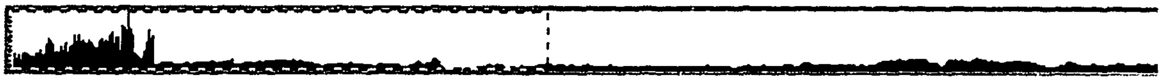


Figure 6-5: Data range selection slider.

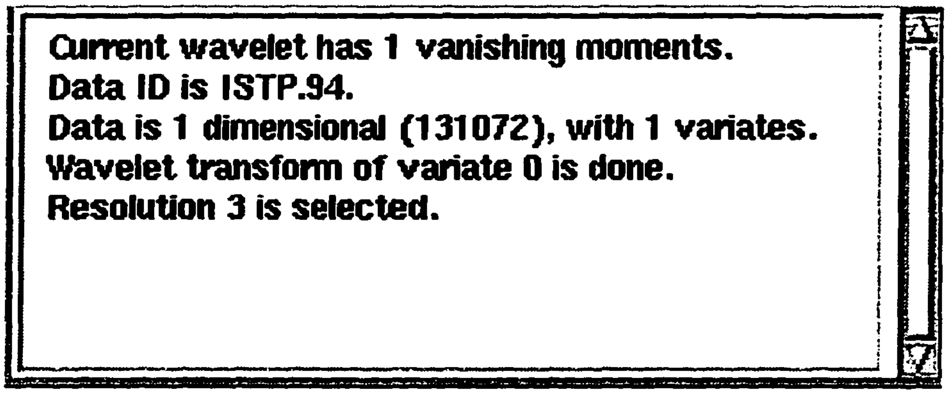
We skip the main data display window pane for now, and move on to the next control slider. The index reflected by the slider is used to map data to color. The current prototype only allows very simple color mapping.

The system provides a scrollable message box, which is depicted in Figure 6-6. It is used to display system status including data ID, metadata, colormap, wavelet, as well as error messages.

The wavelet window pane displays the wavelet currently being used. Figure 6-7 depicts a Daubechies wavelet [Dau92] with five vanishing moments.

The wavelet selection pane contains ten selection buttons, as shown in Figure 6-8. The system currently provides compactly supported orthogonal wavelets with vanishing moments from 1 to 10. This includes the Haar wavelet, and the nine wavelets of the Daubechies family.

The main window pane in Figure 6-1 is the data visualization area of the system. Since one-dimensional time-varying data is the primary target of this prototype, we limit the visualization



Current wavelet has 1 vanishing moments.
Data ID is ISTP.94.
Data is 1 dimensional (131072), with 1 variates.
Wavelet transform of variate 0 is done.
Resolution 3 is selected.

Figure 6-6: System message display.

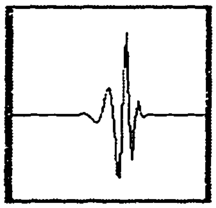


Figure 6-7: Wavelet display.

Figure 6-8: Wavelet selection buttons.

options to simple polyline plots. This basic technique, although very simple, is a powerful tool to convey information and characteristics of time-varying data [Cle93].

6.1.2 Visualization of One Dimensional Error

Figure 6-9 shows a one dimensional dataset extracted from the CD-ROM[1] recorded from the spacecraft GEOTAIL of the ISTP [God92] project. The dataset contains electron average energy data recorded every 64 seconds around the earth for the first three months of 1994. It has a total of $2^{17} = 131,072$ integers.

D_4 is used to generate a total of 7 resolutions, from the 0^{th} resolution with $2^{10} = 1,024$ to the 6^{th} resolution with $2^{16} = 65,536$ items. The display is a one-dimensional line plot with colors

[1] USA_NASA_DDF_ISTP_KP_0003

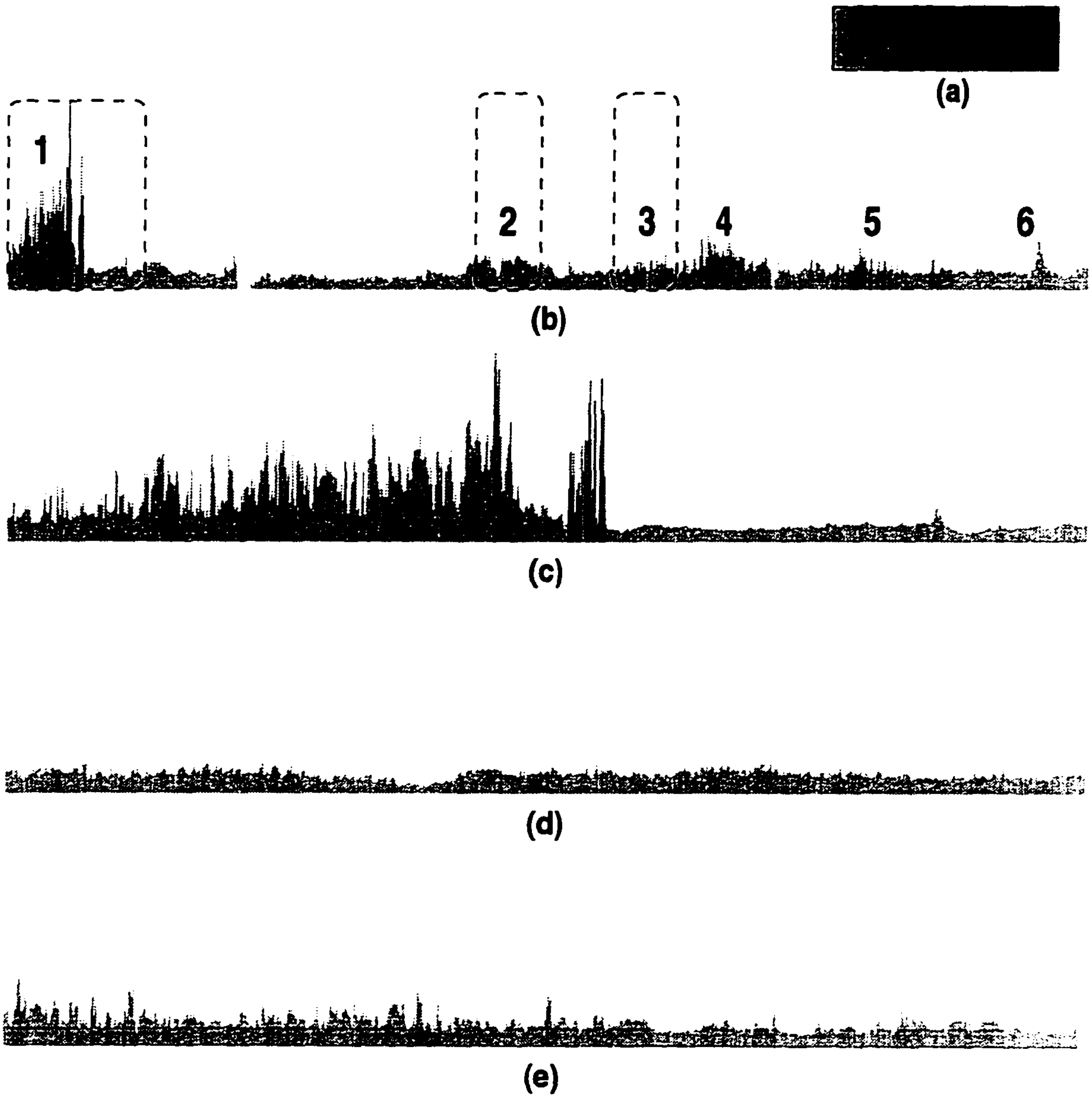


Figure 6-9: a) Rainbow colormap. b) The coarsest approximation of the ISTP average energy data is displayed at the 0^{th} resolution. Interesting features are identified with numbers. Dotted rectangles are zooming windows. The color indicates the accumulated approximation error. c) Feature 1 at the 3^{rd} resolution. d) Feature 2 at the 4^{th} resolution. e) Feature 3 at the 4^{th} resolution. See also Color Plate 2 in Appendix D.

indicating the accumulated data loss (error) of each item. A rainbow colormap shown in Figure 6-9a is used for the display.

We start from the coarsest (0^{th}) resolution with $1,024$ items, an approximation of the original $131,072$ items. From Section 5.5, we know that the energy loss of a wavelet transform depends on a number of factors including the data values and the smoothness of the data. Six features are chosen from the data, as indicated in Figure 6-9b, to illustrate these ideas as well as the importance of the approximation error display.

Feature 1 contains highly fluctuating data with some of the highest data values. Both of them contribute to the very high energy loss of the approximation. This is reflected by the darker colors (green/blue) of the error display in Figures 6-9b and 6-9c.

In terms of data values, features 4, 5 and 6 are more or less close to each other. However, when we look at the smoothness of the data, we notice that feature 6 is smoother than the other two. It implies that feature 6 has the lowest energy loss. This is accurately reflected by the color of the error displayed in Figures 6-9b. Both features 4 and 5 have green spikes while feature 6 is light orange.

The value of the error representation is particularly evident when looking at features 2 and 3. They have very similar values, spreads, and shapes. Feature 3, however, has more green (higher value) spikes than feature 2. These two features (marked by the dotted rectangles) are zoomed to a finer (4^{th}) resolution, as shown in Figures 6-9d and 6-9e. This finer resolution reveals that feature 2 is indeed very smooth data, while feature 3 is relatively non-smooth. Figure 6-9e shows that feature 3 has multiple spikes spread across the area. These narrow spikes fade away during the downsampling process because the *Nyquist* limit (to hold all the spikes) is reached. Their errors are clearly reflected in our display. By using the color to represent accumulated error, we are able to identify areas of the coarsest resolution representation that warrant investigation at finer resolutions.

Chapter 7

Multiresolution Techniques for Multivariate Data

In the past few chapters, we have discussed the concept and implementation issues of multiresolution data hierarchies based on wavelets, and have shown the importance of error information to data exploration. In the next two chapters, we turn to the applications of the techniques underlying the concept. In this chapter, we introduce the notion of multiresolution approximation into a previously described multidimensional multivariate visualization technique known as *brushing* [BC87, War94, MW95] (see Chapter 3), and we discuss how wavelets fit in with previous work on high dimensional brushing. This chapter is based on [WB96a].

7.1 Data Brushing

Becker and Cleveland [BC87] introduced the notion of *brushing* as an interactive data exploration process in which a user visually highlights or deletes subsets of displayed data. This process provides the user an extra dimension of information as well as insight from an otherwise static display (See Chapter 3). A variety of two-dimensional brushing techniques are described by Cleveland [Cle93]. Ward expanded the direct manipulation concept and created *multidimensional* brushing in XmdvTool [MW95, War94]. The system is a collection of four multivariate visualization techniques: scatterplot matrix [Cle93], star glyphs [SFGF72], parallel coordinates [IRC87, ID87, ID90], and dimension stacking [LWW90]. These tools, although powerful, suffer from the size limitation of a display. Loss of information during very large data visualization is seemingly unavoidable.

In this chapter, we incorporate the concept of brushing into our visualization model. One of the distinctive features of wavelet brushing is that the brushed data is displayed at a different resolution, usually higher, than the non-brushed data. Figure 7-1 shows a simplified example of wavelet brushing on a bivariate scatterplot. As we can see, the coarse background outside the brush gives the *overall* structure of the context surrounding the brushed area while the fine brushed data provides the *local* structure inside the brushed area. The approach relies on the notion of visual inspection of the data approximation.

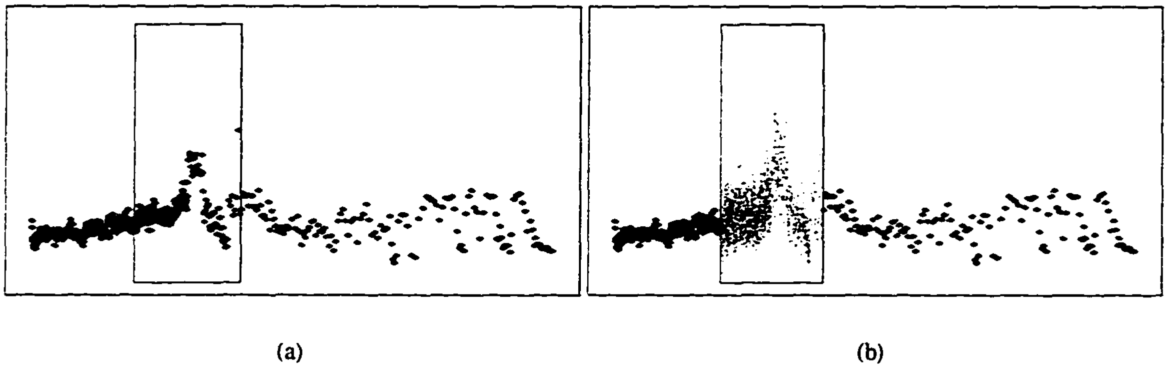


Figure 7-1: a) The brush area is defined. b) Fine brushed data is painted over a coarse data background.

We implemented multiresolution brushing in an enhanced version of XmdvTool [MW95, War94]. The applications to scatterplot matrix and parallel coordinates are emphasized, although the concept is applicable to any multivariate visualization of very large data.

7.2 General Approach and Problems

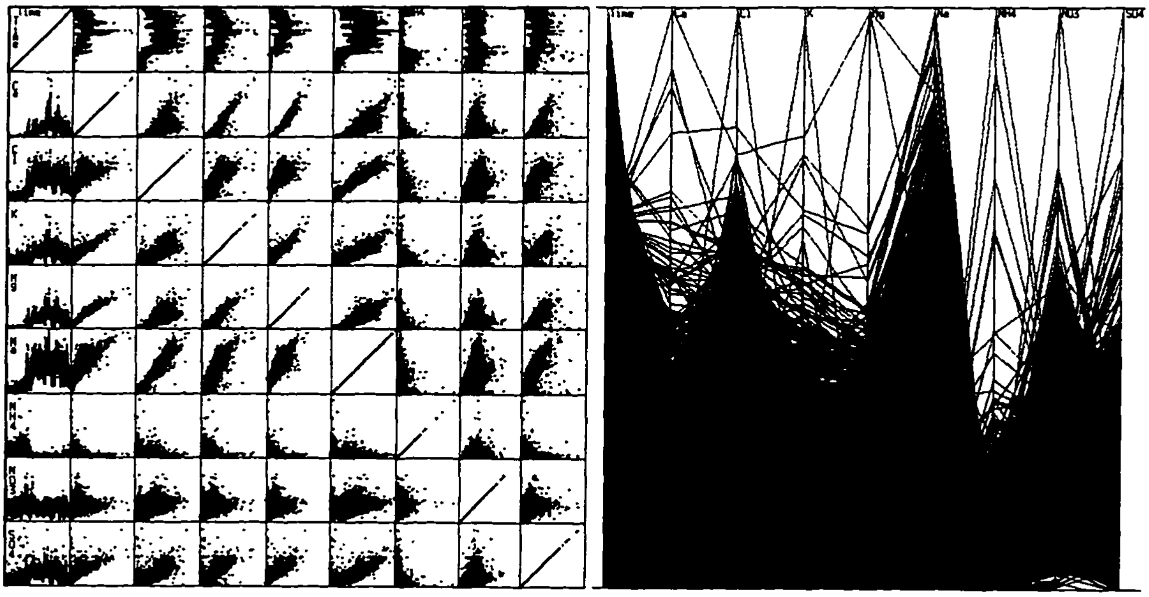
A significant characteristic of any data visualization technique is the relationship between the number of available display pixels, P, and the number of data values to be displayed, D. For a single static image the best we can expect to achieve is $(D = P)$.[1] In fact, Keim et al. [KK94, KKS93]

[1] Note that the effectiveness of any visualization technique cannot be measured solely by the number of values displayed, but is a very complicated evaluation based on ill-defined perceptual and technical issues.

have developed such a visualization technique in the domain of database visualization.

For very large data visualization, however, D is much larger than P, so we cannot rely on any simple static image technique. Instead, we must develop mechanisms that allow us to present authentic *abstractions* of the data and use *time* as a presentation parameter to increase the effective number of pixels used to complete a visualization task. The goal of our research is to support scientific data visualization with $(D \geq P)$. The prevailing techniques which support large data visualization can generally be characterized as *hierarchical visualization* in which multivariate data is represented as a visualization hierarchy. A user needs to click the display and follow the hierarchical path to browse the data one projection view at a time. An example of this class of visualization techniques is *worlds within worlds* [FB90]. These techniques require not only extra screen space to browse the data, but also the user's ability to mentally integrate different views over time.

Problems with visualization of very large data are easily shown when considering the scatterplot matrix and the parallel coordinate representations of large multivariate datasets. Figure 7-2 depicts



(a) Scatterplot matrix

(b) Parallel coordinates

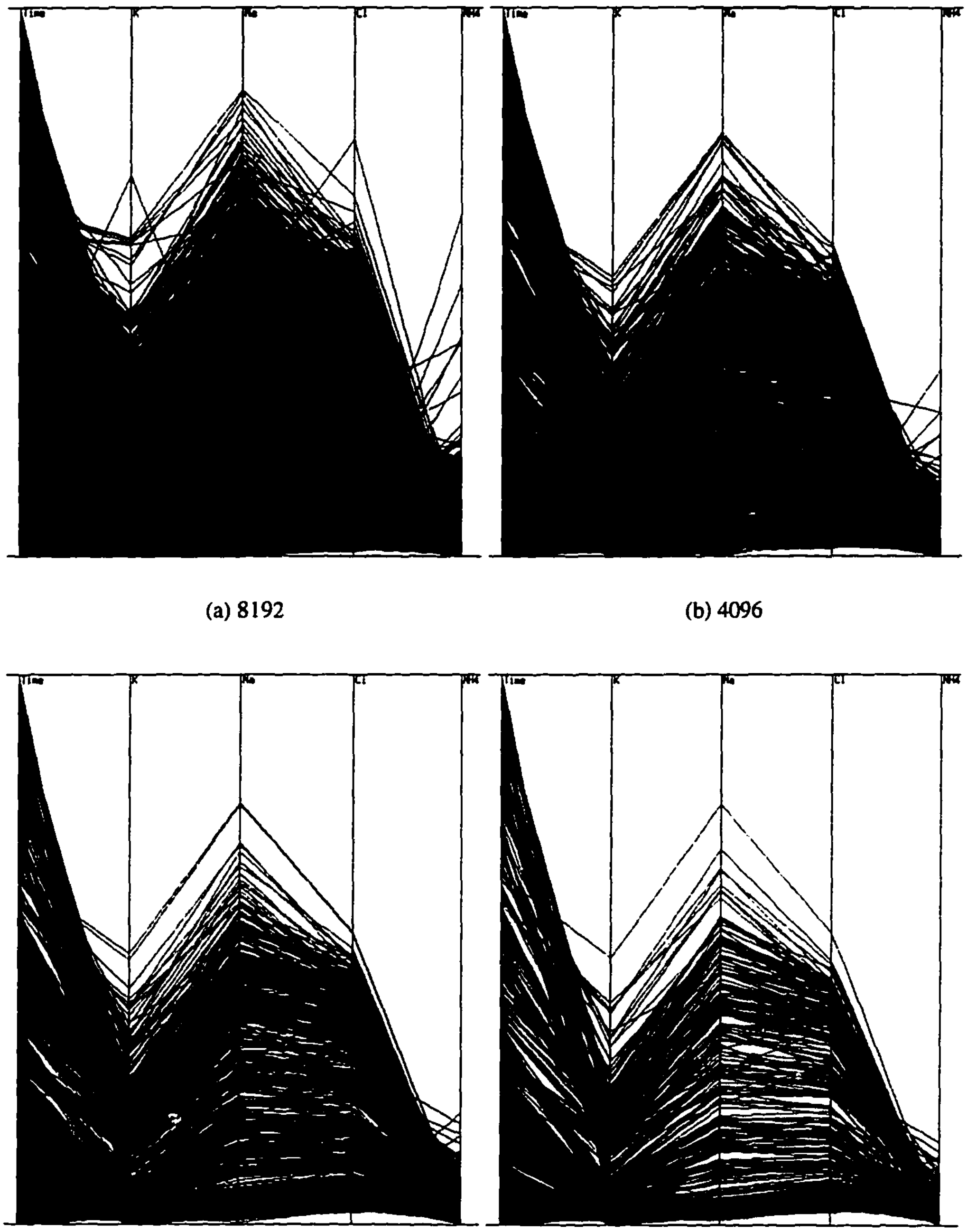
Figure 7-2: Problems of very large multivariate data visualization using a) the scatterplot matrix and b) the parallel coordinates representations.

two multivariate plots of a nine-variate time series dataset using scatterplot matrix and parallel coordinates. Although the scatterplot matrix shows the *overall* structure of the data, it fails to show its *local* structure as neighboring data points (over $660K$ scatterdots) are packed together into display pixels. Problems with the parallel coordinate plot are even more severe: we cannot see anything in the lower half of the plot. The data size has to be reduced before we can apply these two techniques effectively. In the following discussion, we present results of wavelet approximations being applied to these techniques, and explain why it is desirable to have wavelet support in high dimensional brushing.

7.3 Multivariate Visualization Examples

In our data approximation hierarchy, the coarse resolutions highlight the *overall* structure of the original data while the fine resolutions provide the *local* structure. This works nicely with most, if not all, of the multivariate visualization techniques. Figure 7-3 shows a series of parallel coordinate plots of a five-variate dataset. The data resolution starts from 8192 and ends at 1024. The termination point of this resolution reduction process is very data dependent: the goal is to reach the lowest resolution that is still an authentic representation of the original data.

Figure 7-4 shows scatterplot matrices of a fine to coarse data hierarchy of the same five-variate dataset. Since each scatterplot displays 8192 data points, there are over $200K$ scatterdots displayed in Figure 7-4a. As we can see, the visual improvement of the lower resolution scatterplot matrix plots is not as dramatic as with parallel coordinates. This is largely due to the fact that each data point is represented by only a scatterdot compared to a polyline in parallel coordinates. The highly localized time-series data used in this example also plays an important role for this phenomenon. In Figure 7-4, many neighboring points are quietly packed together into display pixels instead of creating chaotic patterns. A more dramatic result of using the scatterplot matrix to display *multidimensional* data is presented in Section 7.5.4.



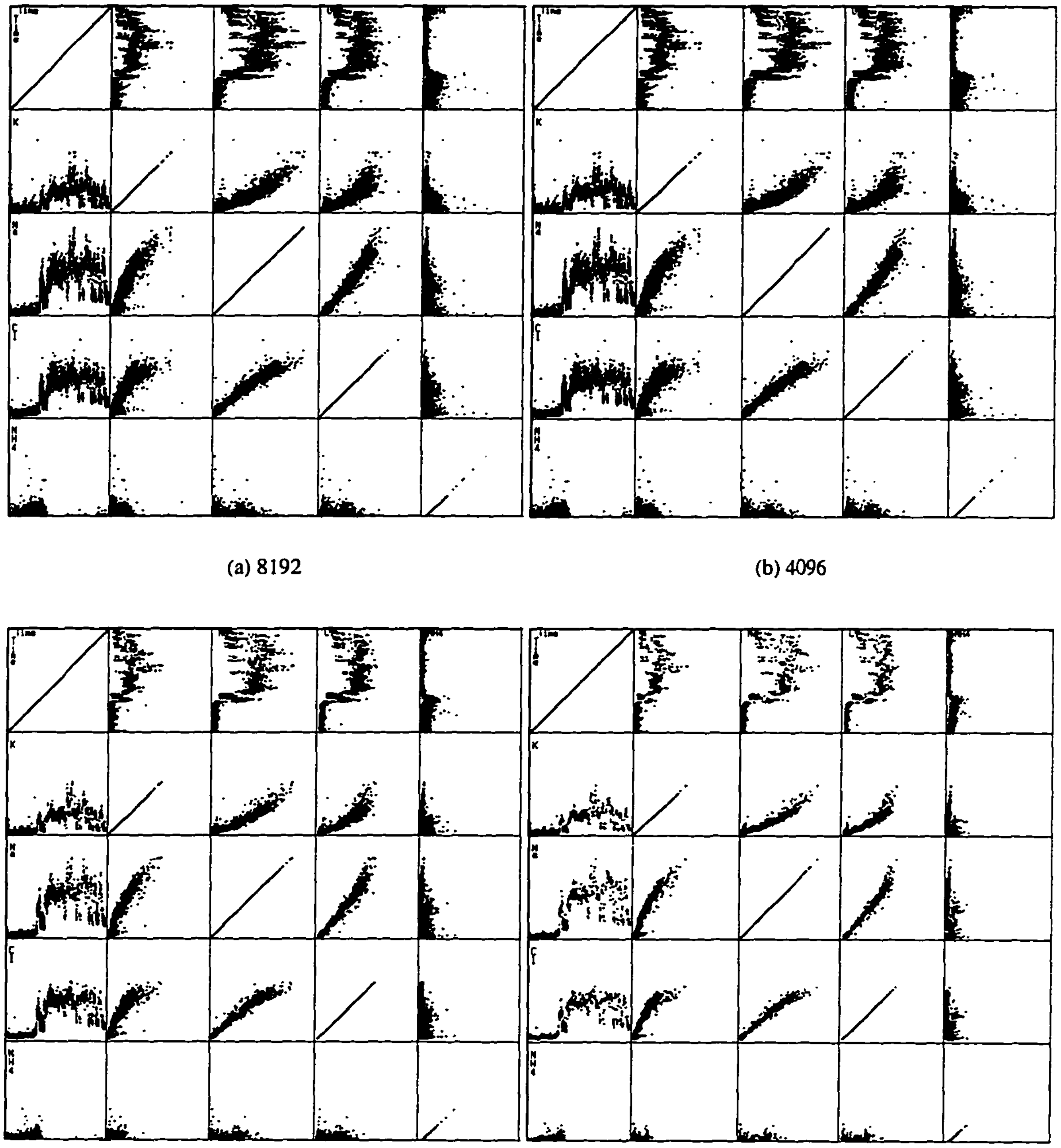
(a) 8192

(b) 4096

(c) 2048

(d) 1024

Figure 7-3: Parallel coordinate plots from fine resolution with 8192 data points to coarse resolution with 1024 points.



(a) 8192

(b) 4096

(c) 2048

(d) 1024

Figure 7-4: Scatterplot matrices from fine resolution with 8192 data points to coarse resolution with 1024 points.

7.4 XmdvTool

Ward [MW95, War94] gives many compelling arguments for the use of high dimensional brushing in exploratory data analysis. The essence of Ward's design is the concept of providing a visual representation of multidimensional database queries as well as their results. With the addition of wavelet transforms, we argue that the operations of multidimensional brushing can be improved to support very large data visualization.

7.4.1 Wavelet Support

Our work follows the system design of XmdvTool. The new functions are written in C using X/Athena Widgets. The system currently supports orthogonal wavelets with up to ten vanishing moments. When a data set is transformed, a fine to coarse hierarchy is generated according to the number of vanishing moments selected. Once the data brush is defined, the user is allowed to independently control the display resolutions of the brushed data and the non-brushed data.

7.4.2 Brushing with Wavelets

The process of browsing multiresolution data suffers from the same problem as hierarchical visualization, both of them need multiple displays to browse the data. This requires not only extra screen swapping, but also the user's ability to mentally integrate different views during the process. We argue that if we allow users to control the data resolutions of the *brushed* and the *non-brushed* data, they can physically merge multiple views into one display for data exploration. Our notion of multidimensional brushing extends Ward's design to include multiresolution support.

7.4.3 Implementation Strategies

Two implementation strategies can be used to construct wavelet brushing. The first one contains multiple layers of approximations with the resolution decreasing (or increasing) according to its distance from the brush. Given a two-dimensional display with a square brush defined near the lower right corner of Figure 7-5a, the system creates multiple layers of display data surrounding the brushed area using the wavelet approximations (as shown in Figure 7-5b.) This design gives

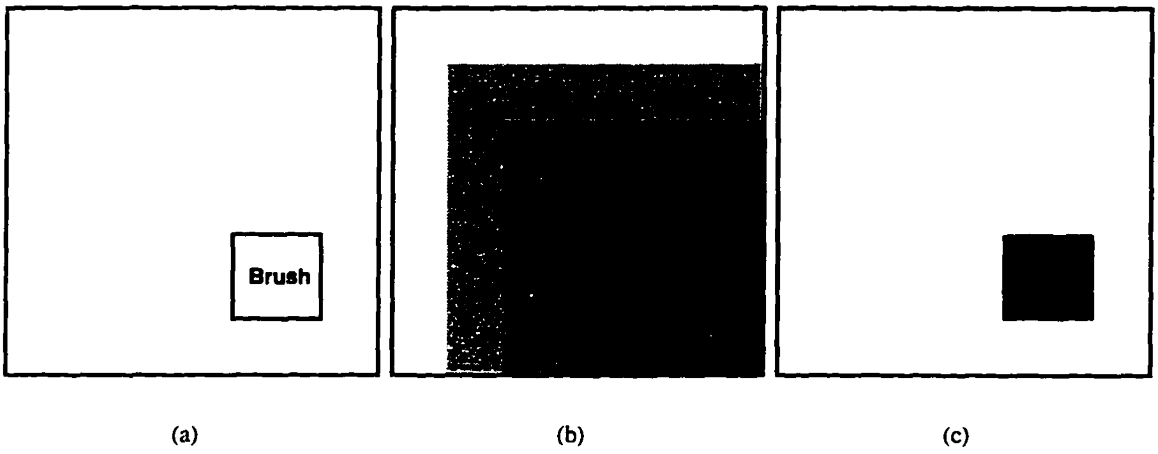


Figure 7-5: (a) A two-dimensional display with a square brush. (b) Brushing with multiple layers of approximations. (c) Brushing with two layers of approximations.

users a smoother view of brushing. However, it also creates confusion for users to distinguish the brushed multivariate data points from the non-brushed ones. The problem is even worse for parallel coordinates as the polylines of brushed data almost always overlap with those of non-brushed data. We have not yet found an effective way to clearly display data with multiple data resolutions in one display.

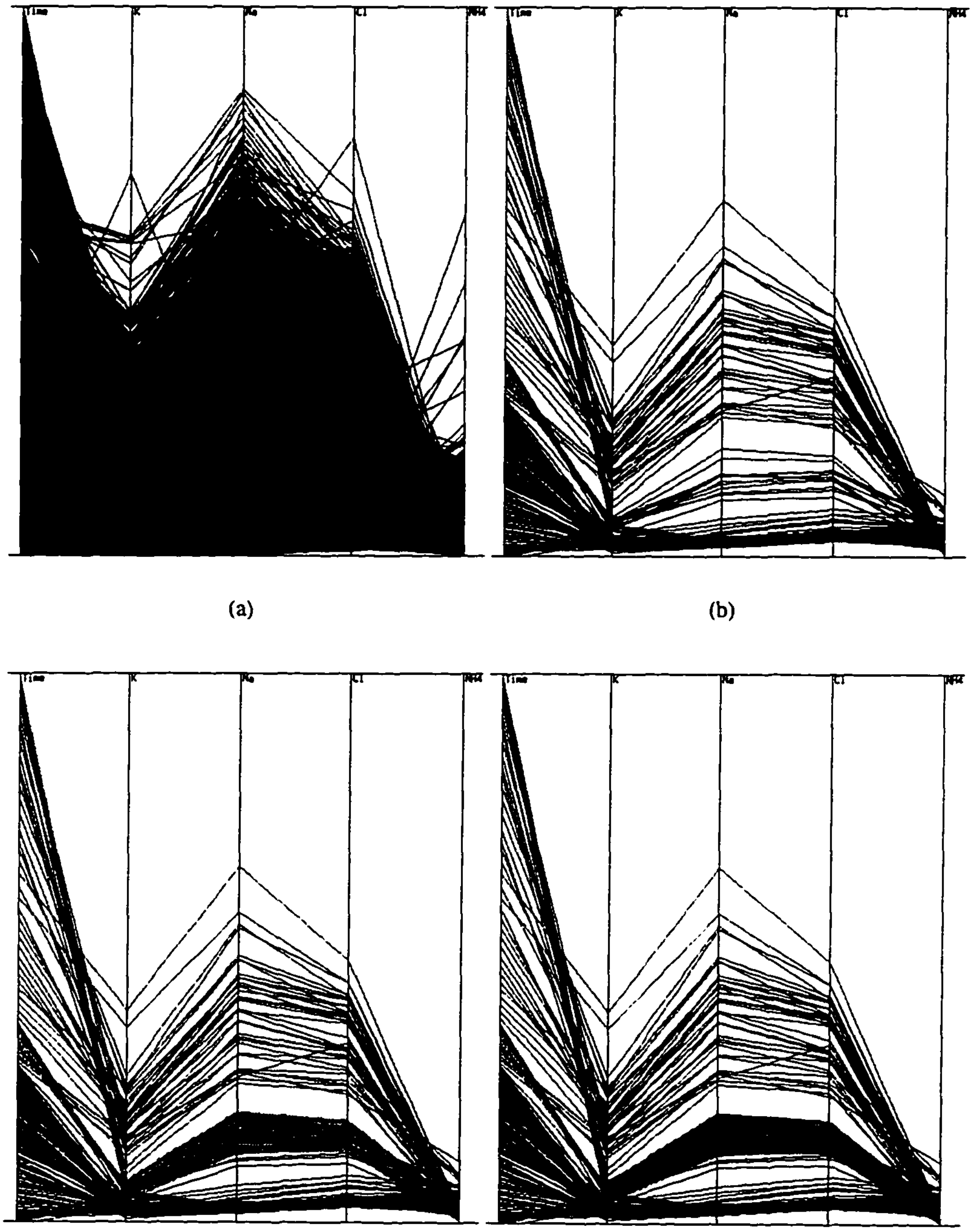
The second implementation, as shown in Figure 7-5c, contains only two layers of approximations. Once the brushing values are decided, both the brushed and the non-brushed data resolutions can be controlled independently. The user can increase or decrease the resolution of each area by clicking the corresponding button. Not only is this approach simpler to implement, it is easier to use and it responds faster compared to the previous implementation. In our experiments, we achieve real-time responses on a DECstation 5000 for data with up to about 700K data items. The time to paint a scatterplot matrix is usually longer than for parallel coordinates. The current version of wavelet brushing supports operations of a single data brush at a time.

7.5 Applications

Four applications of wavelet brushing stand out in our experiments. Three of them (resolution magnification, authenticity analysis, and outlying data identification) are multivariate visualizations while the fourth one involves high dimensional data. All of them take advantage of the multiresolution property, which may otherwise be very difficult to carry out because of the large amount of data that must be visualized. In the following discussion, the color plates which describe the applications are painted in three basic colors: red for the brushed data, blue for the non-brushed data, and light blue for the data brush. The non-brushed data (in blue) is always painted ahead of the brushed data (in red). Their resolutions are displayed in the wavelet toolbox window.

7.5.1 Resolution Magnification

The first application is to use the multiresolution brush as a magnifying glass to increase the resolution of a subset of the data. Figure 7-6 depicts a series of parallel coordinate plots of the same time-series dataset at different resolutions. The first one is the original data. Obviously there is no visible pattern to be seen. So, the user reduces the overall display resolution gradually until the individual line patterns can barely be seen (Figure 7-6b.) From this resolution, the user finds out that the concentration of the potassium ion (the second axis on the left) drops sharply in recent times (the upper portion of the first axis on the left.) In order to study the local details of this time period, the user brushes all the data connected to this time period. The brushed data (in red), the non-brushed data (in blue), and the brush itself are shown in Figure 7-6c. Since the interesting spots are identified, there is no need to study the details of the non-brushed data. The user can then increase the resolution of the brushed data gradually to study the local details, as depicted in Figure 7-6d. With only a few simple mouse movements, the user is able to establish evidence to show that the overall sea salt concentration (mainly potassium chloride and sodium chloride) of modern times is indeed lower than before. In addition, the last axis on the right hand side indicates a minor reduction of the overall ammonium concentration during the same time period. This can be explained by the diminishing of major volcanic eruptions recorded in modern history. This information, which is hidden behind a massive number of polylines in Figure 7-6a, can easily be revealed with our wavelet brushing.



(a)

(b)

(c)

(d)

Figure 7-6: An example of resolution magnification. (a) The original data. (b) A coarse approximation. (c) A brush is defined. (d) Fine data is painted inside the brush. See also Color Plate 3 in Appendix D.

7.5.2 Authenticity Analysis

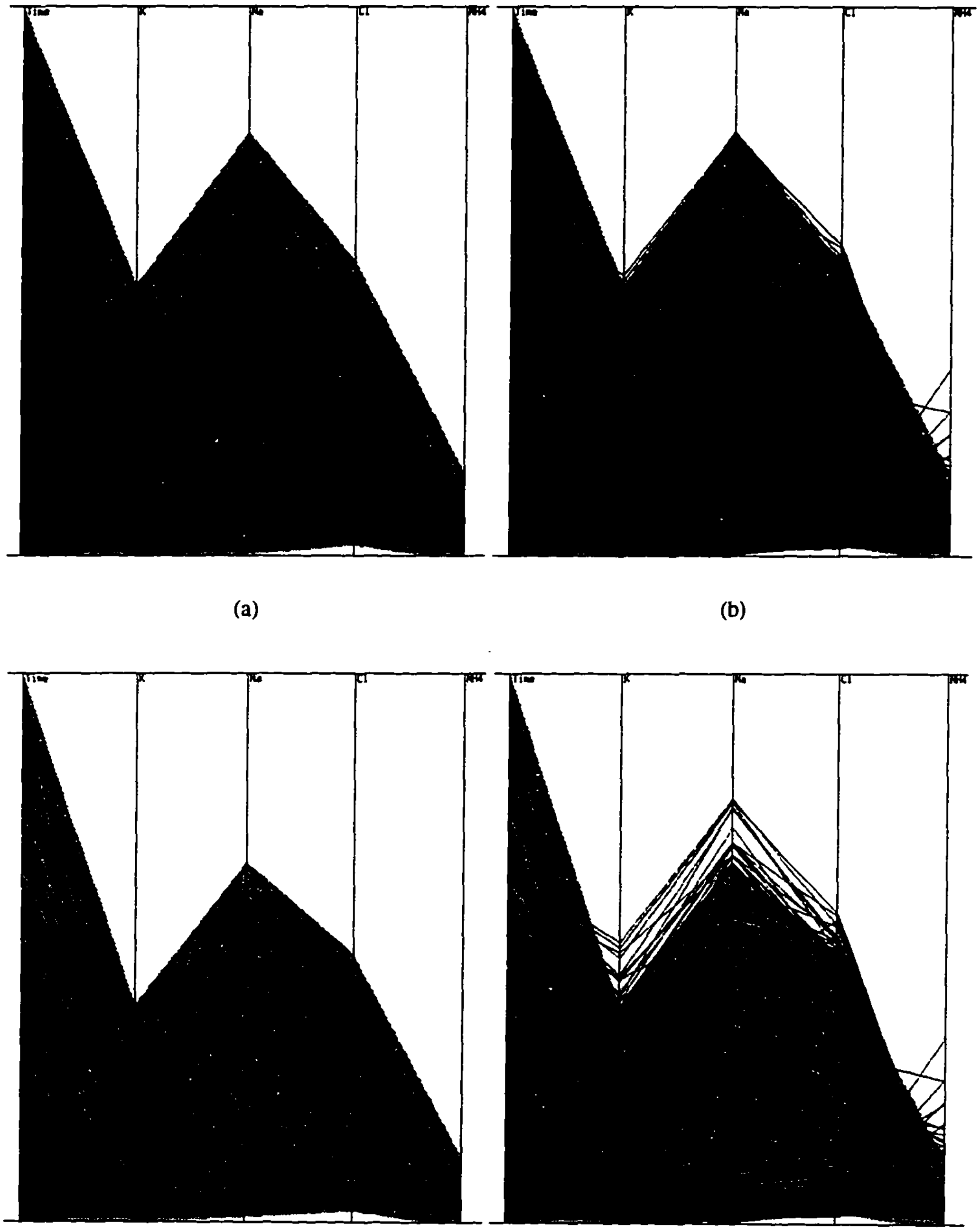
The second application involves the authenticity study of the wavelet approximation itself. When the data size of a very large dataset is reduced by a wavelet transform, there is always the question whether the approximation is a faithful representation of the original. In Section 5, we propose the use of color to display the error information generated by wavelet transforms. With the use of multiresolution brushing, we present a short-cut to interactively verify the accuracy of the wavelet approximation during data analysis.

During any course of analysis using wavelet brushing, the user can always lock the brushed data and bring back the high resolution non-brushed data to the same display. For example, in Figure 7-7, the user wants to compare the current display with the higher resolution data. This can be achieved by creating a brush which just encloses all the low resolution data as depicted in Figure 7-7a. The rest of the high resolution data is then brought back by increasing the resolution of the *non-brushed* data for comparison. In our example, only a few non-brushed data items (in blue) show up (Figure 7-7b) when the resolution of the non-brushed data increases from 512 to 2048. This provides a measure of the quality of the low resolution representation in terms of how well it encloses the data values of the higher resolution representation. The small number of discrepancies between these two resolutions also indicates that further reduction may be permissible.

The same process is then applied to lower resolution data with 256 items (as shown in Figure 7-7c.) When the resolution of the non-brushed data is brought back to 2048, more blue lines appear in Figure 7-7d. By counting the number of the high resolution non-brushed data items (the blue lines) in Figures 7-7b and d, we conclude that the approximation shown in Figure 7-7a is a better representation than the one in Figure 7-7c.

7.5.3 Outlying Data Identification

It is often desirable for a scientist to identify outlying data, that is, data whose location in $\mathbb{R}^n$ is outside that of the majority of the population. In a conventional parallel coordinate plot, locating this kind of data is not always possible. Our multiresolution brushing design, however, enables the user to display the *differences* between two data resolutions. These differences, in the context of



(a) (b)

(c) (d)

Figure 7-7: An example of authenticity analysis. (a) A coarse approximation with 512 data points. (b) Finer non-brushed data is painted over (a). (c) A coarser approximation with 256 data points. (d) Finer non-brushed data is painted over (b). See also Color Plate 4 in Appendix D.

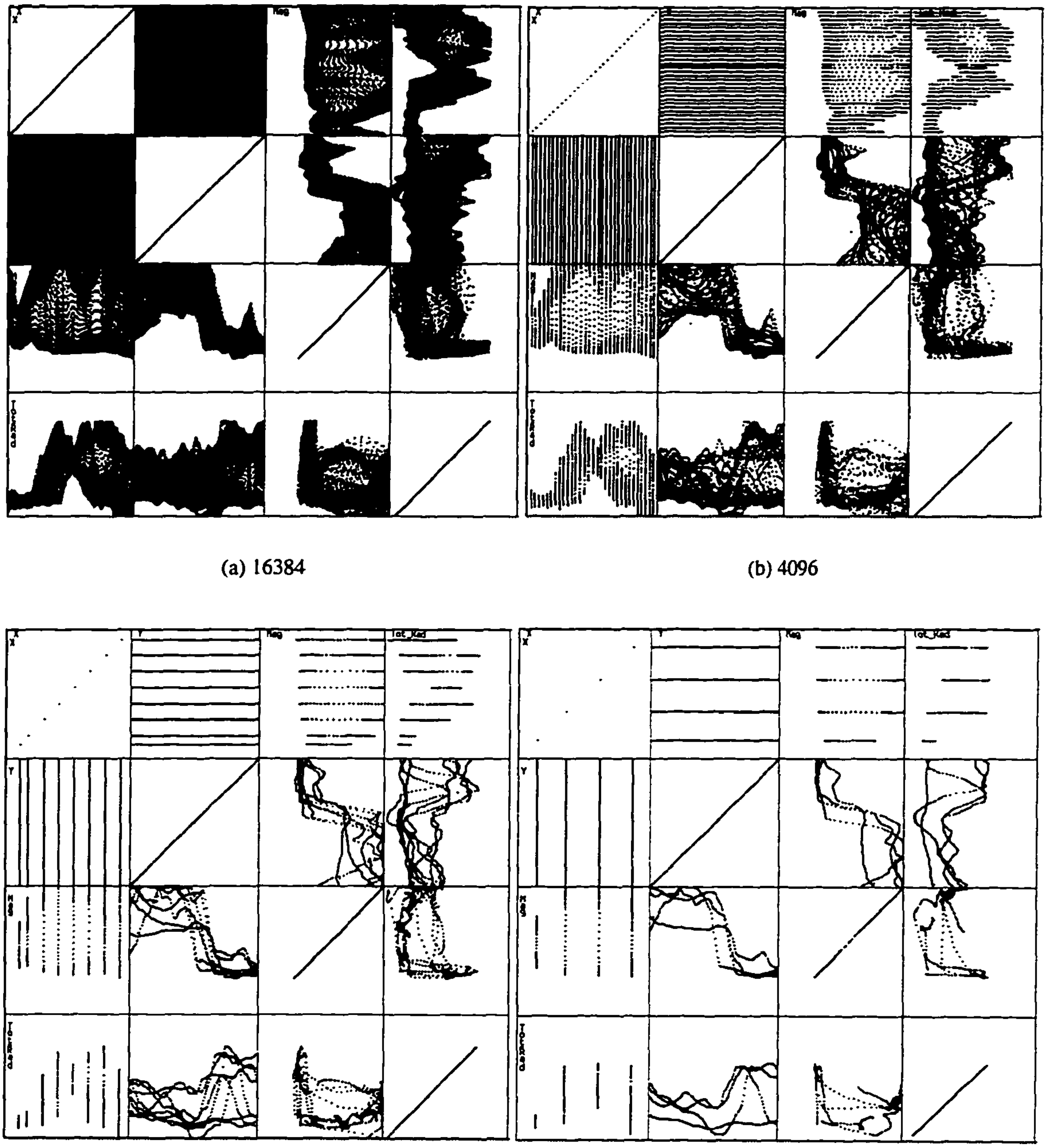
orthogonal wavelet transforms, can be interpreted as the outlying data of the high resolution data. In Figure 7-7b, the blue lines are the outlying data of the high resolution data which do not show up in the low resolution data (in red). When the resolution of the brushed data decreases, more blue lines show up in Figure 7-7d. Our multiresolution wavelet brushing offers another means of providing control over *outlyingness* [RL87] for data analysis. A possible weakness of this approach is that some of the painted outlying data, which is part of the non-brushed data, can be over-painted by the brushed data. The impact of this problem, however, is much less with the scatterplot matrix display.

7.5.4 High Dimensional Data

In scientific visualization, multivariate visualization techniques are seldom applied to high dimensional data. With the support of wavelet brushing, we present such an example to study two-dimensional satellite images. The idea can be extended to three-dimensional volume data without any system modification.

For high dimensional data defined on Cartesian grids, even small dimension sizes can be multiplied into a large amount of data. For example, remotely sensed satellite data usually comes with a set of measurement taken from the same location. Extracting small pieces of information from multiple variates becomes a very challenging task. A multivariate display with wavelet brushing provides a coarse overview of a series of data, and allows the user to brush slices of multivariate data for further investigation.

The XmdvTool distribution includes two sets of remotely sensed images in the file *outmt.okc*. This file contains four variates: x and y coordinates (in y-major order), the magnetics and the accumulated radiometrics channels of an area in Australia. The original data and its wavelet approximations are plotted by scatterplot matrix in Figure 7-8. With 16384 data points in the multidimensional multivariate dataset, the scatterplot matrix in Figure 7-8a displays over $260K$ scatterdots. Together with the x and y coordinates, the first image (i.e., the magnetics channel) occupies nine tiles in the upper left corner of the matrix. A three dimensional plot of the magnetics image is depicted in Figure 7-9. Obviously, the display in Figure 7-8 gets simpler as the data resolution decreases. The hard part is to figure out the meaning of the approximations.



(a) 16384

(b) 4096

(c) 1024

(d) 512

Figure 7-8: (a) The original data with 16384 data points. Wavelet approximations with (b) 4096, (c) 1024, and (d) 512 data points.

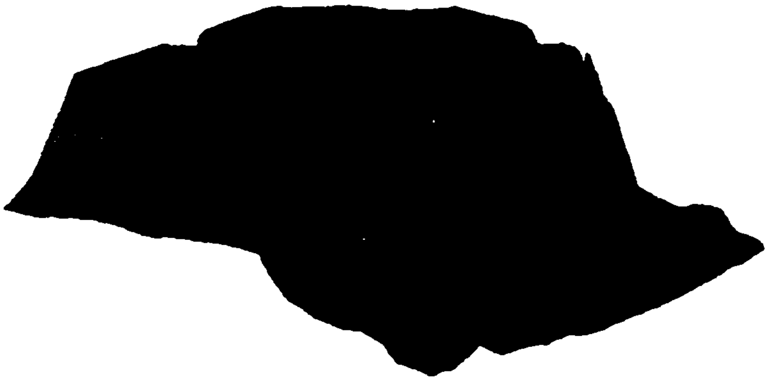
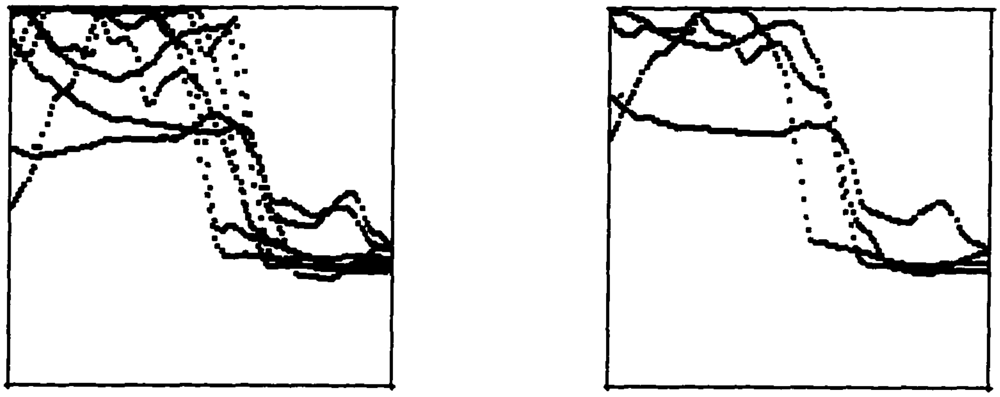


Figure 7-9: A 3-dimensional plot of the magnetics data with shading.

To understand these approximations, we need to go back to the implementation of the approximation process. In this example, the wavelet transform is applied to the *y-magnetics* slices as shown in the (3, 2) tiles of Figure 7-8. Note the XmdvTool implementation of scatterplot matrix reverses the pattern described in Chapter 3. In this case the upper left plot is labelled (1, 1) and the lower right is (4, 4) as shown in Figure 7-12. Among them, the two which contain 1024 and 512 data points are enlarged in Figure 7-10. If we carefully trace the scatterdots, there are exactly eight

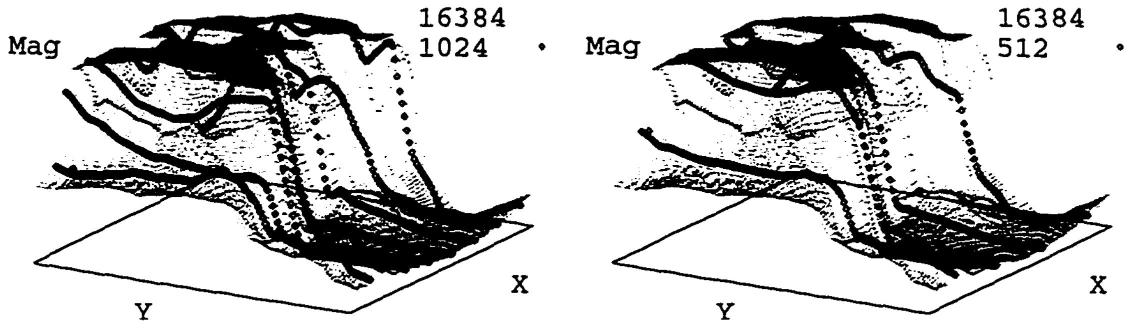


(a) 1024

(b) 512

Figure 7-10: Zoomed displays of the *y-magnetics* view with a) 1024 and b) 512 data points.

curves in Figure 7-10a and four in Figure 7-10b. These are the approximations of the *y-magnetics* slices of the same data depicted in Figure 7-9. Figure 7-11 displays these approximation lines with



(a) 1024

(b) 512

Figure 7-11: The scatterdot background is the original data with 16384 data points. The dark lines are the approximation of the surface with a) 1024 and b) 512 data points.

three-dimensional scatterplots of the original data with 16384 data points. The dark lines are the approximations of the surface with 1024 and 512 data points.

The scatterplot matrix provides a very efficient layout for the user to select data from multi-dimensional multivariate datasets. In Figure 7-12, the data brush covers a subset of the x variate as shown in tile $(1,1)$, the whole y variate in $(2,2)$, the whole magnetics variate in $(3,3)$, and the whole radiometrics variate in $(4,4)$. At this resolution there are four scatterdots in $(1,1)$. A brush movement from one dot to the others implies horizontal brush movements in $(2,1)$, $(3,1)$, and $(4,1)$. These movements result in simultaneous selections of the *y-magnetics* view in $(3,2)$, the *y-radiometrics* view in $(4,2)$, and the *magnetics-radiometrics* view in $(4,3)$. The combination of wavelet brushing and scatterplot matrix allows the user to study slices of the coarse approximations of the magnetics image in $(3,2)$ and the radiometrics image in $(4,2)$ and compare their relationships in $(4,3)$. In our example, the brushed data in $(4,3)$ (in red) clearly indicates that the magnetics data value is not directly dependent on the accumulated radiometrics data value.

Bivariate comparison is the strength of the scatterplot matrix technique, and wavelet brushing makes it possible to apply this visualization technique to very large multidimensional data. It is easier, especially for novices, to understand the three dimensional plots in Figures 7-9 and 7-11, but each of these displays only shows one variate. For scientists who do exploratory data analysis with very large multidimensional multivariate data, the wavelet supported scatterplot matrix provides a

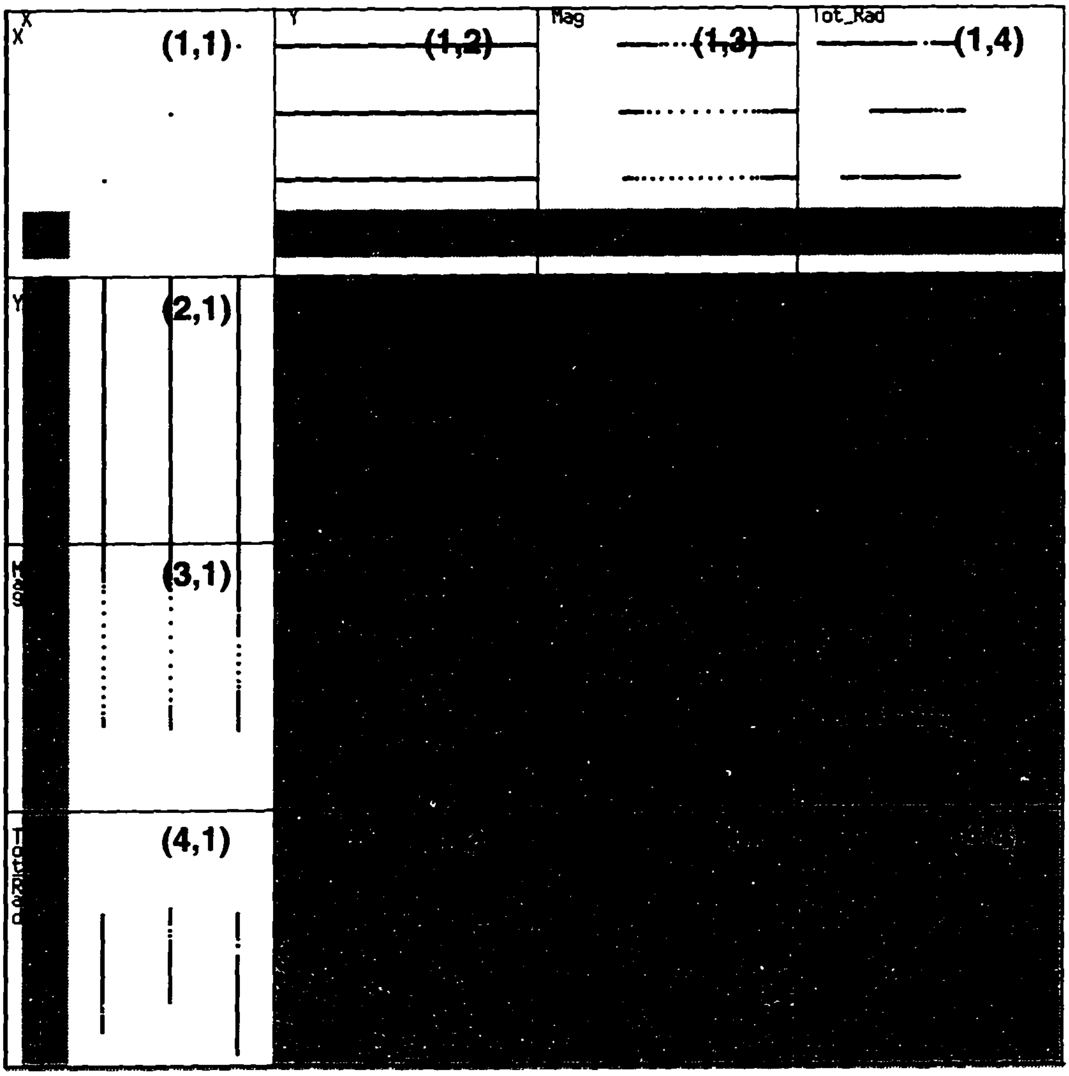


Figure 7-12: An example of multidimensional data visualization. See also Color Plate 5 in Appendix D.

unique and powerful mechanism to highlight the data from *multiple* variates and investigate the *relationships* among them.

Chapter 8

Brushing Techniques for Multidimensional Data

We introduced the notion of ***multiresolution brushing*** to browse very large one dimensional multivariate data in Chapter 7. The idea is to query interesting portions of a very large dataset by brushing a coarse but authentic approximation of the data before the fine resolution data is retrieved. Traditionally, volume visualization faces greater difficulty and requires more computation resources than its lower dimension counterparts. This chapter studies several visualization techniques specially designed to tackle some of these problems. We have implemented a set of volume visualization tools using the *vtk* library [SML96] running on a 133 MHz Linux machine with 32MB of memory. These tools show that very large scientific volume datasets can be accessed and utilized without expensive hardware.

8.1 Introduction

Scientific volume visualization is currently gaining popularity but the cost of rendering has hindered the development of general applications for widespread use. Although visualization researchers have developed new techniques to improve the performance of the underlying hardware [PK96, YRL$^+$96] and software [BPS96, SMK96], more powerful machines with ever improving calibrations are producing larger and larger volume datasets everyday. For example, the Visible Woman dataset produced by the Visible Human Project[1] occupies over 480MB of memory

[1] http://www.nlm.nih.gov/research/visible

in compressed format. We believe that the best way to make this precious information available to the research community at large is to provide low cost versions of the technology so that more researchers can use it.

8.2 Problem and Solutions

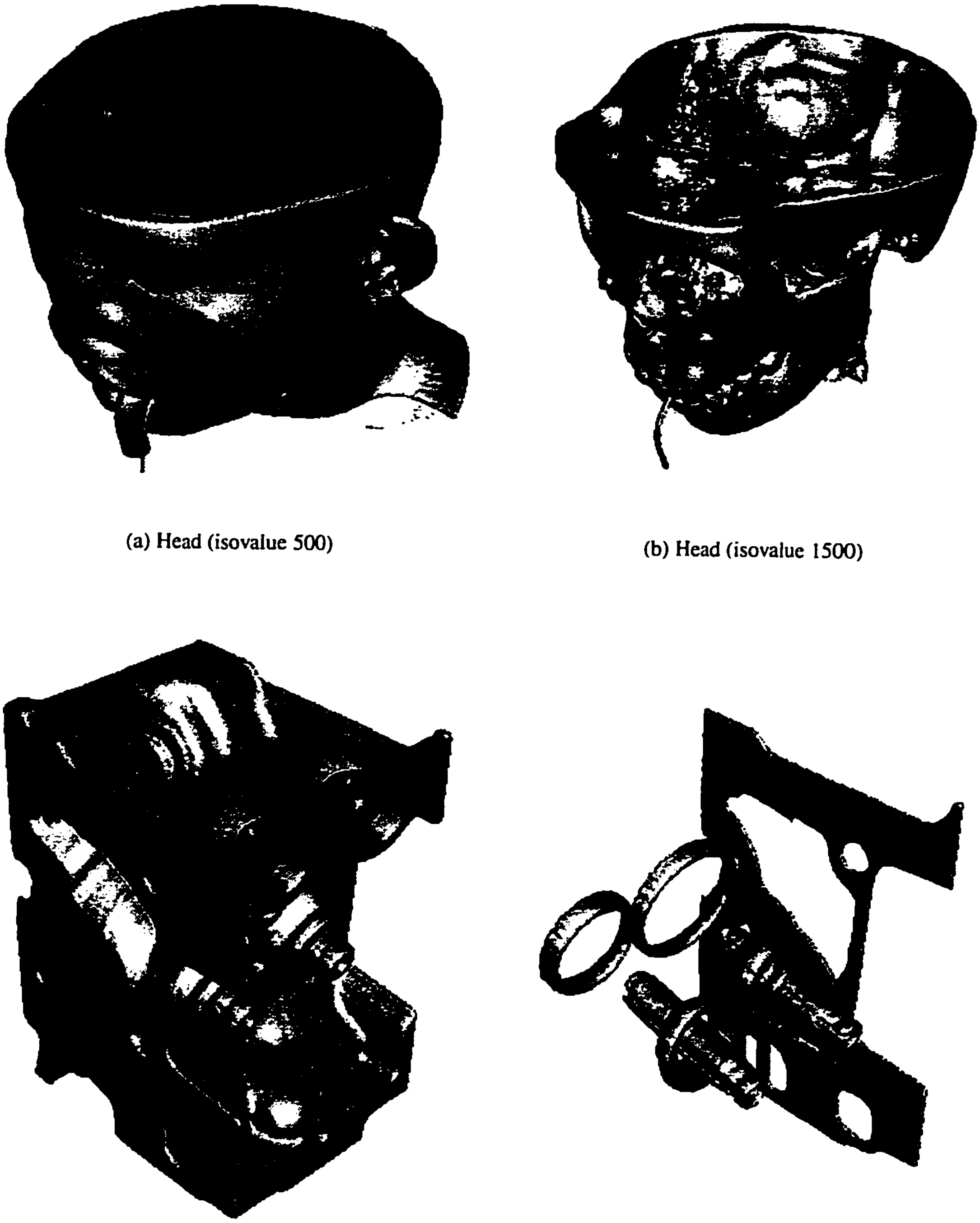
Scientific volume data is large. An ordinary computed tomography (CT) dataset, which contains 128 slice images at 256^2 resolution, occupies more than 16MB of memory (if 16-bit format is used). Two other medical imaging procedures which require similar memory are Magnetic Resonance Imaging (MRI) and ultrasound scanning. Figure 8-1 depicts four isosurface renderings of two CT datasets[2] using the marching cubes algorithm [LC87]. Unfortunately, large amounts of data usually implies high computational costs. Table 8.1 provides characteristics of our test datasets and statistics of the isosurface renderings shown in Figure 8-1. The *Time(s)* column specifies the CPU time of the rendering process including input, marching, and isosurfacing. It is important to realize that the time data presented in this paper is not meant to measure the performance of the rendering process as implemented in the *vtk* library. Rather it is used to reflect the *differences* in performance between using the original large datasets and their coarse approximations.

Dataset	*Resolution*	*Isovalue*	*$\sharp$ of $\triangle$*	*Times(s)*	*Figure*
Head	$256 \times 256 \times 128$	500	482302	278.08	8-1a
		1500	575272	435.82	8-1b
Engine	$256 \times 256 \times 128$	50.5	622196	534.55	8-1c
		200.5	144716	36.92	8-1d

Table 8.1: Characteristics of the datasets and statistics of their isosurface renderings.

Figures 8-1c and 8-1d are two isosurface renderings (with different isovalues) of the same dataset. From Table 8.1, we see that the processing time of Figure 8-1d is much shorter than that of

[2] The CT head dataset is courtesy of Will Schroeder, Ken Martin, and Bill Lorensen of General Electric Corporate R&D Center. The CT engine dataset was obtained by anonymous FTP from Stanford University.



(a) Head (isovalue 500)

(b) Head (isovalue 1500)

(c) Engine (isovalue 50.5)

(d) Engine (isovalue 200.5)

Figure 8-1: Isosurface renderings of the head dataset and the engine dataset.

Figure 8-1c because fewer isosurface triangles are generated using isovalue 200.5. This implies that we may substantially speed up the rendering process if we can reduce the number of data points. The idea was explored by Muraki [Mur92, Mur93] who introduced the use of wavelets to compress three dimensional volume datasets. This approach helps to improve the performance of many three dimensional rendering techniques including the marching cubes algorithm [LC87].

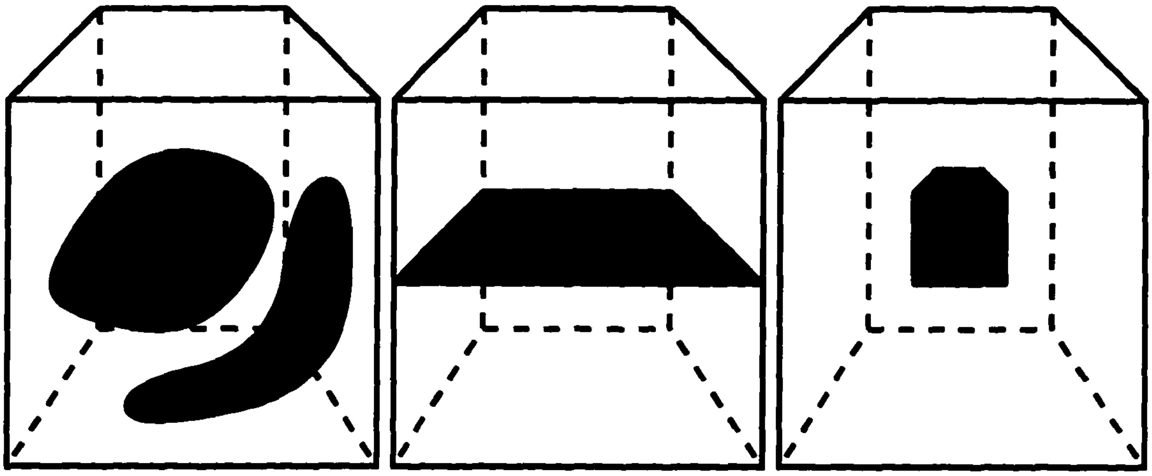
8.2.1 Three Dimensional Brushing

As with other kinds of very large datasets, scientists dealing with volume datasets are often only interested in analyzing subsets of the data. For example, a heart surgeon is more interested in the heart and the blood vessels than the bones and joints of a human body. If we only bring out the most important portions of a large dataset at the highest possible resolution and fill in the rest with coarser data, we can cut down the costs of data rendering substantially and still provide accurate visualization of the important information. This kind of data extraction process can be considered as a form of database query, which we call *data brushing.*

In the heart surgeon example, we in fact identified two important types of three dimensional data extractions: *qualitative* (X-ray density corresponding to the blood vessels) and *spatial* (the heart region). The latter can further be categorized into *planar* and *volume.* Simplified examples of these brushes are depicted in Figure 8-2. Since data brushing is an interactive process, we need to actually see the display of the dataset before we can brush it. Our next step is to provide a coarse but authentic approximation of the very large dataset to guide brushing.

8.2.2 Authenticity of Wavelet Approximations

Muraki [Mur92, Mur93] used Mallat's multiresolution wavelet algorithm [Mal89] to generate a series of fine to coarse approximations of a large volume dataset for rendering. Our goal is to find the coarsest authentic approximation within the series to support brushing. The prevailing method to determine the accuracy of a volume approximation is the human visual test. In the case of a large approximation hierarchy, this means a lengthy rendering of many volume approximations. Because of the potential hardware requirements, this procedure may not be possible for every scientist who



(a) Qualitative (b) Planar (c) Volume

Figure 8-2: Three dimensional brushing of a volume dataset.

wants to use the data. We describe an objective authenticity evaluation mechanism in Chapter 11 which bypasses the requirements of volume rendering and visual comparison, and is still able to identify authentic approximations of a hierarchy.

8.2.3 Translucent Coarse Data

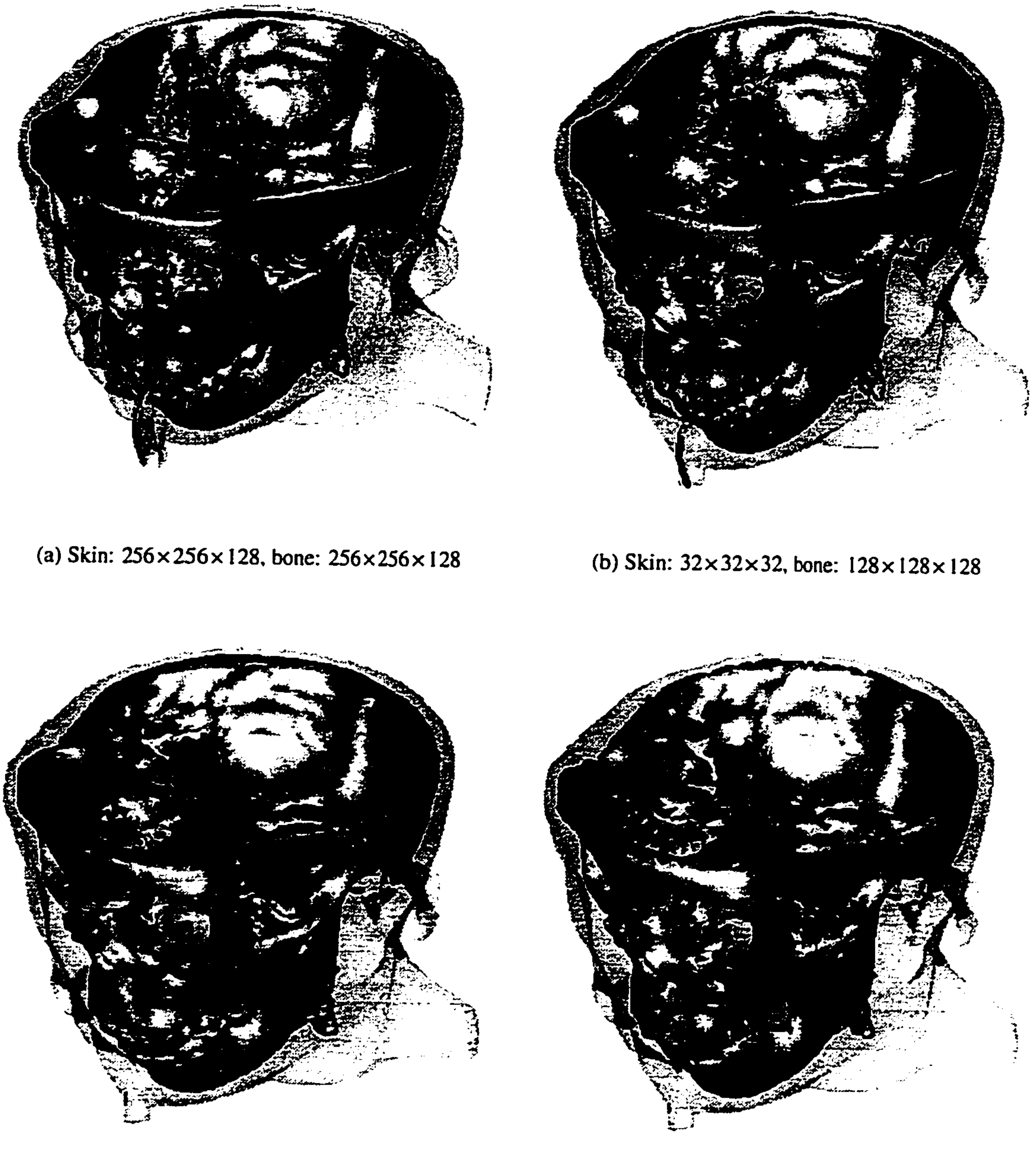
After the interesting data subset of the approximation is identified, our next step is to put both the fine and the coarse data portions together into one visualization. An obvious option to combine two separate data subsets into one display is to make one of them translucent. This can be done by reducing the alpha values of the grey level density. Since a translucent body carries less visible information, it is a logical choice to apply the reduction to the coarser representation instead of the finer one. This translucent display gives a quick but less accurate overview of the data. It helps the viewer to understand the overall structure before finer (but much slower) local detail is brought to the display.

8.3 Examples

We present realistic examples of 3D brushing using the CT head dataset depicted in Figure 8-1. All the following illustrations are rendered using the marching cubes algorithm. Because of heavy memory paging, the full resolution head dataset (256 × 256 × 128) takes up to four hours of clock time to render on a 133 MHz Linux machine with 32MB of memory. By brushing only a partial subset of either the original data or a close approximation, we achieve near real-time responses in almost all cases on the same platform.

8.3.1 Qualitative Brushing

A qualitative brushing process highlights interesting areas according to the values of the data. For example, if the most important part of the figure is the skull, only the data values corresponding to bones and joints are used to generate the isosurfaces. The rest of the data, mainly flesh and skin, can be obtained from a coarser version of the data. Examples of composite displays at different resolutions are shown in Figure 8-3. The translucent regions of these figures show the flesh and skin portion of the body. Figure 8-3a shows the skin and bone structures at original resolutions. Our investigation in Chapter 11 indicates that a Daubechies wavelet with two vanishing moments (i.e., D_4) is a good wavelet candidate to decompose this volume dataset. The translucent skin regions of Figures 8-3b – 8-3d are generated by a D_4 wavelet at 32^3 resolution. For the bone display in Figure 8-3b we apply D_4 to the x and y dimensions of each image plane. The same wavelet decomposition is applied to the dataset twice in Figure 8-3c for the bone display. And in Figure 8-3d, we apply decomposition to the x and y dimensions twice, and to the z dimension once. Table 8.2 gives the characteristics of the datasets (including approximations) and the statistics of their renderings. The contribution of wavelets to the brushing process is obvious: Figure 8-3c takes less than 1% of the CPU time required to process Figure 8-3a. This time difference is translated to over six hours of clock time when using a 133MHz Linux machine with 32MB of memory. The notion of qualitative brushing allows scientists to study particular isovalues of a volume dataset. It is, however, not the end of the story. The front side of the skull blocks all the internal structure behind it. We need more spatially oriented data brushes such as the ones described in the following



(a) Skin: 256×256×128, bone: 256×256×128

(b) Skin: 32×32×32, bone: 128×128×128

(c) Skin: 32×32×32, bone: 64×64×128

(d) Skin: 32×32×32, bone: 64×64×64

Figure 8-3: Qualitative brushing with multiple resolutions.

Skin Resolution	*Bone Resolution*	*♯ of Δ*	*Time(s)*	*Figure*
256 × 256 × 128	256 × 256 × 128	1057559	845.20	8-3a
32 × 32 × 32	128 × 128 × 128	215584	54.18	8-3b
	64 × 64 × 128	82580	8.42	8-3c
	64 × 64 × 64	51016	3.54	8-3d

Table 8.2: Statistics of the isosurface renderings with qualitative brushing.

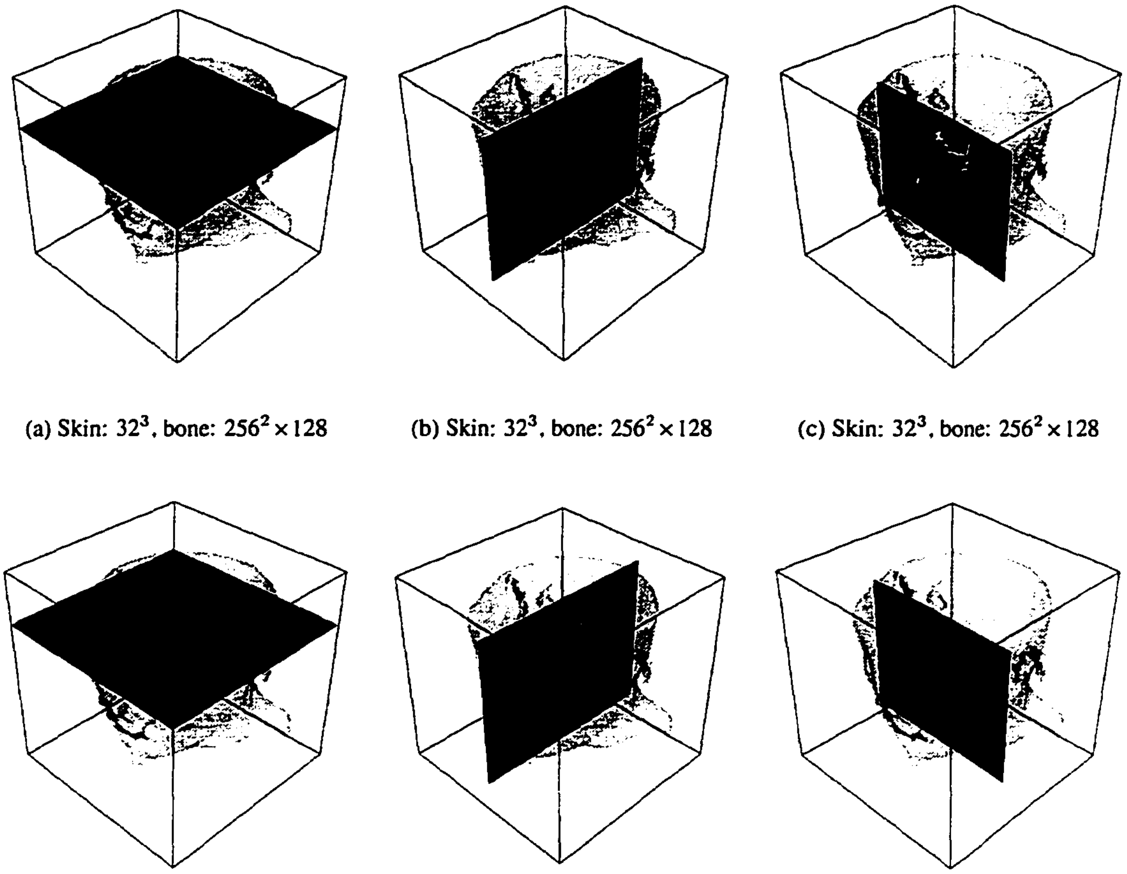
sections to further study the local structure of the human skull.

8.3.2 Planar Brushing

A planar brushing process highlights cross sections of a volume dataset. There are three kinds of planar brushing: axial, sagittal, and coronal. Each of them indicates a different orientation of a plane through a volume dataset. Figure 8-4 shows examples of the three planar brushings using the head dataset. Like the previous brushing examples, the translucent skin portion is generated by a D_4 wavelet at 32^3 resolution. Since the data brush is only a slice of the dataset, we can afford to use the finer (Figures 8-4b, 8-4d, and 8-4f) or even the finest (Figures 8-4a, 8-4c, and 8-4e) data to show the local details of the human head. Since we have already shown the effects of reducing the skin resolution in Figure 8-3, we do not display the full resolution rendering here. However, the processing time of using full resolution skin data are included in Table 8.3 for comparison purposes. The results show that all renderings depicted in Figure 8-4, even with the finest data brushes, can be processed in real-time using a 133MHz Linux machine.

8.3.3 Volume Brushing

A volume brush is a high resolution sub-volume within a coarse volume dataset. Its goal is to brush smaller regions of data values which cannot be achieved solely by a qualitative or a planar brush. For example, the jaw bone area of the skull, which is totally blocked by the forehead in Figure 8-3, can easily be seen in Figure 8-5. Once again, D_4 is used to decompose the dataset in these figures. From Table 8.2 in Section 8.3.1 we realize that it is still too expensive to render the brushed bone structure at full resolution. We can, however, render a coarse resolution bone structure and then



(a) Skin: 32^3, bone: $256^2 \times 128$

(b) Skin: 32^3, bone: $256^2 \times 128$

(c) Skin: 32^3, bone: $256^2 \times 128$

(d) Skin: 32^3, bone: $128^2 \times 128$

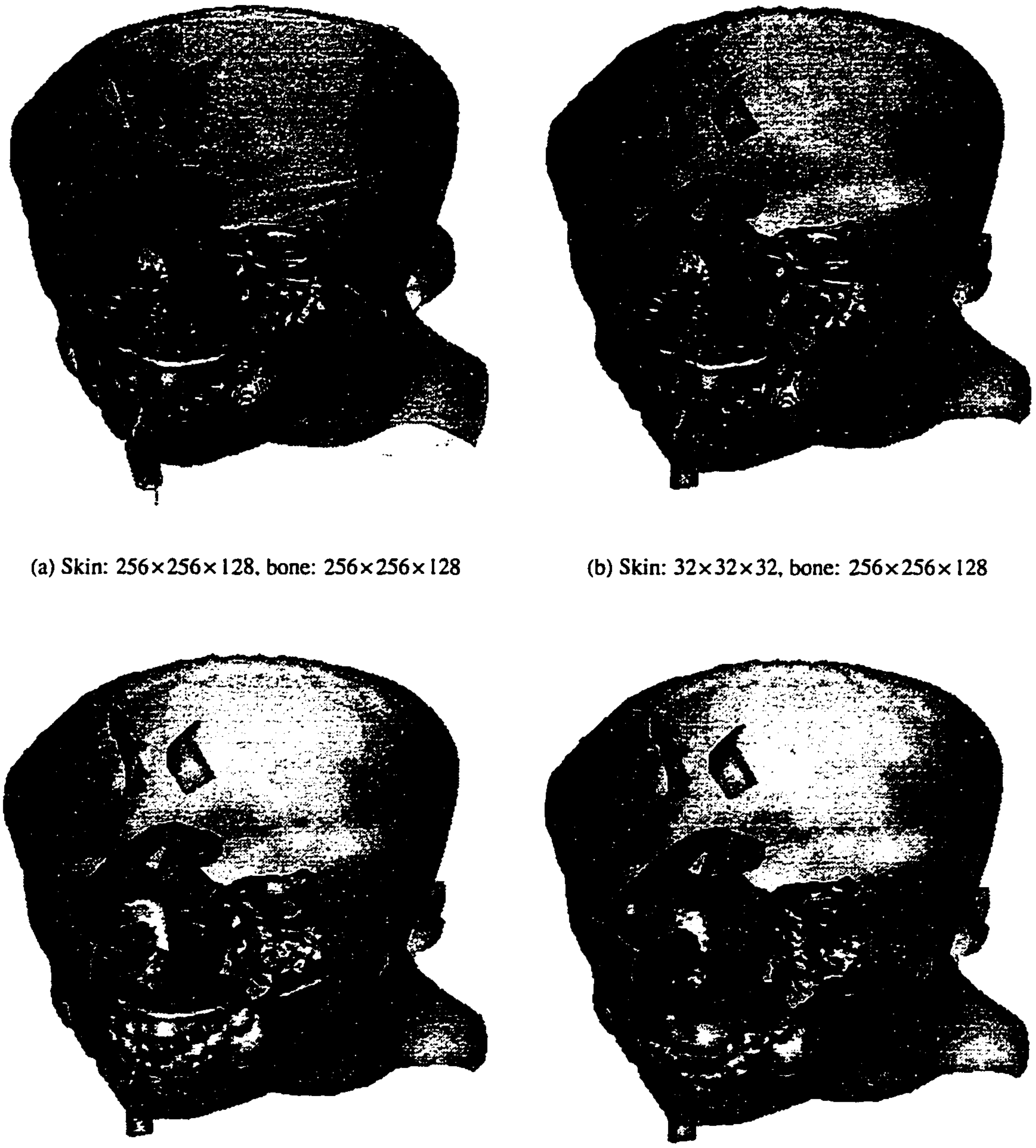
(e) Skin: 32^3, bone: $128^2 \times 128$

(f) Skin: 32^3, bone: $128^2 \times 128$

Figure 8-4: Planar brushing with multiple brush resolutions.

Skin Resolution	*Brush Resolution*	*Brush Size*	*Time(s)*	*Figure*
256 x 256 x 128	256 x 256 x 128	55696	555.22	N/A
		22134	548.96	N/A
		15908	540.94	N/A
32 x 32 x 32	256 x 256 x 128	55696	2.06	8-4a
		22134	1.84	8-4b
		15908	1.57	8-4c
	128 x 128 x 128	13924	0.83	8-4d
		11067	0.75	8-4e
		7954	0.72	8-4f

Table 8.3: Statistics of the isosurface renderings with planar brushing.



(a) Skin: 256×256×128, bone: 256×256×128

(b) Skin: 32×32×32, bone: 256×256×128

(c) Skin: 32×32×32, bone: 128×128×128

(d) Skin: 32×32×32, bone: 64×64×128

Figure 8-5: Volume brushing of the jaw bone area.

retrieve portions of the finest data using a volume brush. The results in Table 8.4 show that it is possible to access the head data at its finest resolution in near real-time. The rendering as well as brushing together may take minutes, but we can access any portion of the very large head dataset at its finest resolution using only an ordinary desktop computer.

Skin Resolution	*Bone Resolution*	$\sharp$ *of* Δ	*Time(s)*	*Figure*
256 x 256 x 128	256 x 256 x 128	734606	555.02	8-5a
32 x 32 x 32	256 x 256 x 128	123520	27.32	8-5b
	128 x 128 x 128	50160	5.78	8-5c
	64 x 64 x 128	23404	1.75	8-5d

Table 8.4: Statistics of the isosurface renderings with volume brushing.

8.4 Discussion

The field of volume visualization research will benefit greatly from putting the data into as many researchers' hands as possible. The rendering hardware previously described in the literature [BPS96, PK96, SMK96, YRL+96] costs more than many scientists can afford. We have shown that with an average-equipped Linux machine, scientists can obtain precious information from the same large datasets described in visualization literature with near real-time response. Many of the brushing ideas presented in this paper are conceptually simple and easy to implement. Our software implementation is based on a public domain visualization library, *vtk*, which is compiled using the public domain version of OpenGL. Based on experience with our experiments, we believe the notion of brushing will play an important role in making very large scientific datasets available to the research community at large.

Chapter 9

Dual Multiresolution Extension

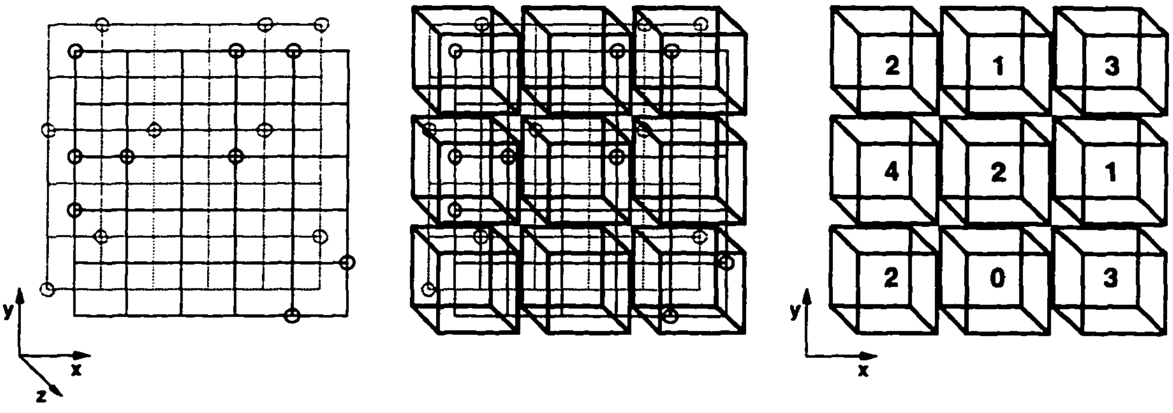
In this chapter, we consider how a wavelet based data hierarchy can further be enhanced to include a second data reduction algorithm. This extension results in a new multiresolution visualization design which allows a user to control the physical data resolution as well as the logical display resolution of multivariate data. Perhaps more importantly, this chapter shows that the multiresolution visualization technique developed in this thesis is very versatile and robust. This chapter is based on [WCB96].

9.1 Dual Multiresolution Exploration

Two data reduction approaches are applied to reduce the size of the data and create a fine to coarse data hierarchy. They are norm-based projection and wavelet transform. Our system, which is based on HyperSlice [vWvL93], combines these two processes and provides a *dual* multiresolution visualization environment to improve browsing and navigation capabilities [Cra96, WCB96].

9.1.1 Display Resolution Through Norm Projection

Norm projections are based on $\{\mathbb{R}^m \rightarrow \mathbb{R}^n : m > n\}$. The goal is to represent a high dimensional dataset by a lower dimensional display. Our definition of norm projection is stronger and more powerful than the similar term which simply describes the *view point projections* of different dimensions in the scatterplot matrix and prosection matrix [STDS95] techniques. In our design, data from higher rank spaces are projected by different *norms* into data of lower rank spaces. By applying the projection to the data repeatedly, we generate a data hierarchy with multiple *display* resolutions. In Figure 9-1a, there are 72 grid points in $\mathbb{R}^3$; suppose that each of the grid points rep-



(a) (b) (c)

Figure 9-1: a) 72 sampling points in $\mathbb{R}^3$. b) Norm projection along the z direction. c) The projected data in $\mathbb{R}^2$ with 9 data points.

resented by a circle has the value 1 and all others grid points have the value 0. A norm projection, $\mathbb{R}^3 \rightarrow \mathbb{R}^2$, is applied to the data along the z-direction in Figure 9-1b. A traditional view projection simply maps data points on top of each other if they project to the same display point, resulting in the loss of valuable information. In our approach, we can apply any desired summarizing operation to the data points that map to the same display location. Figure 9-1c shows the result of applying the L^1 norm to the projected data. Depending on the nature and characteristics of the data as well as the usage of the projection, the L^1 norm can be substituted with the L^2 norm or other aggregate functions such as the maximum, the minimum, the sum or the average.

In a way, the definition of our norm projection shares similarities with the definition of the *attribute projection*, π, in *relational databases*. By manipulating different norms, we can quickly obtain some basic facts about the data. Figure 9-2 shows bivariate scatterplots in three consecutive display resolutions (8192 initially) using the *maximum* aggregate function at the top (Figures 9-2a – 9-2c), and the *average* aggregate function at the bottom (Figures 9-2d – 9-2f). The upper left quadrants of all six plots contain relatively high intensity *spots*, indicating either a sparse or highly centralized population with higher values. The lower left quadrant of Figure 9-2b shows similar bright spots but the intensity fades away in Figure 9-2e, indicating a high population with evenly distributed data values. The cloud patterns between the two right plots further indicate that the lower

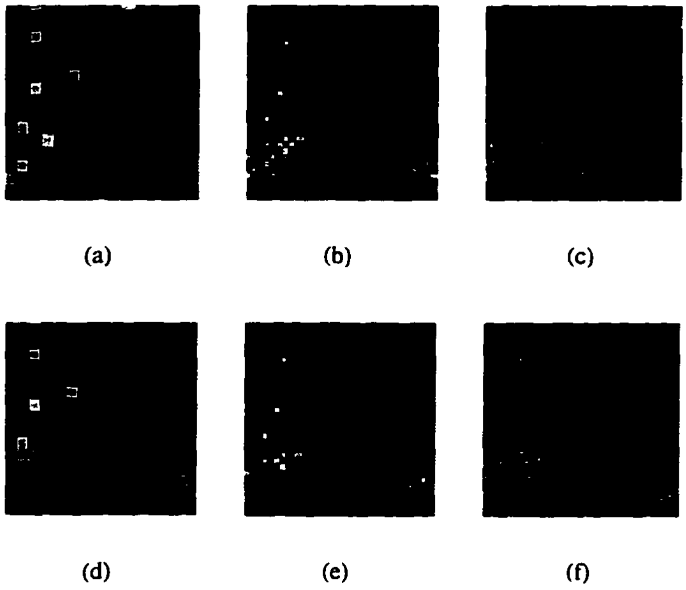


Figure 9-2: Top: Three consecutive display resolutions of a bivariate scatterplot using the *maximum* aggregate function. Bottom: The same three plots using the *average* aggregate function.

right quadrant has relatively sparse population with relatively low data values.

A standard scatterplot does not show data density within a pixel and for this data we can gain only limited insight – the two variates do not seem dependent. The use of norm projection allows a user to gain more insight such as data locality and demography quickly and accurately. However, when data size grows, conventional aggregate functions eventually fail to show the locality of the data *trends*. We need scaling functions which are localized in space and frequency – *wavelets* satisfy these characteristics.

9.1.2 Data Resolution Through Wavelet Decomposition

We show an example of combining wavelet decomposition and norm projection using the same dataset. Figure 9-3 depicts bivariate scatterplots of five resolutions of a dataset with 16384 points. The original data is plotted, together with four coarser resolutions (8192, 4096, 2048, and 1024) generated by a Haar wavelet. The aggregate function used is *count*, which means that a high display intensity implies a high population density. The wavelet approximations visually reflect the trend of the data distribution. More importantly, we can now navigate through a dataset that is only $\frac{1}{16}$ (or

(a) 16384 (b) 8192 (c) 4096 (d) 2048 (e) 1024

Figure 9-3: Scatterplots of the original data with a) 16384 points, wavelet approximations with b) 8192, c) 4096, d) 2048 and e)1024 points. The *count* aggregate function is applied in all graphs to show the data distribution.

even smaller) of the original size. For pixel-based graphics applications, this results in a substantial speed-up in terms of computation time and memory.

9.1.3 Display Resolution versus Data Resolution

A major drawback of orthogonal wavelets is that the reduction rate is fixed at 2^{-n} where n is the number of data dimensions. Conventional aggregate functions, however, can generate more flexible resolutions. Data exploration using wavelet approximations depends upon (both objectively and subjectively) the shapes and the trends being preserved in the coarse approximations. For data mining, conventional aggregate functions are more natural and easier to understand and manipulate. We do not change the physical contents of the data during norm projections, only the visualization. Wavelets, on the other hand, physically replace the data with smoother values.

Our system puts these two mechanisms into one powerful data visualization tool. As depicted in Figure 9-4, a data hierarchy with multiple data resolutions is first generated by wavelet decompositions. For data mining, norm projections are applied to the data of the selected data resolution and data with multiple display resolutions are generated. Wavelet approximations provide a coarse view of a large dataset, which may not otherwise be able to be displayed on screen. Once the interesting patterns are located from this resolution of the data, the user can go to a higher data resolution to do the data mining using norm projections.

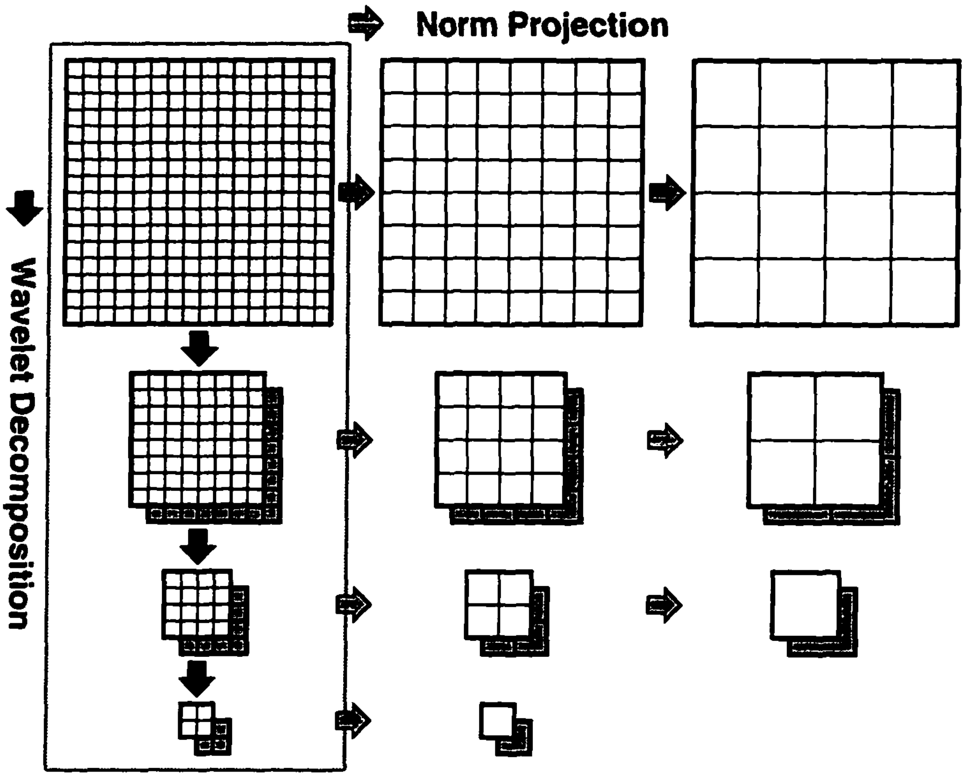


Figure 9-4: Two hierarchies are generated from a very large data through norm projections and wavelet decompositions.

9.2 HyperSlice Enhancements

Nearly all matrix-based visualization representations, including HyperSlice, duplicate mirror images (and/or movements) of the upper left half to the lower right half of the matrix. We, however, believe that this precious space (almost one half of the display area) can be used to provide another dimension of information. In additional to all the standard features to define a HyperSlice representation, our system also provides the *error* information generated by the wavelet decompositions. The approach of using wavelet details as a means of *data authenticity analysis* is discussed in Chapter 5 and [WB95a].

In Figure 9-5, a 9-variate time-series dataset with 16384 data points is plotted with our system. The multivariate data comes from the GISP2 [MMM$^+$93] project which investigates the timing and forcing of climate changes of the atmosphere over the last 200,000 years. A wavelet transform is applied to the data and the finest wavelet approximation is depicted in Figure 9-5. Since each data

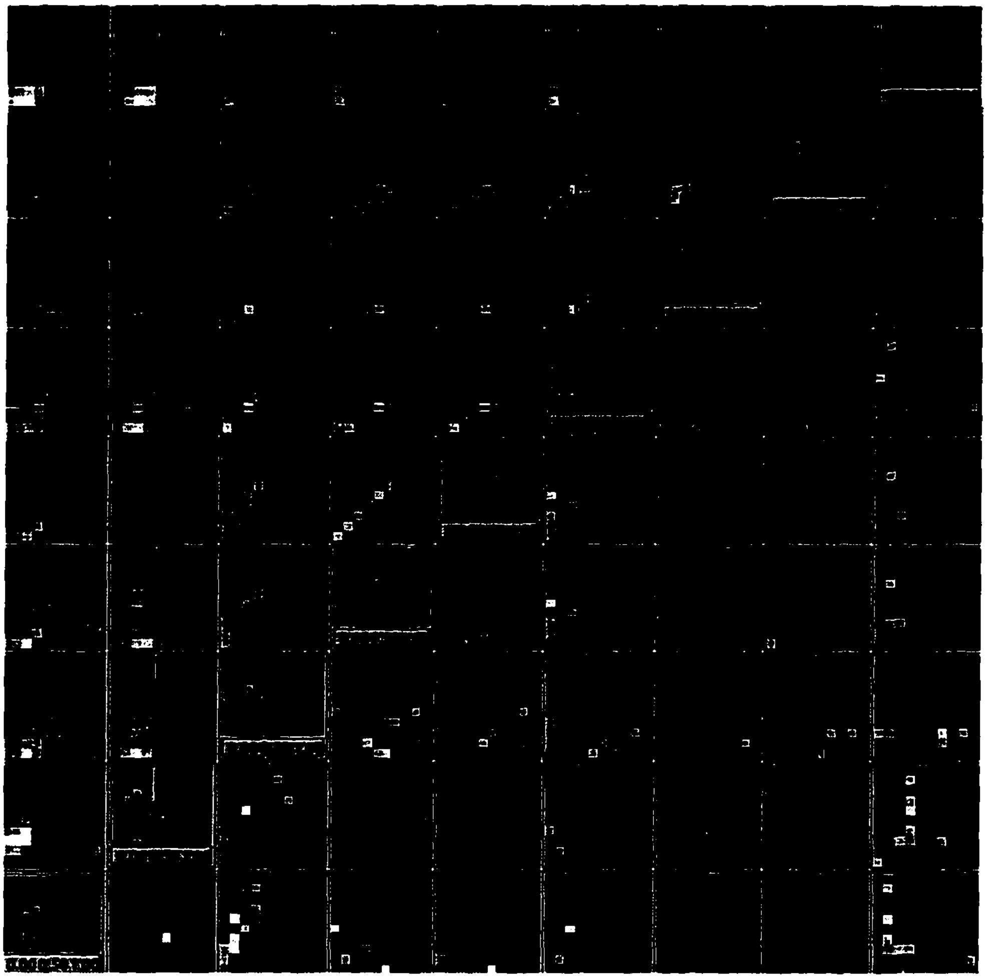
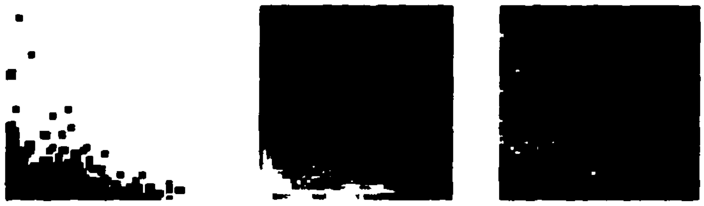


Figure 9-5: A multivariate data in HyperSlice representation.

point consists of 9 variates (Time, NO_3, SO_4, Cl, Na, K, Ca, Mg, and NH_4), Figure 9-5 represents well over one million data values. The upper left half of the matrix uses the aggregate function *count* as the norm projection norm. A pixel block with high intensity implies a dense population of data points. In the lower right half of the matrix where the *maximum* aggregate function is used, a pixel block with high intensity means at least one of the approximation values in that block has large error, which may need further investigation at finer resolutions.

We argue that the error information is crucial for data exploration, especially in a multiresolution environment. It is a fact that data loss is unavoidable during wavelet decomposition. These losses, however, can be presented together with the approximations. Figure 9-6a shows a standard bivariate scatterplot (NH_4 versus Cl) without applying the norm projection; a coarse wavelet approximation



(a) Standard (b) Approximation (c) Error

Figure 9-6: a) A bivariate scatterplot without norm projection. b) A coarse wavelet approximation. c) The error information of the corresponding approximation.

is shown in Figure 9-6b and the error information is shown in Figure 9-6c. The intensity variations in the lower left portion of Figure 9-6b provide distribution information that is entirely lost in the standard scatterplot in Figure 9-6a. Figure 9-6b also clearly shows that the wavelet smoothes away most of the isolated points in the upper left quadrant. These data losses, however, are reflected accurately in Figure 9-6c, which is displayed in the corresponding tile at the lower right half of the HyperSlice. The error display reveals information which does not show up in the approximation plot.

9.3 A Simple Application

In this section, we present an application using a publicly accessible non-scientific dataset[1] containing information about faculty at U.S. universities. We have selected six variates from this dataset – the number of faculty at each faculty rank (i.e., full, associate, and assistant) and the average salaries at each rank. The data is displayed in Figure 9-7 with 1024 (32^2) pixel blocks used in each display tile. The display data, which is the second approximation generated from the wavelet transforms, represents well over 36K data values. The points of interest and the ranges of the display of the tiles are indicated in the corresponding diagonal tiles. We use the matrix coordinate system defined for HyperSlice in Section 3.3.2. The diagonal tiles have the coordinates $\{(i, i), i \in \mathbb{Z}^+\}$ with the lower left tile as the origin, i.e., $(1, 1)$.

[1] http://lib.stat.cmu.edu/datasets/colleges/aaup.data

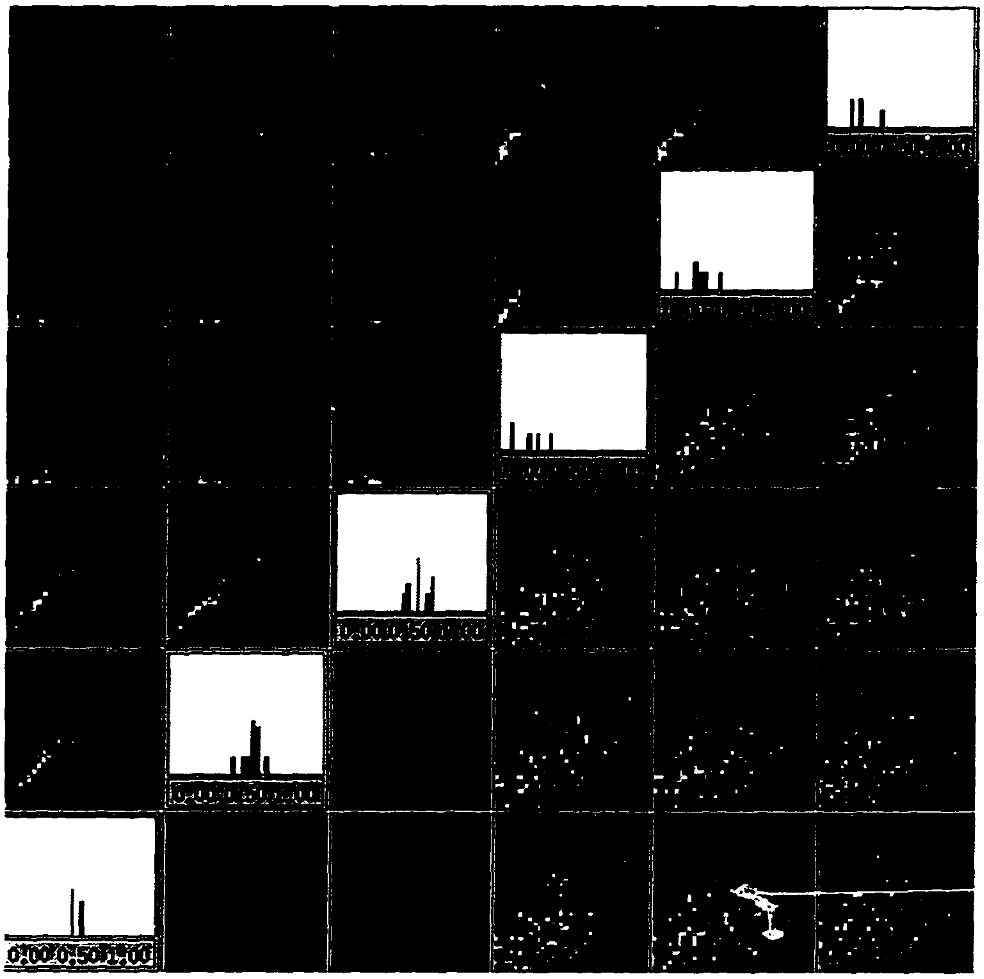


Figure 9-7: A fine projection of a coarse approximation of the faculty salary dataset.

We notice that the overall pixel intensity of the (1, 2), (1, 3), and (2, 3) tiles generated by the *average* norm in Figure 9-7 are much lower than the rest of the tiles in the lower right half of the matrix. This indicates that the approximations depicted in the (2, 1), (3, 1), and (3, 2) tiles (which are the pairwise scatterplots of the salary figures of full, associate, and assistant professors) are more accurate representations than the rest of the approximations in the upper left half of the matrix. Suppose we want to see the distribution of the error in order to show that all (i.e., not just some of) the error values in the mentioned tiles are small. We decrease the display resolution and apply the *maximum* norm projection. As we can see in Figure 9-8, the pixel intensity of the (1, 2), (1, 3), and (2, 3) tiles stays almost the same as in Figure 9-7. This shows that *all* the error data values of the three tiles are indeed very low.

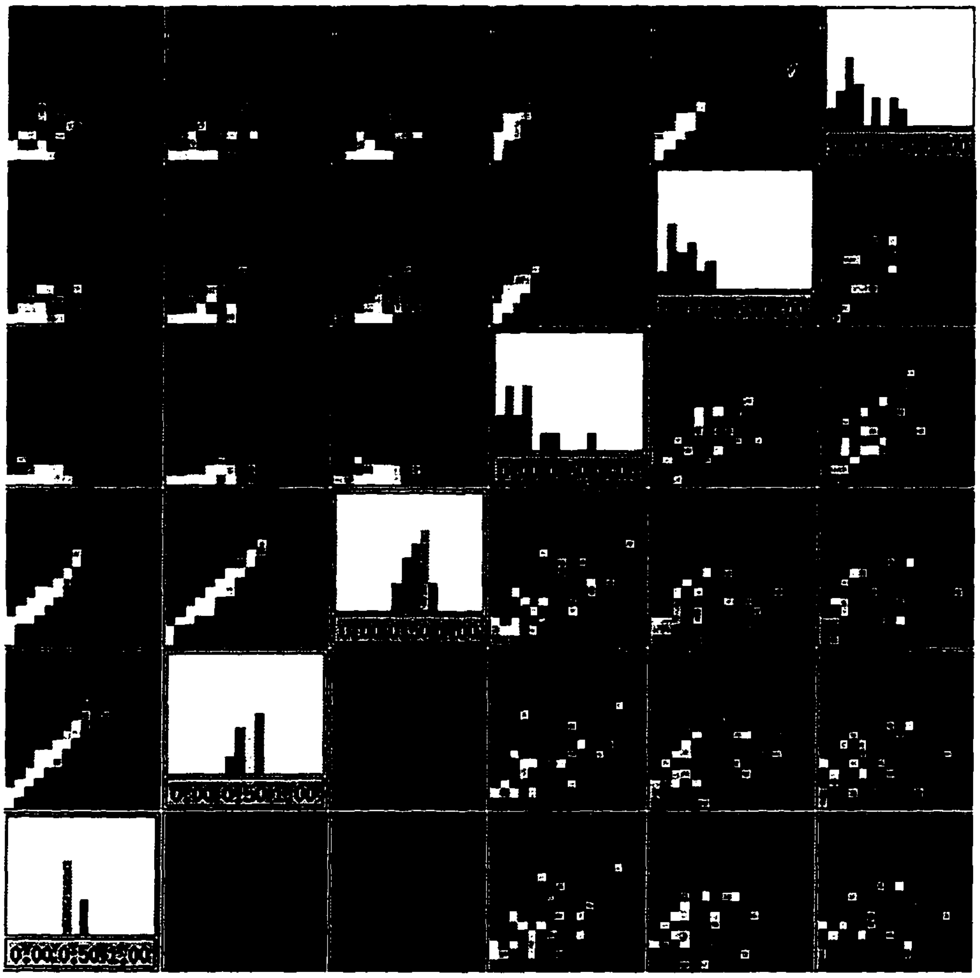


Figure 9-8: A coarse projection of a coarse approximation of the faculty salary dataset.

We now turn our attention to study the relationship between the number of full professors versus their average salaries shown in the (4, 1) tile in Figure 9-8. We can see a group of very high intensity pixel blocks around the lower left corner. This indicates that a substantial number of universities employ a very small number of full professors with relatively low salaries. Since most of the data pixels stay in the lower left corner of the display tile, we redefine the display window of the first variate (average salaries of full professors) to cover the first half, as shown in the (1, 1) tile of Figure 9-9. We also increase the data resolution to 1024 data points in each display tile to obtain finer details of the data. From the (4, 1) tile of Figure 9-9, we can get a better view of the ;henomenon. This is indicated by the horizontal white pixel blocks at the bottom of the (4, 1) tile. This is further evident in the (4, 1) tile of Figure 9-10, which shows the data at fine *data* and *display* resolutions.

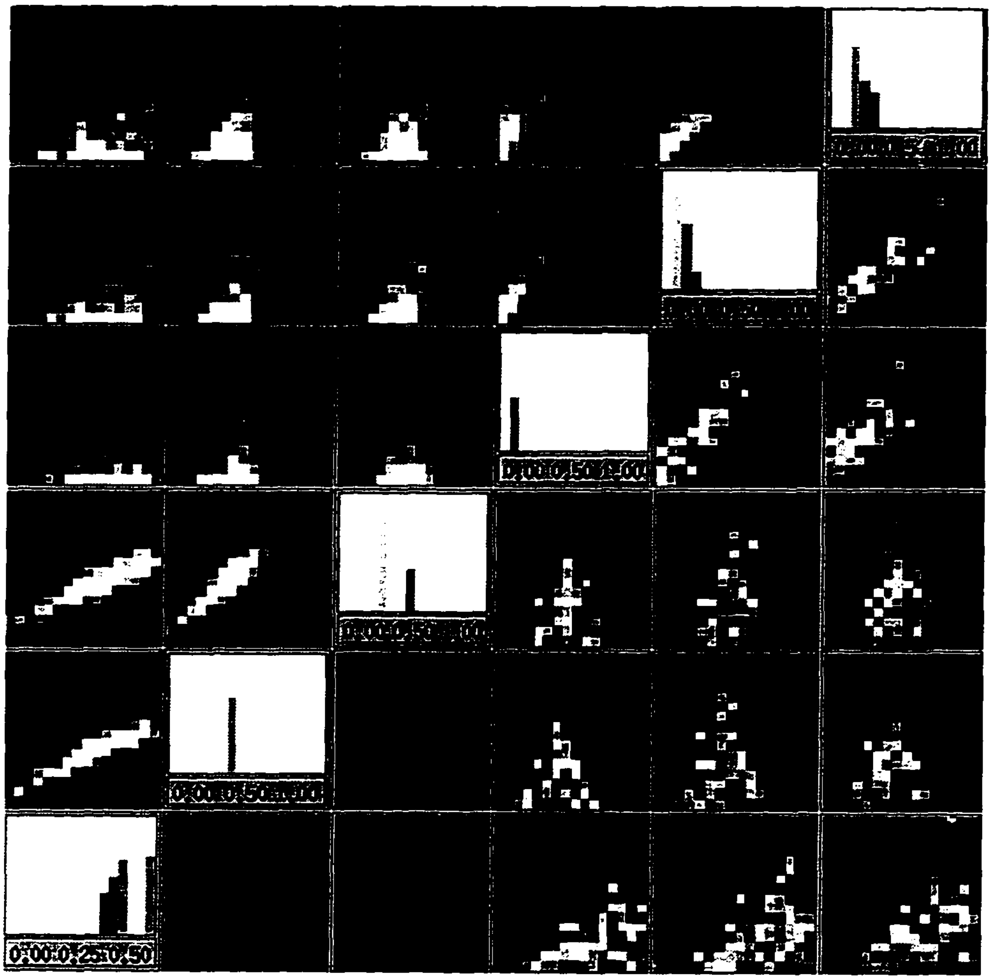


Figure 9-9: A coarse projection of a fine approximation of the faculty salary dataset.

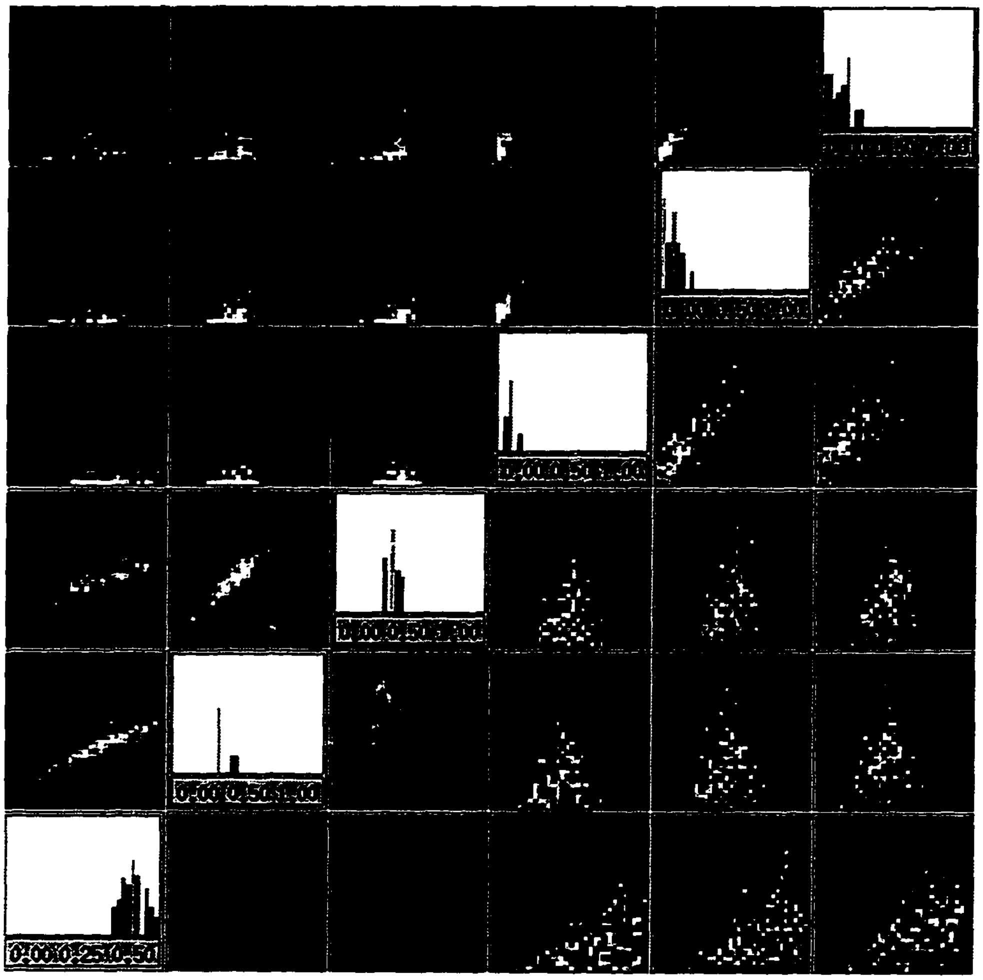


Figure 9-10: A fine projection of a fine approximation of the faculty salary dataset.

Chapter 10

Multidimensional Scaling

The multiresolution techniques described so far are sufficiently general that they can be applied to a variety of multidimensional data. However, these techniques do not address the high variate problem of many multivariate datasets. In this chapter, we suggest another solution designed to reduce the number of variates of a multivariate dataset in a multiresolution fashion.

10.1 Low Dimensional Data Overview

We present an efficient visualization approach to support multivariate data exploration through a simple but effective low dimensional data overview based on *metric scaling* [CC94, Dav83]. This overview can be used to enhance multidimensional data brushing, or arrange the layout of other conventional multivariate visualization techniques. Some of the underlying design concepts have been applied in various visualization tools. Keim et al. [KK94, KKS93] define a *distance function* as a metric to show the *relevance factor* of individual variates of a dataset in *VisDB*. Ward and Bentley [BW96] use multidimensional scaling to *animate* multidimensional datasets in *Mavis*. Hurley et al. use principal components to analyze data with *motion graphics* in *Data Viewer* [HB90] and *XGobi* [CCH95].

We demonstrate our approach using a publicly accessible automobile dataset[1], which contains information about thirty-eight 1978-79 model automobiles including miles per gallon, weight, drive ratio, horsepower, displacement, and number of cylinders. An example using a larger dataset with 329 records is described in Section 10.3.2. Suppose the data has *dissimilarities*, δ_{rs}, measured

[1] http://lib.stat.cmu.edu/DASL/Stories/ClusteringCars.html

between all pairs of automobiles in the 6 dimensional *variate* space. The dissimilarity between two station wagons is expected to be much smaller than the one between a compact and a full size sedan. A graph configuration of 38 vertices, in which the r^{th} vertex represents the r^{th} automobile, is sought in a d dimensional display space such that the distances, d_{rs}, between all pairs of vertices *match* the corresponding dissimilarities, δ_{rs}, in variate space. This configuration is called a low dimensional data overview. Figure 10-1 shows an example of such a graph configuration of the automobile

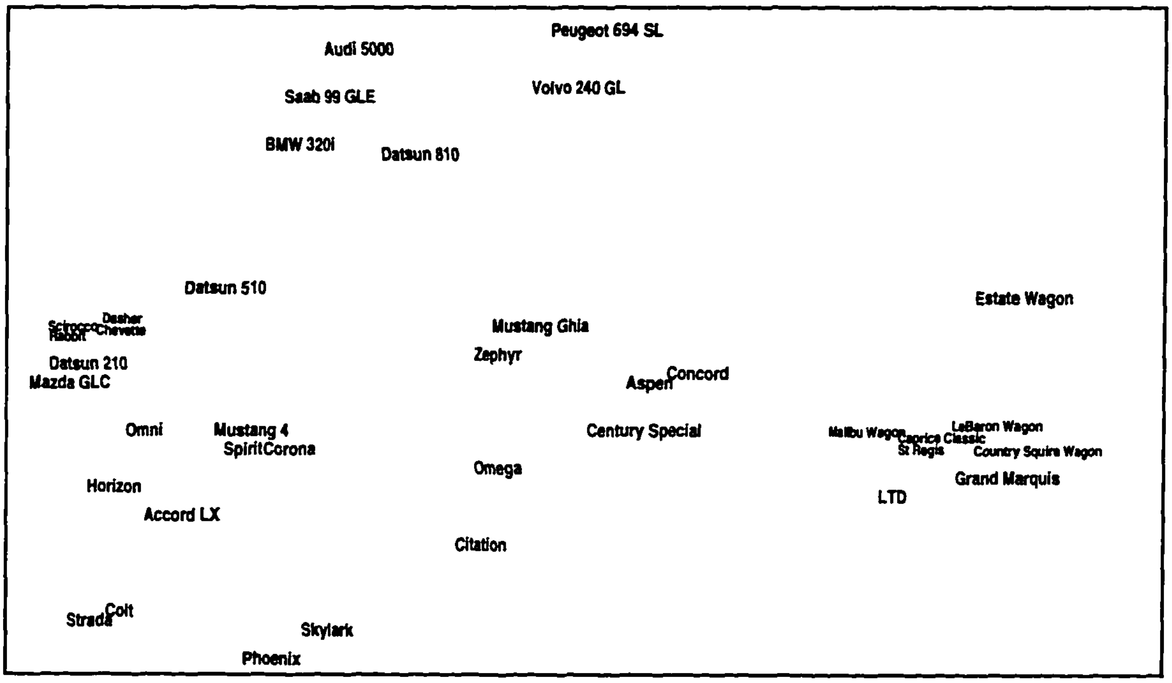


Figure 10-1: A two dimensional display of the six variate automobile dataset.

dataset in a two dimensional space. A quick look at the figure reveals that the full size sedans and wagons are located at the right hand side, the compacts and subcompacts are at the left hand side, the upscale medium size sedans are at the top, and the rest are scattered around the middle of the display. The graph successfully highlights all the major clusters, which reflect the dissimilarities among the six variates of the data.

This example is effective partly because common knowledge is sufficient to identify the nature of the clusters. When there is no obvious meaning coming out of the data overview, further analysis of the original data through other means is necessary. The data overview can then be used to guide the exploration of the local details.

This chapter describes the construction of such a low dimensional display of multivariate data through the use of principal components [Jac91]. The strengths and weaknesses of the approach as compared to other multivariate visualization techniques such as scatterplot matrix are also investigated. An effective visualization environment can be achieved by combining different techniques.

10.2 Metric Scaling

In our discussion, we narrow our data choices to *quantitative* data which includes both *interval* and *ratio* data. Suppose we have a set of n records with v variates and dissimilarities, δ_{rs}, measured between all pairs of records in a v dimensional space. We want to configure a graph of n vertices in a d dimensional display space such that each vertex represents one record and the distances, d_{rs}, measured between all pairs of vertices in display space match δ_{rs} in variate space as closely as possible. This graph configuration problem falls into the broad research area of *metric scaling* studied mostly by mathematical psychologists. The goal is to determine the dissimilarities between all pairs of records by Euclidean distances in v space, and then use them to compute the Euclidean coordinates of the n vertices in the d dimensional display space. Figure 10-1 shows a low dimensional data overview of the 38 automobiles in two dimensional Euclidean space.

10.2.1 Data Dissimilarity Measurement

The first step is to determine the dissimilarities between all pairs of input records, using the Euclidean distance in v space. The dissimilarity, δ_{rs}, between records r and s is given by

$$\delta_{rs} = \sqrt{\sum_{i=1}^{v} (x_{ri} - x_{si})^2}.$$

A dataset with n records generates an $n \times n$ real symmetric dissimilarity matrix. Each element of this matrix contains the dissimilarity, δ_{rs}, between records r and s of the original data. For example, given a dataset with five variates and six records as shown in Figure 10-2a, the Euclidean metric is applied to the data and the result is a 6×6 dissimilarity table, such as shown in Figure 10-2b.

	V1	V2	V3	V4	V5
X1					
X2					
X3					
X4					
X5					
X6					

(a)

	X1	X2	X3	X4	X5	X6
X1	0	δ12	δ13	δ14	δ15	δ16
X2	δ12	0	δ23	δ24	δ25	δ26
X3	δ13	δ23	0	δ34	δ35	δ36
X4	δ14	δ24	δ34	0	δ45	δ46
X5	δ15	δ25	δ35	δ45	0	δ56
X6	δ16	δ26	δ36	δ46	δ56	0

(b)

Figure 10-2: a) A dataset with five variates and six records. b) A dissimilarity matrix of the dataset.

10.2.2 Recovery of Coordinates

Using the approach of principal components, we can represent the data as points in a p dimensional space where $p \leq n$. We create an inner product matrix from the dissimilarities δ_{rs} in variate space, and find its eigenvalues $\lambda_1, \ldots \lambda_p$ and the corresponding eigenvectors to yield the Euclidean coordinates of the n vertices in the p dimensional space[2] (See Appendix C for more details.)

If the eigenvalues are sorted in descending order, i.e., $\lambda_1 > \lambda_2 > \ldots > \lambda_p$, the first principal component associated with λ_1 is more important than the second component, which in turn is more important than the third and so on. (See [Jac91] for details.) The distance, Δ_{rs}, between vertices r and s is given by

$$\Delta_{rs}^2 = \sum_{i=1}^{p} \lambda_i (x_{ri} - x_{si})^2$$

where x is defined in Appendix C. A smaller eigenvalue contributes much less weight to the distance d_{rs}, so these smaller eigenvalues can be truncated with less error. In most cases, the first two to three components can approximate the data very well. Suppose we select the d most significant eigenvalues to display the data overview, the degree of accuracy of the approximation can then be

[2] It is important to realize that the eigenvalue-based principal components are not the only approach to do scaling. Other popular methods include Monte Carlo scaling and Least Squares scaling [CC94].

measured by

$$\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=i}^{p} \lambda_i}.$$

The two dimensional graph of the automobile dataset depicted in Figure 10-1 has a degree of accuracy of 94.61%. It reaches 97.37% if the third principal component is included.

10.3 Strengths and Weaknesses

We investigate the strengths and weaknesses of the low dimensional data overview, and compare it to other multivariate visualization techniques including scatterplot matrix and parallel coordinates. All the figures of scatterplot matrix and parallel coordinates were generated by an enhanced version of XmdvTool [MW95, War94, WB96a].

10.3.1 Data Clustering

Both scatterplot matrix and parallel coordinates are very flexible multivariate visualization techniques which perform well in a wide variety of visualization situations [WB97a]. With the addition of high dimensional brushing [War94, MW95, WB96a], both techniques enable clustering analysis in a limited form. Figure 10-3 depicts the same automobile dataset shown in Figure 10-1 using the low dimensional data overview, scatterplot matrix, and parallel coordinates. The data in red represents the information for the six upscale medium size cars. We notice that the red dots in Figure 10-3a are all by themselves without any blue dots close by. The same data cluster is not so obvious in Figure 10-3c where the red polylines are embedded among the blue polylines. It is very difficult to spot the data cluster without brushing, and it is not clear how easy it would be to figure out how to determine the right brush to use. The situation is worse in Figure 10-3b, where the red dots are scattered all over most of the display tiles.

10.3.2 Display Density

A second advantage of the low dimensional data overview is its relatively low display density. Because of the data reduction during the scaling process, the low dimensional data overview handles

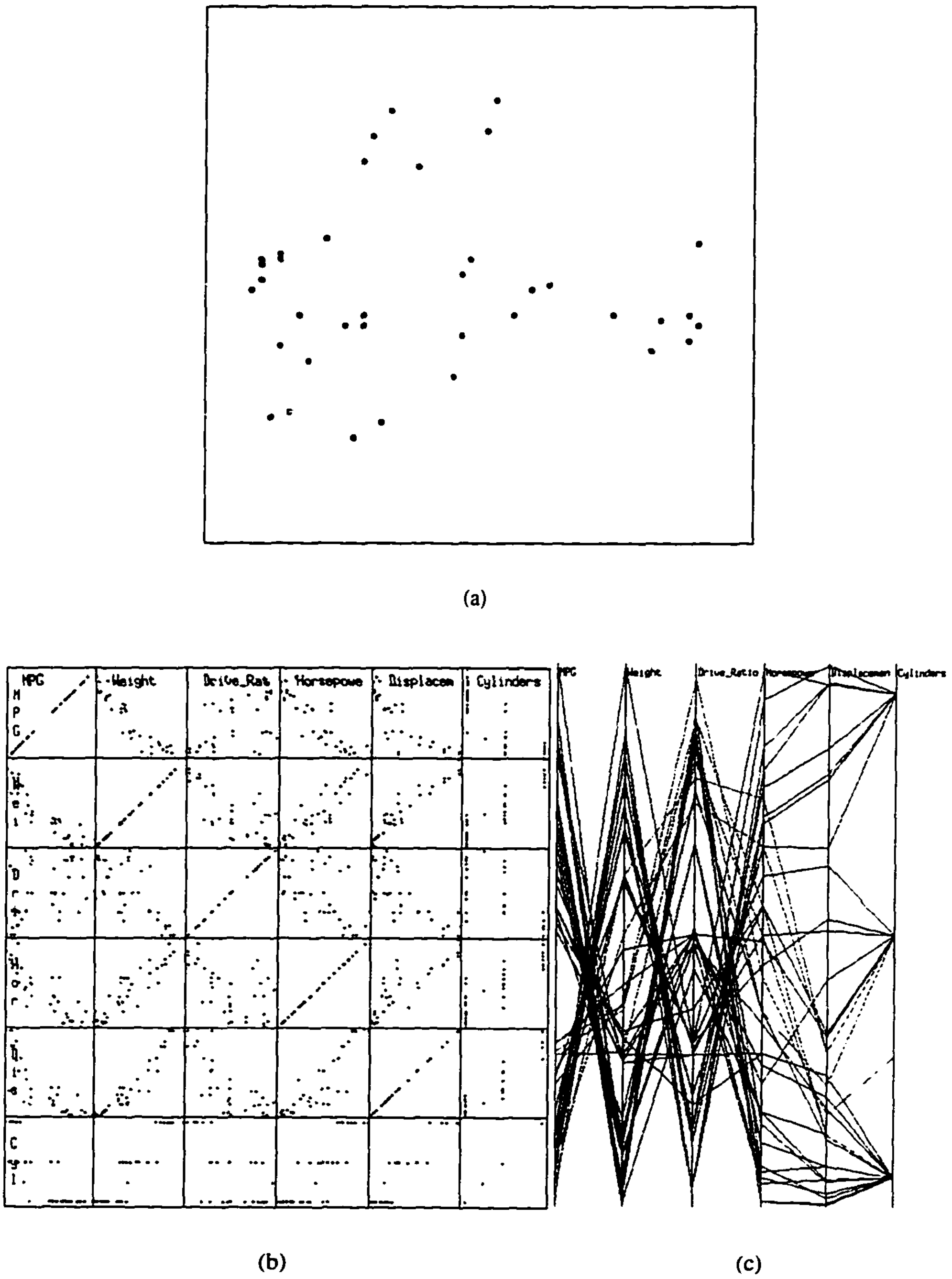


Figure 10-3: The cluster of the upscale medium size cars is displayed in red in the graphs. See also Color Plate 6 in Appendix D.

larger datasets (especially with a high number of variates) much better than the other two techniques. Figure 10-4 depicts a dataset[3] of US cities with 329 records and 10 variates that measure various

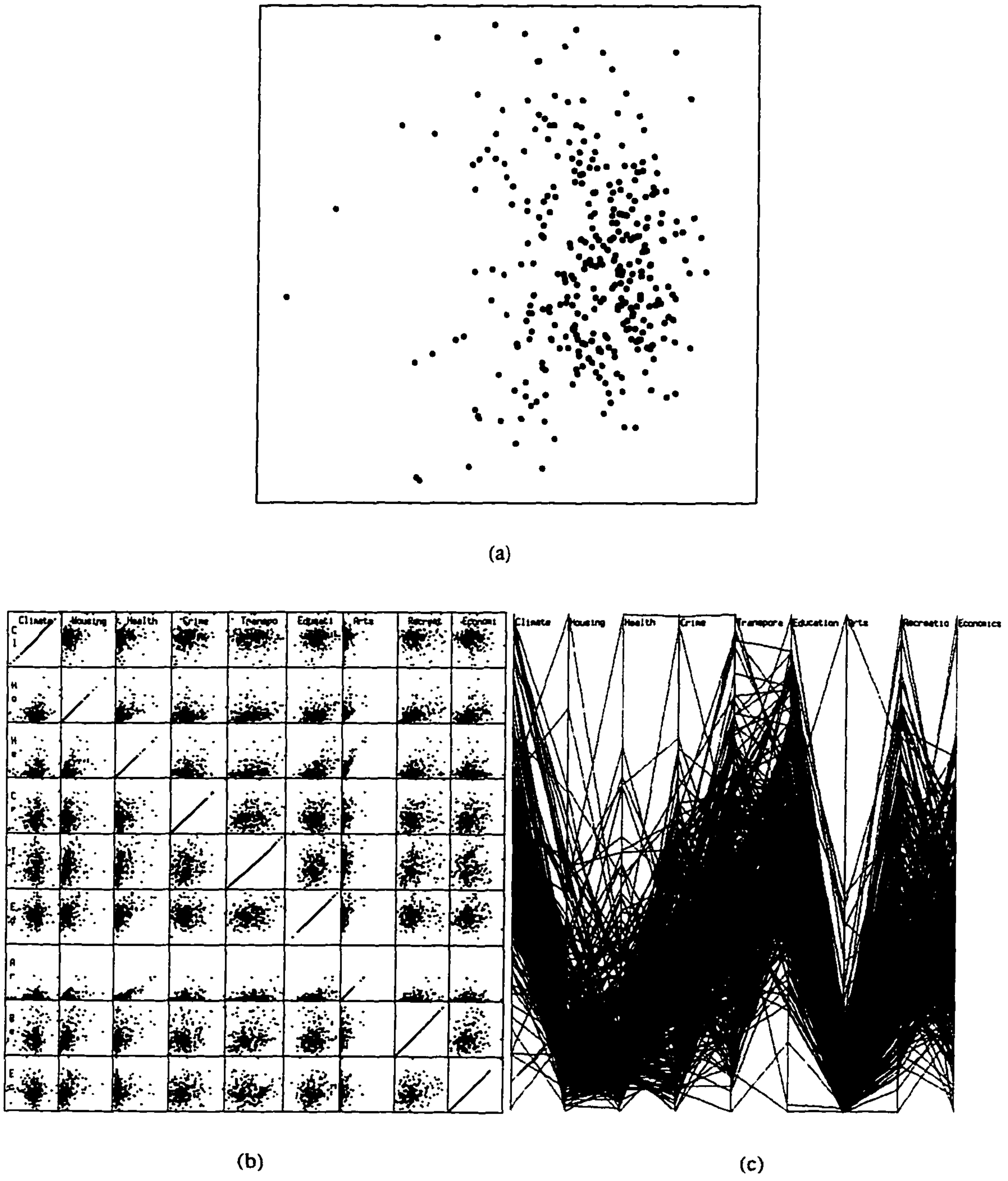


Figure 10-4: A 10 variate dataset with data records of 329 US cites.

quality of living parameters. As we can see, the scatterplot matrix in Figure 10-4b suffers from the high number of data variates, which requires 81 display tiles to cover the data completely. The problem is even more obvious with the parallel coordinate display in Figure 10-4c in which valuable information and data clusters are mostly hidden behind the polylines.

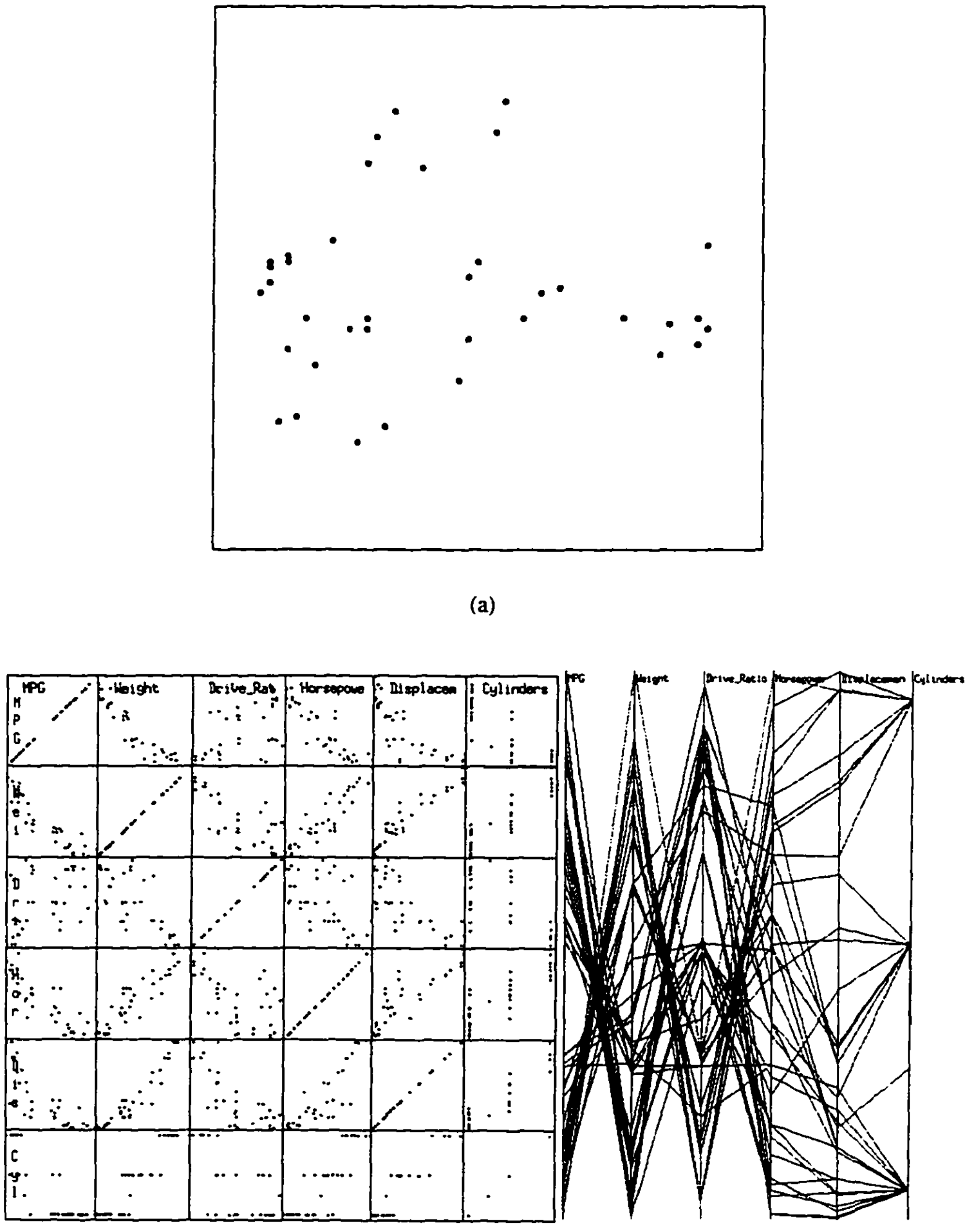
The large number of data points in Figures 10-4b and 10-4c also makes it very difficult to brush the data accurately. That is because a majority of data points are plotted very close to each other while some of the others simply vanish due to overlappings. It is also important to realize that even though two data points are displayed in the same neighborhood in certain tiles in Figure 10-4b (or some axes in Figure 10-4c), the two data records may still be very far away when all variates are considered.

The data points in the low dimensional overview in Figure 10-4a have a significantly lower density than those in Figure 10-4b and there is much less overlap. This makes it much easier to brush neighboring sets of data values than is possible with either of the other displays. In addition, proximity in Figure 10-4a is a measure of the similarity between data records (i.e., cities) whereas proximity in Figure 10-4b and 10-4c only indicate similarity of variate values.

10.3.3 Outlier Detection

The document which comes with the automobile dataset indicates that the Buick Estate Wagon record is an outlier. This is because the data was collected on a test track and the car was operated with a higher than recommended tire inflation pressure, while the rest of the data were collected by EPA under standard test conditions. Since the track condition and the tire pressure are not included in the dataset, the miles per gallon value of the Buick is unexpectedly better than the other cars in the same category. The outlier shows up in the data overview in Figure 10-1 as the Estate Wagon is located away from the full size sedan/wagon cluster. However, the other six variates of the Buick record keep the car fairly close to its peers. Once again, it is much easier to spot the outlying data (shown in red in Figure 10-5) in the low dimensional data overview than in the other two visualization techniques depicted in Figures 10-5b and 10-5c.

[3] http://lib.stat.cmu.edu/datasets/places.data



(a)

(b)

(c)

Figure 10-5: The Buick Estate Wagon (an outlier) is displayed in red in the graphs. See also Color Plate 7 in Appendix D.

10.3.4 Multiresolution Visualization

The visualization of large multivariate datasets continues to be one of the major challenges of visualization research. We suggest another *progressive refinement* solution to visualize datasets with a high number of *variates*.

As we mentioned in Section 10.2.2, the low dimensional data overview based on principal components is only an approximation of the data. Its degree of accuracy is determined by the number of principal components used for display. In most cases, the first two to three principal components of the data overview convey most of the important information of the data. Nonetheless, there are times when a higher number of dimensions is needed.

As an example, we use a 10-variate dataset[4] which contains information about protein consumption in Europe during the 70's. The first five data overviews, which use eigenvectors from the first one, two, three, four, and five principal components, achieve degrees of accuracy of 45.57%, 63.15%, 76.02%, 86.04%, and 91.26% respectively. After the first component, every unit increase in dimension improves the degree of accuracy by about 5–10%. If we consider the 80% size reduction rate, the two dimensional data overview (63.15%) is indeed a very practical result. Nevertheless, error information is always welcome for data visualization which involves data reduction.

We use the glyph [PG88] technique to demonstrate the idea of multiresolution visualization. Bear in mind that our goal is to reduce the number of data dimensions, not the size of the data. Figure 10-6 depicts three coarse-to-fine data overviews of the protein data in two, three, and four dimensional spaces using the glyph representation, followed by an annotated graph to show the identities of the data points. The first two principal components are mapped to the two axes in all three overviews. In Figure 10-6b and Figure 10-6c, the third principal component is mapped to the rainbow colormap with red being the highest. In Figure 10-6c, the fourth component is mapped to the diameter of the glyph, which is a circle. The multiresolution display is particularly important in this dataset because the first two principal components do not reflect a very high degree of accuracy, i.e., 63.15%. This shows up when we look at the data points at the lower left side of Figure 10-6c,

[4] http://lib.stat.cmu.edu/DASL/Datafiles/Protein.html

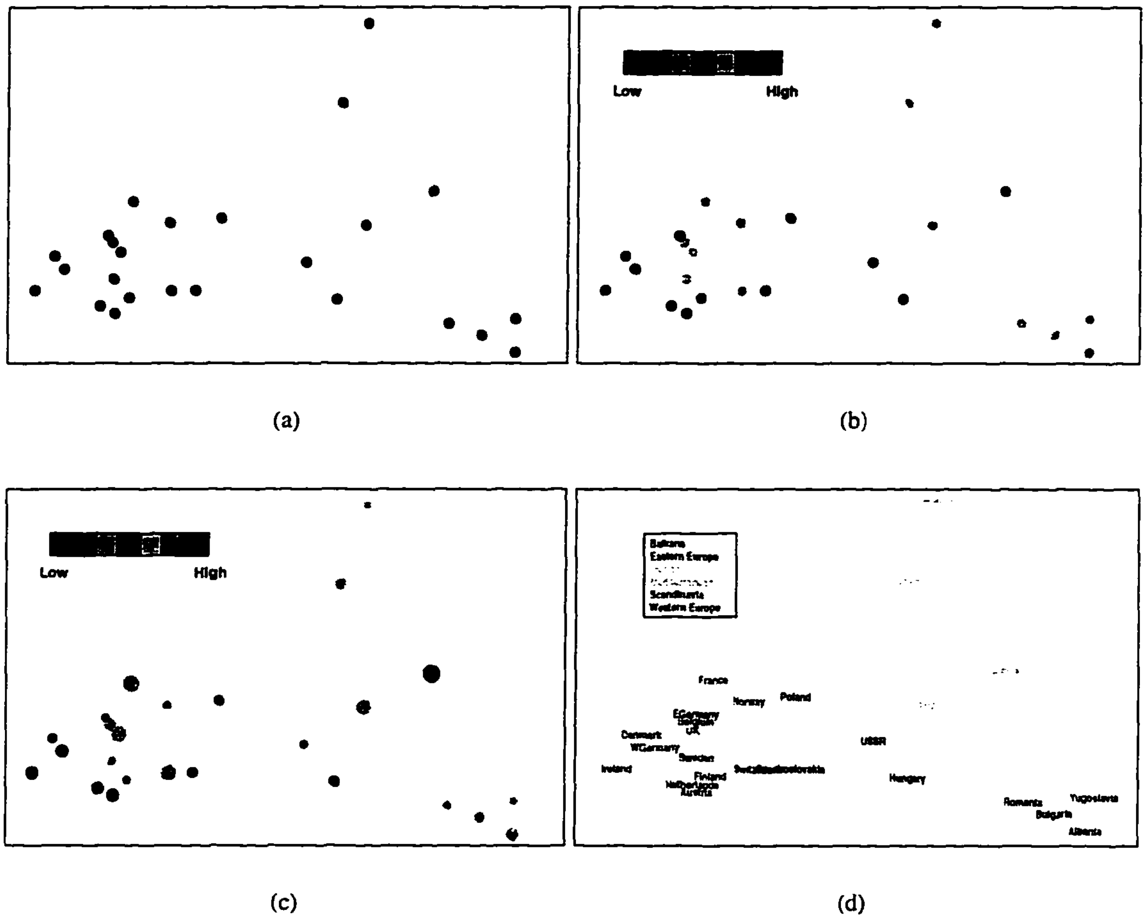


Figure 10-6: The glyph representation of three coarse-to-fine data overviews of the protein data in a) two, b) three, and c) four dimensional spaces. d) An annotated graph to show the identity of the data. See also Color Plate 8 in Appendix D.

where the glyphs include both color and size parameters whose values come from the third and fourth components respectively. Even though the first two major components put these data points close to each other, there are still differences among the neighbors which deserve attention.

From Figure 10-6d, we see that the spatial locations of the glyphs resemble the geographical locations of the corresponding countries. The interpretation is that people who live in neighboring countries share similar diets, which determine the way they consume protein. By looking at the degrees of accuracy achieved by the additions of the third and fourth principal components, the colors and the sizes of the glyphs are expected to play important roles in the interpretations. The display proximity and the similarity of the color of the two data vertices representing the two Germanys is strong evidence that they share very similar cultures, even though they belong to two different data groups, i.e., Eastern and Western Europe.

10.3.5 Shortcomings

It is not entirely fair to compare the effectiveness of visualization tools when we only look at one aspect of the results. The metric scaling technique is good at representing clusters of data points but the individual variate values are lost. For example, a car with more cylinders does not always imply more horsepower than one with less. This can be spotted easily in Figure 10-3c, but not in Figure 10-3a. In the following section, we describe how to integrate the metric scaling technique with techniques that maintain more of the variate value information.

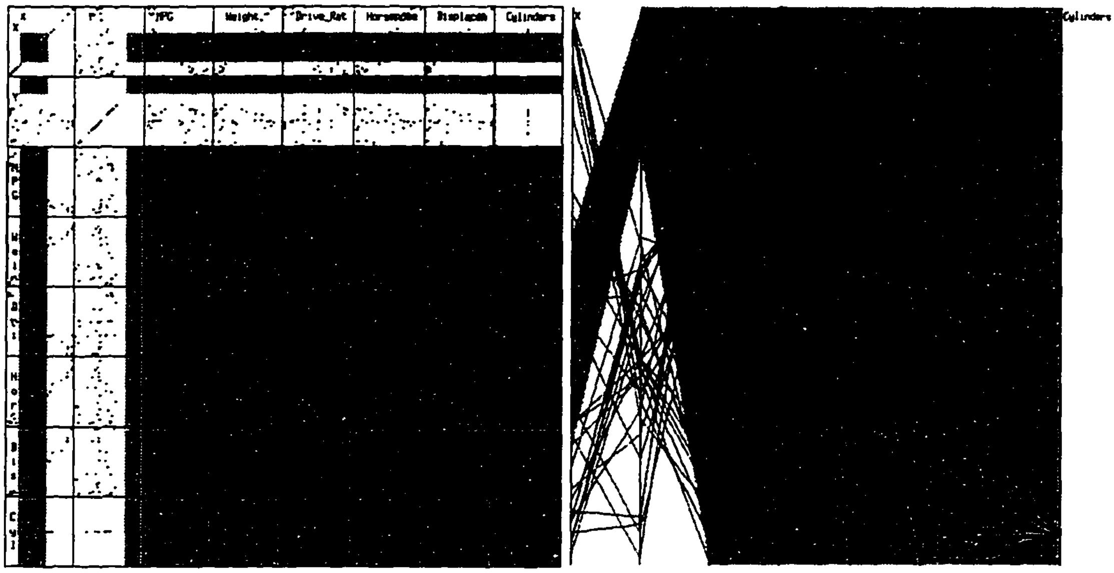
10.4 Integration of Techniques

From the previous examples, we have learned that the low dimensional data overview cannot be used to replace the other visualization tools for effective multivariate visualization. The goal is to combine these tools together easily and productively.

10.4.1 View Linking

Since the size of each principal component of the data overview is the same as the number of records of the data, we can simply treat the components of the data overview as part of the data for visualization. This approach is demonstrated in Figure 10-7 using the same automobile dataset presented in Section 10.1. The first two variates in the upper left corner of Figure 10-7a are the principal components of the data overview. With the use of high dimensional data brushing, we can mark the overview tiles and study the responses in the other display tiles at the same time. As we can see, the brushed data (in red) in the two data overview tiles are spread out in the rest of the scatterplot matrix in Figure 10-7a. In Figure 10-7b, the first two axes effectively pull all the data clusters together. Brushing these clusters is difficult without the support of the data overview.

The idea of linking a data overview to a visualization technique seems simple, but the implications reach far beyond just the point-and-click operations. By applying the Euclidean metric



(a) (b)

Figure 10-7: Multidimensional brushing with a low dimensional data overview. See also Color Plate 9 in Appendix D.

to different subsets of variates, we provide a very realistic overview which we can use to query the data with a far more powerful and flexible language than the conventional database query languages using aggregate functions. The data overview presents a graphic summary with reduced data dimensions, reduced data size, additional data semantics, and most important of all, better user-friendliness for the underlying visualization technique.

10.4.2 Display Merging

The second strategy is to merge the Euclidean coordinates of the data overview and the data into one visualization display. Conventional glyph representations such as the stick figure icon [PG88] can be useful only when the icons are arranged in certain dimensions. The patterns may vanish altogether with even a slight change of the plotting axes. This limits the flexibility and functionality of the technique. The low dimensional data overview, however, brings new perspective to conventional icon visualization. Our approach is to arrange the icons according to the Euclidean coordinates of the data overview determined by the principal components of the data.

An arbitrary glyph, which is defined in Figure 10-8, is used to visualize the protein data described in Section 10.3.4 using the same colormap as in Figures 10-6b and 10-6c. The glyph contains four layers and nine blocks. Each layer approximately represents one food category and each block represents one kind of food. The result is shown in Figure 10-9. As we can see, the glyphs

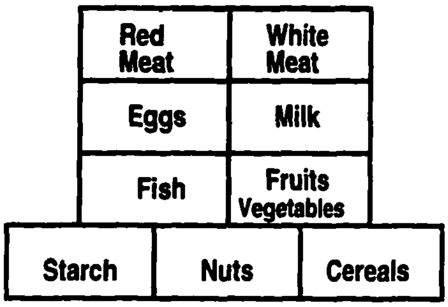


Figure 10-8: The definition of a nine variate glyph.

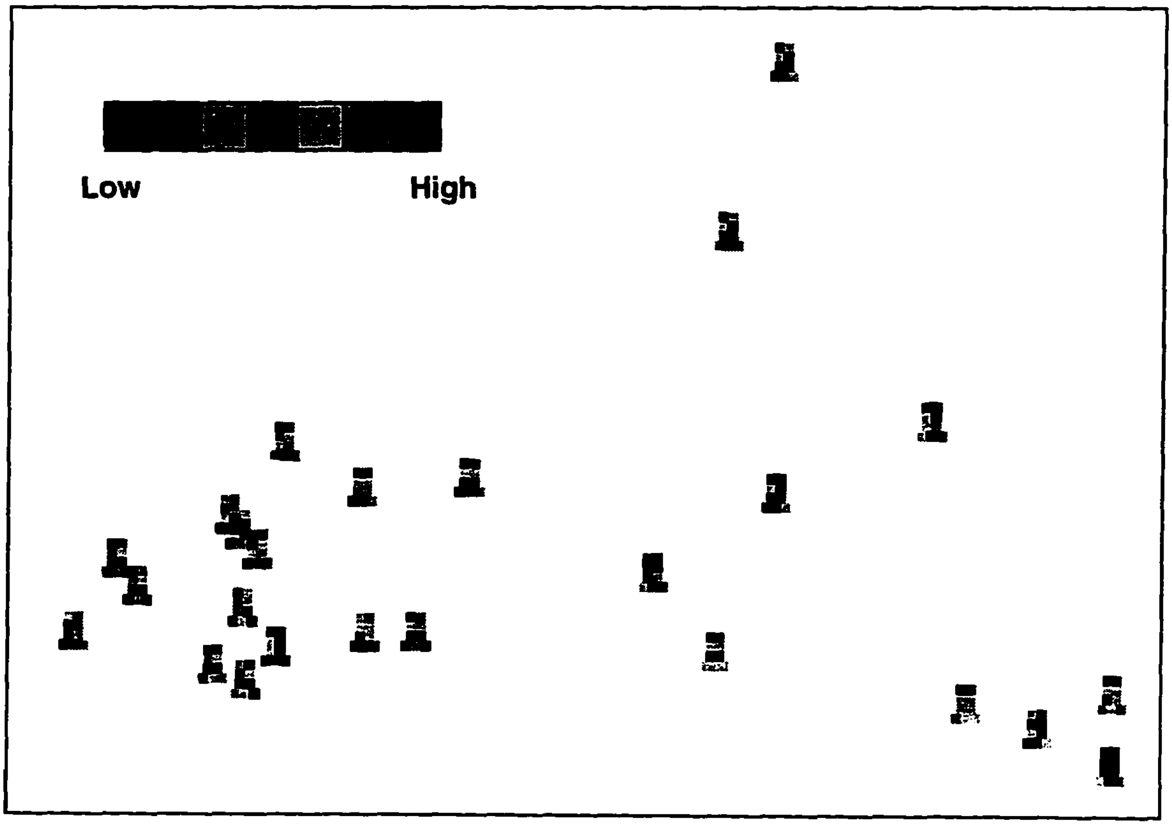


Figure 10-9: The nine variate protein consumption data represented by the glyph representation. See also Color Plate 10 in Appendix D.

representing the Balkans indicate that the countries including Yugoslavia, Romania, Bulgaria, and Albania consume a relatively small amount of meat, milk, fruits, vegetables, and fish, but a large amount of starch and cereals. Scandinavian countries such as Sweden and Finland consume much more meat, milk, and vegetables than starch and cereals. The Iberian countries including Spain and Portugal consume a lot of fruits, vegetables, and fish, and only a small amount of meat and milk. And the Mediterranean countries including Greece and Italy have relatively balanced diets. All this information, however, can also be revealed by the icons themselves. However, if we look closely at the spatial locality of the glyphs, countries with higher meat, egg, and milk consumptions (the top two layers) tend to be located at the lower left hand side; countries with higher starch and cereals consumptions (the bottom layer) tend to be located at the lower right hand side; countries with higher fruits, vegetables, and fish consumptions (the third layer) are at the top; and finally, countries with relatively balanced diets are around the middle of Figure 10-9. These explain why the Balkans are at the lower right hand side, Scandinavian and Western Europe countries are at the lower left hand side, Iberian countries are at the top, and Mediterranean countries are in the middle of Figure 10-9.

The visualization of the data and its principal components together offers a lot more than just the data itself. The principal coordinates of the data also provide new opportunities to create new shape and texture patterns with the conventional iconographic technology.

Chapter 11

Experimental Results

The goal of this chapter is to evaluate the performance of our multiresolution data hierarchy model. We describe a computational study that examines the multiresolution data representation of real life high dimensional datasets. The test datasets are fully public domains. They are often used for performance evaluation within the volume visualization community. The experimental tests were conducted on a 133 MHz Linux box with 32MB of real memory and 256MB of virtual memory.

The remainder of the chapter is divided into four sections. Section 11.1 briefly describes a function library we developed to conduct our experiments. Section 11.2 describes the test data and procedures of our investigation. Section 11.3 gives the results of our study. We conclude with a discussion of the results in Section 11.5.

11.1 Function Library

In the course of running our experiments, it was beneficial to develop a C++ function library that supports wavelet transforms of very large datasets. The library

- provides orthogonal wavelet transforms with up to ten vanishing moments;
- supports wavelet transforms on high dimensional datasets;
- supports progressive refinement data analysis;
- keeps track of the accumulated data loss as well as data loss from individual resolutions during wavelet transforms;
- provides over a dozen statistical functions for data analysis;

- supports numerical analysis functions such as root-mean-square error and norms.

In addition to a complete implementation of wavelet transforms, the heart of the library is a group of multidimensional refinement procedures built on recursive subspace projections. Assume we have a three dimensional volume dataset which contains an approximation subvolume (in grey) and a detail subvolume (in white) as shown in Figure 11-1. The library supports the following *n*-dimensional operations:

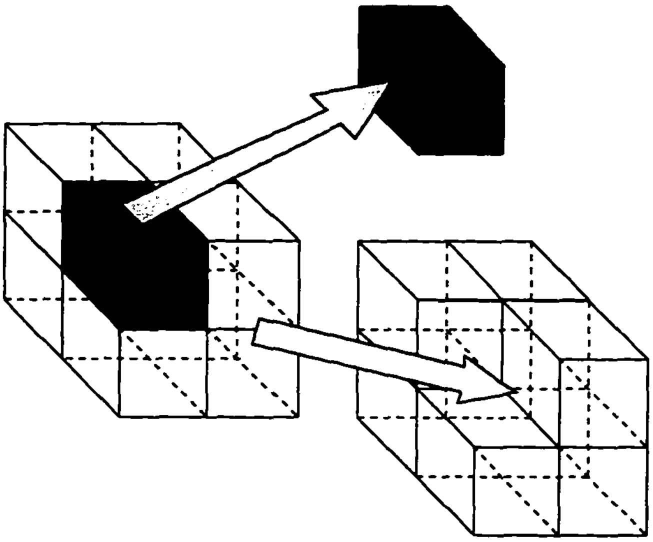


Figure 11-1: The approximation subvolume (in grey) and detail subvolume (white) are extracted from a volume dataset created from one wavelet decomposition.

- extracts the *data approximation* of an *n*-dimensional dataset;
- extracts the *wavelet detail* of an *n*-dimensional dataset;
- calculates the error norm of an individual approximation point by combining the error values of the corresponding details in all dimensions;
- calculates the overall error norm of an approximation volume.

11.2 Test Datasets and Procedures

Given a very large multidimensional dataset, our goal is to find highly authentic approximations whose sizes are substantially smaller than the original in a hierarchical fashion. Since the data size and data authenticity are usually two conflicting goals, the qualities of our results can only be judged by the outcomes of the applications which use the data. In our case, the target application is isosurface rendering of three dimensional volume data.

11.2.1 Datasets

Our test data consists of three volume datasets which include CT scans of a lobster,[1] a partial human head,[2] and an engine block.[3] The lobster dataset has 64 slice images at 128^2 resolution each. The human head dataset contains 128 slice images at the same resolution. The engine dataset consists of 256 slice images at 256^2 resolution. More information about the datasets is listed in Table 11.1. Figure 11-2 shows the isosurface renderings of the three datasets using the marching cubes algorithm [LC87, NH90] as implemented in the *vtk* library [SML96]. The isovalues and the number of generated triangles of the renderings are listed in Table 11.2.

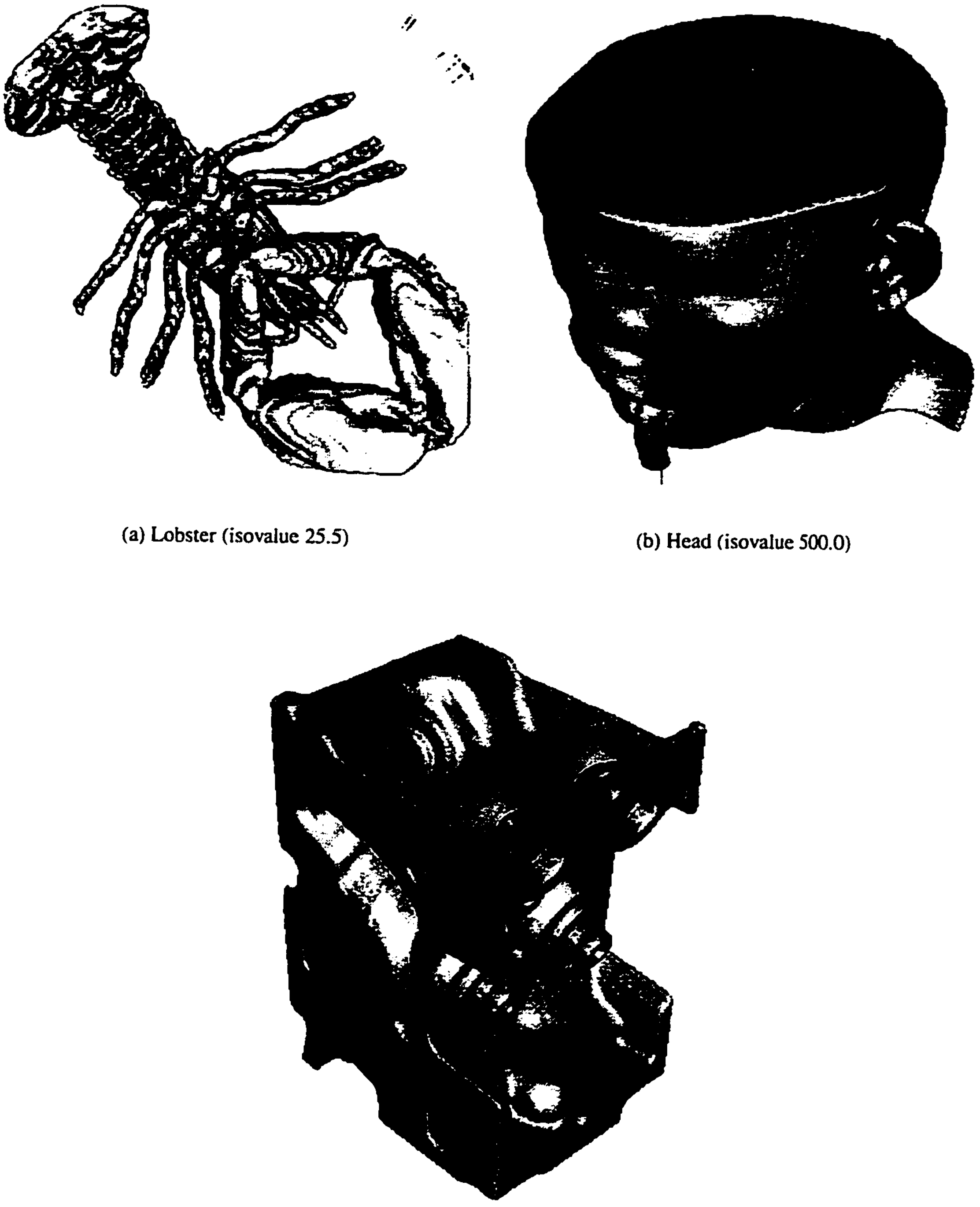
Dataset	*Data Dimensions*	*Data Size*	*Min Value*	*Max Value*	*Mean Value*	*Standard Deviation*
Lobster	128 × 128 × 64	1048576	0	255	8.19	35.97
Head	128 × 128 × 128	2097152	0	4095	370.52	547.92
Engine	256 × 256 × 128	8388608	0	255	21.87	49.28

Table 11.1: A summary of important information about the test datasets.

[1] Courtesy of Advanced Visual Systems, Inc.

[2] Courtesy of Will Schroeder, Ken Martin, and Bill Lorensen of General Electric Corporate R & D Center.

[3] Obtained by anonymous FTP from Stanford University.



(a) Lobster (isovalue 25.5)

(b) Head (isovalue 500.0)

(c) Engine (isovalue 50.5)

Figure 11-2: Isosurface renderings of the a) lobster (isovalue 25.5), b) head (isovalue 500.0), and c) engine (isovalue 50.5) datasets.

Dataset	*Isovalue*	*Number of $\triangle s$*	*Time (s)*
Lobster	25.5	93252	22.70
Head	500.0	171180	278.08
Engine	50.5	622196	534.55

Table 11.2: Statistics of the isosurface renderings.

11.2.2 Procedures

Each test dataset is first decomposed into a multiresolution data hierarchy using wavelet transforms. For each data resolution of the hierarchy, both the L^1 and L^2 norms of the wavelet details are calculated. These norm values are used to decide the better resolutions (in terms of data size and authenticity) of the approximations in the hierarchy. To reduce the storage space of the hierarchy, a substantial number of less significant wavelet coefficients are removed from these selected approximations. Datasets with different degrees of truncation are then reconstructed to their original sizes for comparison. These procedures are repeated for different wavelets. Due to the relatively small dimension sizes (64, 128, 256) of our test datasets, only orthogonal wavelets with less than three vanishing moments are used.

11.3 Results and Evaluations

This section describes the experiments and evaluates the results of our study in detail. We start with the head dataset because it is twice as large as the lobster dataset and its data content is more heterogeneous than the metallic based engine block.

11.3.1 Human Head Dataset

We summarize our main experimental results of the head dataset in a sequence of tables and graphs followed by explanations and discussions. Table 11.3 contains the information about the running time required by different wavelets to fully decompose the human head volume dataset. The columns *Wavelet* and *Vanishing moments* identify the wavelets and their number of vanishing moments used to generate the hierarchy. The *Time* column specifies the running time of the data

Wavelet	*Vanishing Moments*	*Time (s)*
H_2	1	12.70
D_4	2	18.30
D_6	3	24.23

Table 11.3: CPU seconds to fully decompose the head dataset using different wavelets.

decomposition. All running times are measured in CPU seconds on a 5133 Linux Box with 32MB of memory. A plot of the CPU seconds needed to decompose the dataset versus different wavelets is shown in Figure 11-3. Since more vanishing moments imply more computational steps during decompositions, the wavelet which has the smallest filter size (i.e., H_2) has the shortest CPU time.

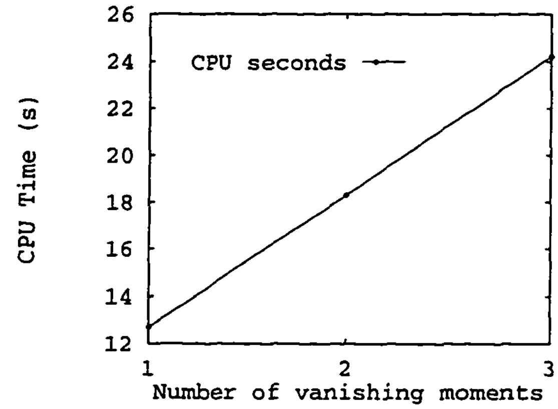


Figure 11-3: CPU times versus number of vanishing moments of the wavelet decompositions of the human head dataset.

Error Estimation

We now turn our attention to the qualities of the wavelet approximations. As in Chapter 5, we use a *norm* to measure the information loss due to wavelet decomposition. In our investigation, two norms (L^1 and L^2) are applied to the wavelet details of each resolution to study the errors of the corresponding approximations.

Let $X = \{x_i\}$ for $i = 1, \ldots, N$ where $x_i \subset \mathbb{R}$. The L^1 norm of X is defined as $L^1(X) = \sum |x_i|$ while the L^2 norm is given by $L^2(X) = \sqrt{\sum x_i^2}$. Since the information loss due to data decomposition is recorded only in the wavelet details, we do not include the approximations in the norm calculations. For the rest of the discussion, we use L^1 to represent the L^1 error of the wavelet decomposition $W(n) \rightarrow W(\frac{n}{2})$, i.e., $L^1 = L^1(Details(W))$. Similarly, the L^2 error is given by $L^2 = L^2(Details(W))$.

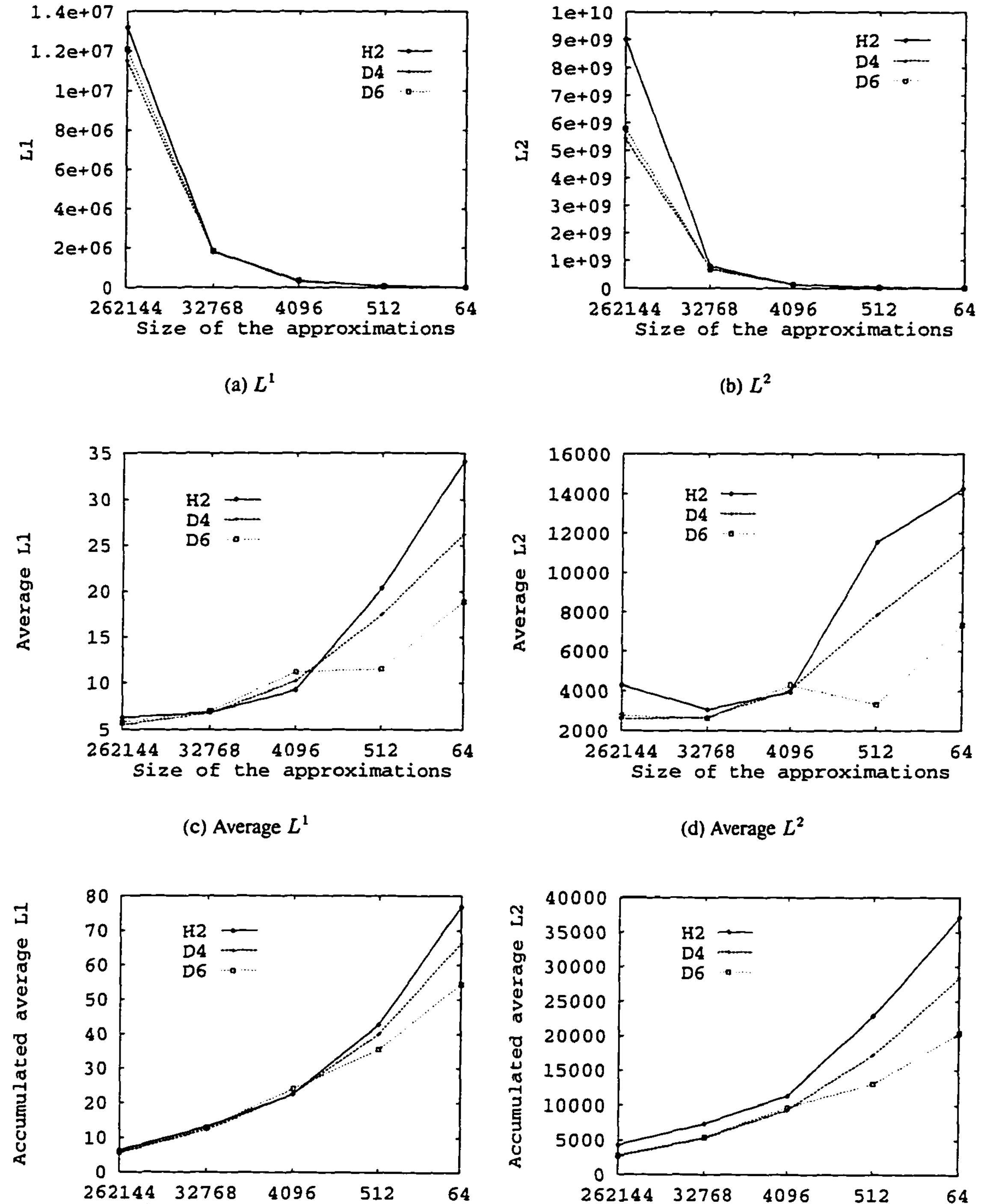
An *average* L^1 generated from a single decomposition is defined as $\overline{L^1} = \frac{L^1}{n}$ where n is the data size *before* the decomposition. Similarly, $\overline{L^2} = \frac{L^2}{n}$. Since a wavelet approximation inherits all the errors generated from previous (i.e., finer) decompositions, an *accumulated* L^1 (L^2) norm, $\sum \overline{L^1}$ ($\sum \overline{L^2}$), is defined as the sum of all the $\overline{L^1}$ ($\overline{L^2}$) norms of the finer resolutions in the multiresolution hierarchy. To compute the percentage error E of an individual approximation point, we define $E = \frac{\sum \overline{L^1}}{\overline{X}}$ where $\overline{X}$ is the mean of the set X.

The norm values and the percentage errors of the wavelet details generated from the human head dataset are listed in Table 11.4. These norm values are plotted against the size of the approximations in Figure 11-4.

W	*Data Size*	L^1	$\overline{L^1}$	$\sum \overline{L^1}$	E (%)	L^2	$\overline{L^2}$	$\sum \overline{L^2}$
H_2	64×64×64	1.32e+07	6.29	6.29	1.70	9.01e+09	4295.48	4295.48
	32×32×32	1.80e+06	6.87	13.17	3.55	8.03e+08	3061.89	7357.38
	16×16×16	3.04e+05	9.29	22.45	6.06	1.29e+08	3946.56	11303.94
	8×8×8	8.35e+04	20.38	42.83	11.56	4.74e+07	11562.01	22865.95
	4×4×4	1.75e+04	34.12	76.95	20.77	7.30e+06	14251.07	37117.02
D_4	64×64×64	1.15e+07	5.49	5.49	1.48	5.44e+09	2593.26	2593.26
	32×32×32	1.79e+06	6.82	12.31	3.32	7.02e+08	2679.41	5272.66
	16×16×16	3.36e+05	10.25	22.56	6.09	1.32e+08	4036.19	9308.86
	8×8×8	7.17e+04	17.52	40.08	10.81	3.22e+07	7871.46	17180.32
	4×4×4	1.34e+04	26.25	66.33	17.90	5.76e+06	11243.05	28423.36
D_6	64×64×64	1.21e+07	5.76	5.76	1.56	5.79e+09	2706.76	2760.76
	32×32×32	1.84e+06	7.03	12.79	3.45	6.89+08	2628.78	5389.54
	16×16×16	3.68e+05	11.22	24.01	6.48	1.41+08	4292.21	9781.75
	8×8×8	4.73e+04	11.54	35.56	11.04	1.36+07	3325.81	13007.56
	4×4×4	9.68e+03	18.92	54.47	19.09	3.76e+06	7338.07	20345.62

Table 11.4: The norms, average norms, accumulated average norms, and percentage errors of the wavelet details generated from the head volume dataset by different wavelets in multiple resolutions.

As one might suspect, the values of L^1 and L^2 are largely dependent upon the sizes of the data approximations, i.e., more data values imply a larger norm. This shows up in Figures 11-4a and 11-



(a) L^1

(b) L^2

(c) Average L^1

(d) Average L^2

(e) accumulated average L^1

(f) accumulated average L^2

Figure 11-4: The a) L^1, b) L^2, c) average L^1, d) average L^2, e) accumulated average L^1, and f) accumulated average L^2 of the wavelet details generated from the human head dataset are plotted against the size of the approximations

4b where the norm values drop sharply after the first decomposition. If we look at Figures 11-4c and 11-4d, the average norms are actually lower at finer (higher) resolutions near the origins of the graphs. From these figures, we conclude that the average information loss of the head dataset is lower when the data resolution is finer.

We also notice that the rates of information loss in Figures 11-4c and 11-4d are very steady in the first few resolutions. This can be detected by looking at the slopes of the data lines in Figures 11-4e and 11-4f. The line slopes become steeper after the third decomposition. This suggests the end of a quiescent state in which major features of the approximations are roughly the same. Ideally, only a few of the lower resolution representations need to be maintained from a set of resolutions that are part of a single quiescent state.

An important observation we made during our investigation is that many wavelet generated *volume* data approximations with percentage errors $E < 5\%$ visually resemble the original datasets very well. However, we will show later in this chapter that if datasets have very homogeneous contents (such as the engine block depicted in Figure 11-2c), the acceptable percentage error can be increased somewhat to about 8%. This observation agrees with our previous suggestion that a quiescent state ends at the second decomposition because the percentage error of the third decomposition is over 5%.

Since the error norms of D_4 are the smallest in most cases (as shown in Table 11.4), the Daubechies wavelet with two vanishing moments may probably be the best candidate to decompose the human head dataset. However, the differences among the norm values within the same resolution are so small that we believe the overall qualities of the final renderings are very close to each other. If processing time is critical to an application, H_2 has the upper hand over the other two.

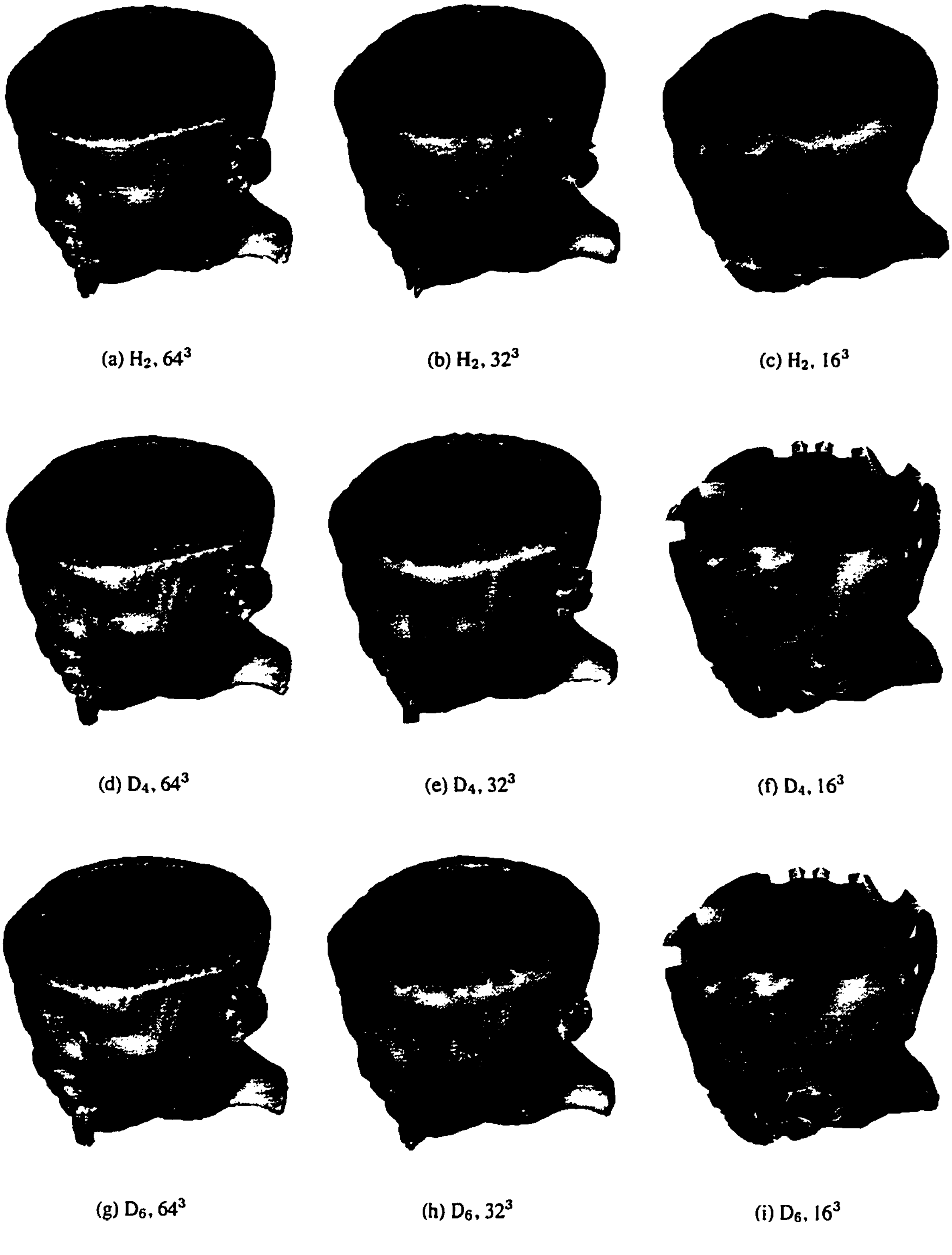
To verify the results, we use both a quantitative (objective) test and a visual (subjective) test to evaluate the performance of our error measure mechanism. First, we created a dataset whose size matched the original data from the approximation using zero values for all detail coefficients. We compared this reconstruction to the original data using a root-mean-square error function. The results are listed in Table 11.5. Our second test uses a marching cubes program to render the approximations. In addition to the visual effects, the number of triangles generated by the rendering

W	*Data Size*	*RMSE*	*Number of* Δs
H_2	64x64x64	149.72	39568
	32x32x32	156.51	8688
	16x16x16	177.69	2944
	8x8x8	304.13	0
	4x4x4	337.65	0
D_4	64x64x64	116.33	39592
	32x32x32	146.41	8644
	16x16x16	179.69	3024
	8x8x8	250.94	0
	4x4x4	299.91	0
D_6	64x64x64	120.02	39376
	32x32x32	145.02	8958
	16x16x16	185.31	3116
	8x8x8	163.12	0
	4x4x4	242.29	0

Table 11.5: The RMSEs between the original and the reconstructed datasets, and the number of triangles required to render the approximations using the marching cubes algorithm.

process more or less reflects the quality of an approximation because more triangles usually give a better surface representation.

The results in Table 11.5 clearly indicate that major data losses exist after the third wavelet decomposition. Our implementation of the marching cubes algorithm actually cannot generate any isosurface triangles at 8×8×8 resolution (see Table 11.5). These figures coincide with our previous suggestion that a quiescent state ends at the 16×16×16 resolution. To perform the visual test, nine isosurface renderings of the approximations generated by H_2, D_4, and D_6 are shown in Figure 11-5. The results are clear. The qualities of the isosurface renderings at 16×16×16 resolution (Figure 11-5c, 11-5f, and 11-5i) are poor because their percentage errors E are all within the 5% to 10% ranges. It is also true that the qualities of the approximations generated by different wavelets are visually very close to each other. However, the processing time of H_2 is only one third of that of D_6 (see Table 11.3). At this point, it is reasonable to consider the approximations of H_2 and D_4 at both 64×64×64 and 32×32×32 resolutions.



(a) H_2, 64^3

(b) H_2, 32^3

(c) H_2, 16^3

(d) D_4, 64^3

(e) D_4, 32^3

(f) D_4, 16^3

(g) D_6, 64^3

(h) D_6, 32^3

(i) D_6, 16^3

Figure 11-5: The wavelets used to decompose the data, resolutions of the approximations, and the number of isosurface triangles generated during the renderings are listed in order: a) H_2, 64^3, 39568 triangles. b) H_2, 32^3, 8688 triangles. c) H_2, 16^3, 2944 triangles. d) D_4, 64^3, 39592 triangles. e) D_4, 32^3, 8644 triangles. f) D_4, 16^3, 3024 triangles. g) D_6, 64^3, 39376 triangles. h) D_6, 32^3, 8958 triangles. i) D_6, 16^3, 3116 triangles.

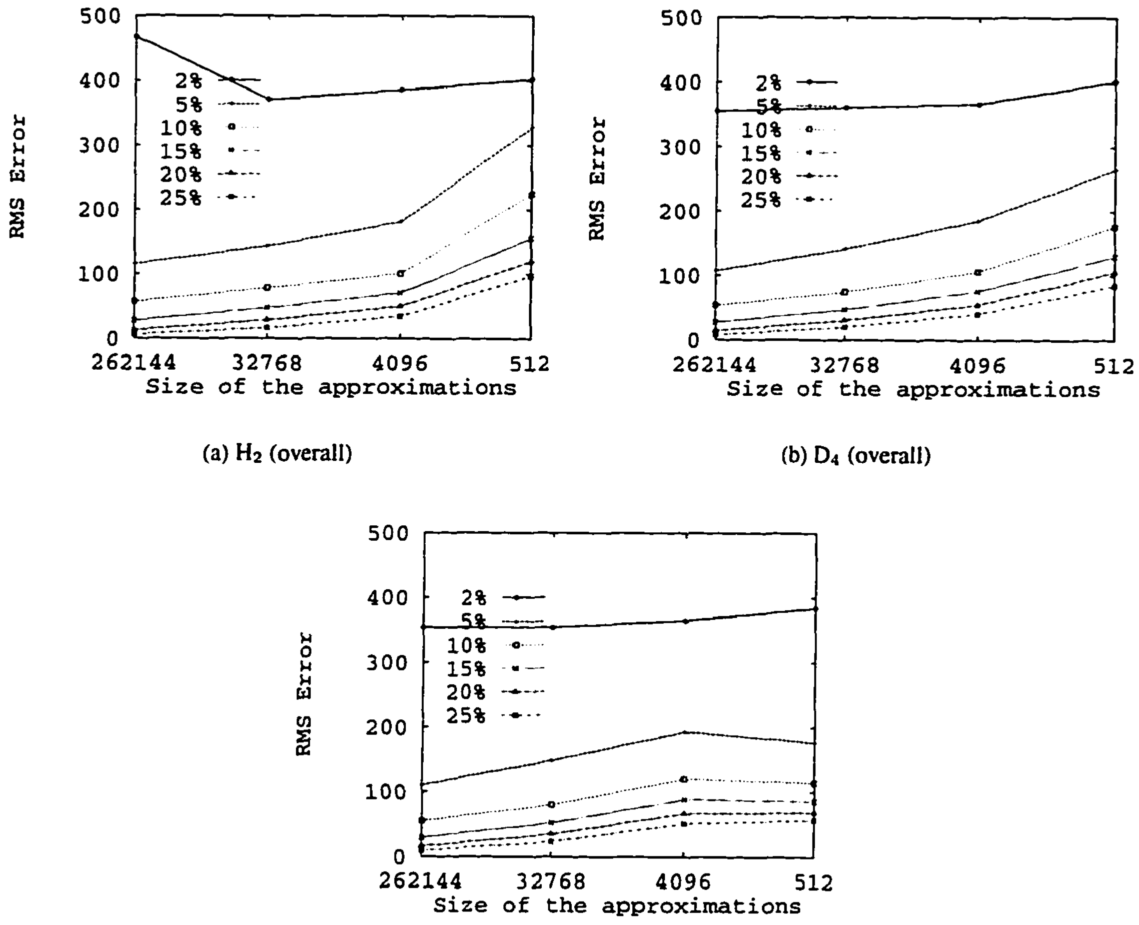
Wavelet Coefficient Truncation

Our next step is to study the effect of data truncations on wavelet approximations. The approximation data values are first sorted according to their absolute values. We then remove 98%, 95%, 90%, 85%, 80%, and 75% of the less significant (i.e., smaller) data values and reconstruct the datasets with the remaining 2%, 5%, 10%, 15%, 20%, and 25% of the data at every resolution. A root-mean-square error (RMSE) function is then used to compare the original datasets with the reconstructed ones and the results are listed in Table 11.6. The RMSE values of different degrees of truncation are plotted against the size of the approximations in Figure 11-6. The results are very consistent with our expectation – more data points give smaller RMSEs. However, the advantages of keeping more data values becomes smaller after the retention percentage reaches 10%. This suggests that at least a 10% retaining rate of the most significant values is desirable in the head dataset hierarchy.

Wavelet	*Data Size*	2%	5%	10%	15%	20%	25%
H_2	64×64×64	347.27	115.37	66.34	40.14	25.57	16.91
	32×32×32	355.91	151.72	92.70	65.81	48.56	36.60
	16×16×16	374.43	190.65	132.39	102.03	81.88	67.03
	8×8×8	332.39	219.69	166.92	134.81	110.551	92.27
D_4	64×64×64	355.29	107.60	53.65	27.22	14.53	7.95
	32×32×32	360.65	141.14	74.44	47.30	30.38	20.17
	16×16×16	366.71	185.39	106.58	75.98	54.96	40.71
	8×8×8	402.07	265.19	176.57	130.30	105.86	84.87
D_6	64×64×64	354.39	110.47	55.27	29.52	16.11	9.22
	32×32×32	354.78	148.60	80.06	52.22	34.75	23.50
	16×16×16	365.28	192.86	120.07	88.40	66.72	51.12
	8×8×8	384.14	176.48	114.36	85.70	68.54	57.05

Table 11.6: The RMSEs between the original approximations and the reconstructions of the truncated approximations.

We would also like to see the impact of data truncation on a particular density of the dataset, especially when we only render the skin surface of the human head. We recompute the RMSEs and this time we only use the isovalue 500, which is the density value of the skin. The results are listed in Table 11.7. These RMSEs of different wavelets are plotted against the size of the



(a) H2 (overall)

(b) D4 (overall)

(c) D6 (overall)

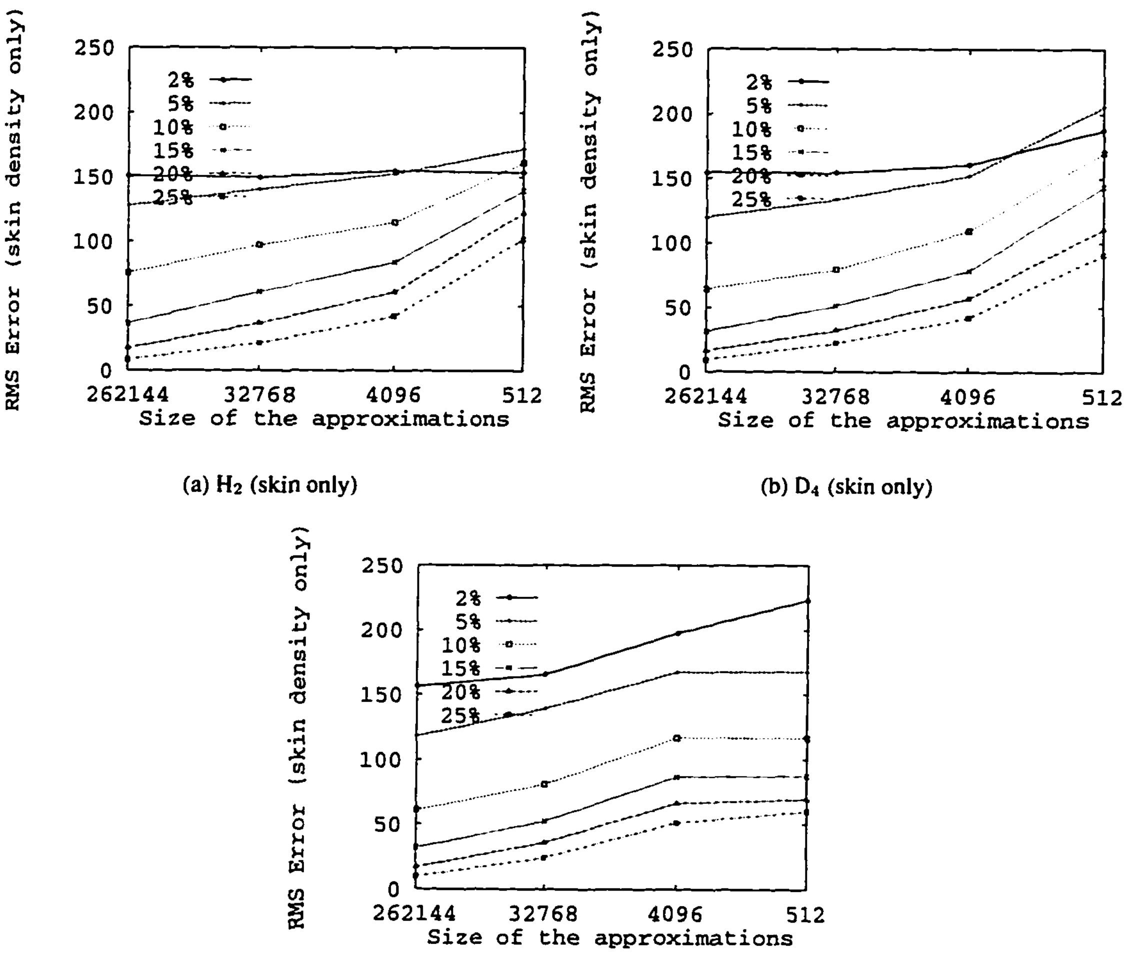
Figure 11-6: The RMSEs between the original approximations and the reconstructions of the truncated approximations are plotted against the size of the approximations.

Wavelet	*Data Size*	2%	5%	10%	15%	20%	25%
H_2	64x64x64	150.79	127.65	75.64	36.45	17.15	8.30
	32x32x32	149.61	140.29	97.12	60.95	36.68	21.11
	16x16x16	154.85	152.51	114.81	84.01	60.88	42.10
	8x8x8	153.57	172.09	161.26	139.23	122.03	101.83
D_4	64x64x64	154.55	119.65	64.19	31.42	16.06	9.02
	32x32x32	154.17	133.44	79.25	51.06	32.02	22.11
	16x16x16	160.66	151.84	109.06	78.36	56.77	41.84
	8x8x8	187.47	205.73	170.11	143.47	110.48	90.86
D_6	64x64x64	156.21	118.01	61.08	31.84	17.02	9.92
	32x32x32	165.39	139.18	80.92	52.43	35.85	24.14
	16x16x16	197.62	167.33	116.77	86.72	66.32	51.25
	8x8x8	223.22	167.74	116.48	87.35	69.48	60.10

Table 11.7: The RMSE between the skin portion of the original approximations and the corresponding reconstructions of the truncated approximations.

approximations in Figure 11-7. There are still clear differences between the RMSEs of the 5% and 10% reconstructions. This is consistent with our previous guideline of at least 10% retention of the coefficients.

Finally, we use a visual test to verify our estimation. All six truncated approximations at both the 32x32x32 and 64x64x64 levels are rendered using the marching cubes algorithm from *vtk*. The results are shown in Figures 11-8, Figures 11-9, 11-10, and 11-11. These figures show that retention rates of 10% (or higher) of the most significant data values (Figures 11-8a – 11-8e, 11-9a – 11-9e, 11-10a – 11-10e, 11-11a – 11-11e) are enough to capture many of the important features of the original data in Figures 11-8f, 11-9f, 11-10f, and 11-11f.

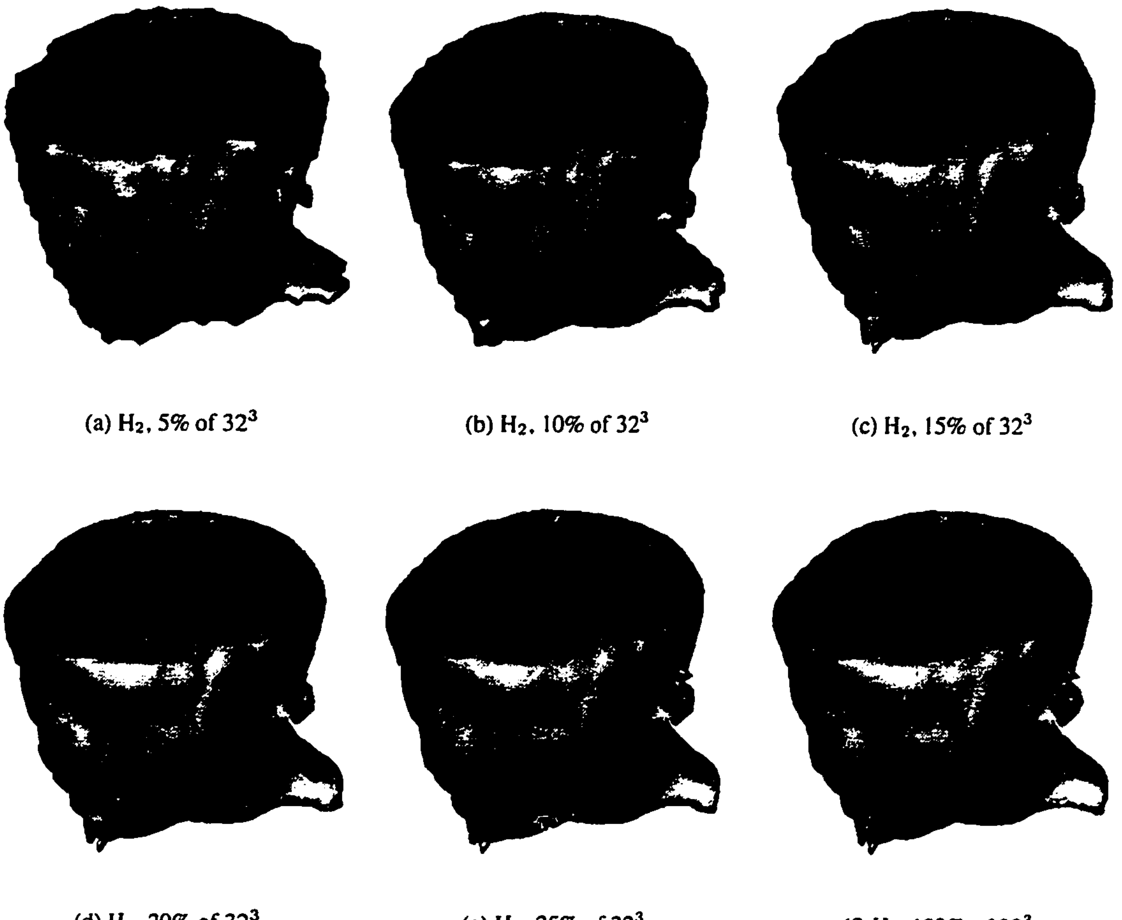


(a) H2 (skin only)

(b) D_4 (skin only)

(c) D_6 (skin only)

Figure 11-7: The RMSEs between the skin portion of the original approximations and the reconstructions of the truncated approximations are plotted against the size of the approximations.



(a) H_2, 5% of 32^3

(b) H_2, 10% of 32^3

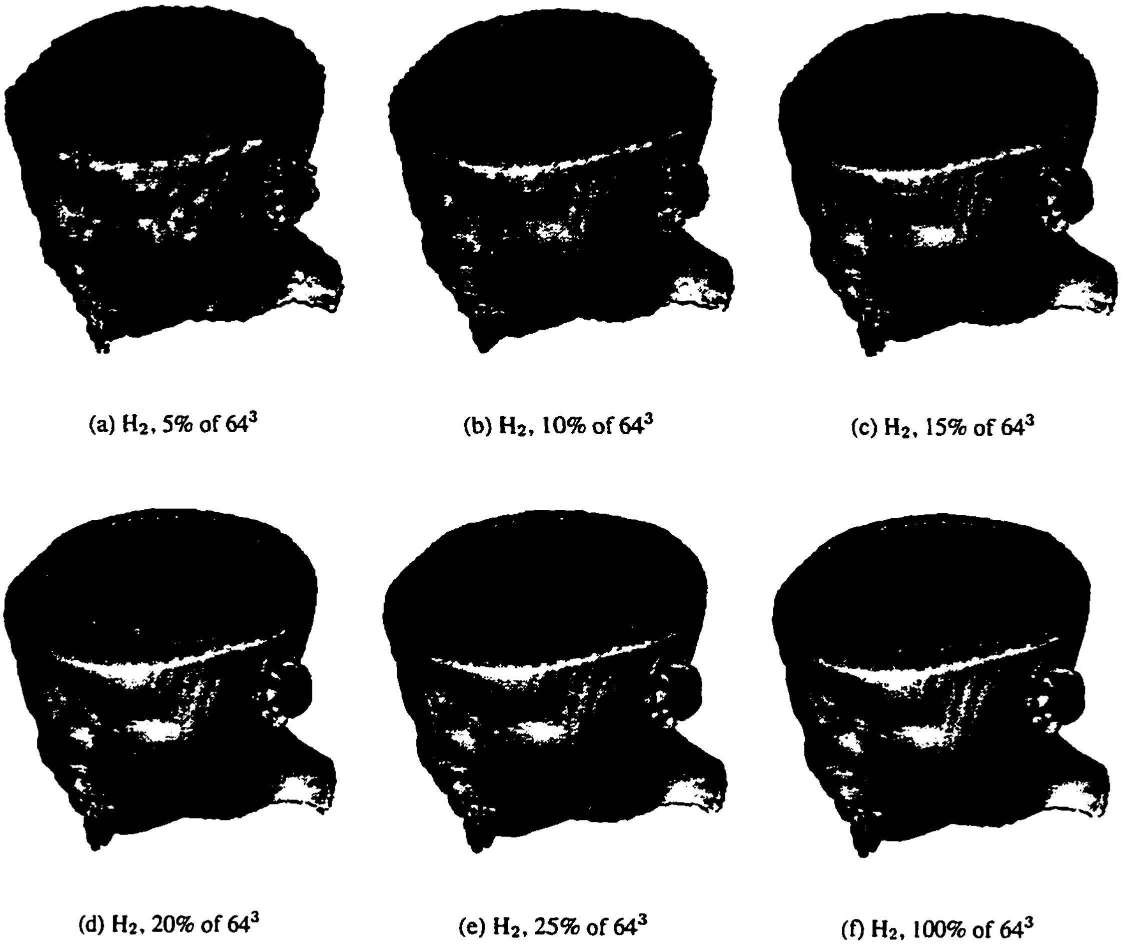
(c) H_2, 15% of 32^3

(d) H_2, 20% of 32^3

(e) H_2, 25% of 32^3

(f) H_2, 100% of 32^3

Figure 11-8: Data approximations of the head dataset generated by H_2 at a) 1635 (5% of 32×32×32), b) 3277 (10% of 32×32×32), c) 4915 (15% of 32×32×32), d) 6554 (20% of 32×32×32), e) 8192 (25% of 32×32×32), f) 32×32×32 resolutions.



(a) H_2, 5% of 64^3

(b) H_2, 10% of 64^3

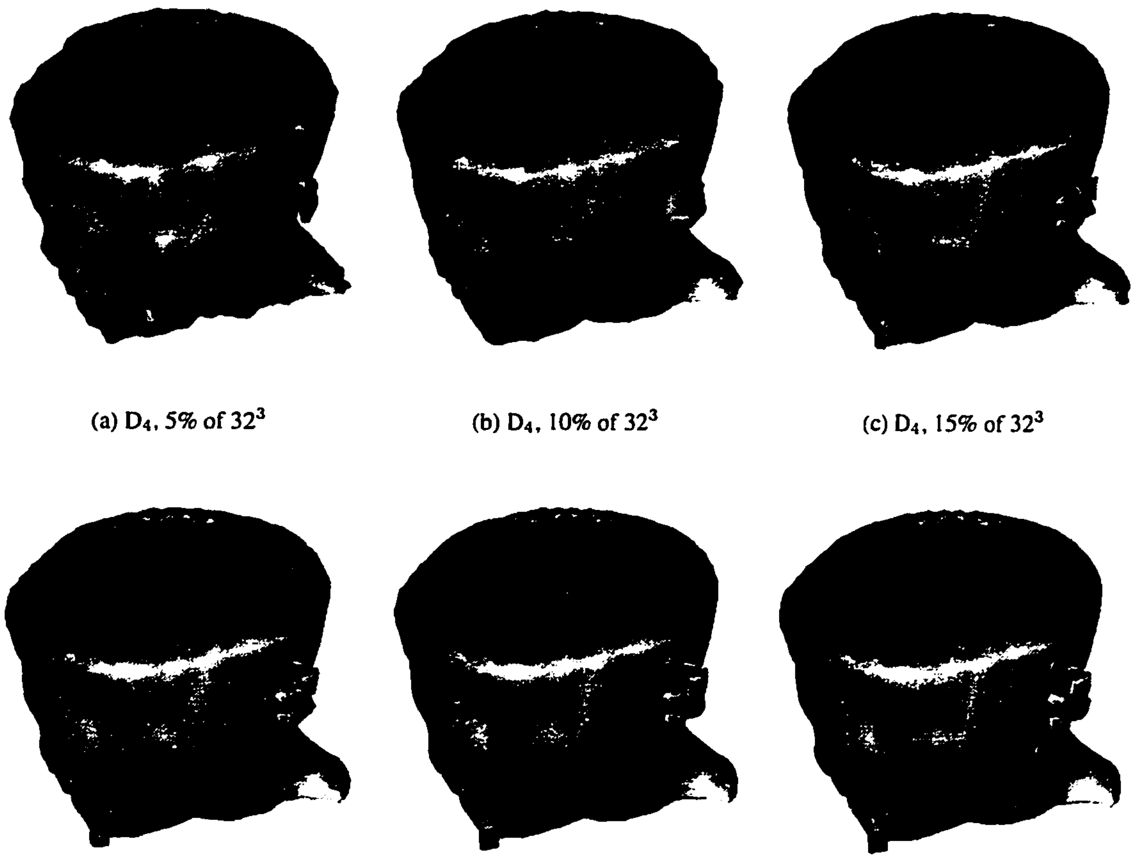
(c) H_2, 15% of 64^3

(d) H_2, 20% of 64^3

(e) H_2, 25% of 64^3

(f) H_2, 100% of 64^3

Figure 11-9: Data approximations of the head dataset generated by H_2 at a) 13107 (5% of 64×64×64), b) 26215 (10% of 64×64×64), c) 39322 (15% of 64×64×64), d) 52428 (20% of 64×64×64), e) 65536 (25% of 64×64×64), and f) 64×64×64 resolutions.



(a) D_4, 5% of 32^3

(b) D_4, 10% of 32^3

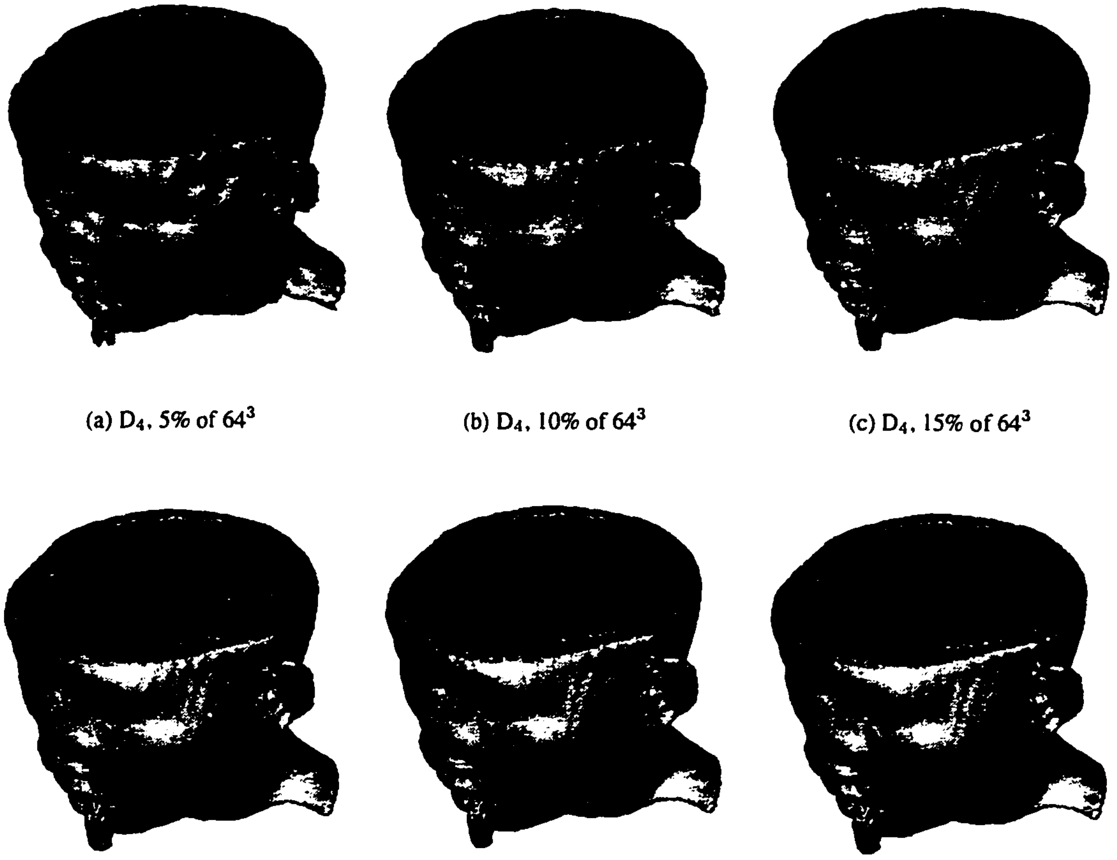
(c) D_4, 15% of 32^3

(d) D_4, 20% of 32^3

(e) D_4, 25% of 32^3

(f) D_4, 100% of 32^3

Figure 11-10: Data approximations of the head dataset generated by D_4 at a) 1635 (5% of 32x32x32), b) 3277 (10% of 32x32x32), c) 4915 (15% of 32x32x32), d) 6554 (20% of 32x32x32), e) 8192 (25% of 32x32x32), f) 32x32x32 (100% of 32x32x32) resolutions.



(a) D_4, 5% of 64^3

(b) D_4, 10% of 64^3

(c) D_4, 15% of 64^3

(d) D_4, 20% of 64^3

(e) D_4, 25% of 64^3

(f) D_4, 100% of 64^3

Figure 11-11: Data approximations of the head dataset generated by D_4 at a) 13107 (5% of 64×64×64), b) 26215 (10% of 64×64×64), c) 39322 (15% of 64×64×64), d) 52428 (20% of 64×64×64), e) 65536 (25% of 64×64×64), and f) 64×64×64 (100% of 64×64×64) resolutions.

11.3.2 CT Lobster Volume Dataset

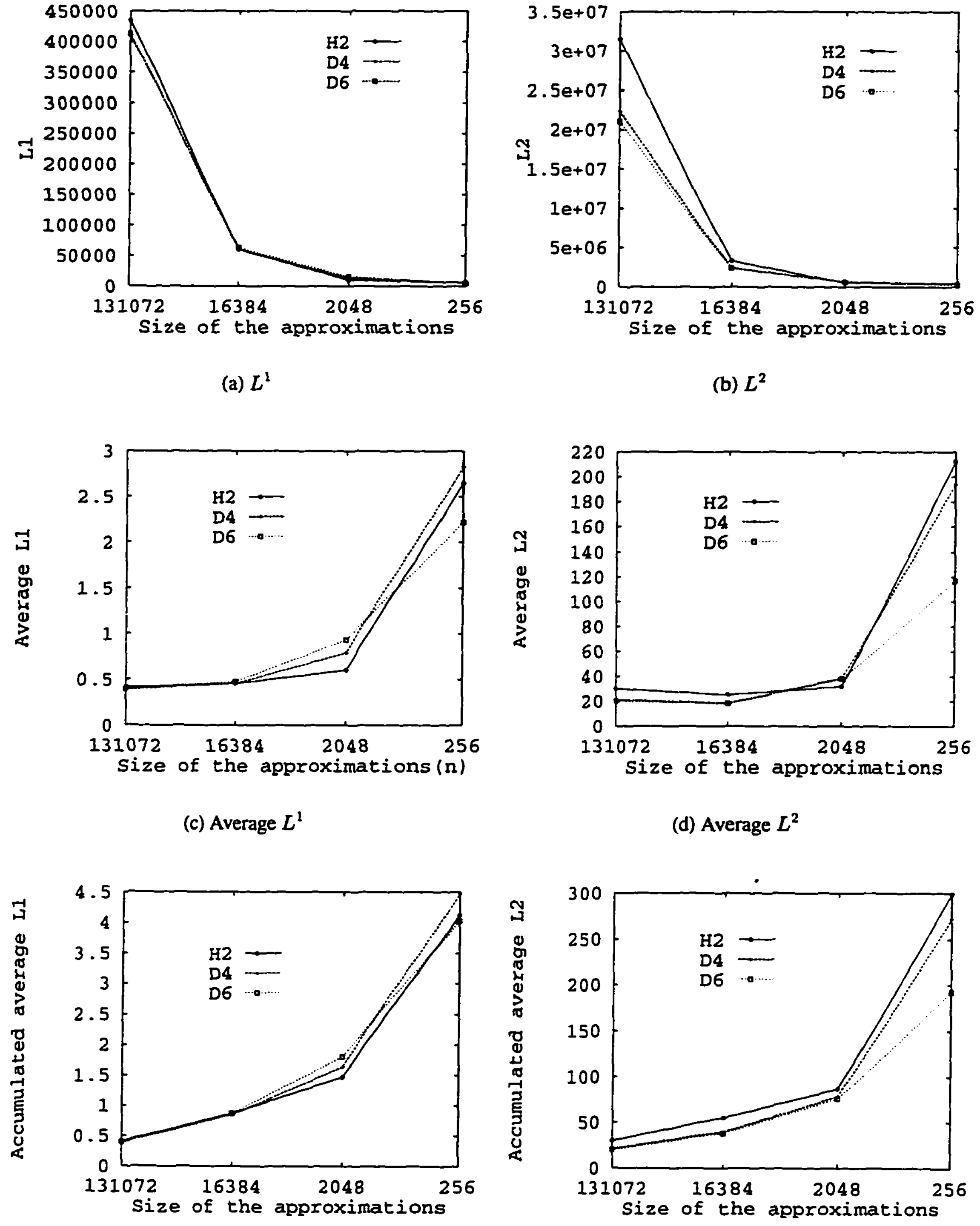
The second dataset of our study is a CT scan of a lobster as shown in Figure 11-2c. Although it is the smallest (128 × 128 × 64) among the three, its small fine details such as the lobster legs make the approximation process more challenging than the previous head dataset.

Error Estimation

After applying the same decomposition process (as the head dataset) to the data, we compute the norm values and the percentage error of the wavelet details. The results are listed in Table 11.8. Different norms are also plotted individually against the size of the approximations in Figure 11-12. It is unrealistic to try to seek a quiescent state when there are only four resolutions available in the hierarchy. However, we can still estimate the quality of an approximation by looking at the percentage error, E, of the corresponding resolution. As we can see, only approximations at the 64×64×32 resolution have percentage error $E \leq 5\%$. From our previous experience, only approximations of this resolution should be retained in the hierarchy. Among the three wavelets at 64×64×32 resolution, D_4 has the lowest percentage error and H_2 has the highest. So it is a logical

W	*Data Size*	L^1	$\overline{L^1}$	$\sum \overline{L^1}$	E (%)	L^2	$\overline{L^2}$	$\sum \overline{L^2}$	Δs
H_2	64×64×32	434969.0	0.41	0.41	5.06	31465200	30.00	30.00	18476
	32×32×16	59748.9	0.46	0.87	10.63	3307170	25.23	55.24	3756
	16×16×8	9886.6	0.60	1.47	18.00	522895	31.91	87.15	744
	8×8×4	5424.3	2.65	4.12	50.33	434919	212.36	299.52	92
D_4	64×64×32	410542.0	0.39	0.39	4.78	22358100	21.32	21.32	18464
	32×32×16	59782.5	0.46	0.85	10.3	2439650	18.61	39.94	3932
	16×16×8	12987.5	0.79	1.64	20.02	635417	33.78	78.72	776
	8×8×4	5795.3	2.83	4.47	54.58	396956	193.83	272.54	148
D_6	64×64×32	412489.0	0.39	0.39	4.8	21057700	20.08	20.08	18816
	32×32×16	62741.9	0.48	0.87	10.6	2366610	18.06	38.14	3978
	16×16×8	15310.9	0.93	1.81	22.06	623179	38.04	76.17	792
	8×8×4	4544.25	2.22	4.03	49.20	238943	116.67	192.85	154

Table 11.8: The norms, average norms, accumulated average norms, and the percentage errors of the wavelet details generated from the lobster volume dataset by different wavelets in multiple resolutions.



(a) L^1

(b) L^2

(c) Average L^1

(d) Average L^2

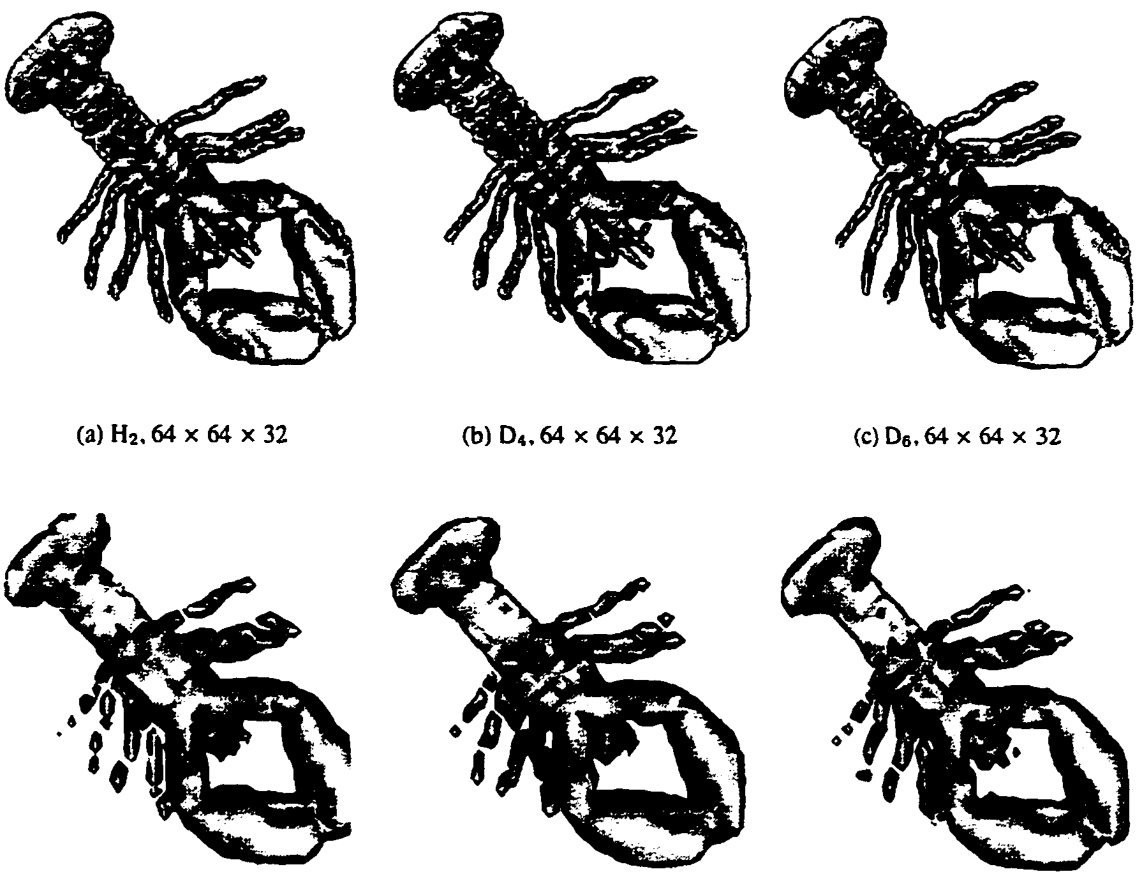
(e) Accumulated average L^1

(f) Accumulated average L^2

Figure 11-12: The a) L^1, b) L^2, c) average L^1, d) average L^2, e) accumulated average L^1, and f) accumulated average L^2 of the wavelet details generated from the lobster dataset are plotted against the sizes of the approximations.

choice to pick D_4 over D_6 with H_2 being considered for fast computation.

The visual test which includes the marching cubes renderings of the finest two resolutions (64×64×32 and 32×32×16) of all three wavelets is shown in Figure 11-13. It is clear that all approximations at the 32×32×16 resolution (Figures 11-13d, 11-13e, and 11-13f) are totally unacceptable. It is also true that the visual differences among the three approximations in Figures 11-13a, 11-13b, and 11-13c are so minor that D_6 becomes the least favorite choice because it requires the



(a) H_2, 64 × 64 × 32

(b) D_4, 64 × 64 × 32

(c) D_6, 64 × 64 × 32

(d) H_2, 32 × 32 × 16

(e) D_4, 32 × 32 × 16

(f) D_6, 32 × 32 × 16

Figure 11-13: The wavelets used to decompose the data, resolutions of the approximations, and the number of isosurface triangles generated during the renderings are listed as follows: a) H_2, 64 × 64 × 32, 18476 triangles, b) D_4, 64 × 64 × 32, 18464 triangles, c) D_6, 64 × 64 × 32, 18816 triangles, d) H_2, 32 × 32 × 16, 3756 triangles, e) D_4, 32 × 32 × 16, 3932 triangles, and f) D_6, 32 × 32 × 16, 3978 triangles.

longest processing time.

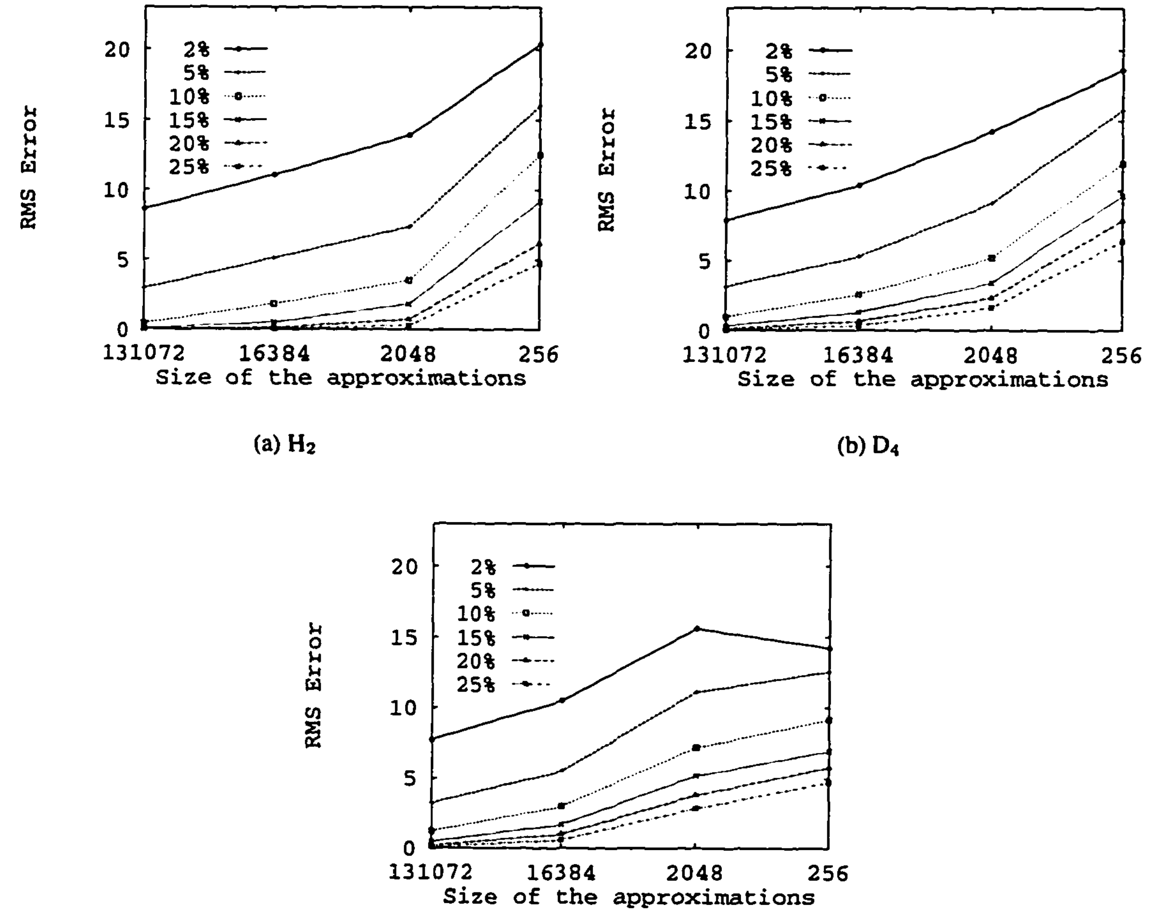
Wavelet Coefficient Truncation

The next phase involves data truncations. From Table 11.9 we are almost certain that retaining 25% (or even 20%) of the details is not necessary because of the zero RMSEs. We also see the large error gaps between retaining 2% and 5% in Figure 11-14. It seems that a retention rate of 5% (or higher) is enough to preserve the major features of the data.

Wavelet	*Data Size*	2%	5%	10%	15%	20%	25%
H_2	64×64×32	8.61	2.94	0.43	0.06	0.00	0.00
	32×32×16	11.08	5.12	1.82	0.49	0.10	0.03
	16×16×8	13.92	7.35	3.52	1.82	0.71	0.27
	8×8×4	20.39	16.06	12.54	9.19	6.19	4.76
D_4	64×64×32	7.89	3.12	0.96	0.29	0.10	0.03
	32×32×16	10.39	5.29	2.56	1.28	0.64	0.33
	16×16×8	14.25	9.13	5.20	3.39	2.32	1.62
	8×8×4	18.62	15.79	11.96	9.66	7.87	6.39
D_6	64×64×32	7.74	3.23	1.22	0.46	0.19	0.09
	32×32×16	10.47	5.52	3.00	1.70	1.00	0.60
	16×16×8	15.60	11.10	7.18	5.19	3.80	2.86
	8×8×4	14.22	12.55	9.14	6.93	5.79	4.72

Table 11.9: The RMSEs between the original lobster approximations and the corresponding reconstructions of the truncated approximations.

Figures 11-15 and 11-16 show reconstructions of truncated approximations at the 64×64×32 resolution. We do not visually detect any major differences between the truncated reconstructions (Figure 11-15a – 11-15e and 11-16a – 11-16e) and the originals (Figure 11-15f and Figure 11-16f). Thus the results agree with our previous estimation that 5% of the most significant data values are enough to preserve the details of the approximation of the lobster dataset at the 64×64×32 resolution.

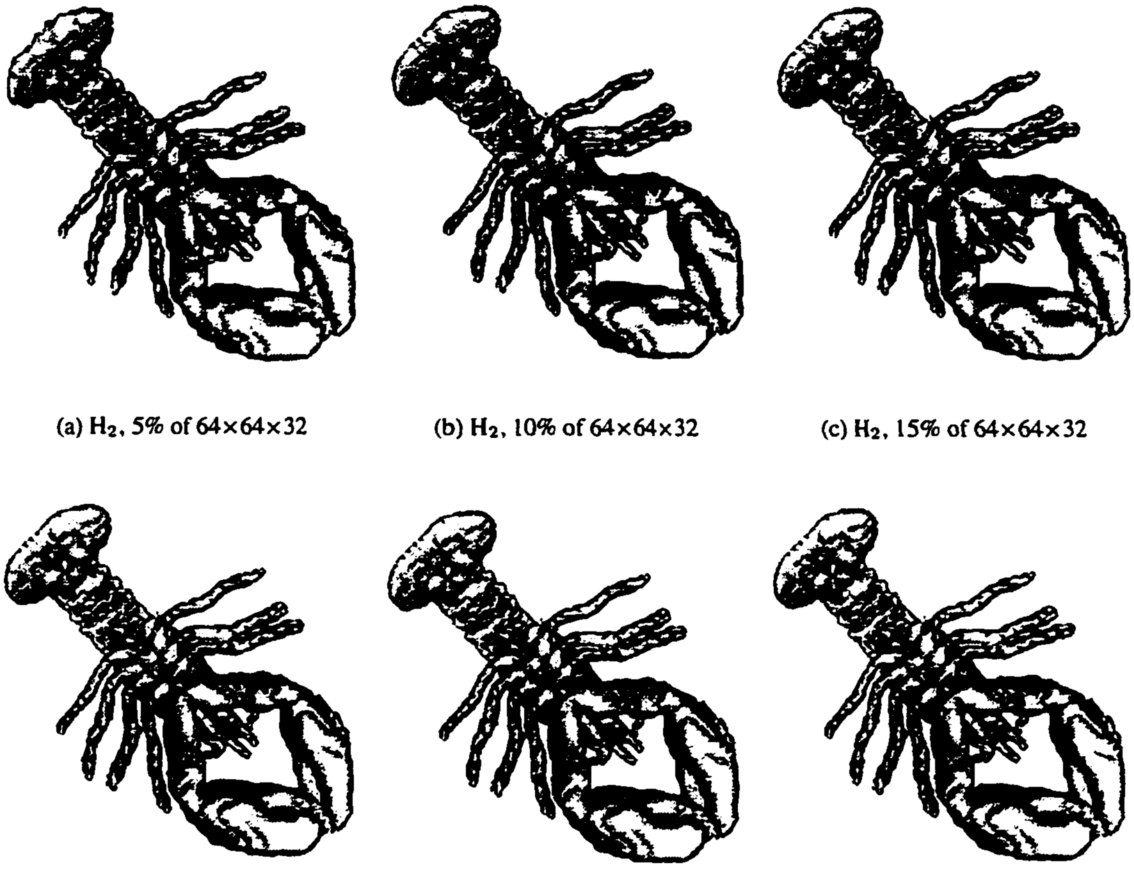


(a) H2

(b) D4

(c) D6

Figure 11-14: The RMSEs between the original lobster approximations and the reconstructions of the truncated approximations are plotted against the size of the approximations.



(a) H_2, 5% of 64x64x32

(b) H_2, 10% of 64x64x32

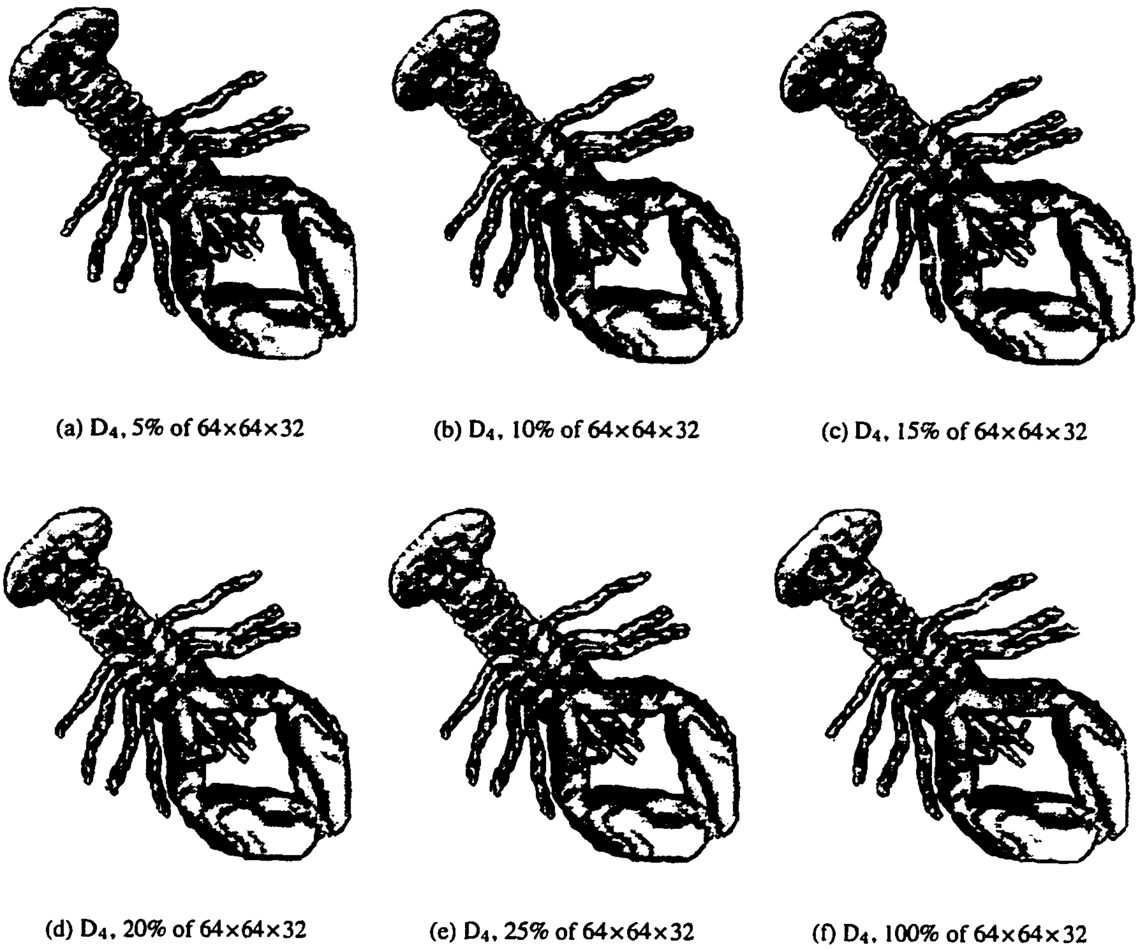
(c) H_2, 15% of 64x64x32

(d) H_2, 20% of 64x64x32

(e) H_2, 25% of 64x64x32

(f) H_2, 100% of 64x64x32

Figure 11-15: Data approximations of the lobster dataset generated by H_2 at a) 6554 (5% of 64x64x32), b) 13107 (10% of 64x64x32), c) 19661 (15% of 64x64x32), d) 26214 (20% of 64x64x32), e) 32768 (25% of 64x64x32), f) 131072 (100% of 64x64x32) resolutions.



(a) D_4, 5% of 64x64x32

(b) D_4, 10% of 64x64x32

(c) D_4, 15% of 64x64x32

(d) D_4, 20% of 64x64x32

(e) D_4, 25% of 64x64x32

(f) D_4, 100% of 64x64x32

Figure 11-16: Data approximations of the lobster dataset generated by D_4 at a) 6554 (5% of 64x64x32), b) 13107 (10% of 64x64x32), c) 19661 (15% of 64x64x32), d) 26214 (20% of 64x64x32), e) 32768 (25% of 64x64x32), f) 131072 (100% of 64x64x32) resolutions.

11.3.5 CT Engine Volume Dataset

Our third test dataset is a CT scan of an engine block. This dataset is twice the size of the head dataset, but the data content is more homogeneous.

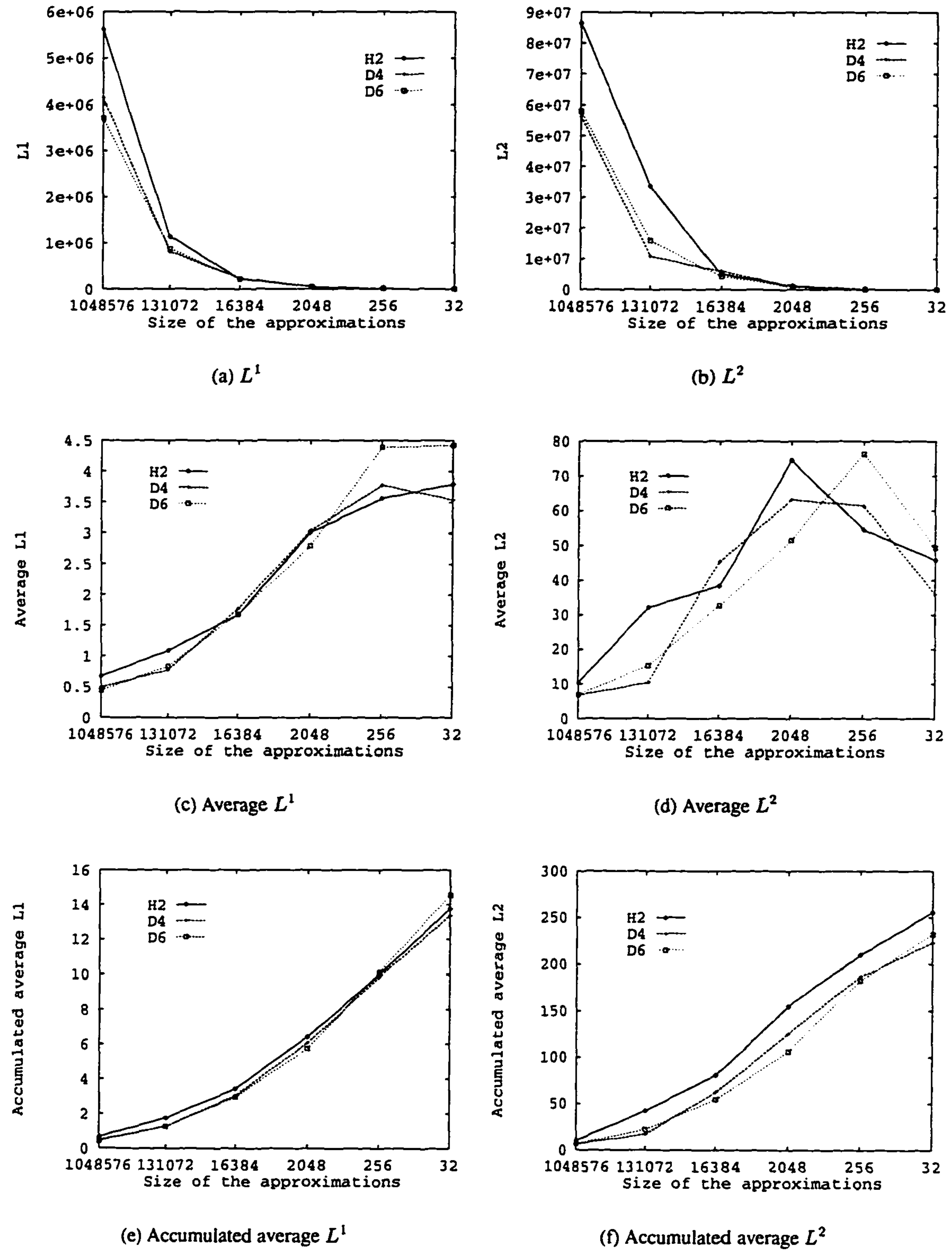
Error Estimation

From Figure 11-2 we see that the rendering of the dataset has many straight lines and shapes on its surface. This presents a different type of challenge for our error estimation process. Once again, we build a multiresolution approximation hierarchy and the corresponding norms and percentage errors are listed in Table 11.10. Individual norms are also plotted against the size of the approximations in Figure 11-17.

W	*Data Size*	L^1	$\overline{L^1}$	$\sum \overline{L^1}$	*E* (%)	L^2	$\overline{L^2}$	$\sum \overline{L^2}$	Δs
H_2	128×128×64	5615360.0	0.67	0.67	3.06	86405800.0	10.30	10.30	153668
	64×64×32	1139840.0	1.09	1.76	8.03	33636300.0	32.08	42.38	38080
	32×32×16	219226.0	1.67	3.43	15.68	5024770.0	38.34	80.71	8124
	16×16×8	49260.8	3.01	6.44	29.43	1220650.0	74.50	155.22	1590
	8×8×4	7285.9	3.56	9.99	45.70	111764.0	54.57	209.79	208
	4×4×2	970.2	3.79	13.78	63.02	11323.6	45.80	255.58	
D_4	128×128×64	4146780.0	0.49	0.49	2.26	56341500.0	6.72	6.72	155396
	64×64×32	808520.0	0.77	1.27	5.79	10866800.0	10.36	17.08	37684
	32×32×16	232733.0	1.78	3.04	13.90	59513900.0	45.41	62.49	8748
	16×16×8	49728.0	3.04	6.08	27.78	1034230.0	63.12	125.61	1666
	8×8×4	7726.2	3.77	9.85	45.03	125754.0	61.40	187.01	184
	4×4×2	905.0	3.53	13.38	61.20	9182.2	35.87	222.89	
D_6	128×128×64	3698740.0	0.44	0.44	2.01	57986800.0	6.91	6.91	153920
	64×64×32	871570.0	0.83	1.27	5.82	15969300.0	15.23	22.14	37536
	32×32×16	220664.0	1.68	2.96	13.51	4276650.0	32.63	54.77	8796
	16×16×8	45699.1	2.79	5.74	26.27	842795.0	51.44	106.21	1610
	8×8×4	8985.8	4.39	10.13	46.33	156529.0	76.43	182.64	260
	4×4×2	1132.1	4.42	14.55	66.55	12621.9	49.30	231.94	

Table 11.10: Different norm values of the wavelet details generated from the engine volume dataset by different wavelets in multiple resolutions.

Unfortunately we do not see any obvious quiescent state in this hierarchy. The percentage errors, however, indicate that the approximations smaller than 64×64×32 resolution are probably not acceptable. While D_6 has the lowest E and H_2 the highest, their values are all within reasonable



(a) L^1

(b) L^2

(c) Average L^1

(d) Average L^2

(e) Accumulated average L^1

(f) Accumulated average L^2

Figure 11-17: The a) L^1, b) L^2, c) average L^1, d) average L^2, e) accumulated average L^1, and f) accumulated average L^2 of the wavelet details generated from the engine dataset are plotted against the size of the approximations.

ranges. The finest three approximations of the wavelets are rendered using the marching cubes algorithm. The results are depicted in Figure 11-18.

Our visual test indicates that both approximations at 128×128×64 (Figures 11-18a, 11-18d, 11-18g) and 64×64×32 (Figures 11-18b, 11-18e, 11-18f) resolutions are acceptable while the approximations at 32×32×16 resolution (Figures 11-18c, 11-18f, 11-18i) are not. Since the qualities of the renderings are visually very close to each other in Figure 11-18, we choose the two faster wavelets (H_2 and D_4) to continue the truncation process.

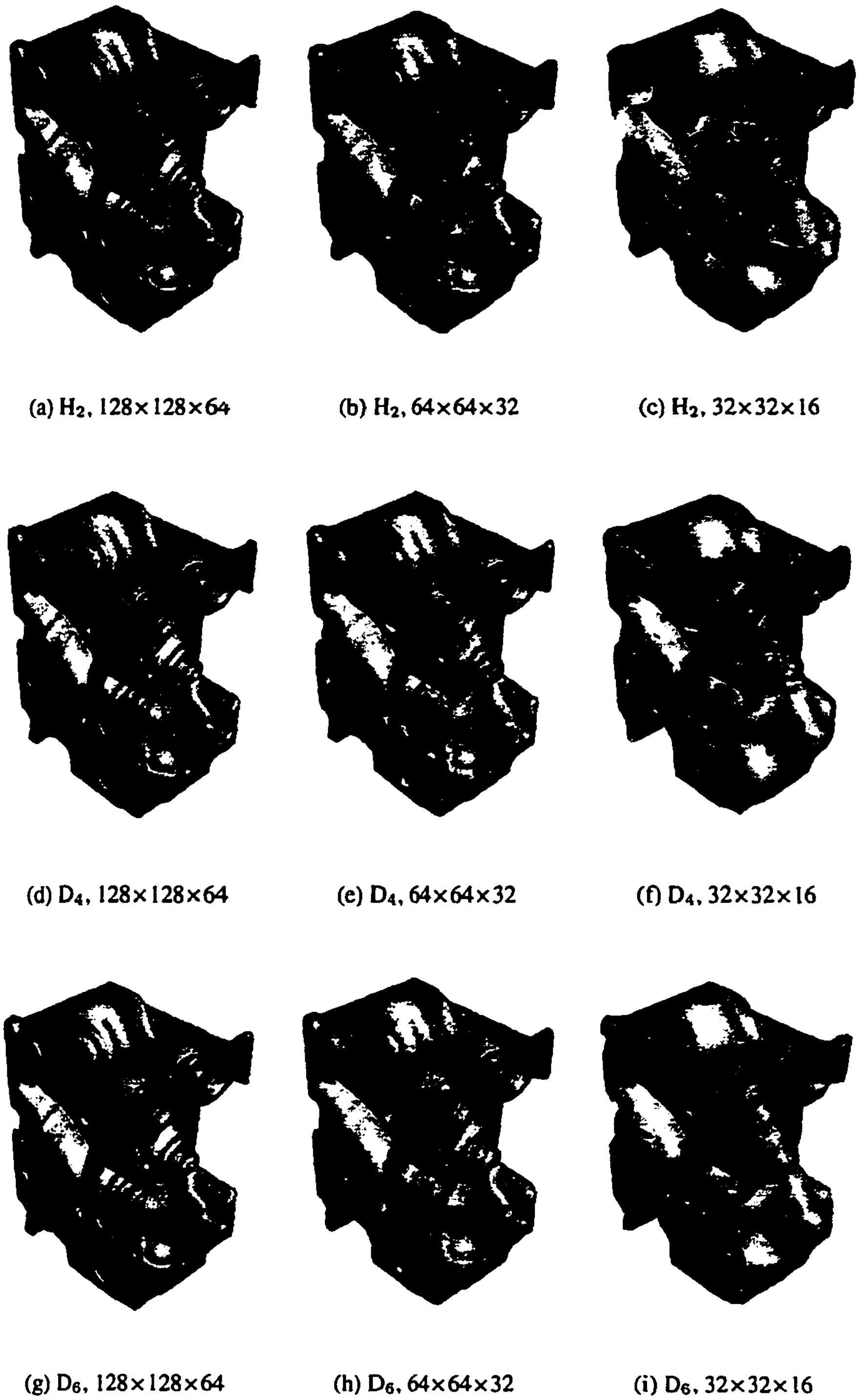
Wavelet Coefficient Truncation

After the approximations are truncated, the results are shown in Table 11.11 and Figure 11-19.

Wavelet	*Data Size*	2%	5%	10%	15%	20%	25%
H_2	128×128×64	15.55	4.57	1.79	0.92	0.56	0.40
	64×64×32	20.89	8.84	3.55	1.72	0.92	0.52
	32×32×16	29.37	18.10	8.92	5.10	3.00	1.77
	16×16×8	7.08	5.11	3.42	2.44	1.73	1.25
	8×8×4	27.77	20.05	14.64	10.96	9.11	7.39
D_4	128×128×64	10.70	3.97	1.65	0.88	0.55	0.39
	64×64×32	20.30	8.69	3.92	2.21	1.29	0.77
	32×32×16	27.27	16.33	9.38	5.98	3.94	2.62
	16×16×8	7.08	5.07	3.26	2.39	1.80	1.37
	8×8×4	25.21	16.83	12.33	9.84	7.84	6.45
D_6	128×128×64	12.60	3.91	1.72	0.93	0.58	0.42
	64×64×32	19.34	8.99	4.38	2.48	1.53	0.94
	32×32×16	25.72	15.05	8.71	5.52	3.71	2.57
	16×16×8	7.84	5.83	3.84	2.82	2.18	1.69
	8×8×4	28.11	22.16	16.16	12.33	10.13	8.16

Table 11.11: The RMSEs between the original engine approximations and the reconstructions of the truncated approximations.

There is an obvious gap between the 5% and 10% truncations at the 128×128×64 and 64×64×32 resolutions. We estimate that a retaining rate of 10% (or above) is sufficient to keep most of the important details of the corresponding approximations.

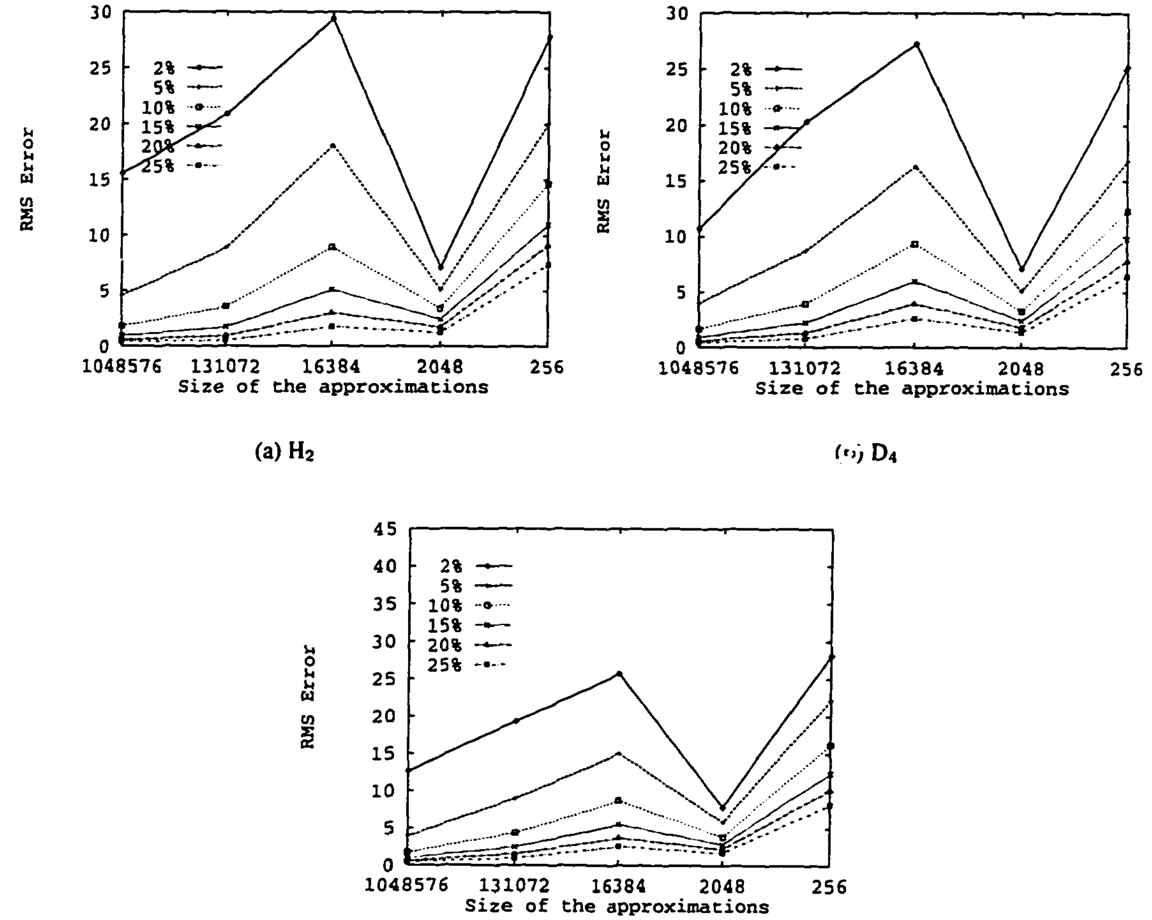


(a) H_2, 128×128×64 (b) H_2, 64×64×32 (c) H_2, 32×32×16

(d) D_4, 128×128×64 (e) D_4, 64×64×32 (f) D_4, 32×32×16

(g) D_6, 128×128×64 (h) D_6, 64×64×32 (i) D_6, 32×32×16

Figure 11-18: The wavelets used to decompose the data, resolutions of the approximations, and the number of isosurface triangles generated during the renderings are listed in order: a) H_2, 128×128×64, 153668 triangles. b) H_2, 64×64×32, 38080 triangles. c) H_2, 32×32×16, 8124 triangles. d) D_4, 128×128×64, 155396 triangles. e) D_4, 64×64×32, 37684 triangles. f) D_4, 32×32×16, 8748 triangles. g) D_6, 128×128×64, 153920 triangles. h) D_6, 64×64×32, 37536 triangles. i) D_6, 32×32×16, 8796 triangles.



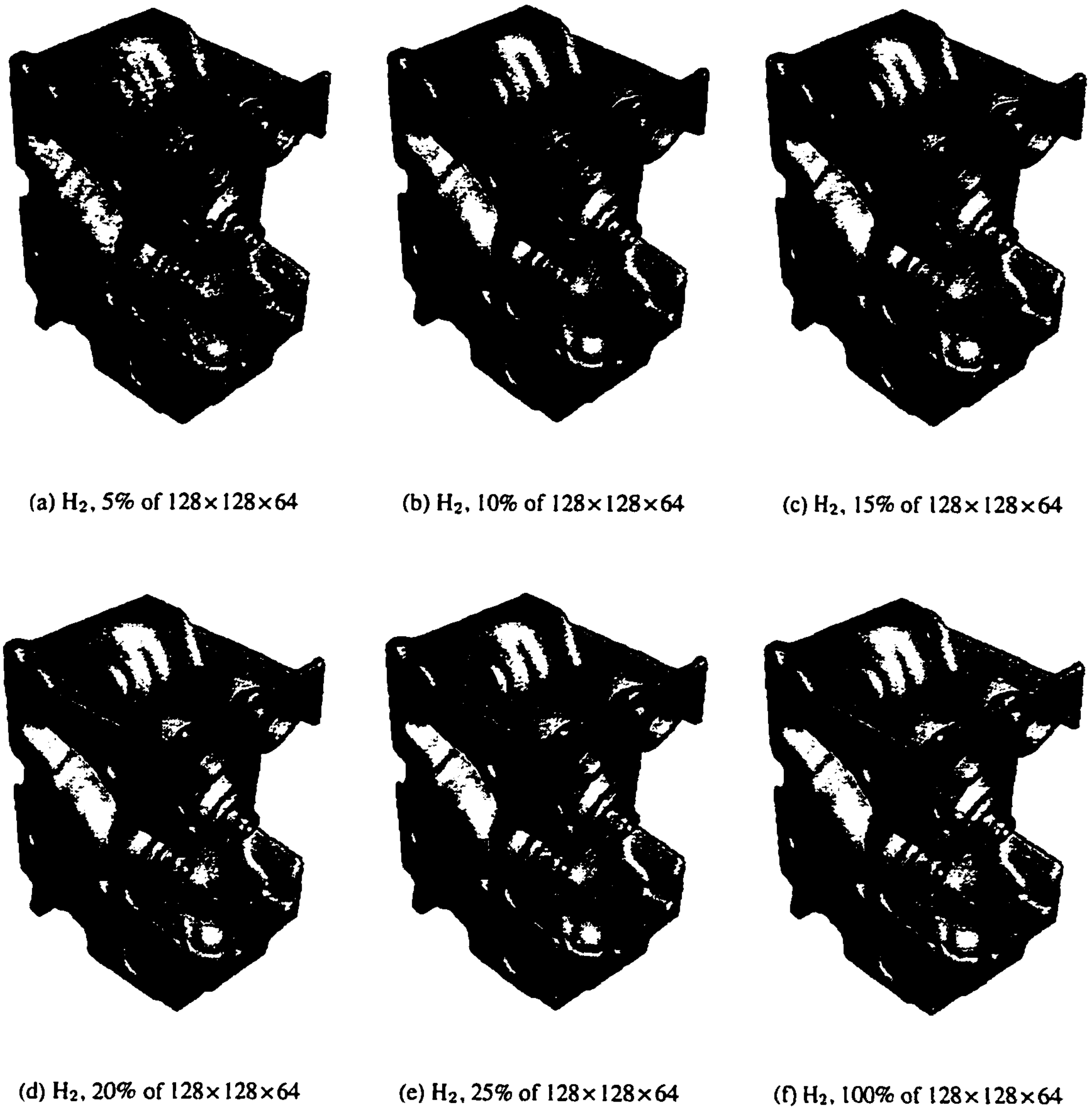
(a) H_2

(b) D_4

(c) D_6

Figure 11-19: The RMSEs between the original engine approximations and the reconstructions of the truncated approximations are plotted against the size of the approximations.

The visual test shows that most of the renderings in Figures 11-20 – 11-23 resemble the corresponding approximations in Figures 11-20f – 11-23f, except the 5% truncations in Figures 11-20a, – 11-23a.



(a) H_2, 5% of 128×128×64

(b) H_2, 10% of 128×128×64

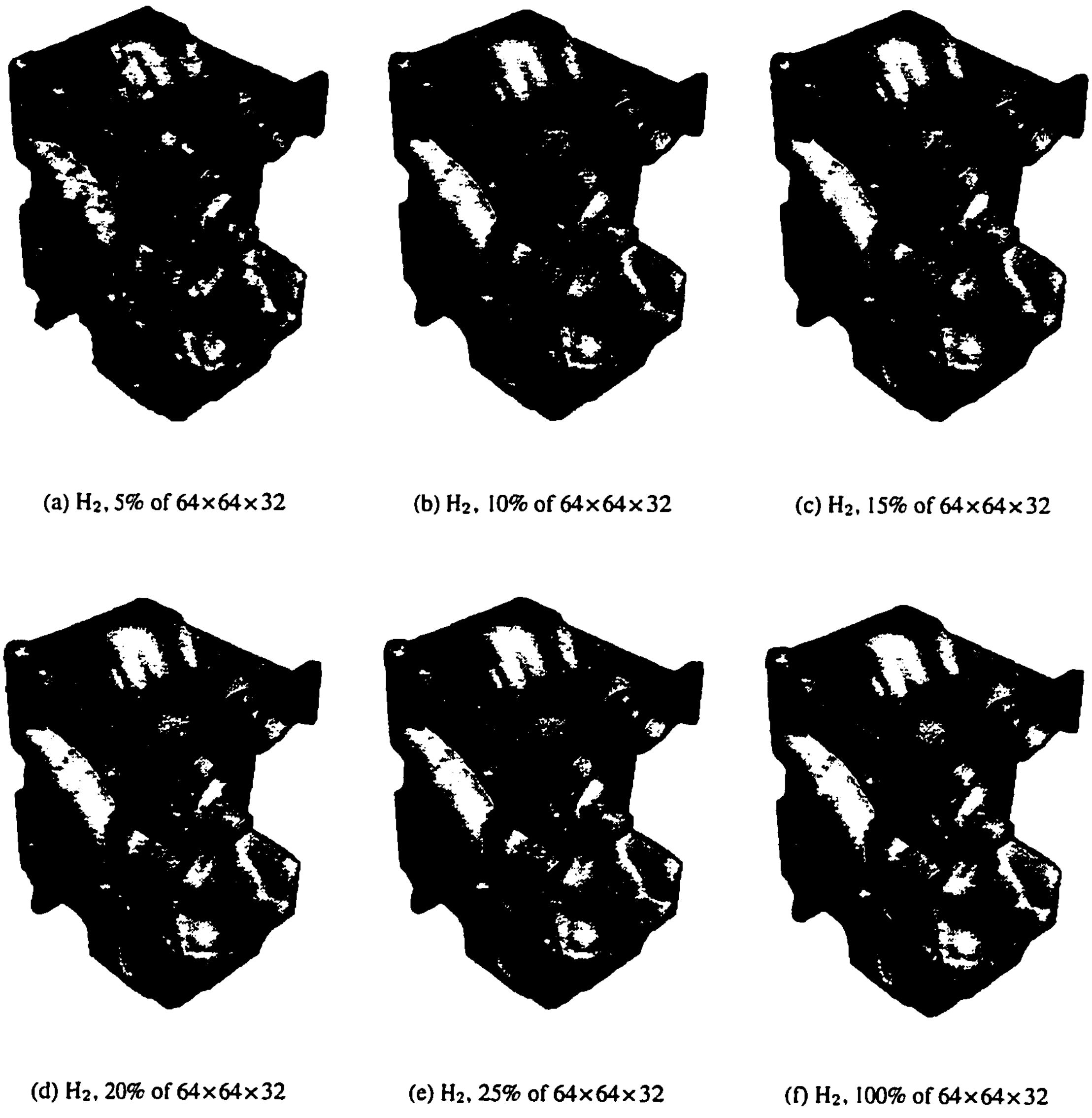
(c) H_2, 15% of 128×128×64

(d) H_2, 20% of 128×128×64

(e) H_2, 25% of 128×128×64

(f) H_2, 100% of 128×128×64

Figure 11-20: Approximations of the engine dataset generated by H_2 at a) 5% (52429), b) 10% (104858), c) 15% (157286), d) 20% (209715), e) 25% (32768), and f) 100% of 128×128×64 resolution.



(a) H_2, 5% of 64×64×32

(b) H_2, 10% of 64×64×32

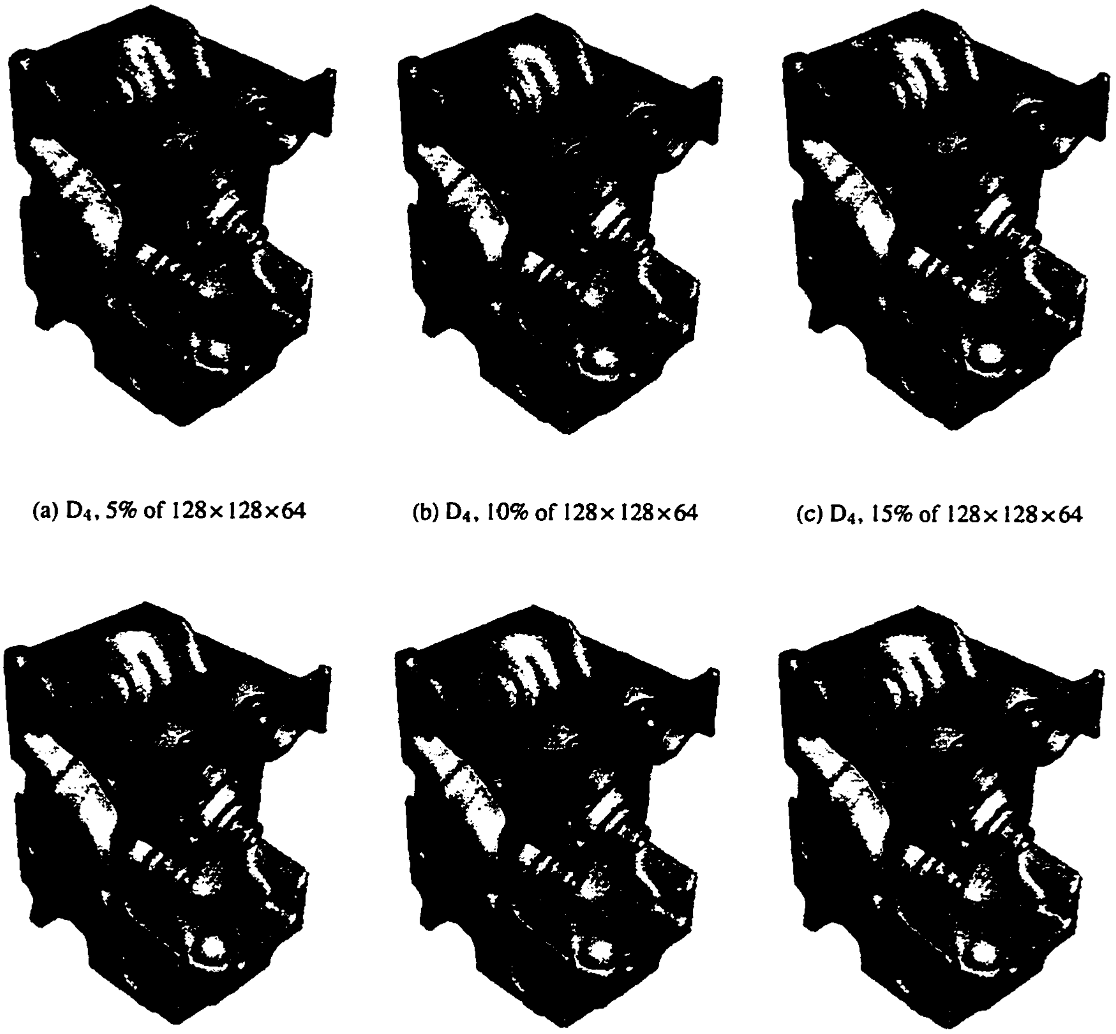
(c) H_2, 15% of 64×64×32

(d) H_2, 20% of 64×64×32

(e) H_2, 25% of 64×64×32

(f) H_2, 100% of 64×64×32

Figure 11-21: Approximations of the engine dataset generated by H_2 at a) 5% (6554), b) 10% (13107), c) 15% (19661), d) 20% (26214), e) 25% (32768), f) 100% of 64×64×32 resolutions.



(a) D_4, 5% of 128×128×64

(b) D_4, 10% of 128×128×64

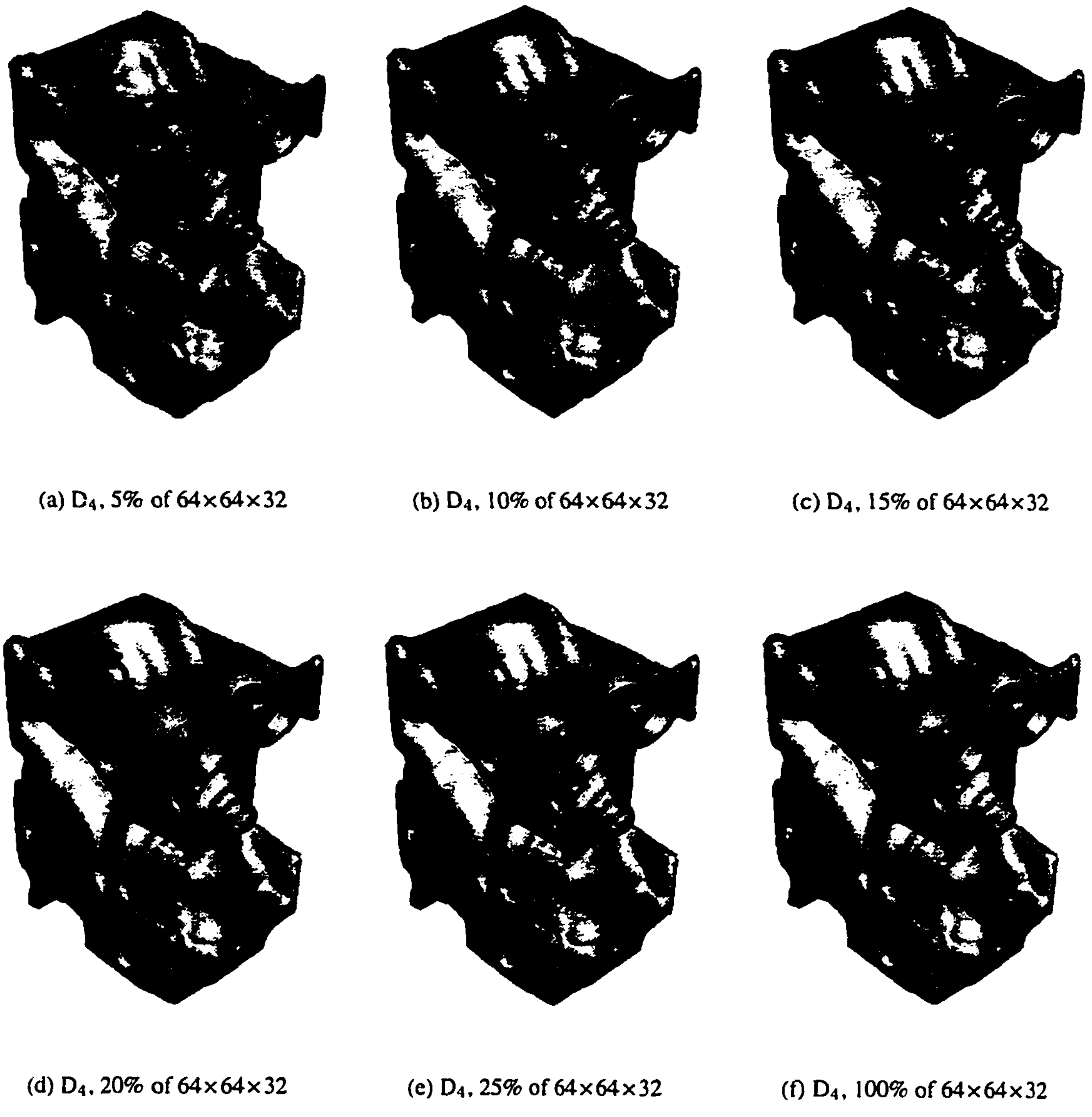
(c) D_4, 15% of 128×128×64

(d) D_4, 20% of 128×128×64

(e) D_4, 25% of 128×128×64

(f) D_4, 100% of 128×128×64

Figure 11-22: Approximations of the engine dataset generated by D_4 at a) 5% (52429), b) 10% (104858), c) 15% (157286), d) 20% (209715), e) 25% (32768), and f) 100% of 128×128×64 resolution.



(a) D_4, 5% of 64×64×32

(b) D_4, 10% of 64×64×32

(c) D_4, 15% of 64×64×32

(d) D_4, 20% of 64×64×32

(e) D_4, 25% of 64×64×32

(f) D_4, 100% of 64×64×32

Figure 11-23: Approximations of the engine dataset generated by D_4 at a) 5% (6554), b) 10% (13107), c) 15% (19661), d) 20% (26214), e) 25% (32×32×32), f) 100% of 64×64×32 resolutions.

11.4 Space/Time Trade-off

This section discusses the space/time trade-offs of approximation reconstructions within a multiresolution hierarchy. Since the wavelet filters used for the transforms are designed to be orthogonal, the filter matrix used for reconstruction is simply the transposed version of the one used for decomposition. Consequently, the time for data decomposition is about the same as the time for data reconstruction at the corresponding resolution. In general, however, we consider the time required for decomposition to be a pre-processing step whose cost is not significant. On the other hand, we can save space by reconstructing the approximation component of intermediate resolutions from the lower level details. Since this is needed during interactive exploration, the decision to eliminate resolution levels represents a classic time/space trade-off.

Table 11.12 contains the CPU and the real clock times of the data reconstructions of the three volume datasets. Suppose the data is stored in 32-bit floating point format, the largest approximation in Table 11.12 (i.e., the engine dataset at 128×128×64 resolution) occupies about 4MB of disk space. However, the reconstruction of the same dataset requires almost 10s of CPU time and almost 12 minutes of real clock time on a 133MHz Linux machine with 32MB of memory. This is clearly unacceptable to support real time data exploration. Even reconstruction times on the order of 1–2 minutes (real time) are inadequate for most interactive purposes. On the other hand, we can develop

Data	*Reconstruction Dimensions*	*CPU (s)*			*Real (s)*		
Set	*From → To*	H_2	D_4	D_6	H_2	D_4	D_6
Lobster	32x32x16 → 64x64x32	0.71	0.93	0.93	58.8	62.4	76.8
	16x16x8 → 32x32x16	0.13	0.17	0.17	10.2	10.8	24.6
Head	32x32x32 → 64x64x64	1.42	1.89	2.91	120.0	138.6	213.6
	16x16x16 → 32x32x32	0.20	0.25	0.40	16.8	22.2	31.8
	8x8x8 → 16x16x16	0.07	0.08	0.10	4.2	5.4	6.0
Engine	64x64x32 → 128x128x64	5.74	7.67	9.76	373.2	483.6	712.2
	32x32x16 → 64x64x32	0.77	1.02	1.28	48.0	62.4	121.2
	16x16x8 → 32x32x16	0.13	0.15	0.20	11.4	9.6	15.6

Table 11.12: Data sizes and CPU times of the multiresolution reconstructions of the volume datasets.

heuristics for an adaptive multiresolution hierarchy that supports a variety of storage life times for different levels of the hierarchy. For example, at the beginning of an exploration session, we could generate more *temporary* intermediate stages to support smooth interactive transitions of a multiresolution hierarchy. These approximations may be discarded, or further compressed, by the end of the exploration session.

11.5 Discussion

Empirically our multiresolution data hierarchy approach seems to deliver very good results. The sizes of the approximations in Figure 11-23b and 11-22b are only 0.156% and 1.25% respectively of the original dataset in Figure 11-2c. Each of these figures is a reasonably good approximation at their corresponding resolution levels.

In a client-server network environment, a coarse representation can increase the efficiency of a visualization process by giving users at the server ends a rough idea of the dataset very quickly. They can then make a decision based on this approximation about whether to continue the transmission or to abort it. The data transmission time over the network can further be reduced by truncating the coarse approximations. In a stand-alone environment, it is also possible to speed up the modeling and rendering time by using a coarse approximation. Even though a truncated approximation will eventually increase the processing time by "inflating" the data back to its original size, it does save disk space. For example, the storage cost of Figure 11-22b is about the same as Figure 11-23f. Although the authenticity of Figure 11-22b is better, the cost of rendering is also higher because the data must be expanded to 128×128×64 before rendering.

The use of percentage error to choose approximations out of a multiresolution hierarchy requires extensive knowledge of the datasets. In our investigation, we have conducted tests on specific datasets with very different characteristics. For example, the lobster dataset has many small and fine details such as the lobster legs which can fade away easily. Although the engine dataset has a very homogeneous data distribution, it also contains many geometric shapes and lines which can easily be distorted. The human head, on the other hand, is very heterogenous and contains many fine details. Our results show that approximations of heterogenous datasets can still be useful with about 5% error while approximations of more homogeneous datasets can tolerate up to about 8%. Since our design performs reasonably well with these datasets, we expect the performance will be improved if we apply it to smoother datasets such as those collected from computational fluid dynamics (CFD) simulations.

Chapter 12

Conclusion

Much of the work presented in this thesis centers on interactive visualizations which bring out information associated with very large datasets. Our contributions include adaptive representations of large datasets, an authenticity measure for data approximation, and interactive techniques to visualize large amounts of data. Our methods are especially applicable to the emerging scientific data needs of medical, atmospheric, oceanographic, and geographic studies.

12.1 Model Overview

Most of the work to develop a progressive visualization environment involves finding a reasonable set of approximations which represent a large dataset at multiple resolutions. After an approximation data hierarchy is generated, the first step is to reduce the number of resolutions of the hierarchy by discarding the unnecessary ones. It starts with the calculations of norm values and percentage error of the wavelet details of each resolution. The goal is to find quiescent states of a hierarchy in which major features of the approximations are roughly the same. Ideally, we only need to maintain a few lower resolution representations from a set of approximations that are part of a single quiescence. If such a phenomenon cannot be identified, we can look at the calculated percentage error of the approximations and discard the less important resolutions.

The second step is to reduce the size of the selected approximations from the previous step. This process retains the most significant data values of the approximations and discards the rest. While the first step reduces the application size of a dataset, the second step only reduces the storage size. In a client-server visualization environment, they both can speed up the data transmission time and improve the efficiency of the visualization process.

We have seen in Chapter 11 that a remarkable number of data values can be eliminated from a volume dataset using our framework. It seems natural to expect that our approach can be extended to other types of data defined on non-Cartesian grids. This might happen by an implementation of a stronger error estimation mechanism which can handle datasets with contents from very smooth to very rough. In any case, we think that this approach will become a useful part of very large data management research.

12.2 Contributions

We have applied our concept of multiresolution data representation to many very large data visualization applications. For most cases, we have new, practical results.

After showing that most of the multidimensional multivariate visualization techniques perform pcorly on large datasets in Chapter 3, we developed a general data representation solution that is independent of visualization technique in Chapter 7. Besides the straightforward increases in the number of data values being explored, this result addresses a long-standing open problem in which the visualization of large datasets is restricted by screen resolution.

In Chapter 8 we have found that many prevailing scientific volume visualization techniques are supported by expensive hardware and software. Such a costly requirement has hindered the development of general volume visualization applications for widespread use. We have implemented a set of visualization tools based on the notion of multiresolution brushing using public domain libraries running on an average Linux machine. Empirically, our system allows scientists to obtain information from a large dataset with over 8.3 million numbers in real time. These tools show that practical volume visualization can be done without expensive software and hardware.

Our research on wavelet based multiresolution visualization has led to some interesting extensions regarding the building of data hierarchies. We developed a second multiresolution hierarchy based on norm projections of the data in Chapter 9. We have observed that combining a norm projection hierarchy with a wavelet hierarchy can improve the exploratory power by providing more options for querying the data.

We have also applied the first general formulation of wavelet error estimation to the field of multivariate visualization in Chapters 5 and 11. Our algorithm, which requires few extra computational steps, produces a multiresolution error representation that reflects the information loss due to data decomposition at every level. Our solution has numerous applications to many wavelet based processes in graphics and visualization because of its general nature and applicability.

We have investigated metric scaling of large multivariate data, where we seek a low dimensional data overview with reduced data dimensions, reduced data size, and additional data semantics. We have successfully combined this data overview with various multivariate techniques to form a powerful visualization environment guided by the principal coordinates of the data in Chapter 10. Preliminary results have shown that this data overview also provides new opportunities to create new shape and texture patterns for conventional iconographic technology.

12.3 Future Research

Many interesting problems present themselves for future work. Will the quiescence phenomenon be repeated regularly within a hierarchy? Is it possible to develop error estimation mechanisms for non-orthogonal wavelets? These seem like especially interesting directions to try, given the results of this thesis.

Is it possible to apply wavelets to non-Cartesian datasets such as curvilinear data? How does the performance of wavelet decomposition compare to other cell decimation techniques applied to three dimensionai datasets? Can we use wavelets to improve the performance of data connectivity and decimation in isosurface rendering? These would seem to be potential tasks.

Can the multiresolution data representation model be extended to other data types to which it is seemingly difficult to apply any scale measures? Can we develop a new distance formulation which can be applied to such diverse data types as binary, nominal, and ordinal data to produce multiresolution data representation? Multidimensional scaling is definitely a beginning. The theory of multiresolution data representation looms large behind the design and analysis presented in this thesis, and approximating datasets without explicit connectivity will likely require significantly different techniques. This area of research has tremendous prospects, with potential applications to a wide variety of datasets such as text, audio, and video in the age of the information superhighway.

Bibliography

[BC87] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29:127–142, 1987.

[BCH$^+$95] R. Daniel Bergeron, William Cody, William Hibbard, David T. Kao, Kristina Miceli, Lloyd Treinish, and Sandra Walther. Database issues for data visualization: developing a data model. In John P. Lee and Georges G. Grinstein, editors, *IEEE Visualization '93 Workshop on Database Issues for Data Visualization*, pages 3–15. Springer Verlag, October 1995.

[Bed92] Jeff Beddow. An overview of multidimensional visualization: Elements and methods. In Jeff Beddow and Cliff Beshers, editors, *Designing a Visualization Interface for Multidimensional Multivariate Data.* IEEE Visualization '92 Tutorial 8, October 1992.

[BMMS91] Andreas Buja, John A. McDonald, John Michalak, and Werner Stuetzle. Interactive data visualization using focusing and linking. In Gregory M. Nielson and Larry Rosenblum, editors, *Proceedings of IEEE Visualization '91*, pages 156–163, Los Alamitos, California, October 1991. IEEE Computer Society Press.

[BPS96] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. Fast Isocontouring For Improved Interactivity. In Roger Crawfis and Charles Hansen, editors, *Proceedings of 1996 Symposium on Volume Visualization*, pages 39–46, New York, NY, October 1996. ACM SIGGRAPH.

[Bre69] Norman M. Brenner. Fast Fourier Transform of externally stored data. *IEEE Transactions on Audio and Electroacoustics*, AU–17(2):128–132, June 1969.

[BW96] Chris L. Bentley and Matthew O. Ward. Animating Multidimensional Scaling to Visualize N-Dimensional Data Sets. In Stuart Card, Stephen G. Eick, and Nahum Gershon, editors, *Proceedings of IEEE Information Visualization '96*, pages 72–73, Los Alamitos, California, Oct 1996. IEEE Computer Society Press.

[CC94] Trevor F. Cox and Michael A. Cox. *Multidimensional Scaling*. Monographs on Statistics and Applied Probability. Chapman & Hall, London, 1994.

[CCH95] Dianne Cook, Javier Cabrera, and Catherine Hurley. Grand Tour and Projection Pursuit. *Journal of Computational and Graphical Statistis*, 4(3):155–172, 1995.

[CCKT83] J. M. Chambers, William S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis.* Chapman and Hall, New York, 1983.

[Cle93] William S. Cleveland. *Visualizing Data.* Hobart Press, Summit, New Jersey, 1993.

[Cra96] Andrew H. Crabb. Slice: A High-Dimensional Visualization Program. Master's thesis, Department of Computer Science, University of New Hampshire, Durham, NH, May 1996.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets.* SIAM, Philadelphia, Pennsylvania, 1992.

[Dau93] Ingrid Daubechies. Wavelet transforms and orthonormal wavelet bases. In Ingrid Daubechies, editor, *Different Perspectives on Wavelets*, volume 47 of *American Mathematical Society Short Course*, pages 1–33. American Mathematical Society, San Antonio, Texas, January 1993.

[Dav83] Mark L. Davison. *Multidimensional Scaling.* Wiley series in Probability and Mathematical Statistics. Wiley, New York, 1983.

[DM90] Lokenath Debnath and Piotr Mikusinski. *Introduction to Hilbert Spaces with Applications.* Academic Press, 1990.

[FB90] Steven Feiner and Clifford Beshers. Visualizing n-dimensional virtual worlds with n-Vision. *Computer Graphics*, 24(2):37–38, March 1990.

[Flo72] Robert W. Floyd. Permuting information in idealized two-level storage. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 105–109. Plenum Press, 1972.

[FvDFH90] James Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics – Principles and Practice.* The Systems Programming Series. Addision-Wesley, second edition, 1990.

[Gal95] Richard S. Gallagher, editor. *Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis.* CRC Press, 1995.

[God92] Goddard Space Flight Center, National Aeronautics and Space Administration, Greenbelt, Maryland. *International Solar–Terrestrail Physics (ISTP) Key Parameter Generation Software (KPGS) Standards and Conventions – Version 1.1*, December 1992.

[GW87] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing.* Addision Wesley, 2 edition, 1987.

[HB90] Catherine Hurley and Andreas Buja. Analyzing High-Dimensional Data with Motion Graphics. *SIAM Journal on Scientific and Statistical Computing*, 11(6):1193–1211, 1990.

[ID87] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates for visualizing multidimensional geometry. In T. L. Kunii, editor, *Proceedings of Computer Graphics International '87*, Tokyo, 1987. Springer-Verlag.

[ID90] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In Arie Kaufman, editor, *Proceedings of IEEE Visualization '90*, pages 361–375, Los Alamitos, California, October 1990. IEEE Computer Society Press.

[IRC87] A. Inselberg, M. Reif, and T. Chomut. Convexity algorithms in parallel coordinates. *Journal of ACM*, 34(4):765–801, October 1987.

[Jac91] J. Edward Jackson. *A User's Guide to Principal Components.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1991.

[KK93] Peter R. Keller and Mary M. Keller. *Visual Cues, Practical Data Visualization.* IEEE Computer Society Press, 1993.

[KK94] Daniel A. Keim and Hans-Peter Kriegel. VisDB: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications,* 14(5):40–49, September 1994.

[KKS93] Daniel A. Keim, Hans-Peter Kriegel, and Thomas Seidl. Visual Feedback in Querying Large Database. In Gregory M. Nielson and R. Daniel Bergeron, editors, *Proceedings IEEE Visualization '93,* pages 158–165, Los Alamitos, California, October 1993. IEEE Computer Society Press.

[Knu73] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison-Wesley, Reading, MA, 1973.

[LC87] William E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics,* 21(3):163–169, July 1987.

[LH92] Haim Levkowitz and Gabor T. Herman. Color scales for image data. *IEEE Computer Graphics and Applications,* 12(1):72–80, January 1992.

[LWW90] Jeffrey LeBlanc, Matthew O. Ward, and Norman Wittels. Exploring n-dimensional databases. In Arie Kaufman, editor, *Proceedings of IEEE Visualization '90,* pages 230–237, Los Alamitos, California, October 1990. IEEE Computer Society Press.

[Mal89] Stephane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 11(7):674–693, July 1989.

[MDB87] Bruce H. McCormick, Thomas A. DeFanti, and Maxine D. Brown. Visualization in Scientific Computing. *Computer Graphics,* 21(6):1–14, November 1987.

[MGTS90] Ted Mihalisin, E. Gawlinski, John Timlin, and John Schwegler. Visualizing a scalar field on an n-dimensional lattice. In Arie Kaufman, editor, *Proceedings of IEEE Visualization '90,* pages 255–262, Los Alamitos, California, October 1990. IEEE Computer Society Press.

[MMM$^+$93] P. A. Mayewski, L. D. Meeker, M. C. Morrison, S. Whitlow, K. K. Ferland, D. A. Meese, M. R. Legrand, and J. P. Steffensen. Greenland icecore signal characteristics offer expanded view of climate change. *Journal of Geophysics,* 98:12839–12847, 1993.

[Mur92] Shigeru Muraki. Application and rendering of volume data using wavelet transforms. In Arie E. Kaufman and Gregory M. Nielson, editors, *Proceedings IEEE Visualization '92,* pages 21–28, Los Alamitos, California, October 1992. IEEE Computer Society Press.

[Mur93] Shigeru Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics and Applications,* 13(4):50–56, July 1993.

[MW95] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In Gregory M. Nielson and Deborah Silver, editors, *Proceedings IEEE Visualization '95*, pages 271–278, Los Alamitos, California, October 1995. IEEE Computer Society Press.

[MZ92] Stephane G. Mallat and Sifen Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, July 1992.

[NH90] Gregory M. Nielson and Bernd Hamann. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. In Gregory M. Nielson and Larry Rosenblum, editors, *Proceedings of IEEE Visualization '90*, pages 83–91, Los Alamitos, California, October 1990. IEEE Computer Society Press.

[NMH97] Gregory M. Nielson, Heinrich Mueller, and Hans Hagen, editors. *Scientific Visualization Overviews, Methodologies and Techniques*. IEEE Computer Society Press, February 1997.

[NR91] Gregory M. Nielson and Larry Rosenblum, editors. *Proceedings IEEE Visualization '91*, Los Alamitos, California, October 1991. IEEE Computer Society Press.

[PG88] Ronald M. Pickett and Georges G. Grinstein. Iconographics displays for visualizing multidimensional data. In *Proceedings IEEE Conference on Systems, Man, and Cybernetics*, pages 514–519, Beijing and Shenyang, PRC, May 1988.

[PK96] Hanspeter Pfister and Arie Kaufman. Cube-4 – A Scalable Architecture for Real-Time Volume Rendering. In Roger Crawfis and Charles Hansen, editors, *Proceedings of 1996 Symposium on Volume Visualization*, pages 47–54, New York, NY, October 1996. ACM SIGGRAPH.

[REE$^+$94] L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors. *Scientific Visualization: Advances and Challenges*. Academic Press, 1994.

[RL87] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, 1987.

[SFGF72] J. H. Siegel, E. J. Farrell, R. M. Goldwyn, and H. P. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery*, 72:126–141, 1972.

[Sin67] Richard C. Singleton. A method for computing the Fast Fourier Transform with auxiliary memory and limited high-speed storage. *IEEE Transactions on Audio and Electroacoustics*, AU–15(2):91–98, June 1967.

[SMK96] Claudio Silva, Joseph S. B. Mitchell, and Arie Kaufman. Fast Rendering of Irregular Grids. In Roger Crawfis and Charles Hansen, editors, *Proceedings of 1996 Symposium on Volume Visualization*, pages 15–22, New York, NY, October 1996. ACM SIGGRAPH.

[SML96] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit*. Prentice Hall PTR, New Jersey, 1996.

[STDS95] Bob Spence, Lisa Tweedie, Huw Dawkes, and Hua Su. Visualization For Functional Design. In Nahum Gershon and Steve Eick, editors, *Proceedings of IEEE Information Visualization '95*, pages 4–9, Los Alamitos, California, Oct 1995. IEEE Computer Society Press.

[Str89] Gilbert Strang. Wavelets and dilation equations: A brief introduction. *SIAM Review*, 31(4):614–627, December 1989.

[TSDS96] Lisa Tweedie, Robert Spence, Huw Dawkes, and Hua Su. Externalising Abstract Mathematical Models. In Michael J. Tauber, editor, *Proceedings of CHI 96*, pages 406–412, New York, April 1996. IEEE Computer Society Press.

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[vWvL93] Jarke J. van Wijk and Robert van Liere. HyperSlice. In Gregory M. Nielson and R. Daniel Bergeron, editors, *Proceedings IEEE Visualization '93*, pages 119–125, Los Alamitos, California, October 1993. IEEE Computer Society Press.

[War94] Matthew O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings IEEE Visualization '94*, pages 326–336, Los Alamitos, California, October 1994. IEEE Computer Society Press.

[WB93] Pak Chung Wong and R. Daniel Bergeron. Visualization of 200,000 year history of global change. Technical Report 93–09, Computer Science Department, University of New Hampshire, Durham, New Hampshire, 1993. Available at ftp://ftp.cs.unh.edu/pub/vis/gisp2.ps.gz.

[WB94] Pak Chung Wong and R. Daniel Bergeron. A child's garden of wavelet transforms. Technical Report 94–24, Computer Science Department, University of New Hampshire, Durham, New Hampshire, 1994. Available at ftp://ftp.cs.unh.edu/pub/vis/waveletReading.ps.gz.

[WB95a] Pak Chung Wong and R. Daniel Bergeron. Authenticity analysis of wavelet approximations in visualization. In Gregory M. Nielson and Deborah Silver, editors, *Proceedings of IEEE Visualization '95*, pages 184–191, Los Alamitos, California, October 1995. IEEE Computer Society Press.

[WB95b] Pak Chung Wong and R. Daniel Bergeron. A multidimensional multivariate image evaluation tool. In Georges Grinstein and Haim Levkowitz, editors, *Perceptual Issues in Visualization*, pages 95–108. Springer, 1995.

[WB96a] Pak Chung Wong and R. Daniel Bergeron. Multiresolution multidimensional wavelet brushing. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings of IEEE Visualization '96*, pages 141–148, New York, NY, October 1996. ACM Press.

[WB96b] Pak Chung Wong and R. Daniel Bergeron. A tool for hierarchical representation and visualization of time–varying data. In David Banks, Tom Crockett, and Kathy Stacy, editors, *Proceedings of NASA ICASE/LaRc Symposium Visualizing Time–Varying Data*, pages 21–29, Hampton, Virginia, 1996. NASA Langley Research Center.

[WB97a] Pak Chung Wong and R. Daniel Bergeron. 30 years of multidimensional multivariate visualization. In Gregory M. Nielson, Heinrich Mueller, and Hans Hagen, editors, *Scientific Visualization: Overviews, Methodologies & Techniques*, Los Alamitos, California, 1997. IEEE Computer Society Press. In press.

[WB97b] Pak Chung Wong and R. Daniel Bergeron. Brushing Techniques for Exploring Scientific Volume Datasets. Technical report, Computer Science Department, University of New Hampshire, Durham, New Hampshire, 1997. Submitted for publication.

[WB97c] Pak Chung Wong and R. Daniel Bergeron. Hierarchical representation of very large data sets for visualization using wavelets. In Gregory M. Nielson, Heinrich Mueller, and Hans Hagen, editors, *Scientific Visualization: Overviews, Methodologies & Techniques*, Los Alamitos, California, 1997. IEEE Computer Society Press. In press.

[WB97d] Pak Chung Wong and R. Daniel Bergeron. Multivariation visualiation by metric scaling. Technical report, Computer Science Department, University of New Hampshire, Durham, New Hampshire, 1997. Submitted for publication.

[WCB96] Pak Chung Wong, Andrew H. Crabb, and R. Daniel Bergeron. Dual multiresolution hyperslice for multivariate data visualization. In Nahum Gershon and Steven Eick, editors, *Proceedings of IEEE Information Visualization '96*, pages 74–75, Los Alamitos, California, October 1996. IEEE Computer Society Press.

[YRL+96] Roni Yagel, David M. Reed, Asish Law, Po-Wen Shih, and Naeem Shareef. Hardware Assisted Volume Rendering of Unstructured Grids by Incremental Slicing. In Roger Crawfis and Charles Hansen, editors, *Proceedings of 1996 Symposium on Volume Visualization*, pages 55–62, New York, NY, October 1996. ACM SIGGRAPH.

Appendix A

List of Symbols

$\mathbb{Z}$	$\{\cdots,-2,-1,0,1,2,\cdots\}$
$\mathbb{N}$	$\{1,2,\cdots\}$
$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers
$\mathbb{F}$	field of scalars, $\mathbb{R}$ or $\mathbb{C}$
$span\ A$	vector subspace spanned by A, where A is a set of vectors
$\|\cdot\|$	any norm
$cl\ S$	closure of S
$\mathbb{R}^N$	vector space of N-tuples of real numbers
$L^1(\mathbb{R})$	space of integrable functions on $\mathbb{R}$
$L^2(\mathbb{R})$	space of square integrable functions on $\mathbb{R}$
$L^1(\mathbb{R}^N)$	space of integrable functions on $\mathbb{R}^N$
$L^2(\mathbb{R}^N)$	space of square integrable functions on $\mathbb{R}^N$
$\langle\cdot,\cdot\rangle$	inner product
$f * g$	convolution of f and g
$\bar{z}$	complex conjugate of z
$x \perp y$	x is orthogonal to y
$S^{\perp}$	orthogonal complement
$H_1 \oplus H_2$	direct sum
P_S	projection onto S

Appendix B

Mathematical Definitions

This appendix contains mathematical definitions on vector spaces, normed linear spaces, Hilbert spaces, and other definitions needed for a good understanding of wavelets. General mathematical symbols and notations are defined separately in Appendix A.

Definition 1 (Vector Space) By a *vector space* we mean a nonempty set E with two operations:

- a mapping $(x, y) \to x + y$ from $E \times E$ into E called *addition*;
- a mapping $(\lambda, x) \to \lambda x$ from $\mathbb{F} \times E$ into E called *multiplication by scalars*;

such that the following conditions are satisfied:

- $x + y = y + x$;
- $(x + y) + z = x + (y + z)$;
- For every $x, y \in E$ there exists a $z \in E$ such that $x + z = y$;
- $\alpha(\beta x) = (\alpha\beta)x$;
- $(\alpha + \beta)x = \alpha x + \beta x$;
- $\alpha(x + y) = \alpha x + \alpha y$;
- $1x = x$;

Definition 2 (Function Spaces) Let X be an arbitrary nonempty set and let E be a vector space. Denote by F the space of all functions from X into E. Then F is a vector space if the addition and multiplication by scalars are defined in the following way:

- $(f+g)(x) = f(x) + g(x)$,
- $(\lambda f)(x) = \lambda f(x)$.

Definition 3 (Linear Combination) Let E be a vector space and let $x_1, \cdots, x_k \in E$. A vector $x \in E$ is called a *linear combination* of vectors $x_1, \cdots, x_k$ if there exist scalars $\alpha_1, \cdots, \alpha_k$ such that $x = \alpha_1 x_1 + \cdots + \alpha_k x_k$.

Definition 4 (Linear Independence) A finite collection of vectors $\{x_1, \cdots, x_k\}$ is called *linearly independent* if $\alpha_1 x_1 + \cdots + \alpha_k x_k = 0$ only if $\alpha_1 = \alpha_2 = \cdots = \alpha_k = 0$.

Definition 5 (Span) Let A be a subset of vector space E. The *span* A is the set of all finite linear combinations of vectors from A, i.e.,

$$span\ A = \{\alpha_1 x_1 + \cdots + \alpha_k x_k : x_1, \cdots, x_k \in A, \alpha_1, \cdots, \alpha_k \in \mathbb{F}, k = 1, 2, \cdots\}.$$

Definition 6 (Basis) A set of vectors $B \subseteq E$ is called a *basis* of E if B is linearly independent and $span\ B = E$.

Definition 7 (Norm) A real function $\|\cdot\|$ on a vector space E (a function which assigns a real number $\|\cdot\|$ to a vector $x \in E$) is called a *norm* if

- $\|x\| = 0$ if and only if $x = 0$;
- $\|\lambda x\| = |\lambda|\ \|x\|$ for every $x \in E$ and $\lambda \in \mathbb{F}$;
- $\|x + y\| \leq \|x\| + \|y\|$ for every $x, y \in E$.

Definition 8 (Normed Space) A vector space with a norm is called a *normed space.*

Definition 9 (Open and Closed Sets) A subset S of a normed space E is called *open* if for every $x \in S$ there exist an $\varepsilon > 0$ such that $\{y \in E : \|y - x\| < \varepsilon\} \subset S$. A subset S is called *closed* if its complement is open, i.e., if $E - S$ is open.

Definition 10 (Closure) Let S be a subset of normed space E. By the *closure* of S, clS, we mean the intersection of all closed sets containing S.

Definition 11 (Dense Subsets) A subset S of a normed space E is called *dense* in E if *cl* $S = E$.

Definition 12 (Cauchy Sequence) A sequence of vectors $\{x_n\}$ in a normed space is called a *Cauchy sequence* if for every $\varepsilon > 0$ there exists a number M such that $\|x_m - x_n\| < \varepsilon$ for all $m, n > M$.

Definition 13 (Complete Normed Space) A normed space E is called *complete* if every Cauchy sequence in E converges to an element of E.

Definition 14 (Step Functions) A *step function* on the real line $\mathbb{R}$ is a finite linear combination of characteristic functions of semi-open intervals $[a, b) \subseteq \mathbb{R}$.

Definition 15 (Integrable Function) A real valued function f defined on $\mathbb{R}$ is called *integrable* if there exists a sequence of step functions $\{f_n\}$ and a natural number D such that the following two conditions are satisfied:

- $\sum_{n=1}^{\infty} |f_n| < D$;

- $f(x) = \sum_{n=1}^{\infty} f_n(x)$ for every $x \in \mathbb{R}$ such that $\sum_{n=1}^{\infty} |f_n(x)| < D$.

Definition 16 ($L^1(\mathbb{R})$) The space of all integrable functions defined on $\mathbb{R}$ is denoted by $L^1(\mathbb{R})$.

Definition 17 (Locally Integrable Functions) A function f defined on $\mathbb{R}$ is called *locally integrable*, if for every $-\infty < a < b < \infty$ the integral $\int_a^b f$ exists.

Definition 18 ($|f|$) If a complex valued function f is integrable, the the real valued function $|f|$ is integrable and

$$\left|\int f\right| \leq \int |f|.$$

Definition 19 (Square Integrable Functions) The space of all complex valued locally integrable functions f such that $|f|^2 \in L^1(\mathbb{R})$ is denoted by $L^2(\mathbb{R})$. Elements of $L^2(\mathbb{R})$ are called *square integrable functions*.

Definition 20 (Inner Product) Let E be a complex vector space. A mapping

$$\langle \cdot, \cdot \rangle : E \times E \to \mathbb{C}$$

is called an *inner product* in E if for any $x, y, z \in E$ and $\alpha, \beta \in \mathbb{C}$ the following conditions are satisfied:

- $\langle x, y\rangle = \overline{\langle y, x\rangle}$ (the bar denotes the complex conjugate);
- $\langle \alpha x + \beta y, z\rangle = \alpha\langle x, z\rangle + \beta\langle y, z\rangle$;
- $\langle x, x\rangle \geq 0$, and $\langle x, x\rangle = 0$ implies $x = 0$.

Definition 21 (Inner Product Space) A vector space with an inner product is called an *inner product space* or a *pre-Hilbert space.*

Definition 22 (Norm in an Inner Product Space) By the *norm* in an inner product space E we mean the functional defined by

$$\|x\| = \sqrt{\langle x, x\rangle}.$$

Definition 23 (Orthogonal Vectors) Two vectors x and y in an inner product space are called *orthogonal* (denoted by $x \perp y$) if $\langle x, y\rangle = 0$.

Definition 24 (Hilbert Space) A complete inner product space is called a *Hilbert* space.

Definition 25 (Orthogonal and Orthonormal Systems) Let E be an inner product space. A family S of non-zero vectors in E is called an *orthogonal system* if $x \perp y$ for any two distinct elements of S. If, in addition, $\|x\| = 1$ for all $x \in S$, S is called an *orthonormal system.*

Definition 26 (Orthonormal Sequence) A finite or infinite sequence of vectors which forms an orthonormal system is called an *orthonormal sequence.*

Definition 27 (Orthogonal Complement) Let S be a nonempty subspace of a Hilbert space H. An element $x \in H$ is said to be *orthogonal* to S, denoted by $x \perp S$, if $\langle x, y\rangle = 0$ for every $y \in S$. The set of all elements of H orthogonal to S, denoted by $S^{\perp}$, is called the *orthogonal complement* of S. More concisely:

$$S^{\perp} = \{x \in H : x \perp S\}.$$

Theorem 4 (Orthogonal Complement) For any subset S of a Hilbert space H, the set $S^{\perp}$ is a closed subspace of H.

Theorem 5 (Orthogonal Projection) If S is a closed subspace of a Hilbert space H, then every element $x \in H$ has a unique decomposition in the form $x = y + z$, where $y \in S$ and $z \in S^{\perp}$.

Each element of H can be uniquely represented as the sum of an element of S and an element of $S^{\perp}$. This can be stated symbolically as

$$H = S \oplus S^{\perp}.$$

We say that H is the *directed sum* of S and $S^{\perp}$. The union of a basis of S and a basis of $S^{\perp}$ is a basis of H.

Definition 28 (Separable Spaces) A Hilbert space is called *separable* if it contains a complete orthonormal sequence. Finite dimensional Hilbert spaces are considered separable. Since every element x in a separable Hilbert space H with a complete orthonormal sequence $\{x_n\}$ can be represented as

$$x = \sum_{n=1}^{\infty} \langle x, \; x_n \rangle \; x_n,$$

the set $\{x_n\}$ is sometimes referred to as an *orthonormal basis* of a Hilbert space H.

Definition 29 (Projection Operator) Let S be a closed subspace of a Hilbert space H. The operator P on H defined by $P(x) = y$ for $x = y+z, y \in S$, and $z \in S^{\perp}$, is called the *projection operator* onto S. The vector y is called *projection of x onto S*. The projection operator onto a subspace S is often denoted by P_S.

Definition 39 (Inner Product in $L^2(\mathbb{R})$) For $f(x), g(x) \in L^2(\mathbb{R})$, the *inner product* of $f(x)$ and $g(x)$ is

$$\langle g(u), f(u) \rangle = \int_{-\infty}^{+\infty} g(u) \; f(u) \; du.$$

Definition 31 (Norm in $L^2(\mathbb{R})$) The *norm* of $f(x)$ in $L^2(\mathbb{R})$ is given by

$$\|f\|^2 = \int_{-\infty}^{+\infty} |f(u)|^2 \; du.$$

Definition 32 (Convolution in $L^2(\mathbb{R})$) For $f(x), g(x) \in L^2(\mathbb{R})$, the *convolution* of $f(x)$ and $g(x)$ is

$$\begin{aligned} f * g(x) &= (f(u) * g(u)) \; (x) \\ &= \int_{-\infty}^{+\infty} f(u) \; g(x-u) \; du. \end{aligned}$$

Appendix C

Principal Coordinates

This appendix describes the details of generating an inner product matrix from the dissimilarities matrix described in Section 10.2.1 of Chapter 10, and solving the Euclidean coordinates of the vertices in n dimensional space from the inner product matrix by the method of principal components.

Let the Euclidean coordinates of n vertices in n dimensional Euclidean space be a matrix X such that each vector $x_r = [x_{r1}, \ldots, x_{rn}]^T$ where $r = 1, \ldots, n$. The Euclidean distance between vertices r and s is given by

$$\begin{aligned} d_{rs}^2 &= [x_r - x_s]^T[x_r - x_s] \\ &= x_r^T x_r + x_s^T x_s - 2x_r^T x_s. \end{aligned} \tag{C.1}$$

Hence

$$\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 = \frac{1}{n}\sum_{r=1}^{n} x_r^T x_r + \frac{1}{n}\sum_{r=1}^{n} x_s^T x_s - \frac{2}{n}\sum_{r=1}^{n} x_r^T x_s. \tag{C.2}$$

If we standardize the data to have zero mean and unit variance, the center of mass of the vertices is the origin. So we have

$$\frac{1}{n}\sum_{r=1}^{n} x_r^T x_s = 0.$$

Also, since

$$\frac{1}{n}\sum_{r=1}^{n} x_s^T x_s = x_s^T x_s,$$

equation (C.2) becomes

$$\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 = \frac{1}{n}\sum_{r=1}^{n} x_r^T x_r + x_s^T x_s. \tag{C.3}$$

Similarly,

$$\frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 = \frac{1}{n}\sum_{s=1}^{n} x_s^T x_s + x_r^T x_r. \tag{C.4}$$

Furthermore, from (C.4),

$$\begin{aligned}
\frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2 &= \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} x_s^T x_s + \frac{1}{n}\sum_{r=1}^{n} x_r^T x_r \\
&= \frac{1}{n}\sum_{s=1}^{n} x_s^T x_s + \frac{1}{n}\sum_{r=1}^{n} x_r^T x_r \\
&= \frac{2}{n}\sum_{r=1}^{n} x_r^T x_r \qquad \text{(C.5)}
\end{aligned}$$

Defining an inner product matrix B such that

$$[B]_{rs} = b_{rs} = x_r^T x_s,$$

and substituting (C.3), (C.4), and (C.5) into (C.1) gives the inner product matrix B in terms of d_{rs},

$$\begin{aligned}
b_{rs} &= x_r^T x_s \\
&= \frac{1}{2}\left(x_r^T x_r + x_s^T x_s - d_{rs}^2\right) \\
&= \frac{1}{2}\left(\frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{s=1}^{n} x_s^T x_s + \frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 - \frac{1}{n}\sum_{r=1}^{n} x_r^T x_r - d_{rs}^2\right) \\
&= \frac{1}{2}\left(\frac{1}{n}\sum_{r=1}^{n} d_{rs}^2 + \frac{1}{n}\sum_{s=1}^{n} d_{rs}^2 - \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2 - d_{rs}^2\right).
\end{aligned}$$

The next step involves the use of principal components to recover the Euclidean coordinates of the n dimensional space denoted by the matrix X from B. We seek a different coordinate system in the d dimensional display space such that the distances between points in the display space can be approximated by measuring the distances only along some subset of the axes of this new coordinate system. By the definition of an inner product matrix, B can be expressed as

$$B = XX^T. \tag{C.6}$$

Since B is symmetric and positive semi-definite (i.e., only some of the eigenvalues are positive), it has p positive eigenvalues. Let Λ be the eigenvalue matrix in which the diagonal is the sorted eigenvalues $\lambda_1, \ldots, \lambda_p, \ldots, \lambda_n$. Let the corresponding normalized eigenvector of Λ be V. By the definition of eigenvectors, matrix B can be described as

$$B = V\Lambda V^T.$$

Since there are only p positive eigenvalues, B can be expressed as

$$\begin{aligned} B &= V_1\Lambda_1 V_1^T \\ &= V_1\Lambda_1^{\frac{1}{2}}\Lambda_1^{\frac{1}{2}}V_1^T \end{aligned} \tag{C.7}$$

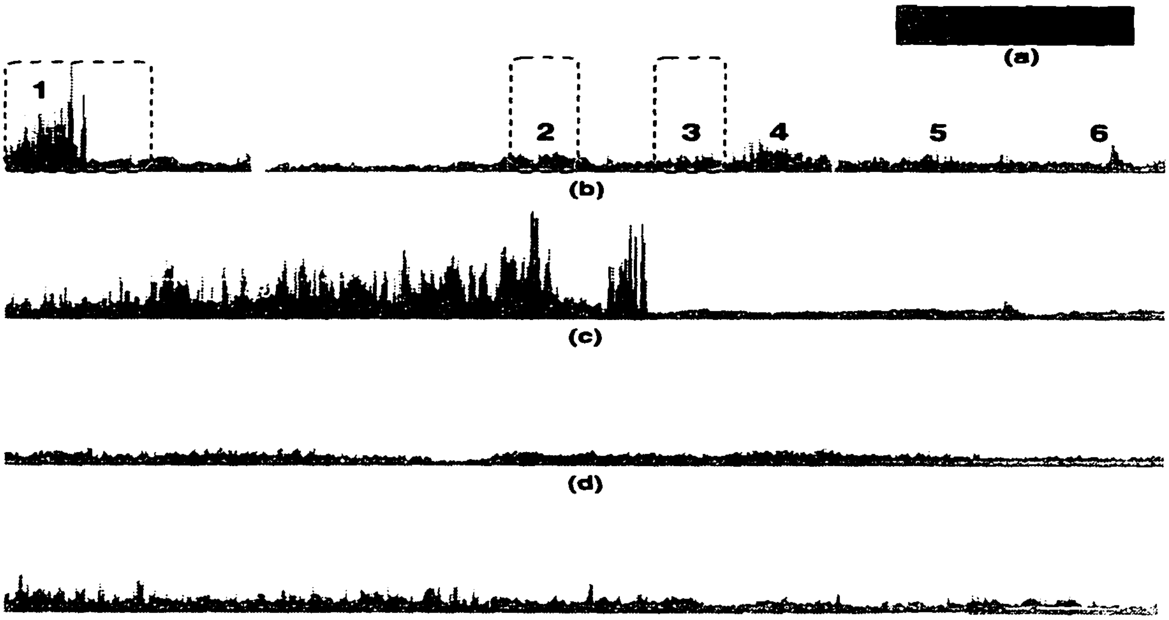
where Λ_1 is the eigenvalue matrix in which the diagonal is the eigenvalues $\lambda_1, \ldots, \lambda_p$ and V_1 is the corresponding eigenvector of Λ_1. From (C.6) and (C.7), therefore

$$X = V_1\Lambda_1^{\frac{1}{2}}.$$

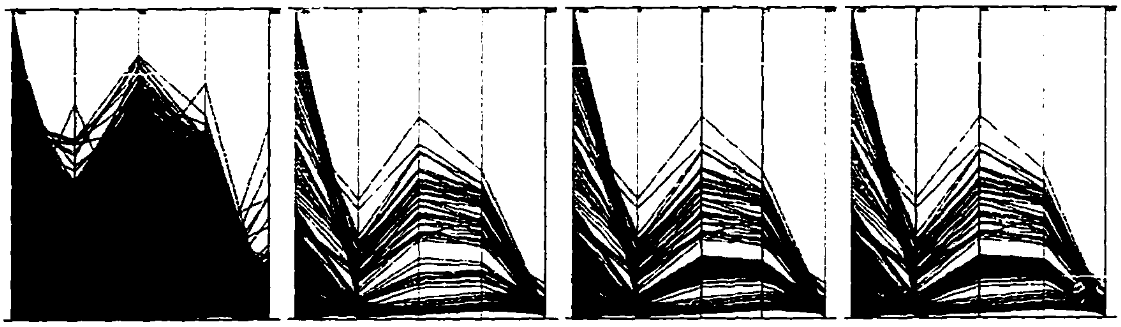
Appendix D

Color Plates

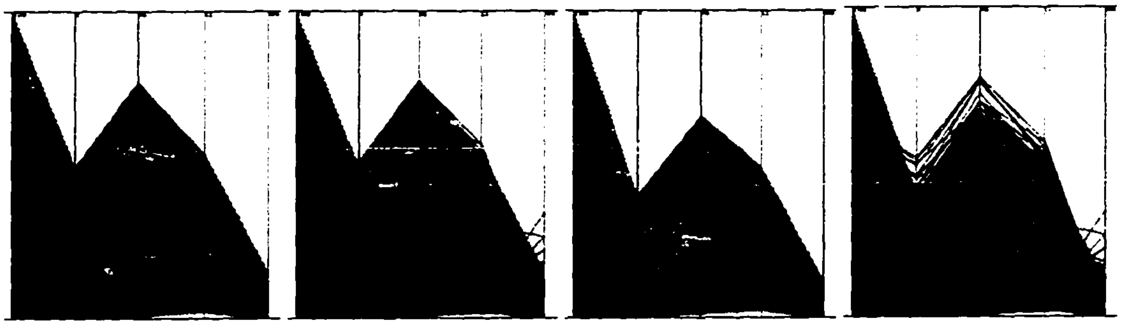
Color Plate 1



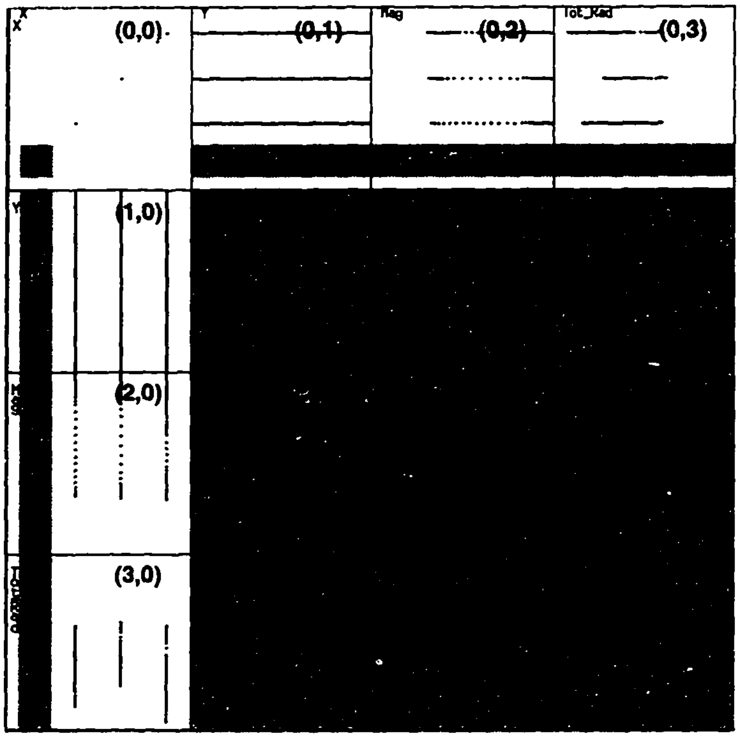
Color Plate 2



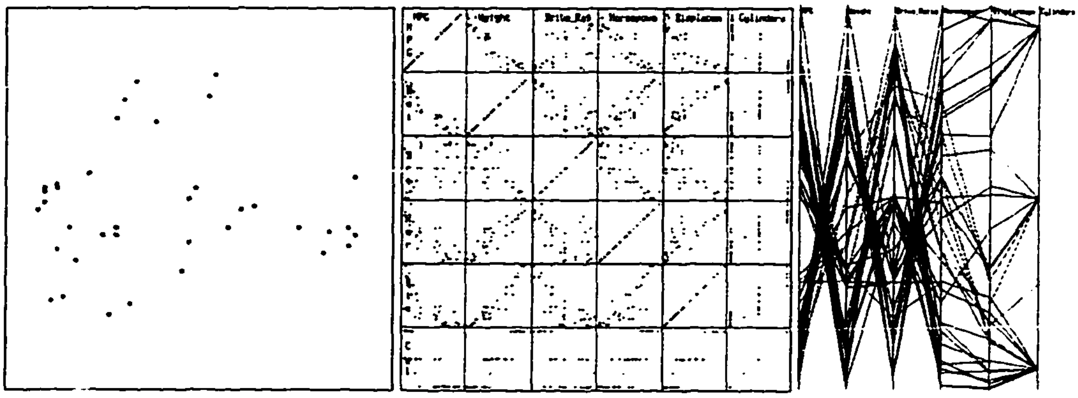
Color Plate 3



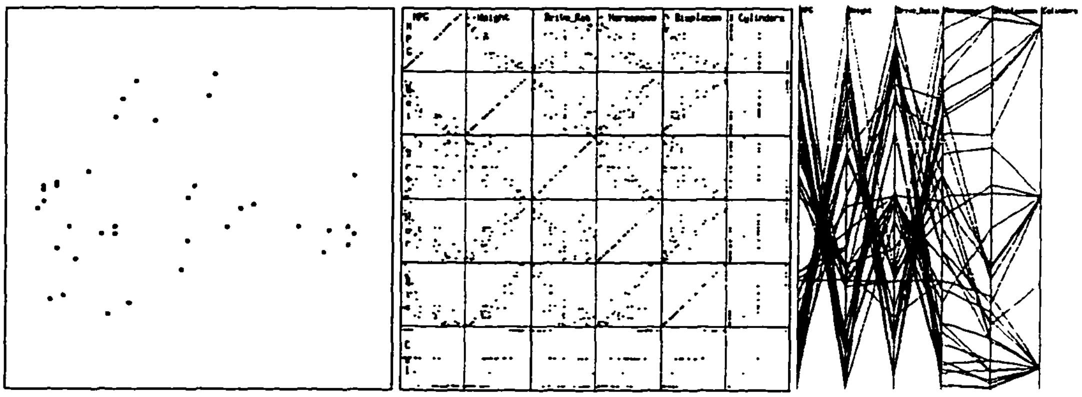
Color Plate 4



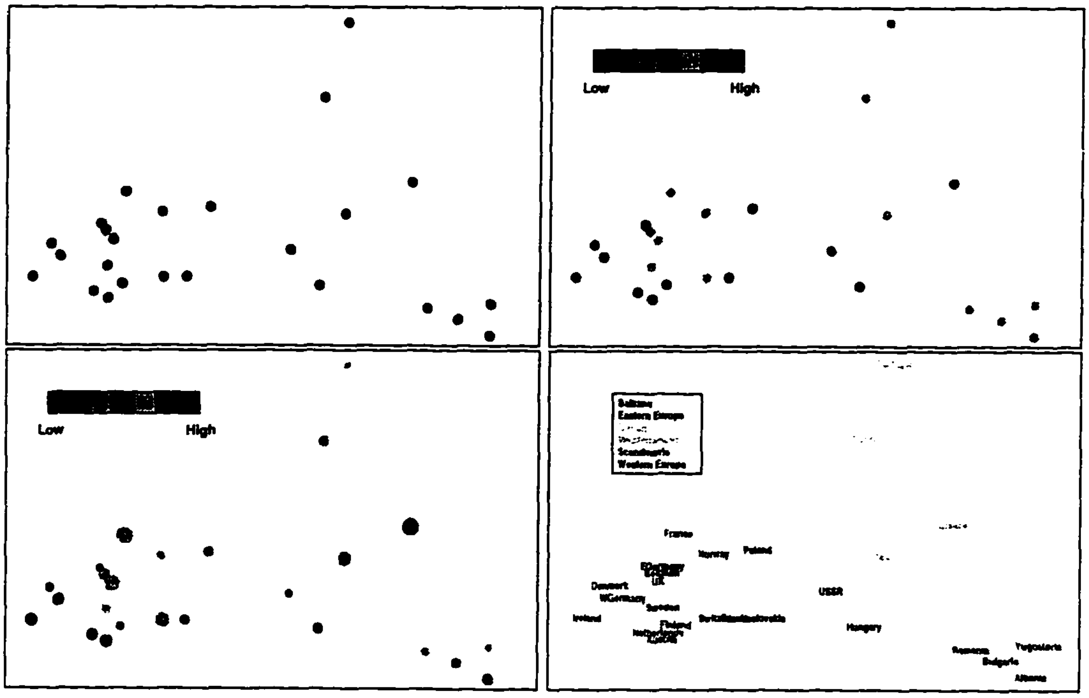
Color Plate 5



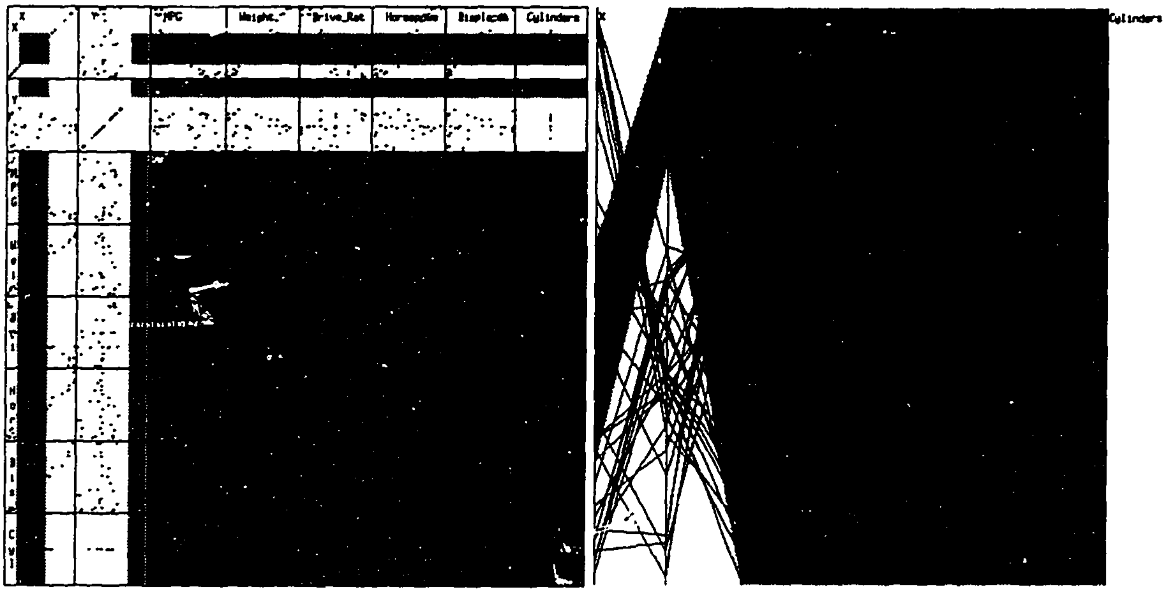
Color Plate 6



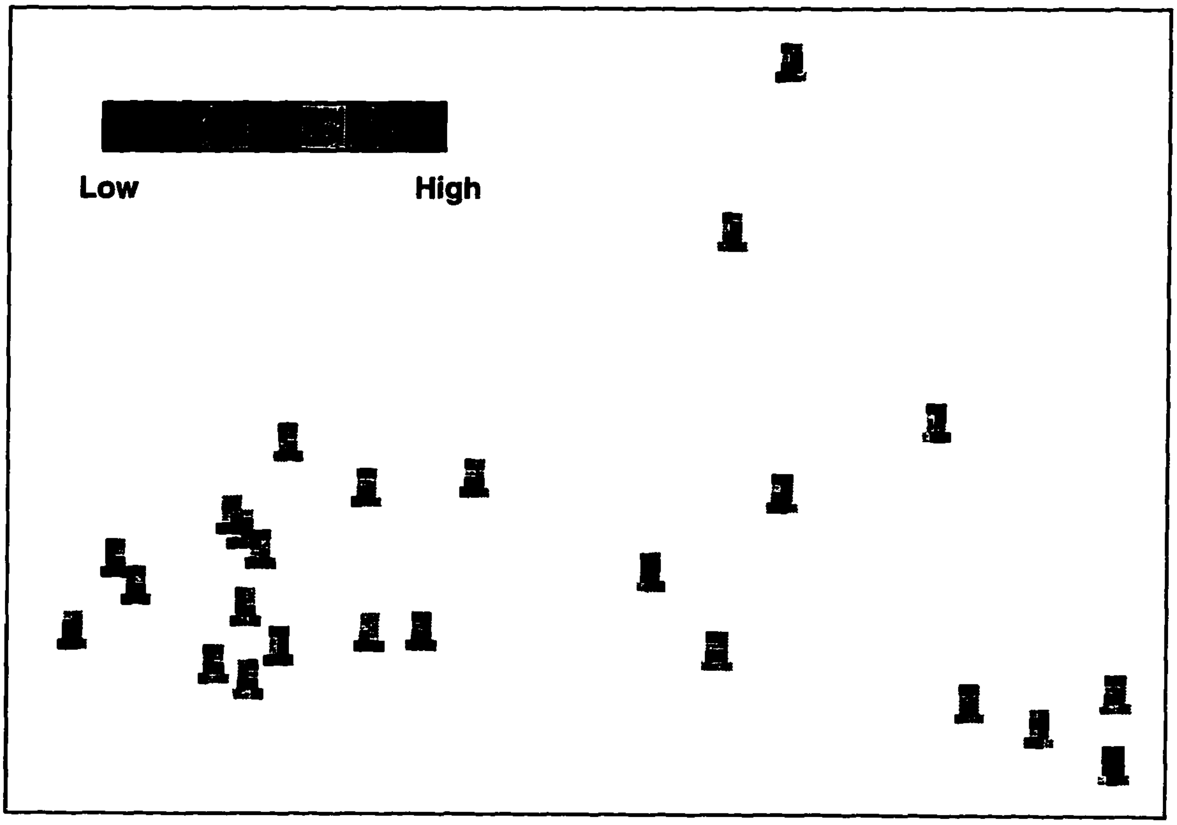
Color Plate 7



Color Plate 8



Color Plate 9



Color Plate 10