Multiresolution Modeling for Scientific Visualization

Martin Bertram

Department of Computer Science,
University of California, Davis

July 14, 2000

Multiresolution Modeling for Scientific Visualization

Martin Bertram

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

_____

_____

_____

Committee in Charge

July, 2000

devoted

to my fiancee
Claudia Hering

and

to my parents
Beate and Horst-Günther Bertram

Acknowledgements

First, I want to thank my committee members, Bernd Hamann, Mark Duchaineau, and Ken Joy for their professional advice and support that made this dissertation possible. During my time in the Visualization Group at the University of California at Davis and in the Center for Applied Scientific Computing (CASC) at Lawrence Livermore National Laboratory, I was inspired and motivated by the ideas of many exceptionally talented researchers and fellow students. Among these are Daniel Laney, Eric LaMar, Shirley Konkle, Björn Heckel, Issac Trotts, Gunther Weber, Antonello Uva, Oliver Kreylos, Falko Küster, Greg Schussman, Fritz Barnes, David Wiley, Kathleen Bonnell, Lenny Tsap, Gerik Scheuermann, Jörg Meyer, and many others. I want to thank especially Nelson Max, Kwan-Liu Ma, Helmut Pottmann, and Chuck Hansen for their ideas and discussions of interesting research topics. Also, I am thankful to the faculty and staff at LLNL and at UC Davis, in particular to Kim Reinking, for her advice that helped me to satisfy the degree requirements without any flaw.

The foundation for this dissertation, was already anticipated in my undergraduate days at the University of Kaiserslautern in Germany, where Hans Hagen, Hans-Christian Rodrian, and Hardy Moock woke up my interest in the fields of Geometric Modeling and Scientific Visualization. Bernd Hamann encouraged me to apply for the Graduate Program at UC Davis and to join his prospering Visualization Group. He arranged a lot of funding opportunities and was never short on time when discussing research ideas and giving invaluable advice. Ken Joy understood, like no one else, how to motivate students and to make research look as easy as it gets. In his lectures, he was often sitting on the school bank as if he was a student, he listened to us giving him lectures, which was a wonderful experience. Mark Duchaineau recovered my interest in wavelets and funded me through a fellowship program at Livermore. Many of his research ideas inspired me and are further developed in this dissertation.

Lastly, I dedicate this dissertation to my parents Beate and Horst-Günther, who raised my brother Matthias and me with so much love and care, provided funding for our undergraduate studies, and, coming from academia, pushed us into the right direction, though being surprised with the outcome. Eventually, there is one very special person to thank, who slipped into my life stealthily, but with elegance and fortitude, giving me love, courage, happiness, and the vision of a satisfied life. I devote this dissertation to her, my adored fiancee Claudia.

Abstract

Interactive visualization and exploration of large-scale scientific data sets is an important application for the analysis of data obtained from computational fluid dynamics (CFD), computerized tomography (CT), and laser-range scans. These data sets are typically defined by discrete samples, either aligned on regular grids or randomly scattered in space, describing an underlying shape or a volumetric field function. Examples are isosurfaces and shock waves in CFD, height maps for terrain data, and three-dimensional scanned objects in reverse engineering. Starting with discrete samples, a continuous geometric model is built, that closely approximates an underlying shape. Multiresolution representations are essential for displaying large-scale surface and volume models within minimal computation time, satisfying certain error bounds or bounds on complexity. This dissertation is concerned with the efficient construction of multiresolution surface and volume models for high-quality approximation of scientific data. We present two adaptive clustering methods used for scattered data approximation. The first method uses hierarchical Voronoi diagrams and Sibson's interpolant for multiresolution surface modeling. The second approach is based on binary space-partition trees and quadratic polynomial approximation used as intermediate representation for the construction of triangulated surfaces. For multiresolution representation and compression of data sets defined on regular grids, we construct biorthogonal lifted B-spline wavelets with small filters. A major contribution is the construction of new symmetric lifted B-spline subdivision-surface wavelets with finite filters for representing surfaces of arbitrary topology defined by irregular polygonal base meshes. Regular mesh subdivision results in smooth limit surfaces with the option of sharp features and boundary curves represented by modified subdivision rules. These new wavelet constructions are used for approximation and compression of isosurfaces taken from a high-resolution turbulent-mixing hydrodynamics simulation. A similar approach is used to define wavelets on planar tessellations.

Contents

1 Introduction **4**

2 Adaptive Clustering **10**

2.1 Clustering Approach 11

2.2 Hierarchical Voronoi Diagrams 13

2.2.1 Algorithm 13

2.2.2 Numerical Examples 16

2.3 Binary Space Partition (BSP) Trees 21

2.3.1 Principal Component Analysis (PCA) 21

2.3.2 Quadratic Polynomial Approximation 23

2.3.3 Affine-invariant Approach 25

2.4 Optimal Approximation with Triangles 27

2.4.1 Related Work 27

2.4.2 Triangulation Approach 28

2.4.3 Principal Axis Transformation 29

2.4.4 The Elliptic Case 30

2.4.5 The Hyperbolic Case 31
2.4.6 The Parabolic Case 33
2.4.7 Merging Triangulations 34
2.4.8 Numerical Examples 37
2.5 Conclusions Concerning Clustering 45

3 Wavelet Representations 47

3.1 Motivation 48
3.2 Wavelets and Signal Processing 52
3.2.1 Fourier Transform 52
3.2.2 Continuous Wavelet Transform 54
3.2.3 Wavelet Frames 57
3.2.4 Discrete Wavelet Transform 58
3.3 Wavelets for Geometric Modeling 64
3.3.1 A Modeling Paradigm 64
3.3.2 Recursive Subdivision 67
3.3.3 Least-Squares Fitting 68
3.3.4 The Lifting Scheme 71
3.3.5 Integer Arithmetic 76
3.4 Symmetric Lifted B-spline Wavelets 78
3.4.1 Linear B-spline Wavelet 81
3.4.2 Cubic B-spline Wavelet 83

3.4.3 Quintic B-spline Wavelet 84
3.4.4 A Remark on Stability 86
3.5 Modeling Scientific Data 89
3.5.1 Arithmetic Coding 91
3.5.2 Lossless Compression Examples 93
3.5.3 Lossy Compression Examples 96

4 Wavelets on Arbitrary Topology **101**

4.1 Multiresolution Representation of Two-Manifolds 102
4.1.1 Related Work 103
4.1.2 Generalized Lifting Operations 109
4.1.3 Sharp Features and Surface Boundaries 118
4.1.4 Subdivision and Fitting Properties 123
4.1.5 Isosurface Fitting 124
4.1.6 Numerical Examples 129
4.2 Wavelets on Tessellations 135
4.2.1 Parametrization 136
4.2.2 Wavelet Transform 138
4.2.3 Results and Applications 141
4.3 Conclusions and Future Work 142

Chapter 1

Introduction

Geometric modeling is concerned with the representation and manipulation of curves and surfaces, required for geometric design, computer graphics, and scientific visualization applications. In the scope of scientific visualization, geometric modeling is used to create continuous, highly detailed mathematical representations for large-scale data sets defined by discrete samples. Geometric models are close approximations to physical shapes or processes, enabling exploration and visualization techniques that could not be applied to discrete sampled representations or to physical processes. Examples for data used in geometric modeling come in two flavours: physical phenomena that can be measured by some technique and numerical simulations that can be reproduced virtually at any resolution.

Applications for modeling physical phenomena include studying the shape of a tumor reconstructed from computerized tomography (CT) scans, visualizing vortex cores in the air flow around an aircraft wing measured from a wind-tunnel experiment, and reverse engineering of a car body scanned from a solid prototype. Applications for numerical simulations are, for example, stress tests of a concrete structure during an earthquake and visualization of turbulent-mixing hydrodynamics simulated on massive parallel supercom-

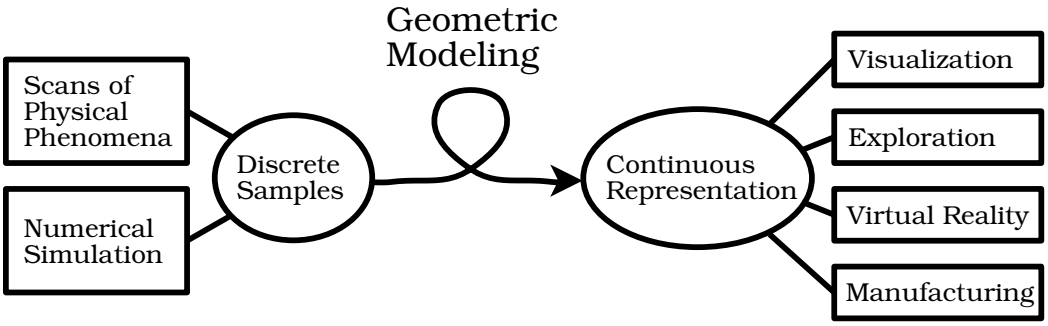

Figure 1.1: Geometric modeling provides continuous approximations of high fidelity to enable techniques that cannot be applied to discrete data or to physical processes.

puters. The role of geometric modeling in these applications is illustrated in Figure 1.1. A broader definition of geometric modeling includes also applications in *computer-aided geometric design* (CAGD), which refers to interactive techniques for designing curves and surfaces.

A geometric model is a multi-dimensional function or a set of functions approximating data with high fidelity and providing efficient evaluation methods for answering queries. In the case of a CT scan, this function defines the material density for every point (x, y, z) in space. In the case of a time-varying compressible flow field, this function has a four-dimensional (x, y, z, t) domain and may have a six-dimensional range (p, ρ, T, u, v, w), encompassing pressure, density, temperature, and three velocity components, respectively. Often, the domain of a geometric model is defined by an irregular finite element mesh that may even be deforming or changing topology over time, which complicates the construction of continuous functions. Also, there may be discontinuities like material boundaries and shock waves that should be explicitly represented in a geometric model.

Another difficult problem is the construction of a surface parametrization when none is given. This problem arises in reverse engineering applications, where a surface is reconstructed form a cloud of sampled three-dimensional

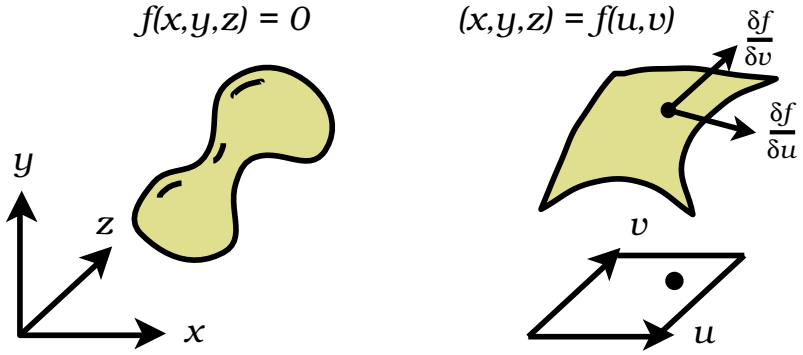

Figure 1.2: Implicit surface (isosurface, left) and parametric patch (right).

points, and in isosurface-extraction applications where a parametric representation for an implicit surface $\{(x, y, z) \mid f(x, y, z) = 0\}$ is sought, see Figure 1.2. Parametric representations are often preferred, since the geometry is represented as the range of a function whose domain can be freely chosen to be a simple topological equivalent, for example a polygonal surface, providing efficient access to the corresponding geometry.

We build geometric models for data sets defined by discrete samples that are often not accurate. The data may be "contaminated" with sampling noise or numerical errors accumulated when solving large systems of equations. Geometric modeling techniques allow the construction of continuous representations that, despite of errors and missing data, closely describe the underlying geometry. From a good geometric model, supplementary information can be derived, like partial derivatives in a flow field and Gaussian curvature of a surface. The model should minimize *aliasing* effects due to discrete sampling as well as anomalies not supported by the data, like "wiggles" caused by interpolation constraints or by using certain basis functions. The representation should be *fair*, *i.e.*, it should be as smooth as suggested by the data, except along sharp features and discontinuities.

Multiresolution representations become essential to process large-scale data sets efficiently. They allow displaying a model at a low-resolution global

view as well as zooming into certain regions at increased resolutions such that the computation and transmission times remain small. If the geometric model can be evaluated locally and progressively, it is possible to satisfy demand-driven queries, *i.e.*, providing a maximal amount of detail within a prescribed processing time or providing a minimal amount of detail satisfying a prescribed error bound. Progressive evaluation means that the resolution of a geometric model adaptively increases while the data is accessed, without processing the same information twice. The concept of multiresolution modeling is surveyed by Heckbert/Garland [63].

In this dissertation, we describe a variety of algorithms that efficiently generate multiresolution models from discrete samples. One class of algorithms are adaptive clustering methods that partition a set of samples recursively into smaller groups (clusters) based on adjacency in space or correlation of associated function values. Every cluster is represented by its most significant sample or by a simple approximation of all samples inside, thus reducing the amount of data and sorting it by decreasing significance. In Chapter 2, we describe two adaptive clustering methods based on hierarchical Voronoi diagrams, see Figure 1.3, and binary space partition (BSP) trees. We provide numerical examples and describe a triangulation method that is based on adaptive clustering.

A second class of multiresolution models are based on wavelet representations [25, 123]. Wavelets have a wide range of applications, including progressive transmission and compression, see Figure 1.4, signal processing, multiresolution design, hierarchical rendering and solving difficult mathematical problems like integrating radiosity kernels and partial differential equations (PDEs). In Chapter 3, we review some fundamental theory of the wavelet transform and describe some highly efficient wavelet constructions based on the lifting scheme [125]. We use these constructions for lossless compression and visualization of large-scale scientific data obtained from simulated turbulent-mixing hydrodynamics.

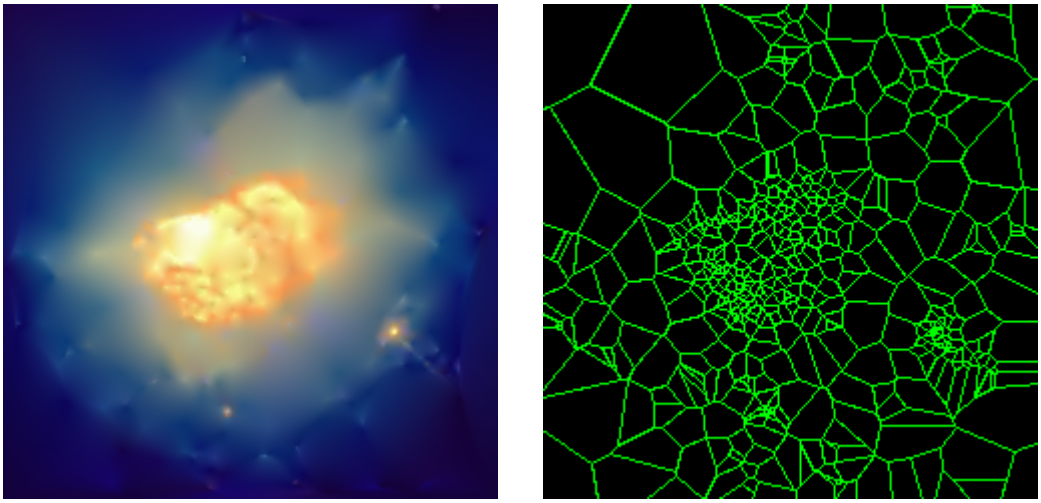

Figure 1.3: An image represented by color values of 500 selected points and the underlying tessellation (Voronoi diagram).

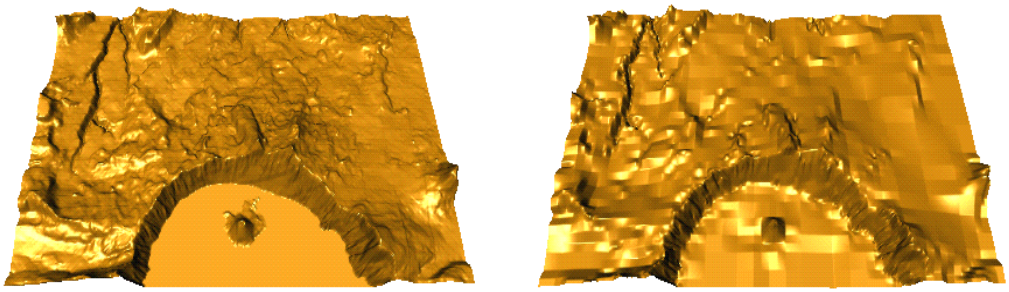

Figure 1.4: Terrain data set (left) and reconstruction from a 1:100 compression (right) using a linear B-spline wavelet transform and arithmetic coding.

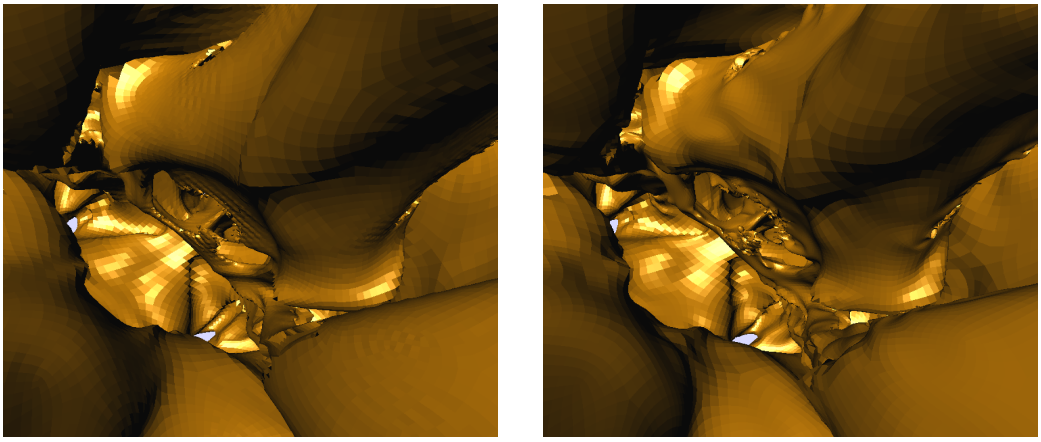

Figure 1.5: Local view of an isosurface re-sampled on a regularly refined polygonal base mesh. The surface model is shown at full resolution (left) and reconstructed from 5 percent of its wavelet coefficients (right).

A major contribution of this dissertation is the construction of lifted B-spline wavelets generalized to arbitrary polygonal mesh domains. Wavelets on arbitrary topology were already described by Lounsbery [90]. Our new construction approach, however, provides the most efficient and comprehensive wavelet transform for smooth two-manifold geometries to date. To the author's knowledge, our approach is the first lifting-style wavelet transform for two-manifold geometries with finite decomposition and reconstruction filters, reproducing tensor-product B-splines on regular, rectilinear meshes. Our wavelet transform satisfies a lot of desirable properties, like vanishing moments, polynomial precision, and tangent-plane continuity at *extraordinary points* (points with other than four adjacent edges in a polygonal mesh domain). Finally, we outline an isosurface fitting method generating meshes and parametrizations for our wavelet transform. We use this wavelet transform for isosurface compression and visualization at different levels of resolution. A re-sampled isosurface reconstructed form a small set of wavelet coefficients is shown in Figure 1.5.

Chapter 2

Adaptive Clustering

Clustering techniques [94] can be used to generate a data-dependent hierarchy representing inherent topological and geometric structures. For example, clustering can be applied to recover topological structures of two-manifold surfaces from scattered data points in 3D space [65]. In contrast to mesh-reduction algorithms, adaptive clustering methods do not require a grid structure connecting the given data points. A cluster hierarchy can be built in a "top-down" fashion, so that coarse levels of resolution require less computation times than finer levels.

In this chapter, we give a general overview of adaptive, data-dependent clustering schemes and introduce two clustering methods, based on hierarchical Voronoi diagrams and BSP trees for adaptive triangulation. We provide numerical examples for these algorithms applying them to adaptive image approximation and terrain modeling.

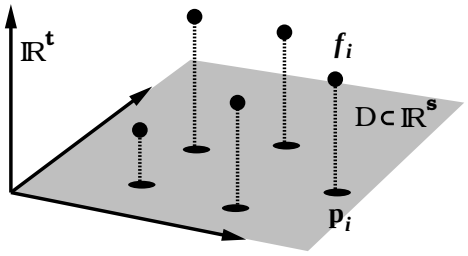

Figure 2.1: Scattered points with associated function values ($s = 2, t = 1$).

2.1 Clustering Approach

Adaptive clustering schemes construct a hierarchy of regions, each of which is associated with a simplified representation for the data points located inside. We assume that a data set is represented at its finest level of detail by a set P of n points in s-dimensional space, $s = 1, 2, \ldots$, with associated t-dimensional function values, $t = 0, 1, \ldots$:

$$P \;=\; \{(\mathbf{p}_i,\; f_i) \mid \mathbf{p}_i \in \mathbb{R}^s,\; f_i \in \mathbb{R}^t,\; i = 1, \ldots, n\}.$$

For our applications, the dimension s is at most four (in the case of time-varying volumetric data). However, the number t of associated function values can be much higher.

The set P is the input for adaptive clustering. It represents a function $f : D \to \mathbb{R}^t$, where $D \subset \mathbb{R}^s$ is assumed to be a compact domain that contains all points $\mathbf{p}_i$. The points $\mathbf{p}_i$ define the associated parameter values for the samples f_i. We do not assume any kind of "connectivity" or grid structure for the points $\mathbf{p}_i$. An example of a point set is illustrated in Figure 2.1.

The output of an adaptive clustering scheme consists of a number of levels $L_j,\; j = 0, 1, \ldots$, defined as

$$L_j \;=\; \{(\tau_k^j,\; \tilde{f}_k^j,\; \varepsilon_k^j) \mid k = 1, \ldots, n_j\},$$

where for every level with index j, the tiles $\{\tau_k^j \subseteq D \mid k = 1, \ldots, n_j\}$ form a partition of the domain D, the functions $\tilde{f}_k^j : \tau_k^j \to \mathbb{R}^t$ approximate the

function values of points located in the tiles τ_k^j, *i.e.*,

$$\tilde{f}_k^j(\mathbf{p}_i) \;\approx\; f_i \quad \forall \mathbf{p}_i \in \tau_k^j,$$

and the residuals $\varepsilon_k^j \in \mathbb{R} \geq 0$ estimate the approximation error. In principle, any error norm can be chosen to compute the residuals ε_k^j. We note that the error norm has a high impact on the efficiency and quality of the clustering algorithm, since it defines an optimization criterion for the approximations at every level of resolution. We suggest the following norm:

$$\varepsilon_k^j \;=\; \left(\frac{1}{n_k^j} \sum_{\mathbf{p}_i \in \tau_k^j} \|\tilde{f}_k^j(\mathbf{p}_i) - f_i\|^p \right)^{\frac{1}{p}}, \quad p \in [1, \infty], \tag{2.1}$$

where $n_k^j = |\{\mathbf{p}_i \in \tau_k^j\}|$ is the number of points located in tile τ_k^j and $\|\cdot\|$ denotes the Euclidean norm (or any other norm) in $\mathbb{R}^t$. In the case of $p = \infty$, the residual is simply the maximal error within the corresponding tile.

A global error ε^j with respect to this norm can efficiently be obtained for every level of resolution from the residuals ε_i^j as

$$\varepsilon^j \;=\; \left(\frac{1}{n} \sum_{k=1}^{n_j} \sum_{\mathbf{p}_i \in \tau_k^j} \|\tilde{f}_k^j(\mathbf{p}_i) - f_i\|^p \right)^{\frac{1}{p}} = \left(\frac{1}{n} \sum_{k=1}^{n_j} n_k^j \; \left(\varepsilon_k^j\right)^p \right)^{\frac{1}{p}}. \tag{2.2}$$

Starting with a coarse approximation L_0, an adaptive clustering algorithm computes finer levels L_{j+1} from L_j until a prescribed number of clusters or a prescribed error bound is satisfied. To keep the clustering algorithm simple and efficient, the approximation L_{j+1} should differ from L_j only in cluster regions with large residuals in L_j. As the clustering is refined, it should eventually converge to a space partition, where every tile contains exactly one data point or where the number of points in every tile is sufficiently low providing zero residuals.

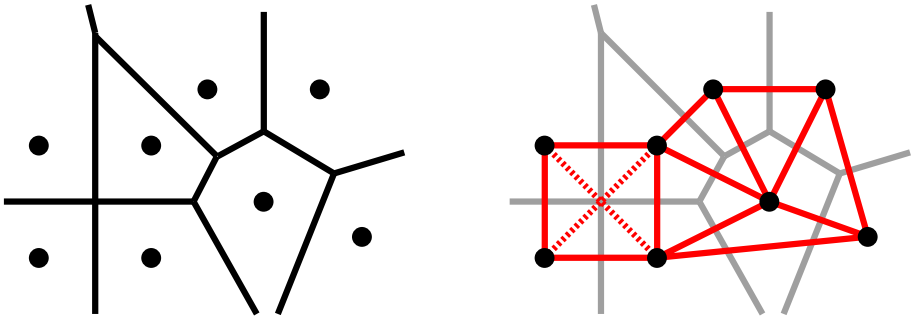

Figure 2.2: Planar Voronoi diagram and its dual, the (not uniquely defined) Delaunay triangulation.

2.2 Hierarchical Voronoi Diagrams

We propose a new adaptive clustering approach for multiresolution representation of scattered data: a hierarchy of Voronoi diagrams [117] constructed from nested subsets of the original set of points.

2.2.1 Algorithm

The *Voronoi diagram* of a set of k-dimensional points $\mathbf{p}_i,\ i = 1, \ldots, n$, is a space partition consisting of n tiles τ_i. Each tile τ_i is defined as a subset of $\mathbb{R}^k$ that contains all points that are closer to $\mathbf{p}_i$ than to any $\mathbf{p}_j,\ j \neq i$, with respect to the Euclidean norm, see Figure 2.2.

A Voronoi diagram can be derived from its dual, the *Delaunay triangulation* [34, 36, 38, 48], see Figure 2.2. The circumscribed circle of every triangle in a Delaunay triangulation does not contain any other data points. If more than three points are located on a such a circle, then the Delaunay triangulation is not unique. The Voronoi vertices are located at the centers of circumscribed circles of Delaunay triangles, which can be exploited for constructing a Voronoi diagram. The Voronoi diagram is unique, in contrast to the Delaunay triangulation.

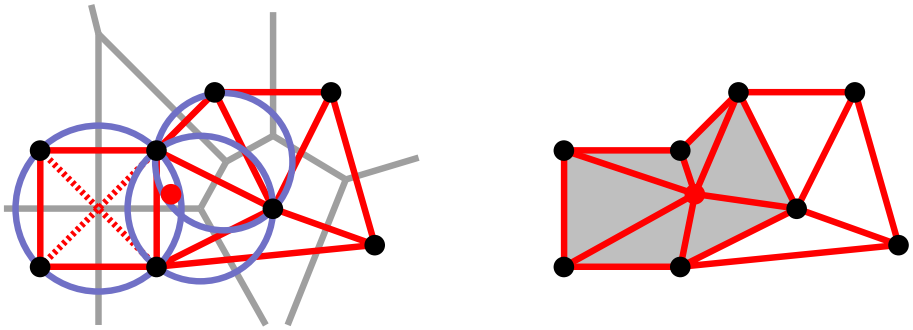

Figure 2.3: Construction of Delaunay triangulation by point insertion. Every triangle whose circumscribed circle contains the inserted point is erased. The points belonging to removed triangles are connected to the new point.

A Delaunay triangulation is constructed in expected linear time, provided the points are evenly distributed [96]. Figure 2.3 illustrates the adaptive construction process in the plane. For every point inserted into a Delaunay triangulation, all triangles whose circumscribed circles contain the new point are erased. The points belonging to erased triangles are then connected to the new point, defining new triangles that automatically satisfy the Delaunay property. Point insertion is an operation performed in expected constant time, provided that the triangles to be removed are identified in expected constant time, which requires the use of some acceleration method. Generalizing the Delaunay triangulation to k-dimensional space is straight-forward: A Delaunay triangulation consists of k-simplices whose circumscribed k-dimensional hyperspheres contain no other point.

The adaptive clustering algorithm uses *Sibson's interpolant* [121] for construction of the functions $\tilde{f}_k^j$, described in Section 2.1. Sibson's interpolant is based on blending function values f_i associated with the points $\mathbf{p}_i$ that define the Voronoi diagram. The blending weights for Sibson's interpolant at a point $\mathbf{p} \in \mathbb{R}^k$ are computed by inserting $\mathbf{p}$ temporarily into the Voronoi diagram and by computing the areas/volumes a_i that are "cut away" from Voronoi tiles τ_i, see Figure 2.4. The value of Sibson's interpolant at $\mathbf{p}$ is

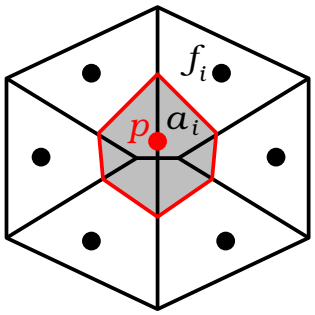

Figure 2.4: Computing Sibson's interpolant at point p by inserting p into a Voronoi diagram and using the areas cut away from every tile as blending weights.

defined as

$$f(\mathbf{p}) \;=\; \frac{\sum_i \; a_i f_i}{\sum_i a_i}.$$

Sibson's interpolant is C^1-continuous everywhere except at the points $\mathbf{p}_i$. To avoid infinite areas/volumes a_i, one clips the Voronoi diagram against the boundary of the compact domain D. A natural choice for the domain D is the convex hull of the given points $\mathbf{p}_i$.

In the following, we provide the clustering algorithm in pseudocode, using notation from Section 2.1. The algorithm performs these steps:

(i) Construct the Voronoi diagram for the minimal point set defining the convex hull of all points $\mathbf{p}_i$, $i = 1, \ldots, n$. The tiles of this Voronoi diagram define the cluster regions τ_k^0 of level L_0.

(ii) From the functions $\tilde{f}_k^j$, defined by Sibson's interpolant and from error norm (2.1) ($p = 2$), compute all residuals ε_k^0. To avoid square root computations, $(\varepsilon_k^0)^2$ is stored.

(iii) Refinement: $L_j \rightarrow L_{j+1}$. Let m be the index of a maximal residual in L_j, *i.e.*, $\varepsilon_m^j \geq \varepsilon_k^j \; \forall k = 1, \ldots, n_j$. Among all $\mathbf{p}_i \in \tau_m^j$, identify a data

point $\mathbf{p}_{max}$ with maximal error $max_{\mathbf{p}_i \in \tau_m^j}\{\|\tilde{f}_m^j(\mathbf{p}_i) - f_i\|\}$. Insert $\mathbf{p}_{max}$ into the Voronoi diagram.

(iv) Update ε_{max}^{j+1} and all residuals associated with tiles adjacent to the new tile τ_{max}^{j+1} with center $\mathbf{p}_{max}$. (All other clusters remain unchanged, *i.e.*, $\tau_i^{j+1} = \tau_i^j$, $\tilde{f}_i^{j+1} = \tilde{f}_i^j$, and $\varepsilon_i^{j+1} = \varepsilon_i^j$.)

(v) Compute the global approximation error ε^j using the error norm (2.2). Terminate the process when a prescribed global error bound is satisfied or when a prescribed number of points has been inserted. Otherwise, increment j and continue with step (iii).

2.2.2 Numerical Examples

We have applied the Voronoi-based clustering approach to represent the two analytical functions $f_1(x, y) = sin\ (x^2)\ sin\ (y^2)$ and $f_2(x, y) = tan\ (x)$, as well as the Hubble image "Dying Sun", courtesy of NASA. All three data sets have been sampled on a regular, uniformly spaced 256×256 grid. The algorithm does not exploit this grid structure. Approximation results are listed in Table 2.1.

Hierarchical voronoi Diagrams can also be used for terrain modeling. We have applied our clustering approach to the Crater Lake terrain data set, courtesy of U.S. Geological Survey. The original dataset contains 159272 samples. A hierarchy of Voronoi-based approximations is shown in Figure 2.7 and the corresponding approximation errors are listed in Table 2.2.

Dataset	No. Voronoi Tiles	Error ($p = 1$) [%]	Error ($p = 2$) [%]
f_1	100	6.0	7.4
	1000	1.0	1.2
f_2	100	7.9	11.3
	200	5.1	6.3
"Dying Sun"	100	4.2	4.6
	200	3.6	4.1
	500	2.8	3.2
	2000	2.0	2.3

Table 2.1: Numerical approximation results. The errors are computed based on (2.2). Figures 2.5 and 2.6 show Sibson's interpolant and the corresponding Voronoi diagrams.

No. Voronoi Tiles	Error ($p = \infty$) [%]	Error ($p = 2$) [%]
100	31.6	3.13
200	17.1	1.96
300	16.3	1.55
400	13.9	1.33
500	11.9	1.21
1000	10.8	0.80

Table 2.2: Approximation errors in percent of amplitude for Crater Lake terrain data set. Figure 2.7 shows the different levels of resolution.

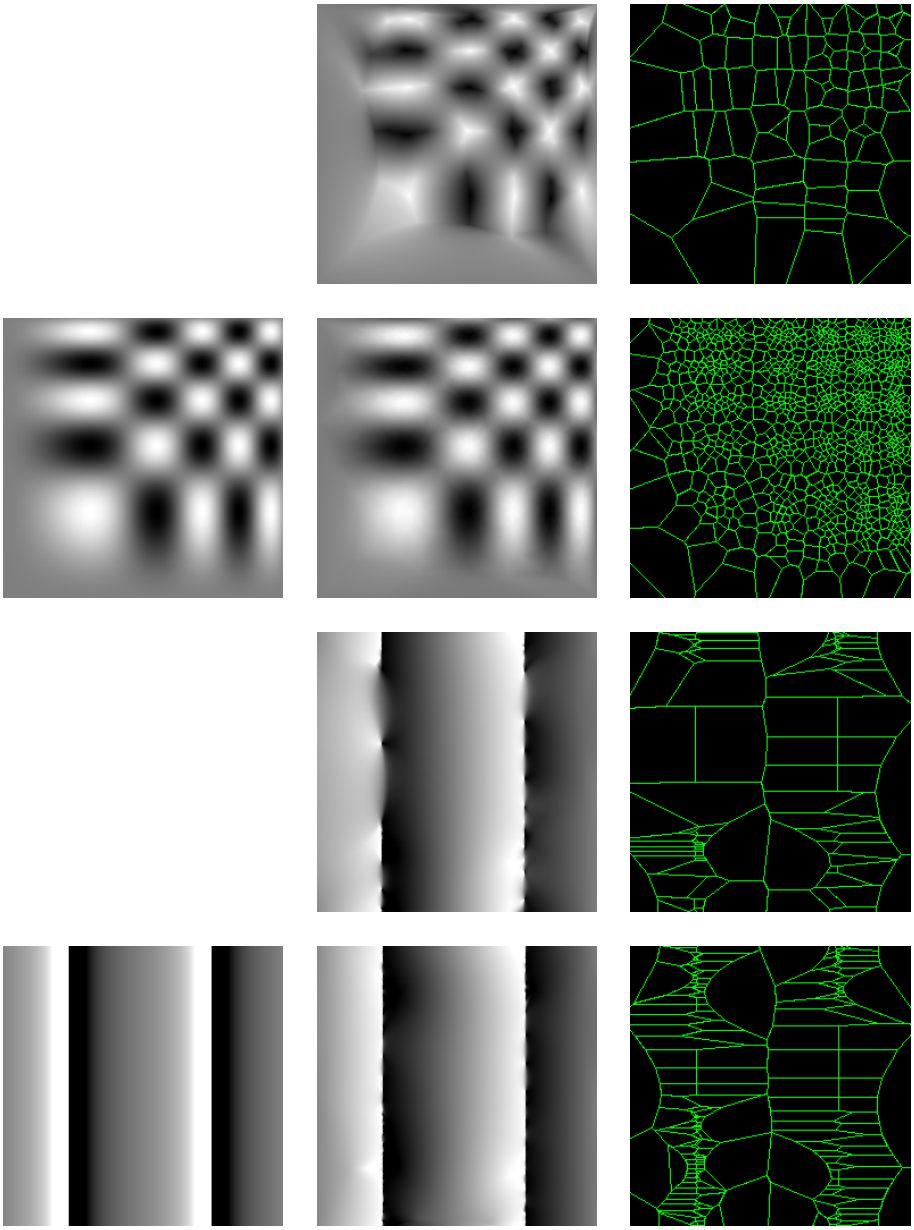

Figure 2.5: Approximations of $f_1(x, y) = sin(x^2)sin(y^2)$ based on 100 and 1000 samples (top) and of $f_2(x, y) = tan(x)$ based on 100 and 200 samples (bottom). Left: original function; middle: Sibson's interpolant; right: underlying Voronoi diagrams. Approximation errors are listed in Table 2.1.

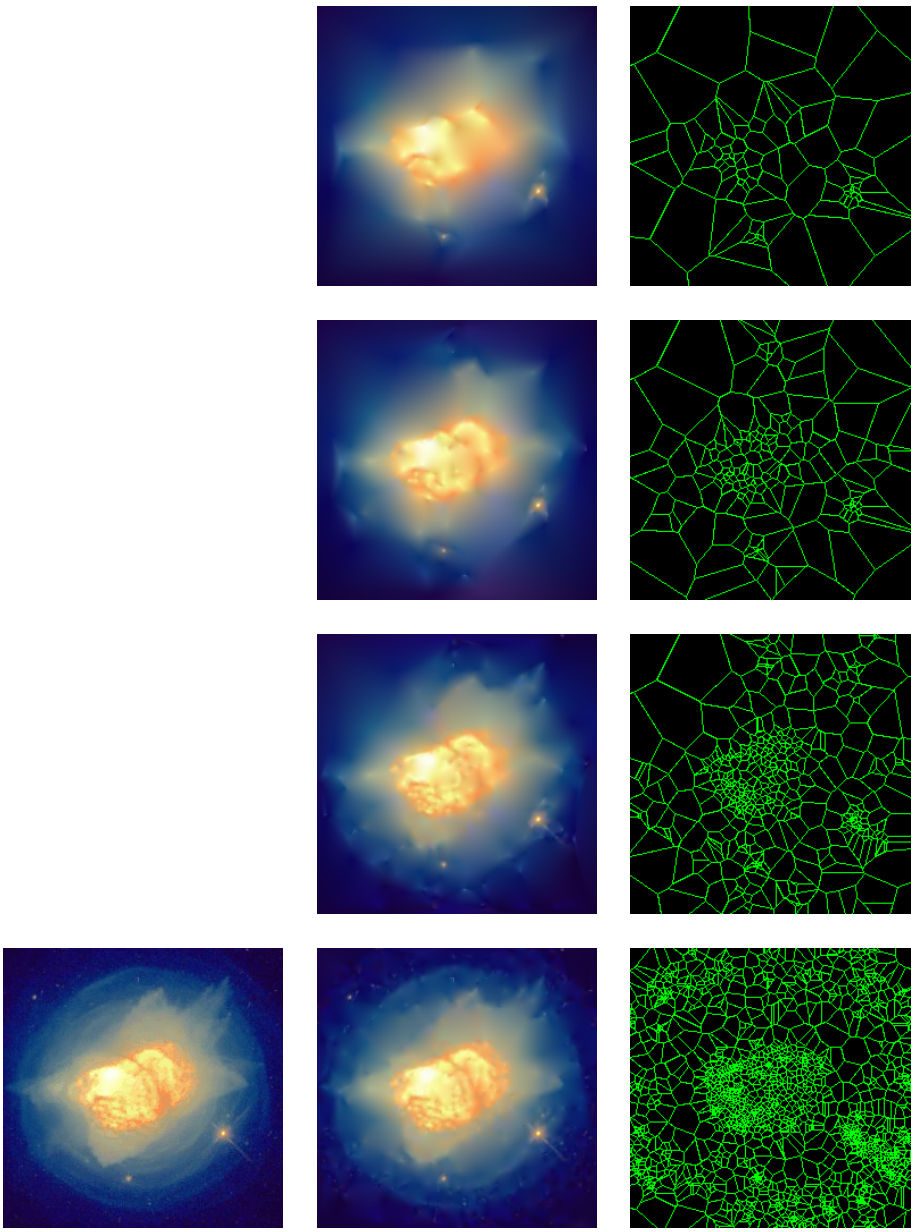

Figure 2.6: Approximations of "Dying Sun" image at different levels of detail, based on 100, 200, 500, and 1000 tiles, see Table 2.1. Left: original image; middle: Sibson's interpolant; right: underlying Voronoi diagrams.

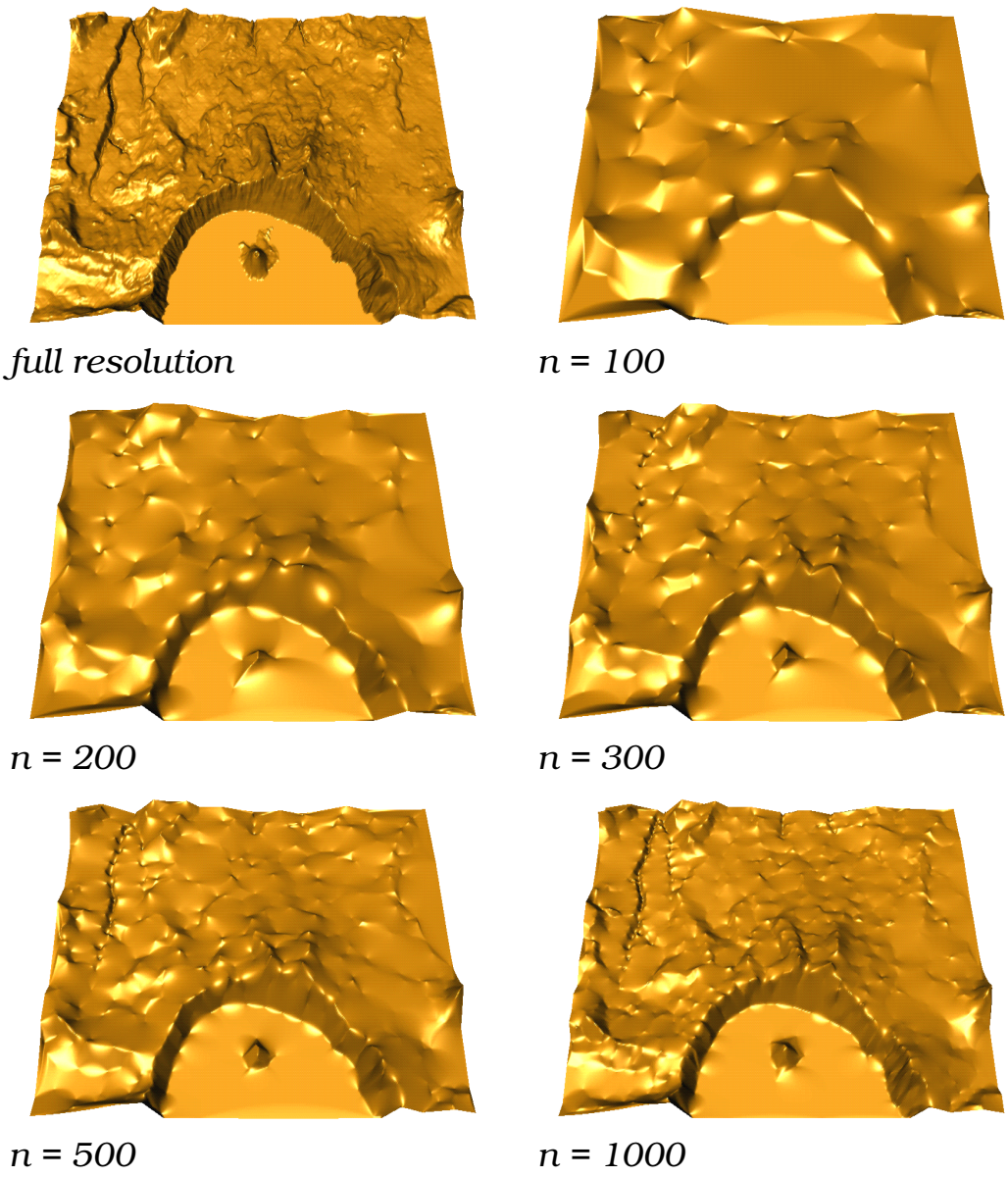

Figure 2.7: Crater Lake terrain data set at different levels of resolution.

2.3 Binary Space Partition (BSP) Trees

BSP trees [34] are generated by clustering schemes that recursively split cluster regions in s-dimensional space along a $(k-1)$-dimensional hyperplane. Every node in the BSP tree corresponds to a splitting step. Adjacency of clusters at any level of resolution can efficiently be recovered by traversing the tree. Thus, BSP trees are well suited to recover topologies of two-manifold surfaces in three-dimensional space or, more generally, to reconstruct $(s-1)$-manifolds in s-dimensional space.

One of the most common problems in scientific visualization is the extraction of surfaces especially isosurfaces of scalar fields. We assume that the geometry of a surface is given by discrete samples in three-dimensional space, possibly with additional function values defining normals, color, etc. To visualize surfaces efficiently, modern graphics hardware requires a surface representation of polygons, preferably triangles. The number of triangles to be rendered has a major impact on the efficiency of the visualization process. Thus, a progressive clustering scheme from which a triangulated surface representation can be derived is desirable.

Although all clustering approaches discussed in the following sections generalize to s dimensions, we emphasize the application of reconstructing a triangulated surface and describe all necessary implementation details. The clustering schemes differ mostly in their use of an approximating function $\tilde{f}_k^j$ and in their choice of a splitting plane for refinement.

2.3.1 Principal Component Analysis (PCA)

A PCA-based clustering approach for surface reconstruction is introduced by Heckel *et al.* [64, 65]. We briefly summarize this work, since it matches exactly our general definition of adaptive clustering. This approach makes use of PCA for the construction of an approximating function $\tilde{f}_k^j$ that pro-

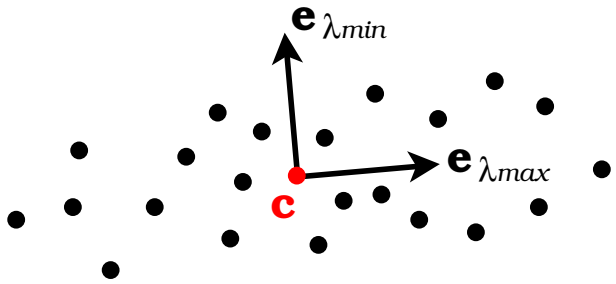

Figure 2.8: PCA provides a coordinate system aligned with scattered points. If these points define a surface, then $\mathbf{e}_{\lambda_{min}}$ is a local estimate for the surface normal.

vides the local distance from a reconstructed two-manifold surface, and for the estimation of a splitting plane.

In the following, we explain how PCA can be used to define a data-dependent coordinate system in s dimensions. Given a "cloud" of points $\mathbf{p}_i = (p_{i1} \cdots \mathbf{p}_{ik})^T,\ i = 1, \ldots, n$, the centroid is defined as

$$\mathbf{c} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{p}_i.$$

We use the centroid as origin for a local coordinate system. Eigenanalysis of the symmetric matrix

$$a_{jk} \ = \ \frac{1}{n} \sum_{i=1}^{n} (p_{ij} - c_j)(p_{ik} - c_k) \tag{2.3}$$

defines a set of orthonormal eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ that serve as local basis vectors. The square-roots of the corresponding real and positive eigenvalues, $\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_k}$, determine the amounts by which the cloud of points is scaled in direction of the corresponding eigenvectors, see Figure 2.8.

If the points $\mathbf{p}_i$ define a s-manifold, then the eigenvector $\mathbf{e}_{\lambda_{min}}$ with smallest eigenvalue is assumed to be locally orthogonal to the manifold. Hence, $\tilde{f}_k^j$ can be defined as signed distance to the manifold:

$$\tilde{f}_k^j(\mathbf{x}) \ = \ (\mathbf{x} - \mathbf{c}) \cdot \mathbf{e}_{\lambda_{min}}.$$

A splitting plane, or a hyperplane in general, is chosen such that it interpolates the centroid $\mathbf{c}$ and is orthogonal to the eigenvector $\mathbf{e}_{\lambda_{max}}$ with maximal eigenvalue. This choice of the splitting plane ensures that every cluster is split orthogonal to its longest axis, resulting in well-shaped clusters.

We note that the clustering process only considers the points $\mathbf{p}_i$ and ignores associated function values f_i. This can be changed by assigning a positive weight w_i, defined by the approximation error, to every point p_i. When performing PCA, the weighted centroid

$$\mathbf{c} = \frac{\sum_{i=1}^{n} w_i \, \mathbf{p}_i}{\sum_{i=1}^{n} w_i}$$

is used as origin for the local coordinate system and the matrix (2.3) is replaced by

$$a_{jk} = \frac{\sum_{i=1}^{n} w_i (p_{ij} - c_j)(p_{ik} - c_k)}{\sum_{i=1}^{n} w_i}. \tag{2.4}$$

It is shown in [65] how to obtain a triangulation for a two-manifold surface from the result of the clustering process.

2.3.2 Quadratic Polynomial Approximation

We present a similar clustering approach for efficient triangulation of scattered data in the plane [10]. This approach can be generalized to triangulate two-manifolds by projecting the data locally onto a plane obtained by PCA. A major difference to the clustering method explained above is that the approximating functions $\tilde{f}_k^j$ are quadratic polynomials representing smooth surfaces much better than piecewise linear approximations.

In the following, we summarize our adaptive clustering approach [10] [10] that is based on quadratic polynomial approximations. The construction of

high quality triangulations outgoing from this representation is discussed in Section 2.4. We assume that the given data set is composed of points $\mathbf{p}_i$ in the plane with associated scalar function values f_i. A generalization to higher dimensions is possible.

For every cluster s at level L_j, a quadratic polynomial

$$\tilde{f}_k^j(x_1, x_2) \;=\; \sum_{l,\; m\geq 0;\; l+m\leq 2} c_{l,m}\; x_1^l\; x_2^m \;=\; \sum_{0\leq |\mathbf{l}|\leq 2} \mathbf{c_l}\; \mathbf{x^l} \qquad (2.5)$$

approximating the function values f_i at the locations $\mathbf{p}_i \in \tau_k^j$ is estimated by least-squares fitting, see Appendix A.

For the splitting process, a straight line is defined along which a cluster is subdivided. Therefore, we construct two weighted centers, called $\mathbf{c}^+$ and $\mathbf{c}^-$, which correspond to the positive and negative errors, respectively. The centers are given by

$$\mathbf{c}^+ \;=\; \frac{\sum_{i=1}^n\; w_i^+\; \mathbf{p}_i}{\sum_{i=1}^n\; w_i^+}, \quad w_i^+ \;=\; \begin{cases} \tilde{f}_k^j(\mathbf{p}_i) - f_i, & \text{if } \tilde{f}_k^j(\mathbf{p}_i) - f_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

and

$$\mathbf{c}^- \;=\; \frac{\sum_{i=1}^n\; w_i^-\; \mathbf{p}_i}{\sum_{i=1}^n\; w_i^-}, \quad w_i^- \;=\; \begin{cases} f_i - \tilde{f}_k^j(\mathbf{p}_i), & \text{if } \tilde{f}_k^j(\mathbf{p}_i) - f_i < 0 \\ 0, & \text{otherwise.} \end{cases}$$

A cluster is subdivided along the bisector of the line segment $\overline{\mathbf{c}^+\mathbf{c}^-}$ (Figure 2.9). If $\mathbf{c}^+$ and $\mathbf{c}^-$ are identical, then the direction of subdivision will be arbitrary.

We provide a description of the clustering algorithm in pseudocode, using notation from Section 2.1. At a high level, these are the steps to be performed:

(i) Establish level L_0 containing a single cluster, $\tau_1^0 = D$. The function $\tilde{f}_1^0$ is a quadratic polynomial resulting from a least-squares fit based on all points p_i, as explained above. The residuals are estimated based on the error norm in equation (2.1), with $p = \infty$ (maximum norm).

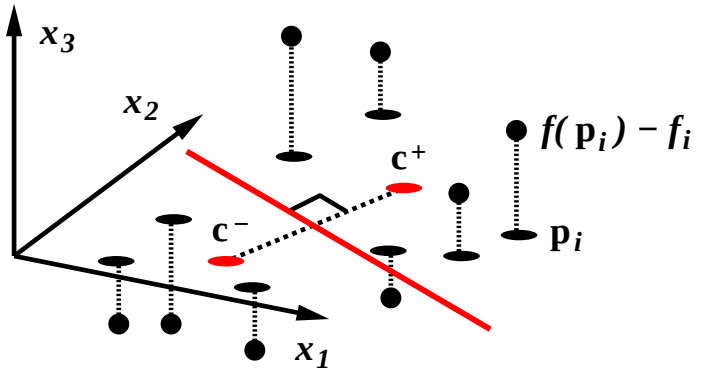

Figure 2.9: Subdivision along perpendicular bisector of $\mathbf{c}^+$ and $\mathbf{c}^-$.

(ii) Refinement: $L_j \rightarrow L_{j+1}$. Let m be the index of a maximal residual in L_j, *i.e.*, $\varepsilon_m^j \geq \varepsilon_k^j \; \forall k = 1, \ldots, n_j$. Split tile τ_m^j as described above. For the two resulting tiles, construct the approximating quadratic polynomials and the residuals using the maximum norm. All other tiles remain unchanged.

(iii) Terminate the process when a prescribed global error bound is satisfied by all residuals ε_k^j or when a prescribed number of tiles is reached. Otherwise, increment j and continue with (ii).

We use the maximum error norm $p = \infty$ to approximate the given data within a prescribed tolerance. In Section 2.4, we construct a triangulation for every approximating quadratic polynomial with a minimal number of triangles so that a prescribed maximal error bound is satisfied. Numerical examples are provided in Section 2.5.

2.3.3 Affine-invariant Approach

Both clustering schemes introduced above provide clusters that remain the same if a data set is translated, rotated, or uniformly scaled, but they are not invariant under affine transformations like a shear.

An affine-invariant norm is discussed in [102]. To exploit this idea for clustering, we can transform a set of points p_i into the coordinate system defined by PCA, see Section 2.3.1, and scale every coordinate corresponding to the eigenvector $\mathbf{e}_i$ by $\frac{1}{\sqrt{\lambda_i}}$. The coordinates obtained from this transformation are now affine-invariant. A splitting hyperplane can be defined in these coordinates and transformed back into the previous system, see Figure 2.10.

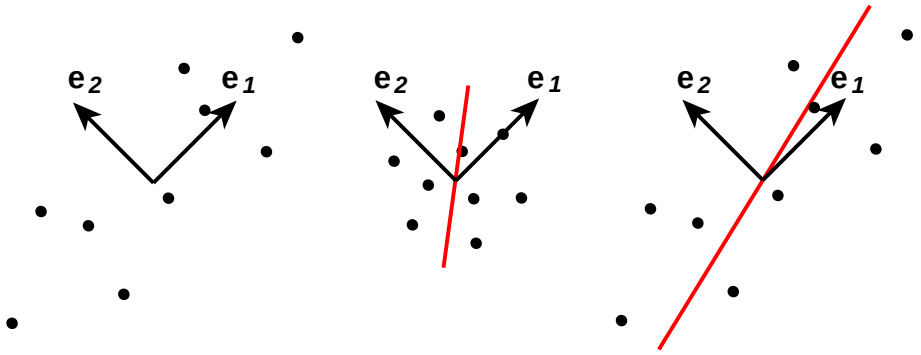

Figure 2.10: Left: set of points p_i and induced normalized eigenvectors; middle: estimated splitting line in affine-invariant coordinates; right: resulting splitting line.

We have implemented this adaptive clustering approach for scattered points in the plane with associated function values. The PCA-based splitting method provides well-shaped clusters, but it does not consider the associated function values. The splitting approach based on bisectors between centers of positive and negative errors typically provides a lower number of clusters than PCA within the same approximation error, because it generates long and skinny clusters when the associated function is curved in only one direction. The affine invariant approach, however, provides poor approximation behavior in some regions where clusters are recursively split along similar directions. It may work well on densely sampled, smooth data sets.

2.4 Optimal Approximation with Triangles

A new hierarchical and data-dependent triangulation approach, based on the adaptive clustering scheme of Section 2.3.2, has recently been published [10]. This approach uses piecewise quadratic polynomial approximation as intermediate data representation. The cost for generating a triangulation depends only on the complexity of the clustering and on the error bound, but not on the number of points in the original data set.

2.4.1 Related Work

A variety of triangulation approaches optimizing different geometrical criteria or analytic cost functions exist, but most of them are not based on clustering. Some early approaches, see for example De Floriani *et al.* [36], apply the Delaunay triangulation that leads to low aspect ratios of the resulting triangles (the ratios between the radii of the circumscribed and inscribed circles for all triangles). However, in the case of data-dependent triangulations the use of long and skinny triangles can reduce approximation errors [19, 43, 95, 111]. To obtain a global optimum for a cost function, multidimensional optimization methods, such as *simulated annealing*, can be applied [79, 116].

A different class of algorithms does not rely on the original data points as triangle vertices. Nadler [100] and D'Azevedo [4] apply a coordinate transformation to generate an optimal triangulation for a certain class of analytical functions. This transformation is based on principal axes, which are also used in our approach. Quak et al. [108] apply least-squares fitting to linear splines to approximate scattered data.

Considering the large amount of data produced by supercomputer simulations, it becomes important to handle data locally and on multiple levels of detail for interactive visualization. A variety of view-dependent triangulation

algorithms, are discussed in [42, 71, 73, 130]. For more general multiresolution triangulations, we refer to [37, 52, 55, 62]. Algorithms for adaptive two-manifold surface reconstruction from scattered points are described in [47, 65, 68].

Once a triangulation is constructed for a given scattered data set, the mesh can be reduced to a coarser level of detail. Mesh reduction strategies are described, for example, in [69, 113]. Not only is elimination of detail possible, but also synthesis. In this context, the concept of the discrete wavelet transform (DWT) [93], which is a highly efficient method for signal processing and data compression on regular grids, has been generalized to arbitrary triangular meshes [33, 58].

2.4.2 Triangulation Approach

Most triangulation approaches operate directly on the data. In contrast, our method is based on adaptive clustering and a piecewise quadratic representation. Quadratic polynomials have the property that their graph surfaces can be approximated by a *regular* triangulation in such a way that all triangles imply the same approximation error. This property provides a method for rapidly generating a set of optimally shaped triangles for every cluster region. The individual triangulations are "stitched" together to a single triangulation by exploiting the connectivity implied by a BSP tree. The overall triangulation method is illustrated in Figure 2.11.

The construction of an optimal triangulation for a quadratic graph surface is discussed in [107], which provides the basis of our constructive approach. We call a triangulation *optimal* when it approximates a quadratic polynomial within a prescribed error bound and consists of triangles with maximal area. The basic idea is to exploit affine invariance of the error norm to transform a quadratic graph surface to a prototype for which a regular triangulation satisfying an error bound is known.

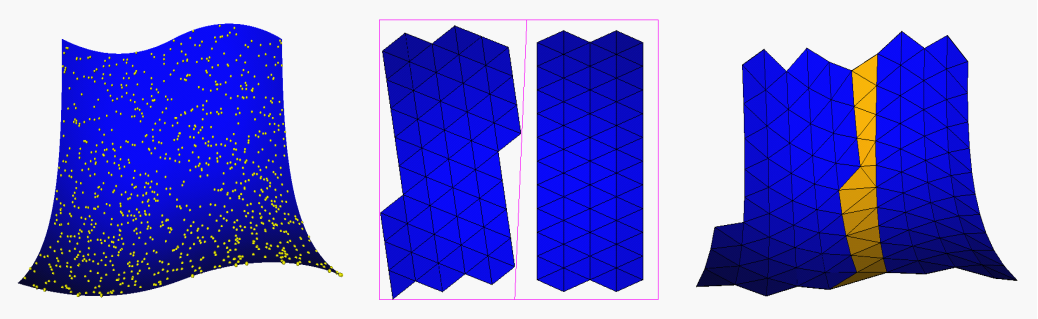

Figure 2.11: Left: scattered points sampled from smooth graph surface; middle: clusters with optimal triangulations; right: final triangulation.

2.4.3 Principal Axis Transformation

In matrix notation a quadratic function can be written as

$$f(x,y) \;=\; (x\ y)\begin{pmatrix} c_{20} & c_{11} \\ c_{11} & c_{02} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \;+\; (c_{10}\ c_{01})\begin{pmatrix} x \\ y \end{pmatrix} \;+\; c_{00}. \qquad (2.6)$$

To analyze its principal axes, *i.e.*, the directions of minimal and maximal second derivative (or curvature), we can neglect the constant and linear terms that do not affect approximation error. Thus, we only need to consider the quadratic form

$$(x\ y)\begin{pmatrix} c_{20} & c_{11} \\ c_{11} & c_{02} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \mathbf{x}^T C\mathbf{x} \;=\; 0. \qquad (2.7)$$

The graph surface's principal curvatures are implied by the eigenvalues λ_1 and λ_2 of the matrix C and the principal axes are determined by the corresponding eigenvectors $\mathbf{e_1}$ and $\mathbf{e_2}$ [16]. Depending on the sign of the eigenvalues, we have to distinguish between three different surface types:

- The surface is elliptic, *i.e.*, $\lambda_1\lambda_2 \;> 0$.
- The surface is hyperbolic, *i.e.*, $\lambda_1\lambda_2 \;< 0$.

- The surface is parabolic (or planar), *i.e.*, $\lambda_1\lambda_2 = 0$.

For every type, a different optimal triangulation scheme is used.

2.4.4 The Elliptic Case

The graph surface of a quadratic function f of elliptic type is equivalent to the graph surface of the paraboloid

$$g(\overline{x},\overline{y}) \;=\; \overline{x}^2+\overline{y}^2, \tag{2.8}$$

i.e., there exists a linear transformation for the arguments of g that produces the same graph surface as f, neglecting linear and constant terms. In the case of positive eigenvalues, this transformation is given by

$$\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}}\, e_{11} & \frac{1}{\sqrt{\lambda_2}}\, e_{21} \\ \frac{1}{\sqrt{\lambda_1}}\, e_{12} & \frac{1}{\sqrt{\lambda_2}}\, e_{22} \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} \;=\; G \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}, \tag{2.9}$$

where the matrix entries are determined by the normalized eigenvectors $\mathbf{e_1} = (e_{11}, e_{12})^T$ and $\mathbf{e_2} = (e_{21}, e_{22})^T$. This result can be verified by inserting (2.9) into the quadratic form (2.7). Considering the orthogonality of eigenvectors, G^TCG is the identity matrix, and it therefore reproduces the quadratic form of g.

We need to construct an optimal triangulation for g, which will also be optimal for f, after applying the above transformation. Due to the rotational symmetry of g, all triangles that share the same circumscribed circle of radius $\sqrt{2\varepsilon}$ have the same maximal error ε. Hence, an equilateral triangle, with arbitrary orientation in the xy-plane — like the one in Figure 2.12 given by the points $(-\sqrt{2\varepsilon}, 0)^T$, $(\sqrt{0.5\varepsilon}, \sqrt{1.5\varepsilon})^T$, and $(\sqrt{0.5\varepsilon}, -\sqrt{1.5\varepsilon})^T$ — has maximal area. The maximal error occurs at the center and at the vertices of the triangle.

In the case of positive eigenvalues, the vertices must be placed below the quadratic surface by a distance ε to obtain the error profile shown in

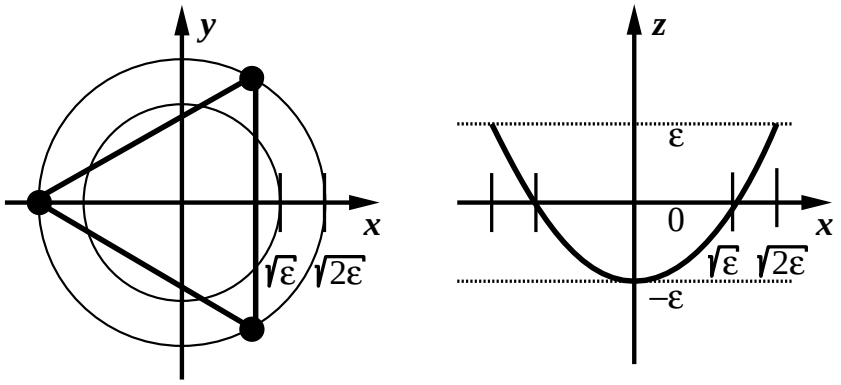

Figure 2.12: Optimal triangle and error profile for $g(x, y) = x^2 + y^2$.

Figure 2.12. For negative eigenvalues, the function f can be replaced by $-f$, which implies positive eigenvalues. The only difference in this case is that the vertices need to be placed above the surface by ε.

To obtain the final triangulation only two of the edges need to be transformed according to (2.9). The transformed edges define a regular triangulation that is established for the convex region corresponding to the cluster, emanating from its centroid, see Figure 2.11. Only triangles that lie entirely in the cluster region are generated, leaving an untriangulated gap along the cluster boundaries.

2.4.5 The Hyperbolic Case

In analogy to the elliptic case, we can choose a single prototype for a hyperbolic quadratic polynomial. We choose the polynomial

$$h(\overline{x}, \overline{y}) \;=\; \overline{x}^2 - \overline{y}^2. \tag{2.10}$$

To transform the quadratic form of h into the quadratic form of a given quadratic hyperbolic function f, with eigenvalues $\lambda_1 > 0$ and $\lambda_2 < 0$ and normalized eigenvectors $\mathbf{e_1} = (e_{11}, e_{12})^T$ and $\mathbf{e_2} = (e_{21}, e_{22})^T$, we apply the

map

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}}\, e_{11} & \frac{1}{\sqrt{-\lambda_2}}\, e_{21} \\ \frac{1}{\sqrt{\lambda_1}}\, e_{12} & \frac{1}{\sqrt{-\lambda_2}}\, e_{22} \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} = H \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}. \qquad (2.11)$$

The transformation is correct, since it reproduces the quadratic form of h:

$$H^T C H = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Due to the nature of hyperbolic surfaces, it seems necessary to place the triangle vertices exactly on the surface. Offsetting a vertex may reduce the error of a single triangle, but, at the same time, will increase the error of an adjacent triangle. As shown in [107], an optimal triangle for the polynomial h has the vertices $(\sqrt{\varepsilon}, \sqrt{\varepsilon})^T$, $(\sqrt{\varepsilon}, -\sqrt{\varepsilon})^T$, and $\left((1-\sqrt{5})\sqrt{\varepsilon}, 0\right)^T$, see Figure 2.13.

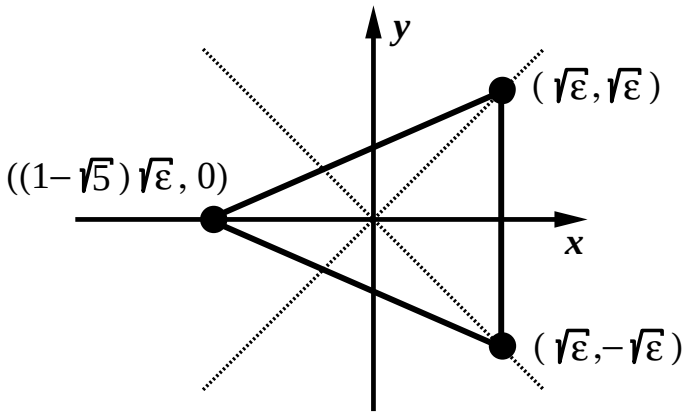

Figure 2.13: Optimal triangle for hyperbolic polynomial $h(x, y) = x^2 - y^2$.

Since the vertices interpolate the surface, the maximal error is located at the midpoint of every triangle edge. We note that the graph surface h has zero curvature (and zero second derivatives) along the lines $x + y = 0$ and $x - y = 0$, which implies that the error inside a triangle cannot be greater

than the maximal error on its edges. The maximal error for any edge $\overline{\mathbf{ab}}$ is given by

$$\varepsilon_{\mathbf{ab}} \;=\; \frac{1}{4}\left|(a_x - b_x)^2 - (a_y - b_y)^2\right|. \tag{2.12}$$

The final triangulation is established in the same way as in the case of an elliptic surface, using two edges transformed according to (2.11). (In the case of $\lambda_1 < 0$ and $\lambda_2 > 0$, the triangulation can be obtained by considering the quadratic form of $-f$.)

2.4.6 The Parabolic Case

In the parabolic case, one of the two eigenvalues, and consequently one of the principal curvatures is zero. (In the case of two zero eigenvalues the surface is a plane, and the cluster region can be triangulated by only considering the vertices of its convex boundary polygon.) We now consider the case $\lambda_1 > 0$ and $\lambda_2 = 0$. We can transform the quadratic form of the function

$$p(\overline{x}, \overline{y}) \;=\; \overline{x}^2 \tag{2.13}$$

into the quadratic form of f by using the map

$$\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}}\, e_{11} & 0 \\ \frac{1}{\sqrt{\lambda_1}}\, e_{12} & 0 \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}. \tag{2.14}$$

The error profile for a line segment is uniquely determined by its projection onto the first normalized eigenvector $\mathbf{e}_1$ of the quadratic form (Figure 2.14). The length of this component should be at most $\sqrt{4\varepsilon/\lambda_1}$. As in the elliptic case, the vertices need to be placed below the surface by the distance ε for $\lambda_1 > 0$ and above it for $\lambda_1 < 0$. In the case that $\lambda_1 = 0$ and $\lambda_2 \neq 0$ the eigenvalues and eigenvectors simply change their roles. The cluster region is finally triangulated by two, typically skinny triangles for every subregion of width $\sqrt{4\varepsilon/\lambda_1}$. To leave some space at the cluster boundary

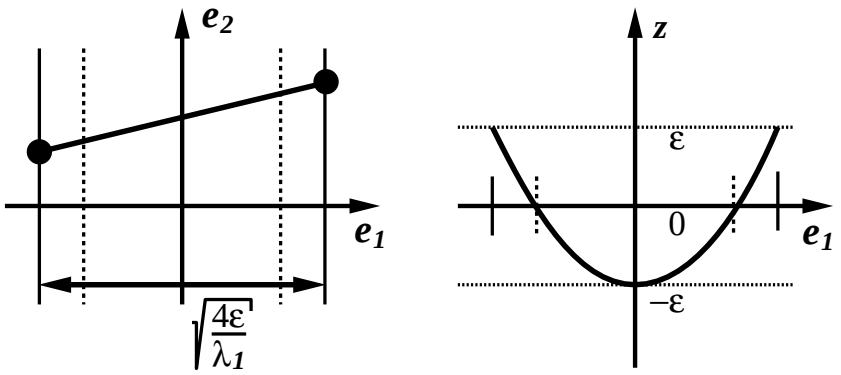

Figure 2.14: Line segment with error ε on parabolic surface.

for the stitching process, we shrink the cluster region by a small percentage towards its centroid.

2.4.7 Merging Triangulations

Based on the BSP tree defining the cluster hierarchy and the set of individual optimal triangulations, the final step of the algorithm stitches the triangulations together by triangulating the gaps along the cluster boundaries. This is done in reverse order of the cluster subdivision process, which allows us to stitch exactly two triangulations at a time along the bisecting lines that separate the corresponding cluster regions, see Figure 2.15.

To efficiently identify the polygon strips that need to be connected, we keep track of the boundary of every triangulation. We initially need to construct a closed boundary polygon for every optimal triangulation. This is straightforward in the case of a triangulation resulting from a parabolic or planar graph surface. In the elliptic and hyperbolic cases, the triangulations inside the cluster region may be disconnected or contain line segments that are not part of any triangle. Some special cases are shown in Figure 2.16.

The boundary of an optimal triangulation is defined as a single "loop"

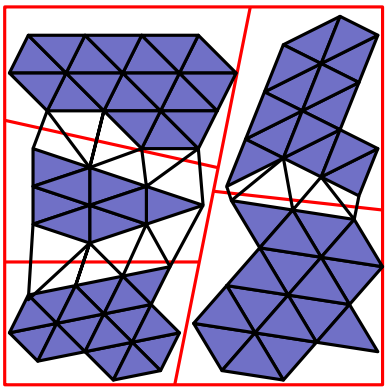

Figure 2.15: Pairs of triangulations are stitched together along straight-line cluster bisectors.

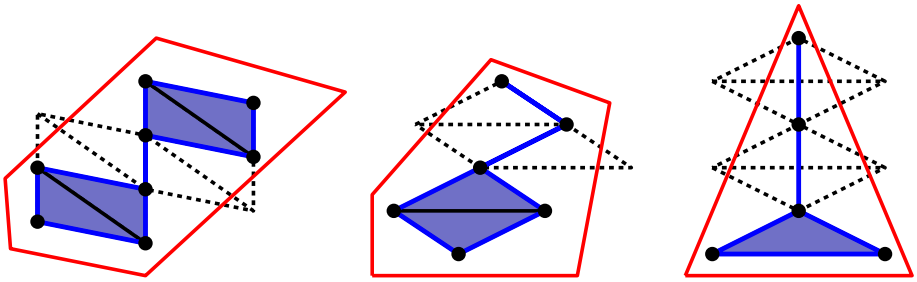

Figure 2.16: Special stitching cases: All simplices — triangles, edges, and vertices — must be enclosed by the boundary polygon.

that encloses all triangles, edges, and vertices of the optimal triangulation that are inside the cluster region. It does not enclose areas other than optimal triangles. Thus, a vertex can be part of the boundary polygon multiple times.

To efficiently construct a triangulation boundary, we use a scan-line algorithm that identifies the left- and right-most vertices inside the cluster region for every scan line parallel to the longest edge in a triangulation. In a second step, additional vertices inside the cluster region are added to the boundary polygon to ensure that the enclosed area contains only optimal triangles. The boundary construction process is illustrated in Figure 2.17.

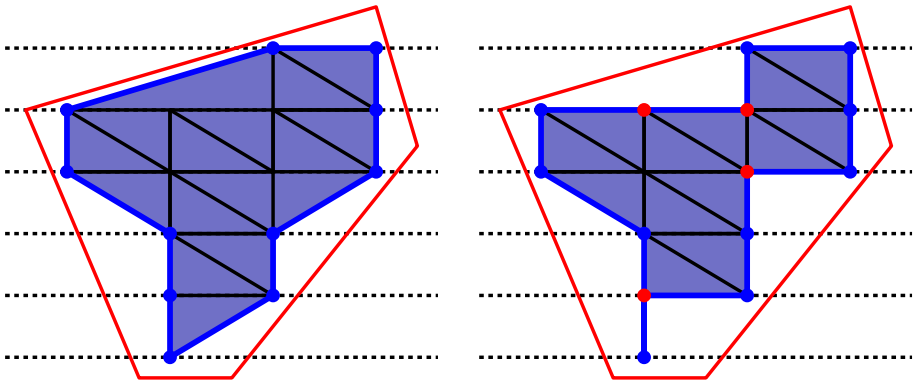

Figure 2.17: Boundary construction for optimal triangulations. Left: boundary vertices for every scan-line are inserted into triangulation boundary; right: additional vertices are used to minimize enclosed area.

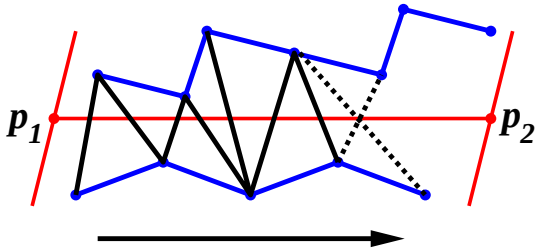

Figure 2.18: Shorter edges are preferred when stitching two triangulations.

In Figure 2.18, two optimal triangulations that need to be merged are separated by the line segment $\overline{\mathbf{p}_1\mathbf{p}_2}$. First, we extract from the two triangulation boundaries two polygon strips along $\overline{\mathbf{p}_1\mathbf{p}_2}$, starting with the vertex closest to $\mathbf{p}_1$ and ending with the one closest to $\mathbf{p}_2$. Second, we construct triangles by marching along the two polygon strips and by connecting point pairs using vertices from opposite sides of $\overline{\mathbf{p}_1\mathbf{p}_2}$. Considering the two possible choices for defining the next triangle, we choose an edge with minimal length, see Figure 2.18. One must not produce triangles with negative area, *i.e.*, triangles that overlap others. In the case that both choices would produce triangles with negative areas, we construct a triangle from three consecutive vertices in one of the polygon stips and thus eliminate the vertex in the

middle. If the stitching process would still produce negative areas for both choices, we insert a triangle from three vertices of the other polygon strip.

The triangles resulting from the stitching process are not guaranteed to satisfy the given error bound. For a rigorous treatment, the final approximation errors must be estimated for all scattered data points located inside the regions in the xy-plane defined by the triangles resulting from the stitching procedure. Triangles that do not satisfy the error bound need to be modified. This can be done by flipping an edge between two triangles or by inserting the sample with greatest approximation error as a vertex into the triangulation. The errors for all modified triangles need to be checked again. In our numerical examples, we use a greater error bound for quadratic approximation than for triangulating polynomials. This implies that the total error bound is relatively loose for the triangles obtained from the stitching process and that violations of the error estimate are extremely rare.

2.4.8 Numerical Examples

We have applied our algorithm to approximate scattered data sets sampled randomly from three different analytical functions and to approximate two terrain data sets sampled on regular grids. The analytical functions are defined as follows:

$$f_1(x,y) \;=\; 0.5\;x \;+\; \begin{cases} x^2 \;+\; y^2 & \text{if } x < 0 \\ -x^2 \;+\; y^2 & \text{otherwise,} \end{cases} \qquad x,\; y \,\epsilon\, [-0.5,\; 0.5]\,;$$

$$f_2(x,y) \;=\; e^{-(x^2 \;+\; y^2)}, \quad x,\; y \,\epsilon\, [-2.2,\; 2.2]\,;$$

and

$$f_3(x,y) \;=\; sin(x^2)\; sin(y^2), \quad x,\; y \,\epsilon\, [-3,\; 0]\,.$$

The numbers of clusters and triangles corresponding to different error bounds are shown in Table 2.3. The error bounds (measured in percents of the distance between minimal and maximal function values) are split in a four-to-one ratio between quadratic approximation and triangulation of the polynomials.

Function	No. Samples	Error Bound [%]	No. Clusters	No. Triangles
f_1	1000	1.0	2	179
		0.3	3	662
f_2	3000	2.0	35	437
		0.5	73	1479
f_3	3000	10.0	38	332
		3.0	79	938

Table 2.3: Approximation results for sets of points sampled from analytical functions. Figures 2.19–2.21 show the resulting triangulations.

Dataset, No. Samples	Error [%] Our Method	Error [%] Regular Tri.	No. Clusters	No. Triangles	Comp. Time [sec] (i)	(ii)	(iii)
"St.	10.0	18.7	46	464	0.732	0.046	1.232
Hellens",	5.0	11.5	173	1,799	1.347	0.057	1.539
151,728	1.0	5.4	2,330	28,881	4.361	0.246	3.284
"Crater	10.0	22.9	114	1,149	1.029	0.054	1.464
Lake",	5.0	16.4	400	4,316	1.861	0.077	1.951
159,272	1.0	10.1	2,798	36,456	4.503	0.288	3.799

Table 2.4: Approximation results for terrain data sets. Figures 2.22 and 2.23 show the resulting triangulations.

Approximation results for the two terrain data sets "St. Hellens" and "Crater Lake" are shown in Table 2.4. The corresponding triangulations are depicted in Figures 2.22 and 2.23. We compared the error bounds for our

triangulations with the approximation errors of regular triangulations that have approximately the same number of triangles and that are obtained from a rectilinear grid. The error bounds for our method are split in a nine-to-one ratio between quadratic approximation and triangulation of polynomials. Since the terrain data sets are more "noisy" than samples from analytical functions used above, we need to allocate a larger fraction of the error bound for the quadratic approximation. The computation times shown in Table 2.4 have been obtained on a 194 MHz MIPS R10000 processor. The computation times are split into three categories:

- (i) Adaptive clustering.
- (ii) Constructing the triangulation.
- (iii) Verifying the error bounds for the triangles obtained from stitching.

In the case of tight error bounds, the clustering step becomes the most expensive part of the algorithm, but it needs to be computed only once for all levels of resolution. The computation time for the triangulation (including stitching) is very small and does not depend on the number of scattered points in the initial data set. Checking the approximation errors, however, requires significant more computation time for dense data sets. We suggest to estimate the approximation error from only a small subset of randomly selected samples.

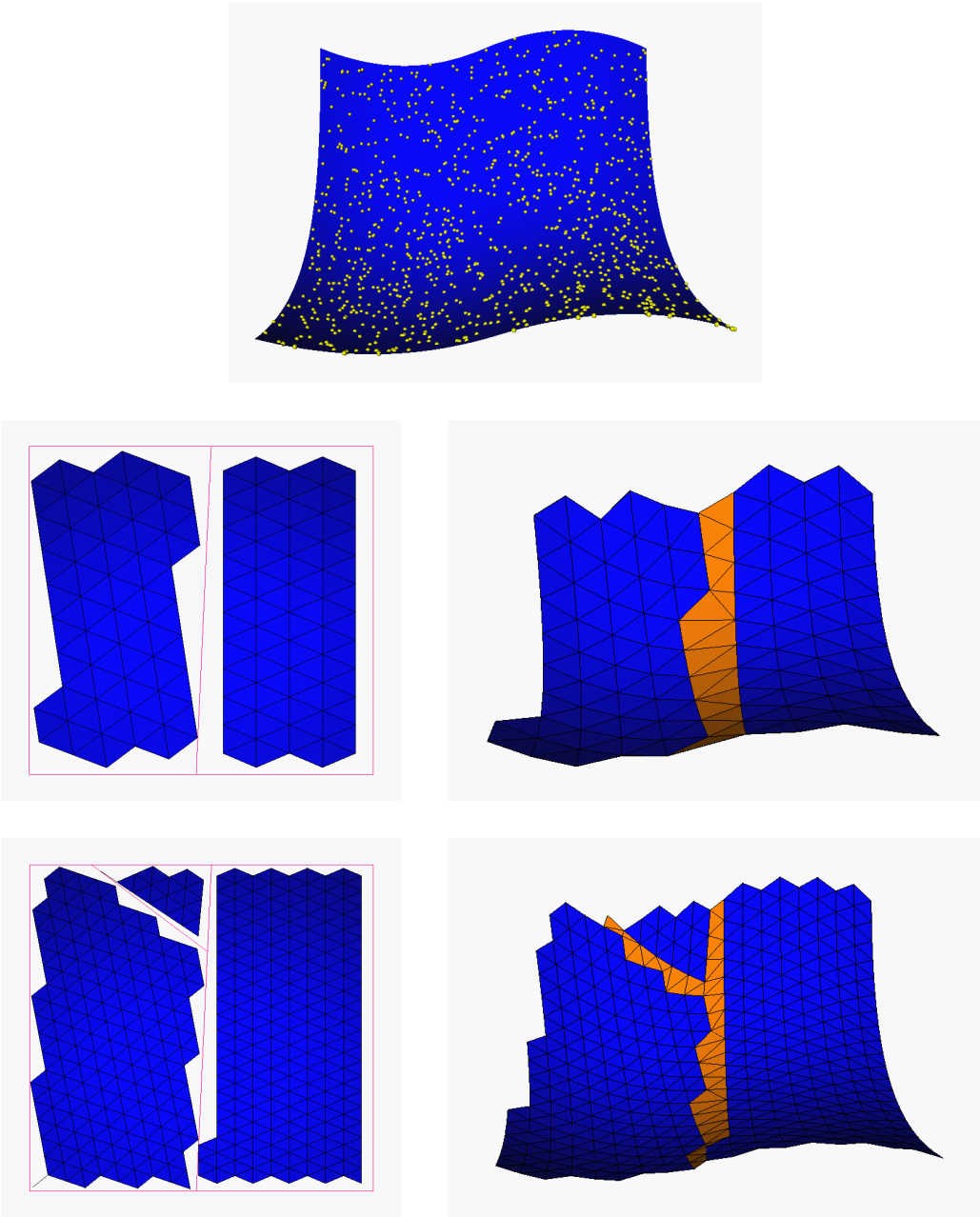

Figure 2.19: Triangulations for f_1. Top: samples on original graph surface; middle: optimal cluster triangulations in xy-plane (left) and final triangulations in xyz-space (right); bottom: same for higher level of resolution.

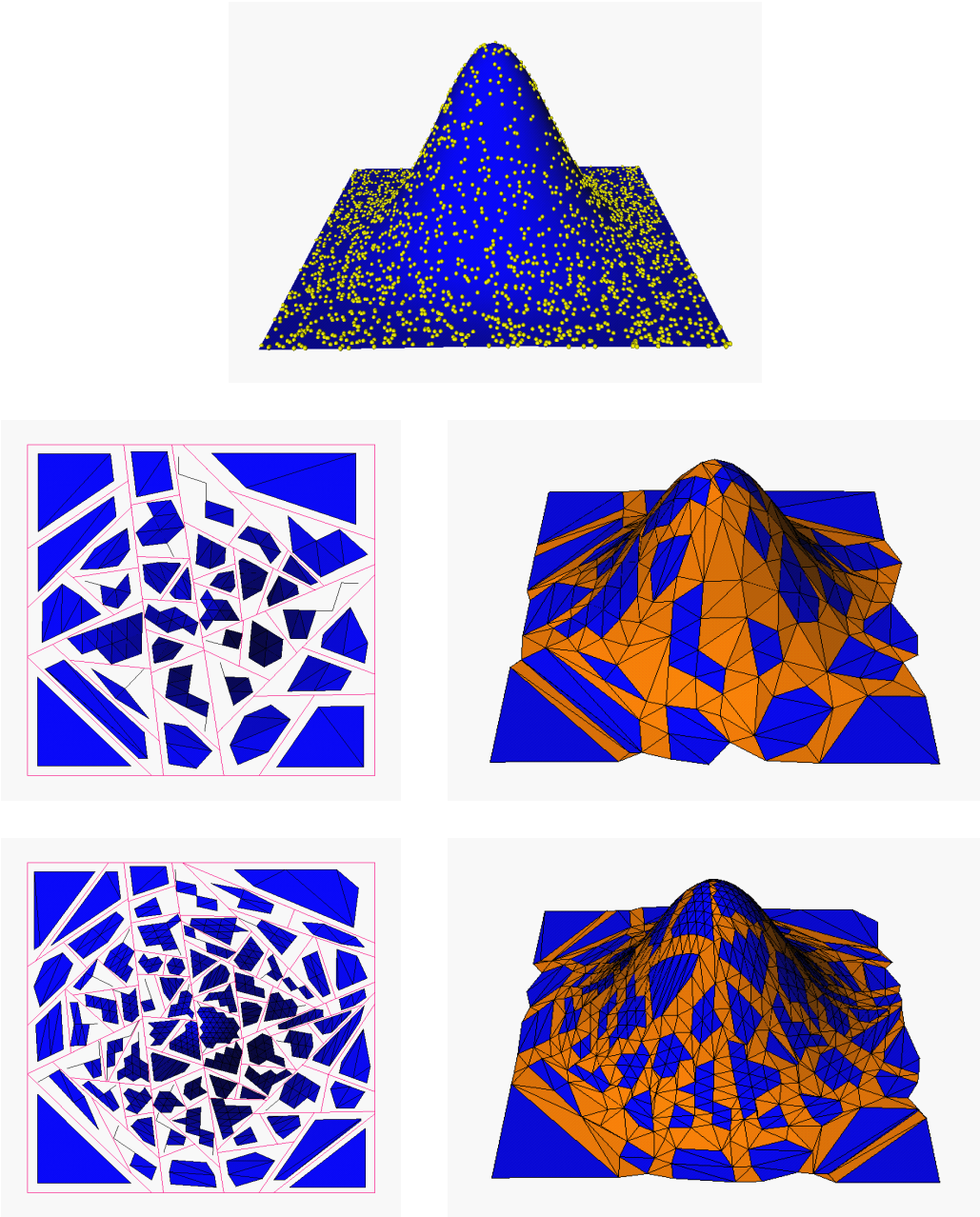

Figure 2.20: Triangulations for f_2. Top: samples on original graph surface; middle: optimal cluster triangulations in xy-plane (left) and final triangulations in xyz-space (right); bottom: same for higher level of resolution.

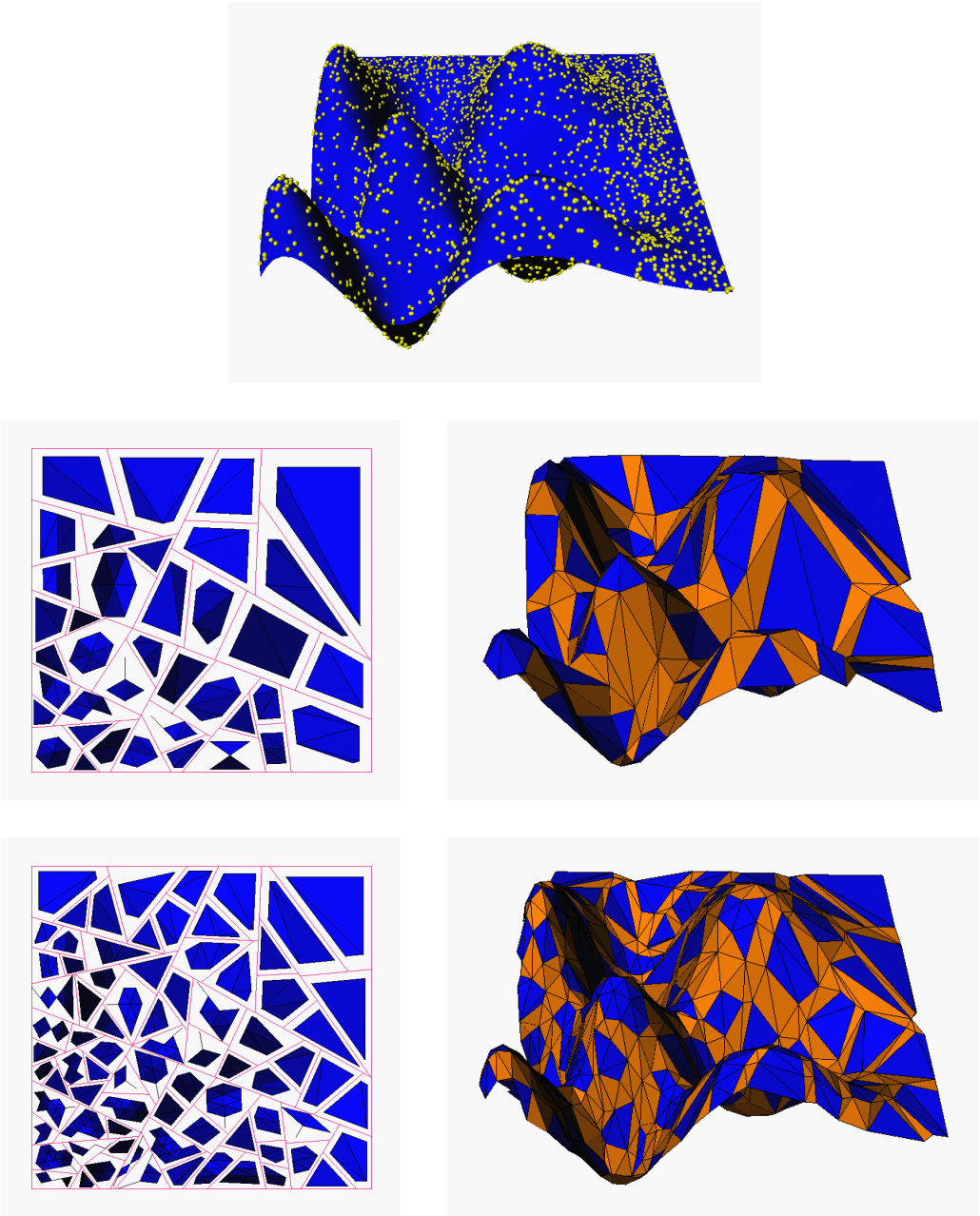

Figure 2.21: Triangulations for f_3. Top: samples on original graph surface; middle: optimal cluster triangulations in xy-plane (left) and final triangulations in xyz-space (right); bottom: same for higher level of resolution.

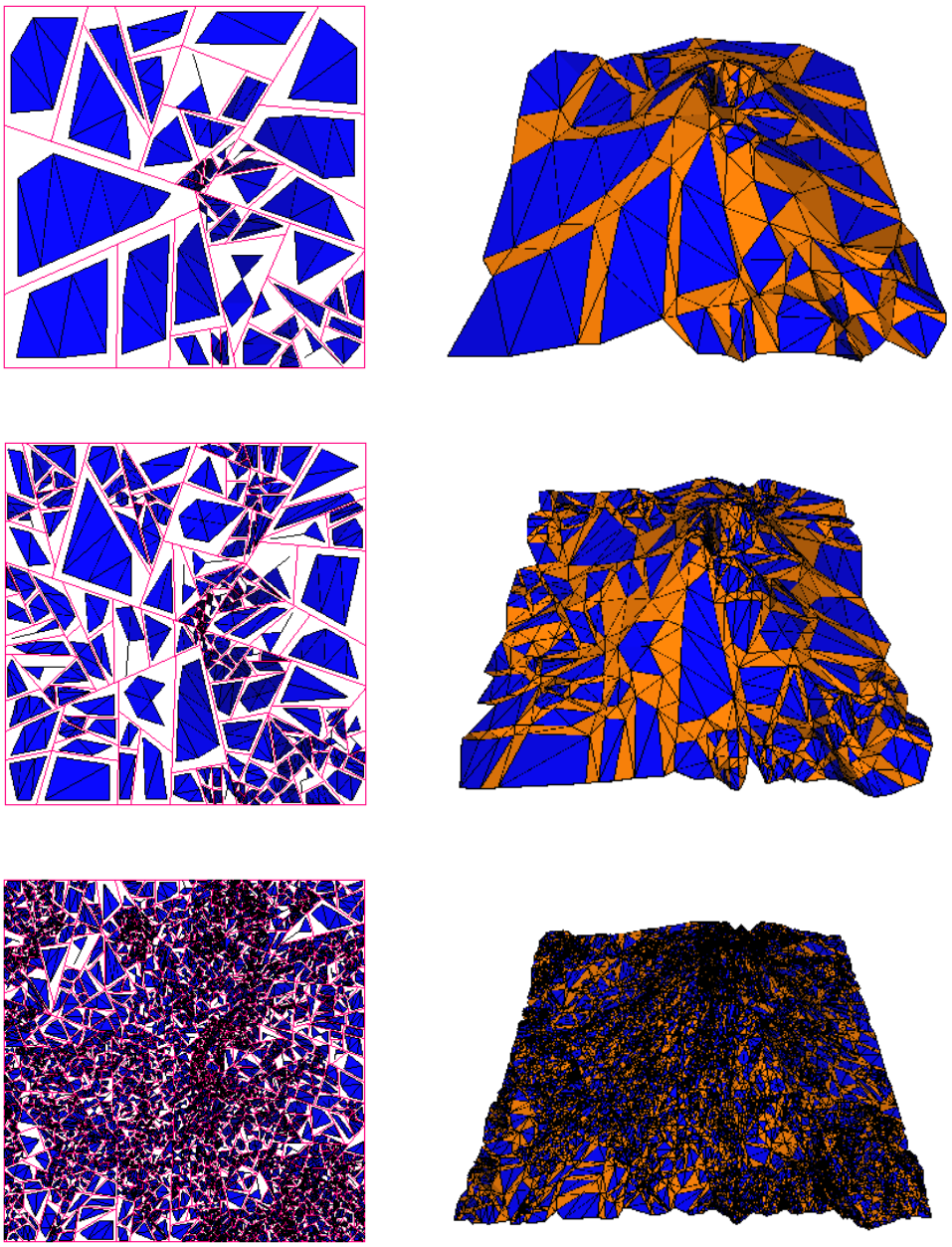

Figure 2.22: Triangulations for data set "St. Hellens." Left: optimal cluster triangulations in xy-plane at three different resolutions, right: corresponding final triangulations in xyz-space.

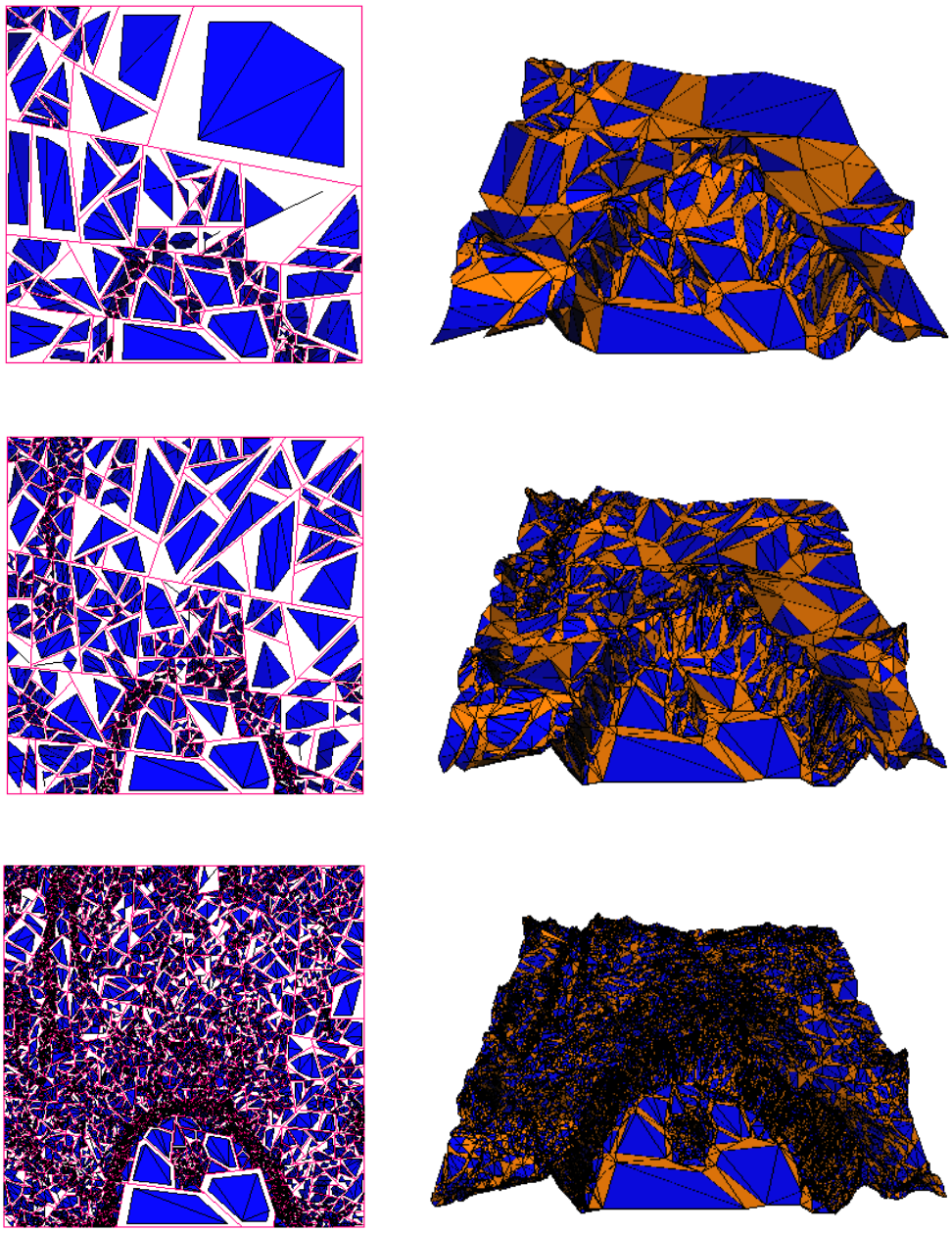

Figure 2.23: Triangulations for data set "Crater Lake." Left: optimal cluster triangulations in xy-plane at three different resolutions, right: corresponding final triangulations in xyz-space.

2.5 Conclusions Concerning Clustering

We introduced a general concept for adaptive clustering and provided numerical results for two different algorithms, hierarchical Voronoi diagrams based on Sibson's interpolant and piecewise optimal triangulations derived from adaptive quadratic approximation. The computational costs for both clustering approaches are $O(n\ log\ m)$ for n data points and m clusters. To make these algorithms applicable to large-scale data sets, some modifications are necessary that reduce the computational cost to expected linear time:

- Considering the refinement step $L_j \rightarrow L_{j+1}$, more than one cluster needs to be refined so that the number of clusters grows exponentially with the level index j. This can be achieved by refining all tiles τ_k^j corresponding to residuals ε_k^j greater than a specified threshold.
- The number of points that determine an approximating function $\tilde{f}_k^j$, see Section 2.1, must be bounded by a constant n_{max}. If a tile contains more than n_{max} points, we select n_{max} of them randomly.
- Also, the residuals ε_k^j can be estimated based on a (different) set of at most n_{max} points. We note that the final approximation error might not exactly satisfy the prescribed error bound.
- For any data point $\mathbf{p}_i$ and any level L_j, the tile containing $\mathbf{p}_i$ needs to be estimated in expected constant time. A list of all points inside a certain tile needs to be available in expected linear time. If the data is evenly distributed, this can be achieved by a regular decomposition of the domain D into squares or voxels (hypercubes, in general), for each of which a list of overlapping tiles is recorded.

In addition to adaptive approximation, hierarchical manipulation of data could be implemented by modifying $\tilde{f}_k^j$ at a coarse level and translating the resulting changes in some way to the next finer representations. However, the

geometry of a modified tile would have a great impact on such a modeling approach. It seems to be more natural to use a uniform or user-defined clustering for hierarchical modeling.

Our adaptive triangulation scheme can be generalized to higher-dimensional spaces. Least-squares fitting with quadratic polynomials and principal axis transformation are available in k-dimensions. The only problem is the construction of a regular simplicial grid. In three dimensions, for example, two types of tetrahedra need to be combined. A tetrahedron can be split into four smaller, similar tetrahedra and one octahedron that can be split into eight tetrahedra, see Figure 2.24.

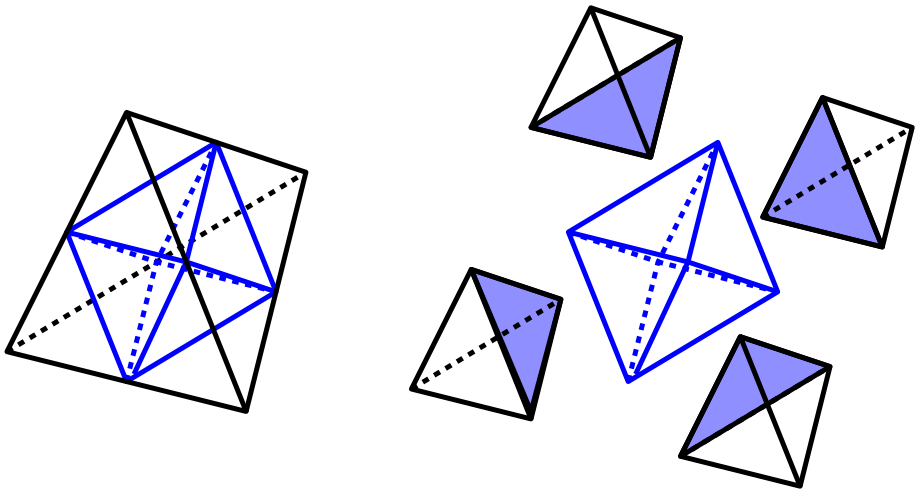

Figure 2.24: Decomposition of a tetrahedron into four smaller, similar tetrahedra and one octahedron.

A challenging research topic for future work is the generalization of these clustering techniques to domains of arbitrary topology, like two-manifolds embedded in three-dimensional space. We want to combine our clustering techniques with smooth basis functions defined by wavelet transforms and subdivision surface schemes to obtain multiresolution surface representations of high quality.

Chapter 3

Wavelet Representations

A wavelet transform [25, 89, 97] provides a sparse representation for highly detailed functions. The corresponding basis functions are dilated and translated versions of one function that, due to its nature, is called *wavelet*. Sparse representations are extremely useful in image compression [66] and for solving difficult mathematical problems, like integrating radiosity kernels [24, 53, 114] and partial differential equations (PDEs) [29, 128]. The discrete wavelet transform is computed in linear time, which makes it superior to most other multiresolution techniques.

Wavelets have their roots in signal processing and pure mathematics. In this chapter, we review the most fundamental results in the theory behind wavelets. Then, we show that the wavelet transform is also a modeling paradigm that naturally combines the concept of recursively generated basis functions with the concepts of fitting and fairing. We use the lifting scheme [125] to construct wavelets for efficiently modeling massive volumetric data sets. Finally, we describe algorithms for hierarchical representation and lossless compression of scientific data obtained from turbulent-mixing hydrodynamics simulations.

3.1 Motivation

Classical image compression algorithms consist of the following three steps [66]:

(i) Compute a linear, non-singular transform of an image.

(ii) Quantize the resulting coefficients (for lossy compression only).

(iii) Apply an entropy coding scheme to compress the coefficients.

The first step de-correlates the information present in an image. Exploiting spacial *correlation*, *i.e.*, similarity of local function values, *e.g.*, RGB color values, leads to a sparse representation in a Fourier basis or a wavelet basis, see Figure 3.1. For lossless compression, the linear transform is computed in integer arithmetic to obtain integer coefficients that can be encoded without quantization. In the case of lossy compression, the coefficients resulting from the transform are rounded to closest numbers on a certain grid to obtain values in a discrete set of symbols represented by integers. Coefficients of small absolute value can be replaced by zero for high lossy compression rates, see Figure 3.2. The integer coefficients are compressed by a coding scheme, for example by arithmetic coding [99], exploiting that small absolute values appear more frequently than greater ones.

To reconstruct an image, the compressed coefficients are decoded. If they have not been quantized in the compression step, the original coefficient values are restored and the inverse transform exactly reproduces the original image. In the case of quantization, the reconstructed image contains an error. In many cases, however, the human eye does not recognize this quantization error since it has been introduced in the range of a transform and is smoothly distributed across the image domain by the inverse transform. When using the wavelet transform, high compression rates in the order of 64:1 [66] are obtained by lossy compression with almost invisible artifacts.

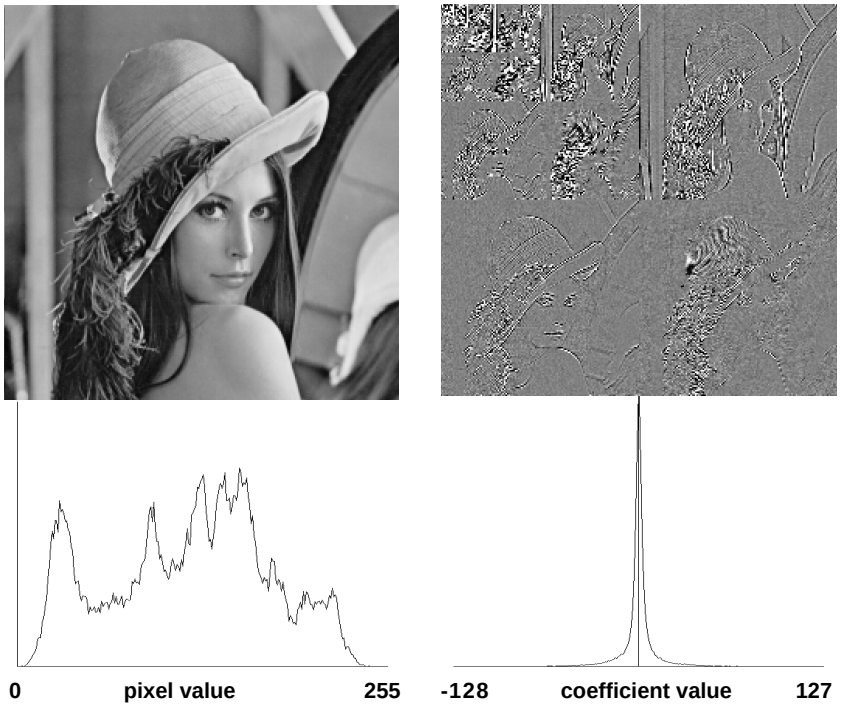

Figure 3.1: Wavelet transform (using a linear spline wavelet) and histogram of pixel and coefficient values.

Figure 3.2: Image reconstructed from ten percent of coefficient values (left) and reconstruction error scaled by a factor of ten (right).

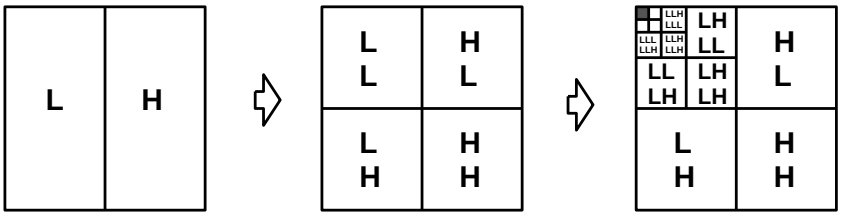

Figure 3.3: The wavelet transform separates details of different frequency bands by recursive band-pass and low-pass filtering.

Early compression standards, like JPEG [3], are based on the *discrete cosine transform* (DCT) using a basis of different-frequency cosine functions. Compared to the *Fourier transform*, the DCT produces no complex coefficients. The problem of the DCT is that all its basis functions have global support, and thus cannot represent local details of an image in a stable way. When the DCT is applied globally to an image, small quantization errors would corrupt local details in an unpredictable way. Therefore, JPEG subdivides an image into frames of 8×8 or 16×16 pixels before transforming it. This fragmentation, however, may become visible in reconstructed images at high compression rates. Newer standards, like JPEG2000, are based on wavelet-like techniques to overcome these problems.

When applying the wavelet transform, fragmentation of images is not necessary, since wavelets have local support in image domain and frequency space. The wavelet transform recursively separates high frequency details from the remaining lower frequencies of an image. This is done by band-pass and low-pass filtering in x- and y-directions separately. The coefficients that are obtained from low-pass filtering in both directions are recursively transformed by the same filtering operations, see Figure 3.3. This treatment, called *decomposition* or *analysis*, separates different frequency bands, referred to as *details* of an image. The inverse process of assembling details is called *reconstruction* or *synthesis*.

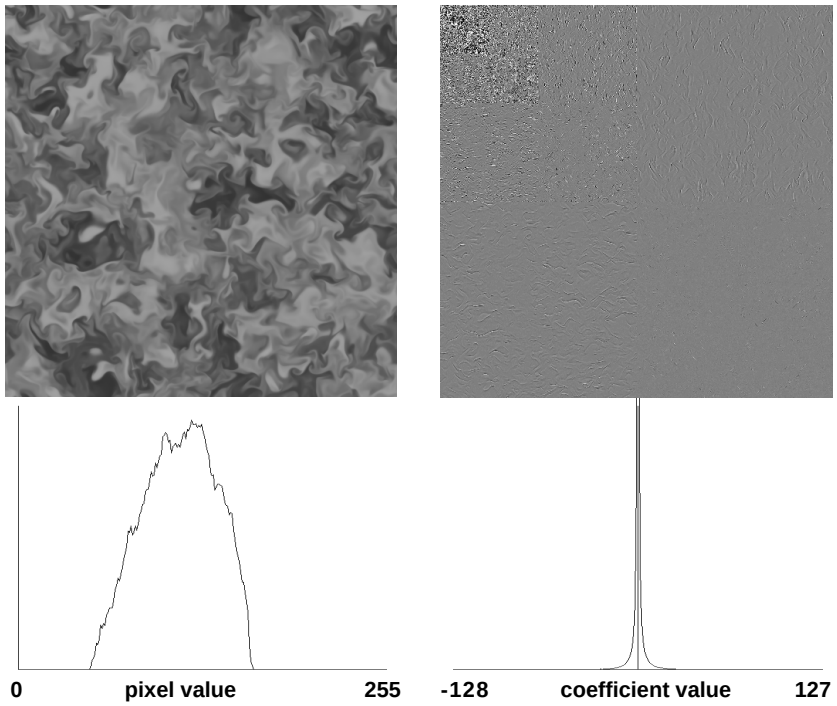

Figure 3.4: Wavelet transform and coefficient histogram for a slice of a Rayleigh-Taylor instability data set, courtesy of Lawrence Livermore National Laboratory.

Modeling scientific volume data for compression, progressive transmission, and multiresolution visualization can be accomplished with the same techniques used for image compression [126]. Generalization to multiple dimensions is straight forward, since the filtering operations are applied independently in every canonical direction. This treatment implies that the corresponding basis functions are tensor products and that the data representations are limited to topologies equivalent to (possibly periodic) hypercubes. Constructions for more general topologies are provided in Chapter 4.

The histogram of a slice taken from a *Rayleigh-Taylor instability* data set, shown in Figure 3.4, suggests that the expected absolute value of wavelet co-

efficients for this data set is even smaller than in the case of images. The entire data set describes a volumetric, time-varying temperature field sampled at a resolution of 512^3 bytes for each of 301 time steps. Applying the filtering operations to all four dimensions exploits much more correlation than in the two-dimensional case.

For compression and visualization of scientific data sets the representation needs to satisfy tight prescribed error bounds. Hence, it is not acceptable to introduce an unpredictable quantization error. Instead, the samples are rounded to a prescribed precision and represented as integer numbers. These are then transformed and compressed without loss. To reconstruct a smooth floating-point approximation, one can use iterative *fairing* techniques fitting B-splines to the tolerance intervals around the integer samples minimizing some kind of fairness function. This is a non-linear optimization process, since the approximation criterion is not defined in terms of *least-squares* errors.

3.2 Wavelets and Signal Processing

We briefly review basics of the wavelet transform and provide a summary of the most significant theoretical results. We start with the continuous wavelet transform that is closely related to Fourier analysis. Then we summarize the most important definitions and results that discretize the wavelet transform and lead to the linear-time algorithm known as the *fast wavelet transform*.

3.2.1 Fourier Transform

Fourier analysis is used to transform functions given in *signal space* into *frequency space* to analyze and manipulate them, *i.e.*, to apply high-pass, low-pass and band-pass filtering. Given a function $f \in L^2(\mathbb{R})$, as defined in

the Appendix. The Fourier transform of f and its inverse are defined as

$$\begin{aligned} \mathbf{F}f(\omega) &= \widehat{f}(\omega) = \tfrac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(x)\, e^{-i\omega x}\, dx \quad \text{and} \\ f(x) &= \tfrac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \widehat{f}(\omega)\, e^{+i\omega x}\, d\omega x, \quad \text{where} \\ i &= \sqrt{-1}. \end{aligned} \tag{3.1}$$

The Fourier transform is an isometry, due to *Parseval's identity*

$$\|f\|_{L^2} = \|\widehat{f}\|_{L^2}. \tag{3.2}$$

One of the most important properties of the Fourier transform is given by the *convolution theorem*,

$$\mathbf{F}(f * g) = \sqrt{2\pi}\ \widehat{f}\ \widehat{g}, \tag{3.3}$$

where the convolution operator "$*$" is defined as

$$f * g(x) = \int_{\mathbb{R}} f(y)\ g(x-y)\ dy. \tag{3.4}$$

For a signal f and a band-pass filter g, the result of filtering f with g corresponds to multiplying $\widehat{f}$ and $\widehat{g}$ in frequency space, see Figure 3.5.

We note that the Fourier transform can be applied to a wider class of functions and *distributions* that are not in $L^2(\mathbb{R})$. A prominent example is the *Dirac* distribution, defined as

$$\begin{aligned} \delta(x) &= \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \\ \int_{\mathbb{R}} \delta(x)\, dx &= 1. \end{aligned} \tag{3.5}$$

It satisfies the equation

$$\widehat{\delta}(\omega) = \tfrac{1}{\sqrt{2\pi}} \tag{3.6}$$

and thus represents the identity of the convolution operator.

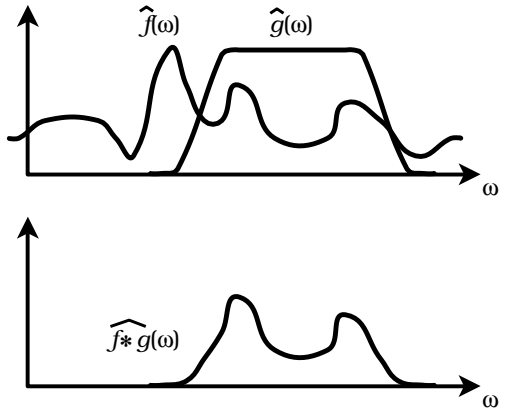

Figure 3.5: Application of the convolution theorem.

3.2.2 Continuous Wavelet Transform

The Fourier transform provides a tool to analyze frequencies that contribute to a given function. However, it fails to localize contributions of these frequencies in signal space, since its basis functions $e^{i\omega x}$ have infinite support. To localize the contribution of a signal at a certain point (ω, x), one needs to construct a filter with local support in both signal and frequency space. A band-pass filter ψ has local support in signal space when its *center* x^* and *radius* Δx exist [25]:

$$\begin{aligned} x^* &= \frac{1}{\|\psi\|_{L^2}^2} \int_{\mathbb{R}} x \, |\psi(x)|^2 \, dx, \quad \text{and} \\ \Delta x &= \frac{1}{\|\psi\|_{L^2}} \sqrt{\int_{\mathbb{R}} (x - x^*)^2 \, |\psi(x)|^2 \, dx}. \end{aligned} \tag{3.7}$$

Local support in frequency space can be described by center ω^* and radius $\Delta\omega$ of the Fourier transform $\widehat{\psi}$. However, this is not a good choice, since ω^* is always zero for real-valued filters. In this case, the Fourier transform is redundant, since

$$\widehat{\psi}(-\omega) = \overline{\widehat{\psi}(\omega)} \quad \text{for} \quad \psi : \mathbb{R} \to \mathbb{R}. \tag{3.8}$$

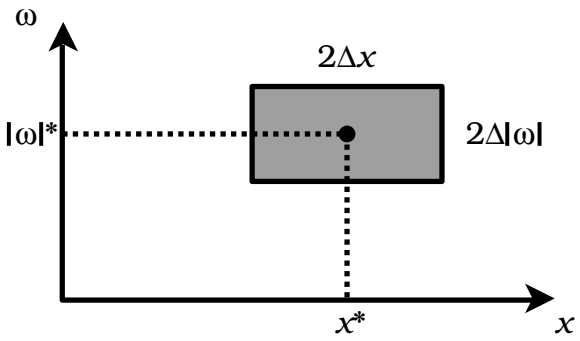

Figure 3.6: Time-frequency localization with an uncertainty proportional to the product of both radii, corresponding to the size of the shaded region.

Hence, we use the quantities $|\omega|^*$ and $\Delta|\omega|$, obtained by integrating over positive frequencies, instead.

To analyze a given function at the point (ω, x), one needs to compute its convolution with a filter ψ that has centers $x^* = x$, $|\omega|^* = \omega$ and radii Δx, $\Delta|\omega|$ as small as possible. Unfortunately, these radii are not independent, due to *Heisenberg's uncertainty principle* [101]. The product of both radii is a constant, depending on the shape of the filter, see Figure 3.6.

For a given filter ψ with known localization properties, we introduce two parameters a and b that modify ψ as follows:

$$\psi_{a,b}(x) \;=\; \tfrac{1}{\sqrt{|a|}}\, \psi\left(\tfrac{x-b}{a}\right), \quad a \in \mathbb{R} \neq 0,\ b \in \mathbb{R}. \tag{3.9}$$

The translation parameter b manipulates the center in signal space and the dilation parameter a accomplishes this in frequency space, since

$$\widehat{\psi}_{a,b}(\omega) \;=\; \sqrt{|a|}\; e^{-i\omega b}\; \widehat{\psi}(a\omega). \tag{3.10}$$

Convolution with this set of translated and dilated filters defines a new transform that provides the desired localization property (in the limits of the uncertainty principle). For the existence of the inverse transform, ψ must satisfy the *wavelet condition*:

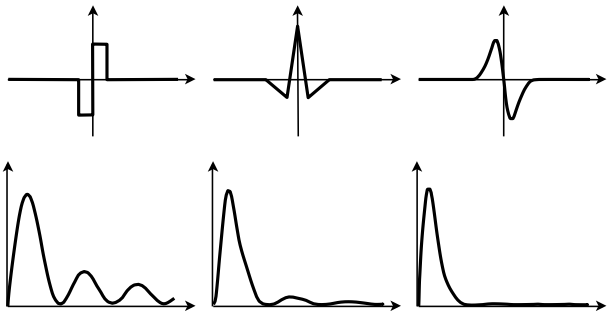

Figure 3.7: Three different wavelets (top) and amplitudes of their Fourier transforms (bottom).

Definition 3.2.1 *A function $\psi \in L^2(\mathbb{R})$ is called wavelet, if there exists a constant $c_\psi \in \mathbb{R} > 0$ with*

$$c_\psi \;=\; 2\pi \int_{\mathbb{R}} \frac{|\widehat{\psi}(\omega)|^2}{|\omega|}\; d\omega.$$

The continuous wavelet transform and its inverse are defined as

$$\begin{aligned} \mathbf{W}_\psi f(a,b) \;&=\; \tfrac{1}{\sqrt{c_\psi}} \langle f, \psi_{a,b} \rangle_{L^2} \;=\; \tfrac{1}{\sqrt{c_\psi}} \int_{\mathbb{R}} f(x)\; \overline{\psi_{a,b}(x)}\; dx, \\ f(x) \;&=\; \tfrac{1}{\sqrt{c_\psi}} \int_{\mathbb{R}^2} \mathbf{W}_\psi f(a,b)\; \psi_{a,b}(x) \frac{da\; db}{a^2}. \end{aligned} \tag{3.11}$$

We note that the inner product in the wavelet transform is equivalent to the convolution operator. In analogy to the Fourier transform, the continuous wavelet transform is an isometry, due to the identity

$$\|f\|_{L^2} \;=\; \|\mathbf{W}_\psi f\|_{L^2\left(\mathbb{R}^2, \frac{da\; db}{a^2}\right)}. \tag{3.12}$$

The wavelet condition implies that $\widehat{\psi}$ decreases quickly approaching $\omega = 0$ and that the average (direct current) of ψ is zero. Examples for wavelets and their Fourier transforms are shown in Figure 3.7.

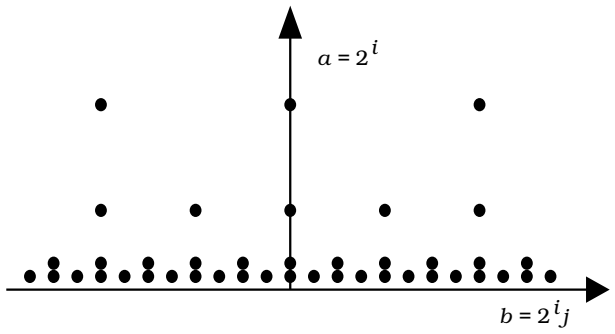

Figure 3.8: Dyadic grid samples that are sufficient for reconstruction.

3.2.3 Wavelet Frames

The continuous wavelet transform is highly redundant. A discrete sampling on a dyadic grid $\{(a,b) = (2^i, 2^i j) \mid i,j \in \mathbb{Z}\}$, illustrated in Figure 3.8, is often sufficient to reconstruct any function in $L^2(\mathbb{R})$ (except for differences of zero $L^2(\mathbb{R})$-norm, see Appendix). A discrete sampling of the Fourier transform, however, can exactly represent only periodical functions.

A sufficient condition for the existence of an inversion formula from discrete samples of the wavelet transform is that the corresponding discrete wavelet basis is a *frame*, as defined in the following.

Definition 3.2.2 *A system of functions $\{\psi_i \mid i \in \mathbb{Z}\}$ is called frame for $L^2(\mathbb{R})$, if constants $0 < A, B < \infty$ exist so that for every $f \in L^2(\mathbb{R})$*

$$A\|f\|_{L^2}^2 \leq \sum_{i\in\mathbb{Z}} \|\langle f, \psi_i\rangle_{L^2}\|^2 \leq B\|f\|_{L^2}^2. \tag{3.13}$$

A frame with constants $A = B$ is called tight.

In the case of a tight frame, the functions ψ_i are mutually orthogonal. It can be shown that a function ψ satisfies the wavelet condition, if the system of functions $\{\psi_{2^i,2^i j} \mid i,j \in \mathbb{Z}\}$ is a frame [89].

The frame condition can be exploited to reconstruct a function f from its wavelet coefficients $c_i := \langle f, \psi_i \rangle_{L^2}$. Therefore, the frame operator S is defined as

$$Sf \;=\; \frac{2}{A+B} \sum_{i \in \mathbb{Z}} \langle f, \psi_i \rangle_{L^2} \; \psi_i. \tag{3.14}$$

The inverse of the frame operator is given by Neumann's series, so that

$$f \;=\; S^{-1}Sf \;=\; \sum_{k=0}^{\infty} (I - S)^k \; Sf, \tag{3.15}$$

where I is the identity. The convergency rate for this series depends on the ratio of the constants A and B. Only in the case of a tight frame, the operator S is the identity and no iteration is necessary to reconstruct f, *i.e.*, $f = Sf$. A more useful and efficient reconstruction algorithm, is described in the next section.

3.2.4 Discrete Wavelet Transform

An algorithm for the *discrete wavelet transform* (DWT), also called *fast wavelet transform* due to its linear computation time, was discovered by Mallat [93]. The DWT is a basis transform implemented by discrete filtering between two individual levels of resolution. This filtering is recursively applied, starting with the finest resolution. High frequency detail is separated by every filtering step, resulting in a coarser, lower-frequency representation. The original algorithm by Mallat is based on orthogonal basis functions without compact support, but with exponentially decreasing amplitude. Since the discrete filters must be finite, the algorithm does not provide perfect reconstruction when using these basis functions. It is possible, however, to obtain perfect reconstruction when using compactly supported basis functions that need not to be orthogonal.

The main concept of the DWT is called *multi-scale analysis*, defined as a sequence of nested spaces V_j, spanned by *scaling functions*. Every space

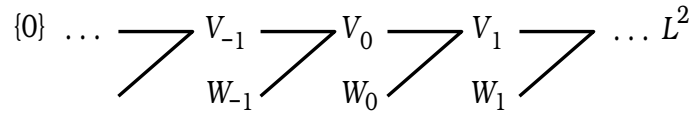

Figure 3.9: Nested spaces V_j and their duals W_j of a multi-scale analysis.

V_j provides the basis functions for one particular level of resolution. The differences (details) between two levels of resolution are represented in spaces W_j spanned by wavelets, see Figure 3.9.

Definition 3.2.3 *A multi-scale analysis is defined by a set of nested spaces V_j and a scaling function ϕ with the following properties:*

1. $V_j \subset V_{j+1} \quad \forall j \in \mathbf{Z}$,
2. $\overline{\bigcup_{j\in\mathbf{Z}} V_j} = L^2(\mathbb{R})$, (the bar denoting closure)
3. $\bigcap_{j\in\mathbf{Z}} V_j = \{0\}$.
4. $f \in V_j \Leftrightarrow f(2\,\cdot) \in V_{j+1} \quad \forall j \in \mathbf{Z}$, and
5. $V_0 = \overline{span\{\phi_k^0 := \phi(\cdot - k) \mid k \in \mathbb{Z}\}}$ *is a Riesz basis, i.e., there exist constants $0 < A, B < \infty$ such that*

$$A \sum_{k\in\mathbb{Z}} c_k^2 \;\le\; \| \sum_{k\in\mathbb{Z}} c_k \; \phi_k^0 \|_{L^2} \le \; B \sum_{k\in\mathbb{Z}} c_k^2 \quad \forall c \in l^2(\mathbb{Z}). \tag{3.16}$$

Starting with a certain multi-scale analysis, we construct wavelet spaces W_j that are complements of V_j in V_{j+1}, *i.e.*,

$$V_j \oplus W_j = V_{j+1}.$$

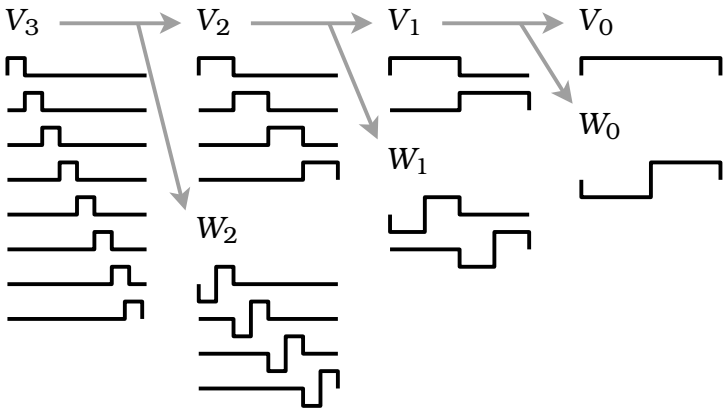

Figure 3.10: Basis functions for multi-scale analysis based on the Haar wavelet.

All spaces V_j and W_j are dilated versions of V_0 and W_0, spanned by translates of a scaling function ϕ and a wavelet ψ, respectively:

$$V_j = \overline{span\{\phi_k^j := \phi\,(2^j \cdot -k) \mid k \in \mathbb{Z}\}}, \quad \text{and}$$
$$W_j = \overline{span\{\psi_k^j := \psi\,(2^j \cdot -k) \mid k \in \mathbb{Z}\}}.$$

A good choice for W_j is the orthogonal complement of V_j in V_{j+1}. In this case, all spaces W_j are mutually orthogonal. If the function ψ is constructed so that all ψ_k^0 are mutually orthogonal, for example by using Gram-Schmidt orthonormalization, then the set of all wavelets ψ_k^j forms an orthogonal basis for $L^2(\mathbb{R})$. Orthogonal bases generally lead to small coefficients for representing functions. An example for an orthogonal wavelet basis is defined by the *Haar wavelet*, shown in Figure 3.10.

Orthogonality, symmetry, smoothness, and compact support are conflicting goals, however. The discontinuous Haar wavelet is actually the only real-valued wavelet that is orthogonal, symmetric, and has compact support [31, 72]. For geometric modeling purposes, however, symmetry, compact support, and certain degrees of smoothness are essential. These considerations lead to wavelet constructions where the spaces V_j and W_j are not orthogonal.

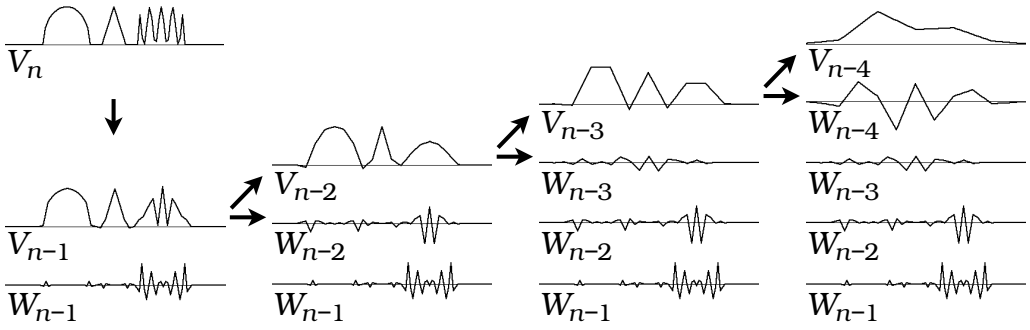

Figure 3.11: Basis transforms for a piecewise linear function.

In this case, there exist sets of *dual spaces* $\tilde{V}_j$ and $\tilde{W}_j$ defined as

$$\tilde{V}_j \perp W_j, \quad \tilde{W}_j \perp V_j, \quad \text{and}$$

$$\tilde{V}_j \oplus W_j = \tilde{W}_j \oplus V_j = V_{j+1}.$$

The dual spaces are spanned by dilates and translates of a dual scaling function $\tilde{\phi}$ and a dual wavelet $\tilde{\psi}$ defining also a multi-scale analysis. Wavelet constructions of this type are thus called *biorthogonal* [25].

The DWT is simply a basis transform from V_{j+1} into V_j and W_j that is recursively applied. Since this basis transform is regular, *i.e.*, it has an inverse, neither ψ, nor $\tilde{\psi}$ need to satisfy the wavelet condition defined in the context of the continuous wavelet transform. An example for the basis transforms between different spaces is illustrated in Figure 3.11.

The basis transform uniquely defines sequences h, l, $\tilde{h}$, and $\tilde{l}$ with the properties

$$\phi_k^{j+1} \;=\; \sum_{i \in \mathbb{Z}} (l_{k-2i}\; \phi_i^j \;+\; h_{k-2i}\; \psi_i^j), \tag{3.17}$$

$$\phi_k^j \;=\; \sum_{i \in \mathbb{Z}} \tilde{l}_{i-2k}\; \phi_i^{j+1}, \quad \text{and} \tag{3.18}$$

$$\psi_k^j \;=\; \sum_{i\in\mathbb{Z}} \tilde{h}_{i-2k}\; \phi_i^{j+1}. \tag{3.19}$$

If the functions ϕ and ψ have compact support, then all four sequences are finite. These sequences represent discrete high-pass and low-pass filters that are used in the fast algorithm for the DWT.

Projections of an analytical function f into spaces V_j and W_j are represented by sets of coefficients c^j and d^j, respectively,

$$f_{V_j}(x) \;=\; \sum_{k\in\mathbb{Z}} c_k^j\; \phi_k^j(x) \quad \text{and} \tag{3.20}$$

$$f_{W_j}(x) \;=\; \sum_{k\in\mathbb{Z}} d_k^j\; \psi_k^j(x). \tag{3.21}$$

Transforming the coefficients c^{j+1} into a coarser representation defined by coefficients c^j plus a representation for the differences (details) by coefficients d^j is called *decomposition* or *analysis*. The inverse operation, *i.e.*, obtaining coefficients c^{j+1} from c^j and d^j, is called *reconstruction* or *synthesis*.

Starting with a coefficient representation c^n for a function f at fine resolution, its DWT is computed by successive decomposition steps, resulting in sets of coefficients $d^{n-1},\ d^{n-2}, \ldots, d^0$, and c^0, see Figure 3.12. The wavelet coefficients d^j are sparse (using integer arithmetic) or small in absolute value (using floating-point arithmetic), since they capture only the differences between two levels of resolution. The inverse DWT reconstructs the coefficient sets $c^1,\ c^2, \ldots, c^n$ again.

The decomposition and reconstruction formulae are based on the four discrete filters defined above. The filter h is a band pass separating the highest frequency band (the details) from a function, l is a low-pass filter generating a coarser approximation without the highest frequency band, and $\tilde{h}$ and $\tilde{l}$ are necessary to invert the transform. These are the standard decomposition and reconstruction formulae:

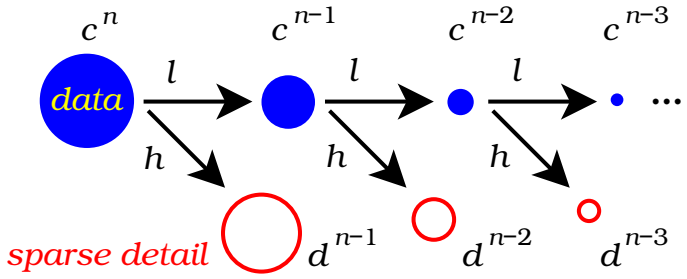

Figure 3.12: Decomposition process.

Decomposition:

$$c_i^j = \sum_{k\in\mathbb{Z}} l_{k-2i}\, c_k^{j+1}, \tag{3.22}$$

$$d_i^j = \sum_{k\in\mathbb{Z}} h_{k-2i}\, c_k^{j+1}. \tag{3.23}$$

Reconstruction:

$$c_k^{j+1} = \sum_{i\in\mathbb{Z}} (\tilde{l}_{k-2i}\, c_i^j + \tilde{h}_{k-2i}\, d_i^j). \tag{3.24}$$

The algorithm known as fast wavelet transform applies this decomposition formula recursively starting with N coefficients c^n that are considered as samples of a function f. Every decomposition step is computed in linear time with respect to the number of coefficients, since the filters l and h have finite length. The number of transformed coefficients decreases by one half for every level of resolution, since the wavelet coefficients d^j are not transformed again. Thus, the overall complexity is $O\left(N + \frac{1}{2}N + \frac{1}{4}N + \frac{1}{8}N + \cdots\right) = O(2N) = O(N)$.

In the two-dimensional case, decomposition is first applied to all rows and then to all columns of a coefficient matrix. The reduction is even more drastic, since only the coefficients that correspond to scaling functions with

respect to both directions, *i.e.*, one quarter of all coefficients, are transformed again. The complexity for the two-dimensional algorithm is linear, since $O\left(N + \frac{1}{4}N + \frac{1}{16}N + \cdots\right) = O\left(\frac{4}{3}N\right) = O(N)$. Generalization to higher dimensional spaces is straight forward and the resulting basis functions are tensor products of the one-dimensional basis functions.

3.3 Wavelets for Geometric Modeling

In the context of geometric modeling, the discrete filters h, l, $\tilde{h}$, and $\tilde{l}$ obtain an entirely different meaning that is not related any longer to signal processing. Wavelets for geometric modeling, sometimes referred to as *second-generation wavelets*, are used for multiresolution representation of functions with sharp features, boundaries, non-uniform parametrizations, and irregular (non-Euclidean) domains. In the remainder of this chapter, we introduce some terminology and wavelet constructions for modeling large-scale data sets. Some of these constructions are generalized in Chapter 4 to represent arbitrary two-manifolds, *i.e.*, surfaces of arbitrary topological genus.

3.3.1 A Modeling Paradigm

Starting with a high-resolution curve or surface represented by a control polygon or control mesh, we want to compute a number of coarser approximations using smaller and smaller sets of basis functions. At the same time, we want to store the differences between any two adjacent levels of resolution compactly in form of wavelet coefficients from which the original resolution can be reconstructed, see Figure 3.13.

The transition to a coarser representation is computed by a linear *fitting* operator $\mathbf{F}$ predicting the coordinates of the reduced set of control points. In contrast to the example shown in Figure 3.13, the coordinates of these

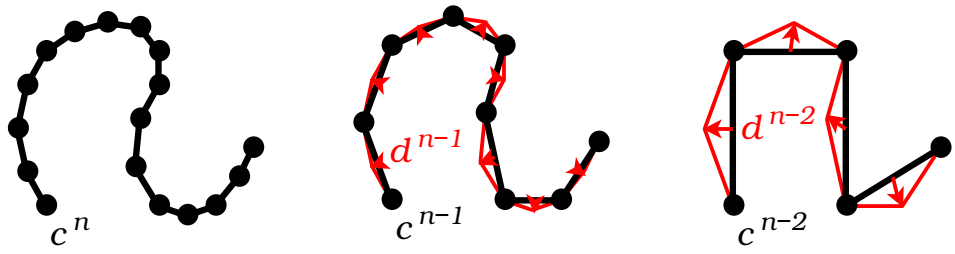

Figure 3.13: Decomposition of a control polygon. The wavelet coefficients d^j are accumulated difference vectors representing the displacements of local control points from a coarser control polygon.

points may be changed, which is necessary to minimize a certain error norm between the limit curves of both control polygons. The wavelet coefficients located at the eliminated control points thus represent the displacements of all control points (and not only the displacements of the removed ones). This is accomplished by a linear *compaction-of-detail* operator $\mathbf{C}$. These two operators define the decomposition formula:

$$\begin{aligned} c^j &= \mathbf{F}\, c^{j+1} \quad \text{and} \\ d^j &= \mathbf{C}\, c^{j+1}. \end{aligned} \tag{3.25}$$

The inverse of these operations is computed by a *subdivision* operator $\mathbf{S}$ and an *expansion-of-detail* operator $\mathbf{E}$. The subdivision operator $\mathbf{S}$ provides a smooth limit curve when recursively applied to a control polygon defining any level of resolution. The operator $\mathbf{E}$ is used to reconstruct detail from the wavelet coefficients that is added to the curve during the subdivision process. The reconstruction formula is defined as

$$c^{j+1} = \mathbf{S}\, c^j + \mathbf{E}\, d^j. \tag{3.26}$$

The decomposition and reconstruction formulae are illustrated in Figure 3.14.

These four operators can be interpreted as matrix multiplications. In the special case of a multi-scale analysis defined by filters h, l, $\tilde{h}$, and $\tilde{l}$, these

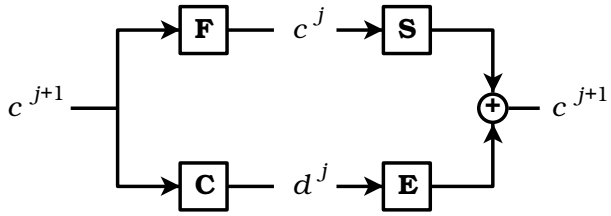

Figure 3.14: Wiring diagram for decomposition (left) and reconstruction (right). We note that the coefficients c^j are recursively transformed, which is not shown in the diagram.

matrices are infinite periodic band matrices, satisfying

$$\begin{aligned} \mathbf{F}_{ij} &= l_{j-2i}, \\ \mathbf{C}_{ij} &= h_{j-2i}, \\ \mathbf{S}_{ij} &= \tilde{l}_{i-2j}, \quad \text{and} \\ \mathbf{E}_{ij} &= \tilde{h}_{i-2j}. \end{aligned} \tag{3.27}$$

We recall that the operators $\mathbf{F}$ and $\mathbf{C}$ compute a regular basis transform from a space V_{j+1} into complementary spaces V_j and W_j, and that the inverse basis transform is computed by $\mathbf{S}$ and $\mathbf{E}$. This imposes the following compatibility constraints on the construction of these operators:

$$\begin{aligned} \mathbf{FS} &= \mathbf{CE} = I \quad \text{and} \\ \mathbf{FE} &= \mathbf{CS} = 0, \end{aligned} \tag{3.28}$$

where I is the identity matrix. We note that in the case of second-generation wavelets the spaces V_j and W_j are not spanned by dilated and translated versions of the same function. However, the hierarchical structure of nested spaces remains still the same and in the regions with uniform parametrizations all other results remain valid.

The subdivision operator $\mathbf{S}$ is the most important one for modeling purposes, since it determines the shape of the basis functions used for representing functions at multiple levels of resolution. Once $\mathbf{S}$ is constructed, we

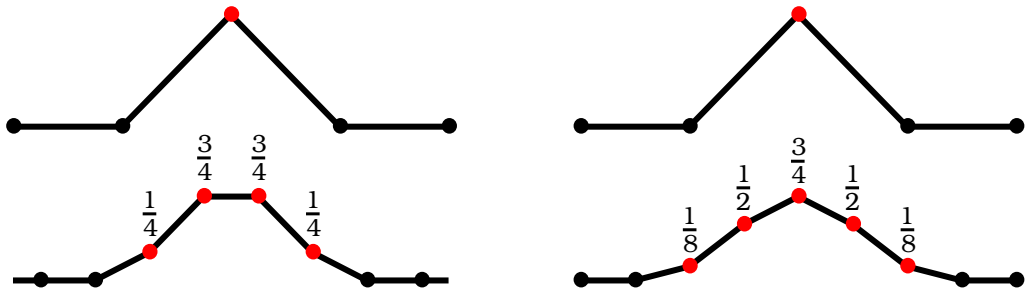

Figure 3.15: Dyadic refinement for quadratic and cubic B-splines.

have various choices for the remaining three operators. The resulting basis functions spanning the spaces V_j and W_j are recursively generated by

$$\begin{aligned} \phi_i^j &= \mathbf{S}^\infty \, \delta_i^j \quad \text{and} \\ \psi_i^j &= \mathbf{S}^\infty \, \mathbf{E} \, \delta_i^j, \end{aligned} \tag{3.29}$$

where δ_i^j is a control polygon at resolution j with the ordinate of the control point with index i set to one and the ordinates of all other control points set to zero.

3.3.2 Recursive Subdivision

A natural choice for scaling functions are uniform B-splines [35], since they provide a continuous, smooth data representation, have compact support, are efficiently computed at arbitrary parameter values, and are widely used in *computer-aided geometric design* (CAGD). For uniform B-splines of any polynomial degree, the sequence l defining the dyadic refinement rules corresponds to a row in Pascal's triangle, normalized such that the sum of all entries is two [41]. These entries are shown for different polynomial degrees in Table 3.1. The subdivision process for quadratic and cubic B-splines is illustrated in Figure 3.15

Smooth interpolating subdivision schemes can also be used for wavelet construction, For example, the subdivision weights for cubic C^1-continuous

degree	$\tilde{l}_{0,1}$	$\tilde{l}_{-1,2}$	$\tilde{l}_{-2,3}$
0	1		
2	$\frac{3}{4}$	$\frac{1}{4}$	
4	$\frac{5}{8}$	$\frac{5}{16}$	$\frac{1}{16}$

degree	$\tilde{l}_0$	$\tilde{l}_{\pm1}$	$\tilde{l}_{\pm2}$	$\tilde{l}_{\pm3}$
1	1	$\frac{1}{2}$		
3	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	
5	$\frac{5}{8}$	$\frac{15}{32}$	$\frac{3}{16}$	$\frac{1}{32}$

Table 3.1: Sequence $\tilde{l}$ defining subdivision operator $\mathbf{S}$ for B-spline scaling functions of different polynomial degrees.

splines are $\tilde{l}_0 = 1$, $\tilde{l}_{\pm1} = \frac{9}{16}$, $\tilde{l}_{\pm2} = 0$, and $\tilde{l}_{\pm3} = -\frac{1}{16}$. Since some of the weights are negative, these splines do not lie within the convex hull of their control polygons. In some cases they show undesirable "wiggles" due to the interpolation constraints. Thus, we use B-spline subdivision as basis for our wavelet constructions. In the case of higher dimensions, considering, for example, surfaces and volumes, we apply the one-dimensional subdivision operator in all canonical directions, resulting in tensor-product B-spline scaling functions.

3.3.3 Least-Squares Fitting

We briefly summarize the results of related work regarding the construction of fitting operators $\mathbf{F}$ based on B-spline subdivision in the following. These approaches minimize certain error norms in every coarsening step.

A prominent example are *semi-orthogonal* wavelets [50, 123, 109]. Starting with spaces V_j and V_{j+1} spanned by uniform B-splines, the space W_j is chosen to be the orthogonal complement of V_j in V_{j+1}. The wavelets spanning W_j are not mutually orthogonal which is the only difference to fully orthogonal wavelet bases. Symmetric wavelets with compact support can be constructed. A regular basis transform from the space V_{j+1} into the orthogonal sum of V_j and W_j is always an orthogonal projection into these spaces.

Since the fitting operator $\mathbf{F}$ is an orthogonal projection, it minimizes the L^2-norm of the error function, *i.e.*,

$$\|f_{V_{j+1}} - f_{V_j}\|_{L^2} \rightarrow \min \tag{3.30}$$

for every function $f \in L^2(\mathbb{R})$. Due to the global nature of this optimization problem, the matrix $\mathbf{F}$ is dense. However, the decomposition can still be computed in linear time using a band-matrix solver [109].

Despite of the fact that the operator $\mathbf{F}$ is uniquely determined by the bases of the spaces V_j and V_{j+1}, the basis for W_j can be chosen, resulting in different wavelet constructions. Semi-orthogonal B-spline wavelets with minimal support were constructed by Chui [25]. A more general framework for subdivision and fitting schemes minimizing certain fairness functions was described by Kobbelt/Schröder [77]. It is also feasible to construct semi-orthogonal wavelets based on a different inner product inducing a fairness norm. If W_j is chosen to be orthogonal to V_j with respect to a certain inner product, then the induced norm is minimized by the resulting fitting operator.

A different least-squares fitting approach based on discrete displacements of control points was described by Duchaineau [41]. Provided that the coefficients c^n at the finest level of resolution are samples from an approximated function, this approach may lead to smaller wavelet coefficients than minimizing least squares between limit functions obtained from recursive subdivision. The idea is to minimize the expanded wavelet coefficients $\mathbf{E}\, d^j$ directly for every function f that can be represented by coefficients c^{j+1}. This optimization problem is equivalent to

$$\|c^{j+1} - \mathbf{S}\, c^j\| \rightarrow min. \tag{3.31}$$

Differentiating with respect to c^j leads to

$$c^j = (\mathbf{S}^T\mathbf{S})^{-1}\, \mathbf{S}\, c^{j+1}. \tag{3.32}$$

The expression $\mathbf{S}^T\mathbf{S}$ corresponds to a discrete filter that can be inverted by discrete Fourier transform and application of the convolution theorem [41]. Hence, the matrix $\mathbf{F}$ and the corresponding sequence l are determined from equation (3.32):

$$\mathbf{F} \;=\; (\mathbf{S}^T\mathbf{S})^{-1}\,\mathbf{S}. \tag{3.33}$$

Unfortunately, due to the global nature of this optimization problem, the sequence l is infinite, but exponentially decreasing. It is possible to approximate l by a finite sequence satisfying the constraint $\mathbf{FS} = I$. The two remaining sequences h and $\tilde{h}$ are constructed such that

$$\begin{aligned} h_i &= (-1)^{i+1}\,\tilde{l}_{i-m}, \quad \text{and} \\ \tilde{h}_i &= (-1)^{i+1}\,l_{i-m}, \end{aligned} \tag{3.34}$$

where $m = 0$ for B-spline scaling functions with even polynomial degrees (point-symmetric wavelets) and $m = 1$ for odd degrees (symmetric wavelets). This construction satisfies the compatibility conditions of the four operators as defined in equation (3.28). An example for a quadratic B-spline wavelet constructed with this approach is provided in the next section.

Another desirable property for the construction of wavelets are vanishing moments. A wavelet ψ has N vanishing moments, if its inner products with N polynomials are zero:

$$\langle \psi, p_k \rangle_{L^2} \;=\; 0, \quad p_k(x) \;=\; x^k, \quad k \;=\; 0, 1, \cdots, N-1. \tag{3.35}$$

Vanishing moments are desirable for the approximation of smooth functions and for solving certain inversion problems like integrating radiosity kernels, since they lead to sparse approximations with improved stability. Vanishing moment conditions are an alternative to global optimization properties, since they allow for wavelet constructions with finite-length filters for both, decomposition and reconstruction. Finite-length filters are required to compute the DWT efficiently at local regions of large-scale data sets.

3.3.4 The Lifting Scheme

The *lifting scheme* described by Sweldens [125] is an important tool for constructing wavelets. A similar technique was introduced independently by Dahmen [28]. The lifting scheme subdivides the computation of the wavelet transform into small local lifting operations updating the value of one coefficient at a time. Using the lifting scheme has several advantages over computing the standard decomposition and reconstruction formulae:

- Increased efficiency. In many cases, the lifting scheme uses fewer operations than the standard formulae, since it combines the computation of two operators, for example $\mathbf{F}$ and $\mathbf{C}$, into one process.
- Lossless compression. Certain lifting operations can be implemented in integer arithmetic with perfect reconstruction. This allows for the construction of integer-to-integer wavelet transforms for lossless compression.
- Simpler wavelet construction. The lifting scheme can be used to design wavelets with certain properties, like vanishing moments. Starting with a simple wavelet prototype, sometimes called *lazy wavelet*, lifting is used to improve the shape of this wavelet. Lifting can also be used to design scaling functions and dual wavelets.

The general lifting scheme for the decomposition formula is illustrated in Figure 3.16. In the simplest case, the operators $\mathbf{F}'$ and $\mathbf{C}'$ defining the lazy wavelet transform are just down-sampling operators. The inverse DWT is defined by the same lifting operations, applied with negated signs in reverse order. In the case of integer arithmetic, the result of every lifting operation is rounded to the closest integer. Since this is done in the same way for decomposition and reconstruction, there is no loss.

We now provide some examples for factoring wavelets into lifting operations. A rigorous mathematical treatment of this topic was done by

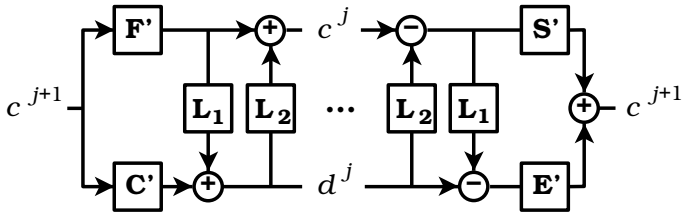

Figure 3.16: Lifting scheme for decomposition and reconstruction formula. The operators $\mathbf{F}'$, $\mathbf{C}'$, $\mathbf{S}'$, and $\mathbf{E}'$ define the lazy wavelet transform. An arbitrary number of lifting operations can be applied to shape scaling functions and wavelets.

wavelet	h_{-1}	h_0	h_1	h_2	l_{-3}	l_{-2}	l_{-1}	l_0	l_1	l_2	l_3	l_4
ψ_H		1	-1					$\frac{1}{2}$	$\frac{1}{2}$			
ψ_Q	$-\frac{1}{4}$	$\frac{3}{4}$	$-\frac{3}{4}$	$\frac{1}{4}$			$-\frac{1}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$-\frac{1}{4}$		
ψ_D	$-\frac{1}{4}$	$\frac{3}{4}$	$-\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{12}$	$-\frac{1}{4}$	0	$\frac{2}{3}$	$\frac{2}{3}$	0	$-\frac{1}{4}$	$\frac{1}{12}$

wavelet	$\tilde{h}_{-3}$	$\tilde{h}_{-2}$	$\tilde{h}_{-1}$	$\tilde{h}_0$	$\tilde{h}_1$	$\tilde{h}_2$	$\tilde{h}_3$	$\tilde{h}_4$	$\tilde{l}_{-1}$	$\tilde{l}_0$	$\tilde{l}_1$	$\tilde{l}_2$
ψ_H				$\frac{1}{2}$	$-\frac{1}{2}$					1	1	
ψ_Q			$\frac{1}{4}$	$\frac{3}{4}$	$-\frac{3}{4}$	$-\frac{1}{4}$			$\frac{1}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{4}$
ψ_D	$-\frac{1}{12}$	$-\frac{1}{4}$	0	$\frac{2}{3}$	$-\frac{2}{3}$	0	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{4}$

Table 3.2: Examples for discrete filters defining wavelet transforms.

Daubechies/Sweldens [32]. Consider the wavelets defined in Table 3.2. ψ_H is the piecewise constant Haar wavelet, ψ_Q is a quadratic B-spline wavelet with small support, and ψ_D is a wavelet constructed by Duchaineau [41].

The decomposition formula for the Haar wavelet ψ_H is being evaluated independently for pairs of adjacent coefficients,

$$\begin{pmatrix} c_i^j \\ d_i^j \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & -1 \end{pmatrix} \begin{pmatrix} c_{2i}^{j+1} \\ c_{2i+1}^{j+1} \end{pmatrix}. \tag{3.36}$$

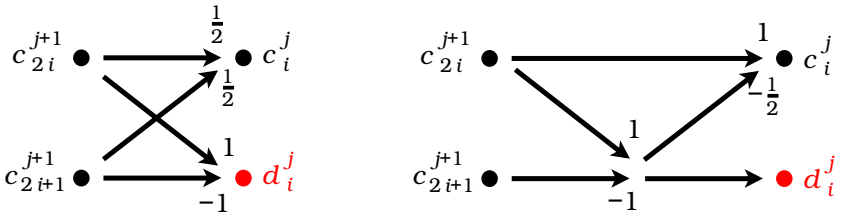

Figure 3.17: Decomposition for Haar wavelet (left) and equivalent lifting operations (right).

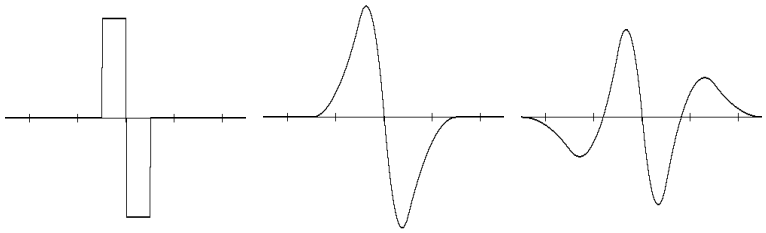

Figure 3.18: Constructed wavelets ψ_H, ψ_Q, and ψ_D.

The matrix in this equation can be split into two operations updating only one coefficient at a time, as illustrated in Figure 3.17:

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}. \tag{3.37}$$

In general, every regular 2×2-matrix M with non-zero entries on the diagonal can be split in this way, based on four constants a_1, b_1, a_2, and b_2 satisfying

$$M = \begin{pmatrix} b_2 & a_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ a_1 & b_1 \end{pmatrix} = \begin{pmatrix} b_2 + a_2 a_1 & a_2 b_1 \\ a_1 & b_1 \end{pmatrix}. \tag{3.38}$$

The lifted decomposition formula for the Haar wavelet can be written in algorithmic notation as

$$\begin{aligned} d'_i &\leftarrow c_i - d_i \\ c'_i &\leftarrow c_i - \tfrac{1}{2} d'_i, \end{aligned}$$

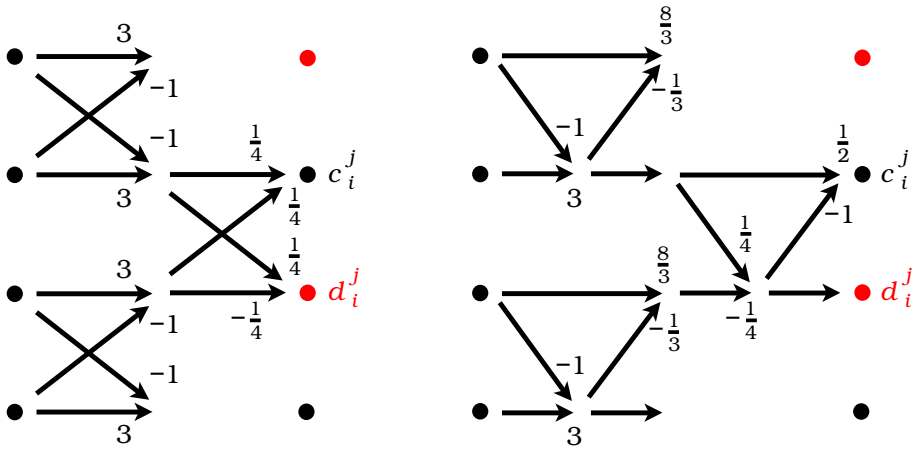

Figure 3.19: Factoring the decomposition for wavelet ψ_Q into two matrix multiplications (left), which can be split into four lifting operations (right).

where we first re-label the coefficients $c_i \leftarrow c_{2i}^{j+1}$ and $d_i \leftarrow c_{2i+1}^{j+1}$. Then, every lifting operation is computed for all indices i. The coefficients for the decomposition formula are the last updated ones, in this case $c_i^j \leftarrow c_i'$ and $d_i^j \leftarrow d_i'$. We use this notation also for the following lifting constructions.

Decomposition for the quadratic B-spline wavelet ψ_Q is defined according to the entries in Table 3.2 as

$$\begin{pmatrix} c_i^j \\ d_i^j \end{pmatrix} = \begin{pmatrix} -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{3}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} c_{i-1}^{j+1} \\ c_i^{j+1} \\ c_{i+1}^{j+1} \\ c_{i+2}^{j+1} \end{pmatrix}. \tag{3.39}$$

As illustrated in Figure 3.19, this formula can be factored into two 2×2-matrix multiplications with unique matrix entries (except for scaling). We have computed these matrix entries form the constraints in equation (3.39) by solving a system of non-linear equations. The corresponding update op-

erations are defined as

$$\begin{pmatrix} d'_{i-1} \\ c'_i \end{pmatrix} \leftarrow \begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} d_{i-1} \\ c_i \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} c''_i \\ d''_i \end{pmatrix} \leftarrow \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} c'_i \\ d'_i \end{pmatrix}. \tag{3.40}$$

Both matrices can be split into two lifting operations each, resulting in the lifting scheme

$$\begin{aligned} c'_i &\leftarrow -\, d_{i-1} + 3c_i \\ d'_{i-1} &\leftarrow \tfrac{8}{3} d_{i-1} - \tfrac{1}{3} c'_i \\ d''_i &\leftarrow \tfrac{1}{4} c'_i - \tfrac{1}{4} d'_i \\ c''_i &\leftarrow \tfrac{1}{2} c'_i - d''_i. \end{aligned}$$

Combining the second and third lifting operations results in the more compact scheme,

$$\begin{aligned} c'_i &\leftarrow -\, d_{i-1} + 3c_i \\ d''_i &\leftarrow \tfrac{1}{4} c'_i - \tfrac{2}{3} d_i + \tfrac{1}{12} c'_{i+1} \\ c''_i &\leftarrow \tfrac{1}{2} c'_i - d''_i. \end{aligned} \tag{3.41}$$

For lifting the wavelet ψ_D, we exploit that it is based on the same compaction-of-difference filter h as the wavelet ψ_Q. We observe that the fitting filters of both wavelets, denoted as l^D and l^Q, satisfy the relation

$$l^D_i = l^Q_i + \tfrac{1}{3}\left(h_{i-2} - h_{i+2}\right).$$

The lifting scheme for ψ_D is thus identical to the one for ψ_Q, with one additional lifting operation, given by

$$c'''_i \leftarrow -\tfrac{1}{3} d''_{i-1} + c''_i + \tfrac{1}{3} d''_{i+1}.$$

We note that the last two lifting operations can again be combined. Despite of its long filters, the wavelet transform for ψ_D requires only one more lifting operation as the transform for the compact Haar wavelet ψ_H.

It can be verified that computing the lifted wavelet transforms requires fewer operations than computing the original decompositon and reconstruction schemes. Another advantage of lifting is that the boundary treatment becomes much simpler. The local lifting operations can individually be adapted to the boundaries. Magnitudes of wavelet coefficients can be reduced at a boundary by reflecting the data at rather than assuming zero function values behind the boundary.

3.3.5 Integer Arithmetic

For certain applications, like lossless compression, it is essential to produce integer wavelet coefficients from which the finest resolution representation can be reconstructed without rounding errors. This is possible for certain wavelet constructions, when the scaling-function coefficients c^n at the finest resolution are integers. If the c^n are finite-precision numbers within a tolerance ε, then we can scale them by $\frac{1}{2\varepsilon}$ and round them to closest integers.

To provide an example for an integer-to-integer transform with perfect reconstruction, we adapt the lifting scheme for ψ_Q, defined in equation (3.41), to integer arithmetic. First, we scale every row such that the variable that is modified is scaled by ± 1 (or by an integer). We note that such scaling may also change the subsequent rows. Scaling the first row by $\frac{1}{3}$ results in the modified lifting scheme

$$\begin{aligned} c'_i &\leftarrow c_i - \tfrac{1}{3} d_{i-1} \\ d''_i &\leftarrow -\tfrac{2}{3} d_i + \tfrac{3}{4} c'_i + \tfrac{1}{4} c'_{i+1} \\ c''_i &\leftarrow \tfrac{3}{2} c'_i - d''_i. \end{aligned}$$

The modified scheme still produces the same coefficients d''_i and c''_i as equation (3.41). We now scale the second row by $\frac{3}{2}$ and the third one by $\frac{2}{3}$, which also

scales the resulting coefficients. The new scheme is given by

$$\begin{aligned} c'_i &\leftarrow c_i - \tfrac{1}{3}d_{i-1} \\ d''_i &\leftarrow -d_i + \tfrac{9}{8}c'_i + \tfrac{3}{8}c'_{i+1} \\ c''_i &\leftarrow c'_i - \tfrac{4}{9}d''_i. \end{aligned}$$

Finally, we must ensure that the computed coefficients are integers. We use the operator $[\,\cdot\,]$ that rounds real numbers to their closest integers. The integer decomposition for wavelet ψ_Q is defined by

$$\begin{aligned} c'_i &\leftarrow c_i - [\tfrac{1}{3}d_{i-1}] \\ d''_i &\leftarrow -d_i + [\tfrac{9}{8}c'_i + \tfrac{3}{8}c'_{i+1}] \\ c''_i &\leftarrow c'_i - [\tfrac{4}{9}d''_i]. \end{aligned} \tag{3.42}$$

The inverse of this decomposition formula is obtained by applying the inverse of every lifting operation in reverse order. The integer reconstruction formula is thus defined by

$$\begin{aligned} c'_i &\leftarrow c''_i + [\tfrac{4}{9}d''_i] \\ d_i &\leftarrow -d''_i + [\tfrac{9}{8}c'_i + \tfrac{3}{8}c'_{i+1}] \\ c_i &\leftarrow c'_i + [\tfrac{1}{3}d_{i-1}]. \end{aligned} \tag{3.43}$$

Every lifting operation has an inverse, provided that the modified variable is scaled by an integer with a fraction of other variables (rounded to an integer) added.

In our example, the scaling-function coefficients loose precision in every decomposition step, which decreases the absolute value of wavelet coefficients on coarser levels of resolution. For compression purposes this is a desired property, since perfect reconstruction is guaranteed. For high-quality representations at coarse levels of resolution, however, it is necessary to maintain a certain precision of the fitting operator, which is done by choosing greater scaling factors when adapting the lifting scheme to integer arithmetic.

3.4 Symmetric Lifted B-spline Wavelets

We now provide a systematic construction approach for symmetric lifted B-spline wavelets with odd polynomial degrees. These wavelets have two vanishing moments and have minimal support. The decomposition and reconstruction schemes are computed with a minimum number of operations, which makes this construction extremely useful for compression of large-scale data sets. It is possible to use integer arithmetic for lossless compression. These symmetric wavelet constructions can be generalized to surfaces of arbitrary topology, as shown in Chapter 4.

We use algorithmic notation to define our wavelet construction, since the coefficients of the wavelet transform are updated multiple times during the process. A decomposition step is computed by re-labeling coefficients according to

$$c_i \leftarrow c_{2i}^{j+1} \quad \text{and} \quad d_i \leftarrow c_{2i+1}^{j+1}, \tag{3.44}$$

followed by a sequence of alternating *s-lift* and *w-lift* operations updating scaling-function and wavelet coefficients, respectively. These operations modify one coefficient at a time, depending on its own and its two neighbors' values. The s-lift and w-lift operations are defined as

s-lift(a, b)**:**

$$c_i \leftarrow a d_{i-1} + b c_i + a d_i \quad \forall i \tag{3.45}$$

w-lift(a, b)**:**

$$d_i \leftarrow a c_i + b d_i + a c_{i+1} \quad \forall i \tag{3.46}$$

After a sequence of s-lift and w-lift operations, we obtain the transformed coefficients

$$c_i^j \leftarrow c_i \quad \text{and} \quad d_i^j \leftarrow d_i. \tag{3.47}$$

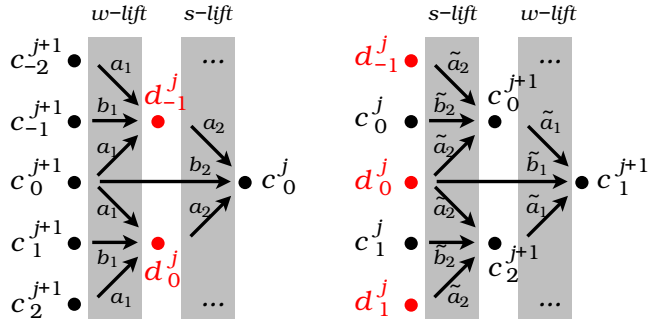

Figure 3.20: Lifting scheme for linear B-spline wavelet. Decomposition (left) and reconstruction (right) are composed of one s-lift and one w-lift operation.

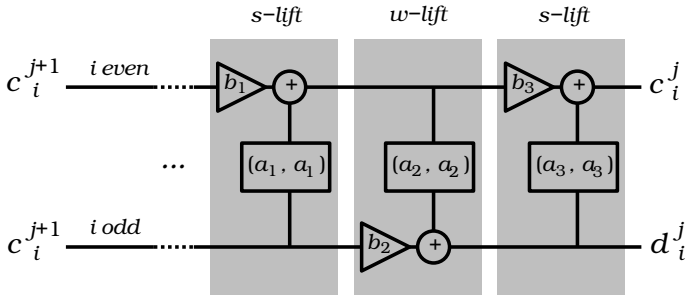

Figure 3.21: Wiring diagram for lifted DWT. Lifting operations that modify coefficients with even indices are called *s-lift* and those that modify coefficients with odd indices are called *w-lift* operations.

Our lifting scheme is depicted in Figure 3.20 for a linear B-spline wavelet. Wiring diagrams for the general approach are shown in Figures 3.21 and 3.22. The last operation for the DWT and the first operation for the inverse DWT is always an s-lift operation, which is necessary to maintain enough degrees of freedom to satisfy the vanishing-moment conditions.

Since we use the same lifting operations to construct both filters h and l at once, we restrict the class of wavelets that can be defined by these operations. However, this wavelet construction uses a minimum of operations and every individual lifting step is inverted simply by replacing a and b by $-\frac{a}{b}$

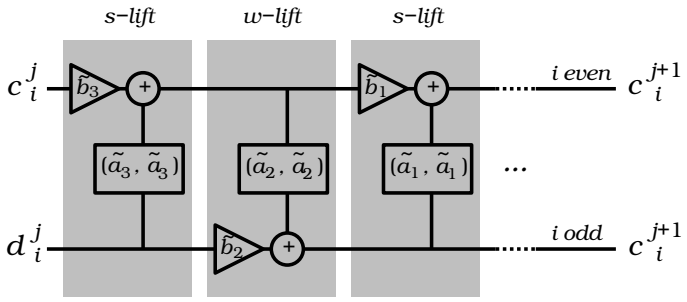

Figure 3.22: Wiring diagram for inverse DWT. The same lifting operations as for DWT occur in reverse order with changed parameters $\tilde{a}_i = -\frac{a_i}{b_i}$ and $\tilde{b}_i = \frac{1}{b_i}$.

and $\frac{1}{b}$, respectively, see Figure 3.22. The only constraint for the existence of the inverse DWT is that the parameter b must be non-zero for all lifting operations. Due to the width of the B-spline subdivision filters l, the number of alternating s-lift and w-lift operations is fixed. Our one-dimensional decomposition rules for symmetric B-spline wavelets are computed by the following operations, where the parameters a_i and b_i are still to be determined:

(i) Linear B-spline wavelet:

$$\begin{aligned} &\text{w-lift}\,(a_1, b_1) \\ &\text{s-lift}\,(a_2, b_2) \end{aligned} \tag{3.48}$$

(ii) Cubic B-spline wavelet:

$$\begin{aligned} &\text{s-lift}\,(a_1, b_1) \\ &\text{w-lift}\,(a_2, b_2) \\ &\text{s-lift}\,(a_3, b_3) \end{aligned} \tag{3.49}$$

(iii) Quintic B-spline wavelet:

$$\begin{aligned} &\text{w-lift}\,(a_1, b_1) \\ &\text{s-lift}\,(a_2, b_2) \\ &\text{w-lift}\,(a_3, b_3) \\ &\text{s-lift}\,(a_4, b_4) \end{aligned} \tag{3.50}$$

In the following, we compute the lifting parameters a_i and b_i defining linear, cubic, and quintic B-spline wavelets. Since the number of non-zero coefficients must be odd for our lifting scheme, we can only construct the DWT for B-spline scaling functions with odd polynomial degrees. The sequence $\tilde{l}$, defining the two-scale relation for B-splines,

$$\phi(x) \;=\; \sum_{i\in\mathbb{Z}} \tilde{l}_i \;\psi(2x-i), \tag{3.51}$$

is already defined in Table 3.1.

To find the constraints imposed by the first vanishing moment condition, we review the two-scale relation between wavelets and scaling functions, given by

$$\psi(x) \;=\; \sum_{i\in\mathbb{Z}} \tilde{h}_i \;\phi(2x-i). \tag{3.52}$$

The integral of the wavelet ψ is zero, if and only if $\tilde{h}$ satisfies

$$\sum_{i\in\mathbb{Z}} \tilde{h}_i \;=\; 0. \tag{3.53}$$

The second vanishing moment condition is satisfied by symmetry of $\tilde{h}$, which implies symmetry of ψ.

3.4.1 Linear B-spline Wavelet

The construction of $\tilde{h}$ is constrained by our specific lifting approach illustrated in Figure 3.20. Figure 3.23 shows the constraints that result from the

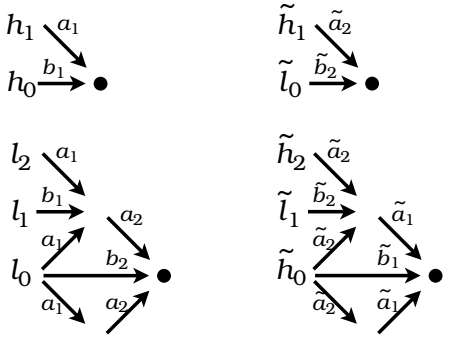

Figure 3.23: Constraints for construction of filters h, l, $\tilde{h}$, and $\tilde{l}$ for linear B-spline scaling functions.

lifting scheme. These constraints for $\tilde{l}$ are

$$\tilde{l}_0 \;=\; \tilde{b}_2 \quad \text{and} \quad \tilde{l}_1 \;=\; \tilde{b}_2\tilde{a}_1. \tag{3.54}$$

Using the values from Table 3.1, we obtain

$$a_1 \;=\; \tfrac{1}{2} \quad \text{and} \quad b_2 \;=\; 1. \tag{3.55}$$

The constraints for $\tilde{h}$ are given by

$$\begin{aligned} \tilde{h}_0 \;&=\; 2\tilde{a}_2\tilde{a}_1 \;+\; \tilde{b}_1 \;=\; \tilde{a}_2 \;+\; \tilde{b}_1, \\ \tilde{h}_1 \;&=\; \tilde{a}_2, \quad \text{and} \\ \tilde{h}_2 \;&=\; \tilde{a}_2\tilde{a}_1 \;=\; \tfrac{1}{2}\tilde{a}_2. \end{aligned} \tag{3.56}$$

We note that $\tilde{l}$ and $\tilde{h}$ are symmetric, and the coefficients with negative indices do not produce additional constraints. Hence, (3.53) becomes

$$\tilde{h}_0 \;+\; 2\tilde{h}_1 \;+\; 2\tilde{h}_2 \;=\; 4\tilde{a}_2 \;+\; \tilde{b}_1 \;=\; 0. \tag{3.57}$$

Since $\tilde{a}_2$ and $\tilde{b}_1$ are proportional and either one of them appears on the right-hand sides of equations (3.56), they do not modify $\tilde{h}$, except for scaling it. Thus, we can choose $\tilde{b}_1 = 1$ and obtain the lifting parameters shown in Table 3.3. The remaining filters h and l can be derived from these lifting parameters, which are summarized in Table 3.4.

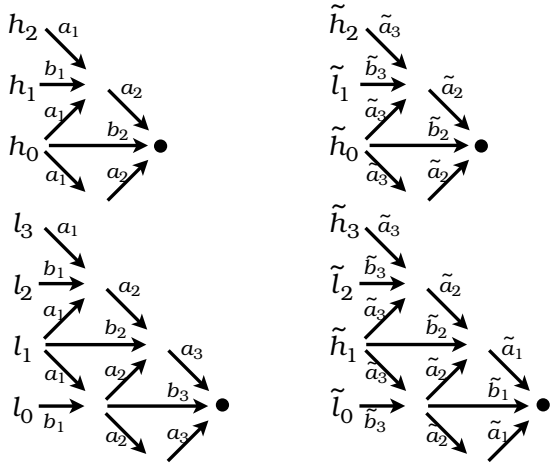

Figure 3.24: Constraints for lifting scheme for cubic B-spline scaling functions.

3.4.2 Cubic B-spline Wavelet

In analogy to linear B-spline wavelets, one can construct wavelets for cubic scaling functions using one additional lifting step. The constraints for the lifting parameters are shown in Figure 3.24. The constraints for $\tilde{l}$ are equivalent to

$$\tilde{a}_1 \;=\; \tfrac{1}{4}, \quad \tilde{a}_2 \;=\; \tilde{b}_1, \quad \text{and} \quad \tilde{b}_1\tilde{b}_3 \;=\; \tfrac{1}{2}. \tag{3.58}$$

The equations for $\tilde{h}$, after eliminating $\tilde{a}_1$ and $\tilde{a}_2$, are given by

$$\begin{aligned} \tilde{h}_0 &= \tilde{b}_2 \;+\; 2\tilde{a}_3\tilde{b}_1, \\ \tilde{h}_1 &= \tfrac{1}{4}\tilde{b}_2 \;+\; \tfrac{7}{4}\tilde{a}_3\tilde{b}_1 \\ \tilde{h}_2 &= \tilde{a}_3\tilde{b}_1, \quad \text{and} \\ \tilde{h}_3 &= \tfrac{1}{4}\tilde{a}_3\tilde{b}_1. \end{aligned} \tag{3.59}$$

Using equation (3.53) we obtain

$$3\tilde{b}_2 \;=\; -16\tilde{a}_3\tilde{b}_1. \tag{3.60}$$

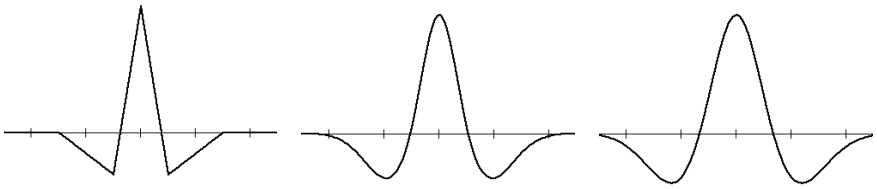

Figure 3.25: Linear, cubic, and quintic B-spline wavelets.

Again, we observe that the remaining lifting parameters do not modify $\tilde{h}$, except for scaling, since $\tilde{b}_2$ and $\tilde{a}_3$ (as well as $\tilde{b}_2$ and $\tilde{b}_1$) are proportional. Hence, we can choose $\tilde{b}_1 = \tilde{b}_2 = 1$ and determine the remaining parameters from the above equations. The resulting values are listed in Tables 3.3 and 3.4.

3.4.3 Quintic B-spline Wavelet

For the quintic case, we only summarize the solution. From the constraints for $\tilde{l}$ we obtain the equations

$$\begin{aligned} \tilde{a}_1 &= \tfrac{1}{6}, \\ \tilde{a}_2 &= \tfrac{9}{16}\tilde{b}_1, \\ \tilde{b}_4\tilde{b}_2 &= \tfrac{1}{4}, \quad \text{and} \\ \tilde{a}_3\tilde{b}_1 &= \tfrac{4}{3}\tilde{b}_2. \end{aligned} \tag{3.61}$$

The constraints for $\tilde{h}$ can be written in terms of $\tilde{b}_3\tilde{b}_1$ and $\tilde{a}_4\tilde{b}_2$. Inserting them into equation (3.53) results in

$$5\tilde{b}_3\tilde{b}_1 = -32\tilde{a}_4\tilde{b}_2. \tag{3.62}$$

We can choose $\tilde{b}_3$, $\tilde{b}_2$, and $\tilde{b}_1$ to be one, since they do not modify $\tilde{h}$. The remaining lifting parameters are then uniquely determined.

degree	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4
1	$-\frac{1}{2}$	1	$\frac{1}{4}$	1				
3	$-\frac{1}{4}$	1	-1	1	$\frac{3}{8}$	2		
5	$-\frac{1}{6}$	1	$-\frac{9}{16}$	1	$-\frac{4}{3}$	1	$\frac{5}{8}$	4

degree	$\tilde{a}_1$	$\tilde{b}_1$	$\tilde{a}_2$	$\tilde{b}_2$	$\tilde{a}_3$	$\tilde{b}_3$	$\tilde{a}_4$	$\tilde{b}_4$
1	$\frac{1}{2}$	1	$-\frac{1}{4}$	1				
3	$\frac{1}{4}$	1	1	1	$-\frac{3}{16}$	$\frac{1}{2}$		
5	$\frac{1}{6}$	1	$\frac{9}{16}$	1	$\frac{4}{3}$	1	$-\frac{5}{32}$	$\frac{1}{4}$

Table 3.3: Lifting parameters for DWT and inverse DWT.

degree	h_0	$h_{\pm1}$	$h_{\pm2}$	$h_{\pm3}$	l_0	$l_{\pm1}$	$l_{\pm2}$	$l_{\pm3}$	$l_{\pm4}$
1	1	$-\frac{1}{2}$			$\frac{3}{4}$	$\frac{1}{4}$	$-\frac{1}{8}$		
3	$\frac{3}{2}$	-1	$\frac{1}{4}$		$\frac{5}{4}$	$\frac{5}{32}$	$-\frac{3}{8}$	$\frac{3}{32}$	
5	$\frac{5}{2}$	$-\frac{15}{8}$	$\frac{3}{4}$	$-\frac{1}{8}$	$\frac{231}{96}$	$-\frac{7}{32}$	$-\frac{21}{24}$	$\frac{15}{32}$	$-\frac{5}{64}$

degree	$\tilde{h}_0$	$\tilde{h}_{\pm1}$	$\tilde{h}_{\pm2}$	$\tilde{h}_{\pm3}$	$\tilde{h}_{\pm4}$	$\tilde{l}_0$	$\tilde{l}_{\pm1}$	$\tilde{l}_{\pm2}$	$\tilde{l}_{\pm3}$
1	$\frac{3}{4}$	$-\frac{1}{4}$	$-\frac{1}{8}$			1	$\frac{1}{2}$		
3	$\frac{5}{8}$	$-\frac{5}{64}$	$-\frac{3}{16}$	$-\frac{3}{64}$		$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	
5	$\frac{77}{128}$	$\frac{7}{128}$	$-\frac{7}{32}$	$-\frac{15}{128}$	$-\frac{5}{256}$	$\frac{5}{8}$	$\frac{15}{32}$	$\frac{3}{16}$	$\frac{1}{32}$

Table 3.4: Filters for DWT and inverse DWT resulting from lifting parameters.

The resulting lifting parameters and filters for our constructions are summarized in Tables 3.3 and 3.4. A desirable property of the lifting parameters b_i is, that they are integers, which makes it possible to compute the lifting scheme in integer arithmetic with perfect reconstruction. In the cubic and quintic cases, we can even scale the last lifting step down by $\frac{1}{2}$ and $\frac{1}{4}$, respectively, which decreases the wavelet coefficients at coarser levels of detail, improving the potential for lossless compression. However, this down-scaling results in a loss of precision when coarser levels of resolution are displayed.

The constructed wavelets are plotted in Figure 3.25. A wider class of wavelets is available by using scaling functions different from B-splines. In particular, it is feasible to construct a wavelet first and obtain a proper scaling function from the constraints given by the lifting scheme.

3.4.4 A Remark on Stability

In the case of large data sets where a number of different levels of resolution are computed, some wavelets may exhibit unstable behavior. This is especially a problem for wavelets with short decomposition and reconstruction filters representing curves and surfaces at high degrees of smoothness. In fact, our quintic wavelet construction fails to represent certain data sets at low resolutions, which is evident in some of the numerical examples provided in the next sections.

Expanding the reconstruction formula for a number of level transitions yields

$$c^n \;=\; \mathbf{S}^n \; c^0 \;+\; \sum_{j=0}^{n-1} \mathbf{S}^j \; \mathbf{E} \; d^j. \tag{3.63}$$

Stability of the reconstruction formula depends on the subdivision operator $\mathbf{S}$ that is applied multiple times. Since we choose dyadic refinement of B-splines for our subdivision schemes, it is guaranteed that the limit curve

or surface is smooth and lies within the convex hull of initial control points when no detail is added. The variation-diminishing property of B-splines guarantees also that the derivatives of a limit curve or surface do not exceed the slope of difference vectors between adjacent control points [49]. Our reconstruction schemes are thus stable in the sense that they produce bounded approximations with bounded derivatives for every level of detail.

The decomposition formula can be expanded as

$$\begin{aligned} c^0 &= \mathbf{F}^n\, c^n \quad \text{and} \\ d^j &= \mathbf{C}\, \mathbf{F}^{n-j-1}\, c^n. \end{aligned} \tag{3.64}$$

Thus, the fitting operator $\mathbf{F}$ controls of stability for decomposition. When applying $\mathbf{F}$ to a control polygon defined by coefficients c^{j+1}, the resulting control points c^j are not necessarily located within the convex hull of the c^{j+1}. However, we can compute a bound for the growth of coefficient values and of differences between adjacent coefficients.

For our linear wavelet construction, we derive a bound from the fitting scheme defined by the filter l:

$$\begin{aligned} c_i^j &= \left(-\tfrac{1}{8} \;\; \tfrac{1}{4} \;\; \tfrac{3}{4} \;\; \tfrac{1}{4} \;\; -\tfrac{1}{8}\right) \begin{pmatrix} c_{i-2}^{j+1} \\ \vdots \\ c_{i+2}^{j+1} \end{pmatrix} \\ &\leq \tfrac{3}{2}\; max\{c_k^{j+1}\}_{k=-2}^{2}. \end{aligned} \tag{3.65}$$

Defining the differences

$$\Delta c_i^j = c_i^j - c_{i-1}^j, \tag{3.66}$$

we can derive a similar bound for

$$\begin{aligned} \Delta c_i^j &= \left(\tfrac{1}{8} \;\; -\tfrac{1}{8} \;\; -1 \;\; -1 \;\; -\tfrac{1}{8} \;\; \tfrac{1}{8}\right) \begin{pmatrix} \Delta c_{i-3}^{j+1} \\ \vdots \\ \Delta c_{i+2}^{j+1} \end{pmatrix} \\ &\leq \tfrac{5}{2}\; max\{\Delta c_k^{j+1}\}_{k=-3}^{2}. \end{aligned} \tag{3.67}$$

This bound on difference vectors limits the growth of the first derivative to $\frac{5}{4}$, since the parametric distance of control points c^j doubles for level $j+1$. The wavelet coefficients d^j are defined in terms of Δc^{j+1}, since they are translation-invariant and thus represent difference vectors between control points, provided that $\sum_i h_i = 0$.

For our cubic fitting scheme, these bounds are defined as

$$\begin{aligned} c_i^j &= \tfrac{1}{32} \begin{pmatrix} 3 & -12 & 5 & 40 & 5 & -12 & 3 \end{pmatrix} \begin{pmatrix} c_{i-3}^{j+1} \\ \vdots \\ c_{i+3}^{j+1} \end{pmatrix} \\ &\leq \tfrac{5}{2}\ max\{c_k^{j+1}\}_{k=-3}^{3}. \end{aligned} \tag{3.68}$$

and

$$\begin{aligned} \Delta c_i^j &= \tfrac{1}{32} \begin{pmatrix} -3 & 9 & 7 & -45 & -45 & 7 & 9 & -3 \end{pmatrix} \begin{pmatrix} \Delta c_{i-4}^{j+1} \\ \vdots \\ \Delta c_{i+3}^{j+1} \end{pmatrix} \\ &\leq 4\ max\{\Delta c_k^{j+1}\}_{k=-4}^{3}. \end{aligned} \tag{3.69}$$

For the quintic wavelet construction, these bounds for the growth of c_i^j and Δc_i^j are 5.6875 and 9.875, respectively. It is possible to construct control polygons such that the coordinates of the control points exactly grow by these bounds for one fitting step. When multiple fitting steps are computed, a tighter bound applies for every single step. This bound can be computed from the matrix norm of $\mathbf{F}^n$,

$$c_i^0 \leq \|\mathbf{F}^n\|_\infty\ max\{c_k^n\}_{k\in\mathbb{Z}}, \tag{3.70}$$

where $\|\mathbf{F}^n\|_\infty$ is the maximal sum of the absolute values of the entries in any row of $\mathbf{F}^n$. This bound is difficult to compute, since the matrix $\mathbf{F}^n$ becomes denser and denser with increasing n. Numerical analysis of this matrix suggests that its norm is bounded independently of n for the linear wavelet. In the cubic case, it grows moderately with n and in the quintic

case it grows rapidly. We note that this situation becomes different when integer arithmetic is used, since the rate of growth is cut down by $\frac{1}{2}$ and $\frac{1}{4}$ for the cubic and quintic wavelets, respectively.

3.5 Modeling Scientific Data

Compressed hierarchical representations are required to store, transmit, and visualize large-scale scientific data sets. The biorthogonal wavelet transforms defined in the previous sections provide ideal tools for computing multiple levels of resolution locally or globally for massive data sets. The transforms are efficiently computed by a small number of local lifting operations. In the following, we describe the use of these tools for modeling scientific data and provide numerical examples for lossless and lossy compression.

Representing scientific data with high fidelity often requires that a prescribed error bound is satisfied. Discretized data is defined by a set of real-valued samples that typically are contaminated with numerical noise of very low amplitude. When this amplitude is known, the samples can be rounded to a certain precision such that most of the noise is cut off and only the significant information is preserved. Choosing this precision is crucial for lossless compression, since it is impossible to compress numerical noise. When it becomes necessary to re-sample a data set to a regular grid, the resolution must be chosen carefully so that the introduced sampling error lies in the same order of magnitude as the data precision.

For lossless compression, the finite-precision samples are scaled to integers from which they can exactly be reproduced. An integer-to-integer wavelet transform maps these integer samples into a set of integer coefficients of low absolute values. These are compressed by arithmetic coding [99, 129], as described in the next section. For compressing large-scale data sets, it is possible to transform small blocks independently for caching purposes. Such

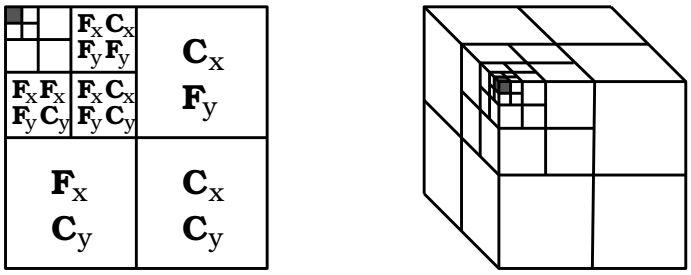

Figure 3.26: Abstraction of wavelet transform applied to surfaces and volumes.

a fragmentation does not cause artifacts, since neither the wavelet transform, nor the arithmetic coder introduce errors.

Higher compression rates are obtained by lossy compression. In this case, the data is transformed using floating-point arithmetic and the resulting coefficients are quantized, *i.e.*, rounded to a finite precision and scaled to integers. Depending on the precision and on the number of level transitions in the wavelet transform, one can still compute a bound for the introduced error. Though this bound is much greater than the rounding error for each coefficient, the resulting accumulated quantization error remains usually low in average. The quantization error can be computed from the reconstructed data set. Again, compression is obtained by arithmetic coding of the integer-valued coefficients.

Generalizing the wavelet transform to higher-dimensional data, like surfaces and volumes, is straight forward. The block structure containing coefficients of different types is illustrated in Figure 3.26 for surfaces and volumes. We note that the lifting scheme does not separate coefficients c^j and d^j into different regions.

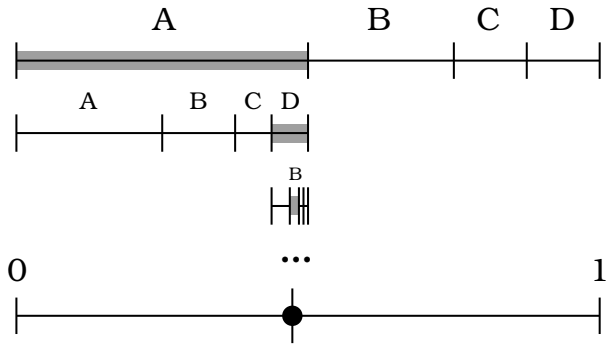

Figure 3.27: Arithmetic coding of the symbol list "ADBAAABC." The average number of bits per symbol is 1.75 for this example.

3.5.1 Arithmetic Coding

In the coding step, the small range of coefficient values and the fact that small coefficients occur more often than larger ones are exploited to store the coefficients compactly. For a range of n possible coefficient values (symbols) with occurance probabilities $p_1, \ldots, p_n$, the minimal average number of bits per symbol is

$$\sum_{i=1}^{n} -p_i \; log_2 \; (p_i). \tag{3.71}$$

This optimum is actually reached by arithmetic coding. A list of symbols is encoded by subdividing the interval $[0, 1]$ according to the probabilities of all symbols. The subinterval corresponding to the current symbol in the list is selected and recursively subdivided according to the succeeding symbols, see Figure 3.27. Finally, the encoded list of symbols is represented as a single, highly precise number in $[0, 1]$ that identifies the final subinterval. From this number, the list of symbols can be uniquely recovered.

Despite its linear computation time, arithmetic coding requires a lot of implementation to avoid numerical problems, since an entire data set is encoded into one single number. Other coding schemes, like *Huffman coding*

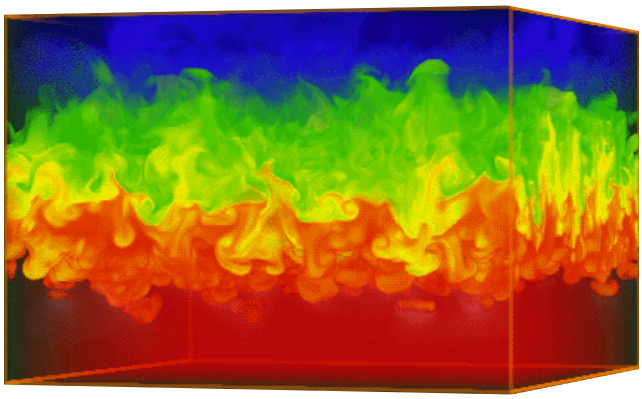

Figure 3.28: Numerical simulation of a Rayleigh-Taylor instability. The data set is composed of 512^3 samples for 301 time steps. Image and data set are courtesy of Lawrance Livermore National Laboratory.

[90], provide a more efficient implementation, but they do not obtain optimal compression rates. An implementation for an arithmetic coder is provided by Witten *et al.* [129].

A major advantage of arithmetic coding is that the probabilities for symbols can be changed adaptively. The local histogram of coefficient values is typically much different in dense and sparse regions of a data set and for coefficients of different types. Starting with a Gaussian distribution of probabilities, one can update the histogram based on the most recently decoded coefficients and thus change it adaptively during the process. An approach exploiting correlation between different levels of a wavelet transform, useful for sparse data, are zerotrees described by Shapiro [120].

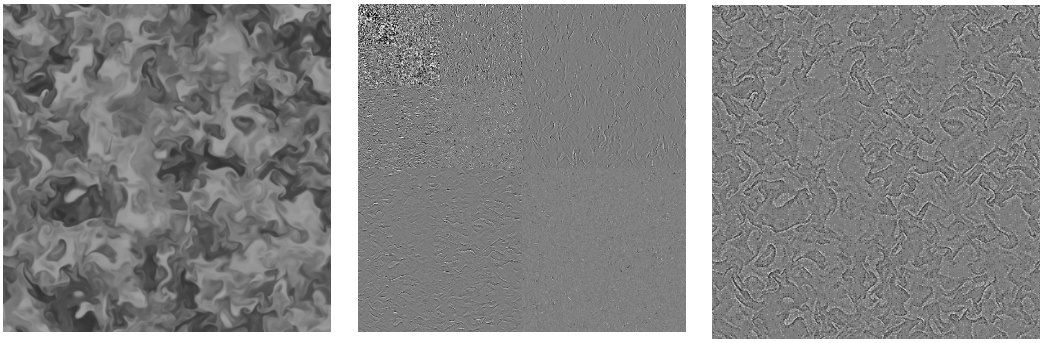

Figure 3.29: 512^2 slice of Rayleigh-Taylor data set (left), linear B-spline integer wavelet transform with coefficients separated in space (middle), and same transform without separating coefficients (right). The wavelet coefficients are scaled by ten, and a grey level was added.

3.5.2 Lossless Compression Examples

We have applied the lossless compression algorithm to a *Rayleigh-Taylor instability* simulation, shown in Figure 3.28. This data set represents a three-dimensional, time-varying temperature distribution in a turbulent flow. This simulation was computed on a massively parallel supercomputer at Lawrence Livermore National Laboratory, California, U.S.A. Its resolution is 512^3 byte samples for each of 301 time steps. Since the data set is sparse in the first time steps, a high lossless compression rate (storage size / compressed storage size) in the order of 20 can be achieved.

Due to the large size of the data set, the compression algorithm needs to be applied locally to smaller blocks. For numerical examples, we have chosen a horizontally aligned 512^2 slice and a 64^3 brick in the middle of the data set at the last, most turbulent time step. The slice and its linear B-spline integer wavelet transform is shown in Figure 3.29. The coefficients can be re-arranged in space according to their types. This may be useful to use different histograms for encoding different groups of coefficients. It also shows, which type of coefficient has large absolute values in average for a

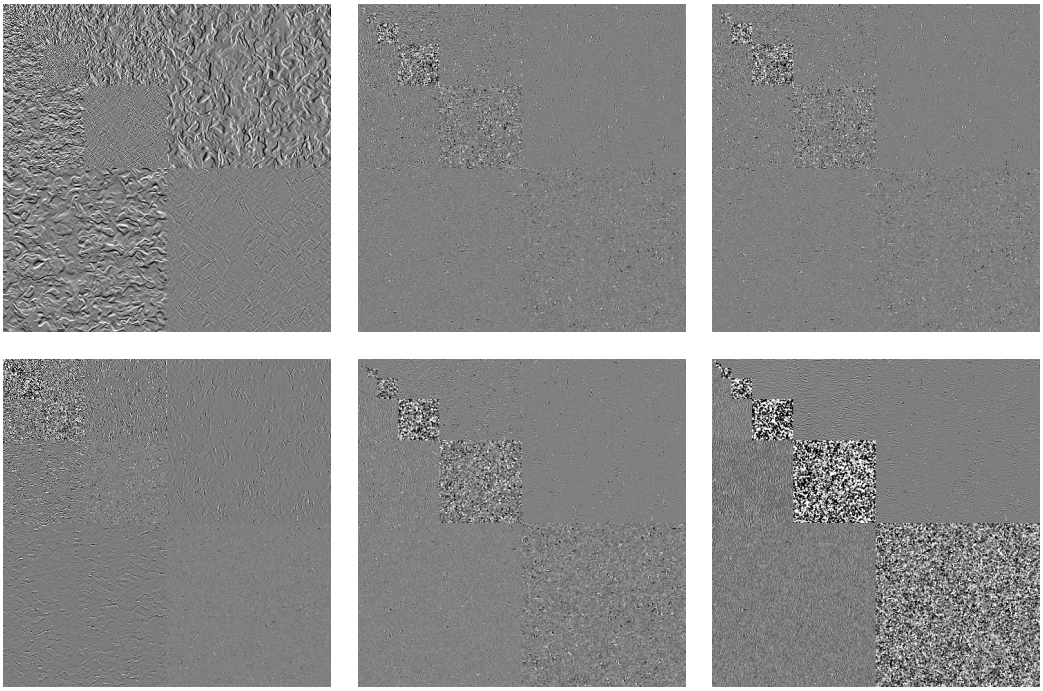

Figure 3.30: Integer wavelet transforms using ψ_H, ψ_Q, and ψ_D (top), and ψ_1, ψ_3, and ψ_5 (bottom).

certain wavelet transform.

We have applied six lifted integer wavelet transforms to the same slice. The following wavelets have been used:

- ψ_H: the Haar wavelet.
- ψ_Q: the quadratic B-spline wavelet with small support and one vanishing moment.
- ψ_D: the quadratic wavelet constructed by Duchaineau [41].
- ψ_1: the linear B-spline wavelet with two vanishing moments.
- ψ_3: the cubic B-spline wavelet with two vanishing moments.

wavelet	comput. time [sec]	arith. code length [%]	compr. rate	number of zeros [%]	coeff. range min	coeff. range max
ψ_H	0.055	51.0	1.96	20.7	−67	112
ψ_Q	0.097	35.6	2.81	35.5	−49	46
ψ_D	0.115	35.8	2.79	35.6	−59	50
ψ_1	0.078	39.5	2.53	37.3	−94	110
ψ_3	0.121	37.6	2.66	31.7	−74	68
ψ_5	0.136	52.6	1.90	16.9	−221	188

Table 3.5: Lossless compression results for 512^2 slice of Rayleigh-Taylor data set.

- ψ_5: the quintic B-spline wavelet with two vanishing moments.

Figure 3.30 shows the different wavelet transforms scaled by ten with a grey level added. Coefficients of same type are grouped together. It can be observed that the discontinuous Haar wavelet represents high frequency parts well but leads to large coefficients on coarse levels of resolution (located in the upper left corner). The quintic wavelet produces large coefficients on the diagonal, due to its instability.

The numerical compression results for the slice of the Rayleigh-Taylor instability data set are summarized in Table 3.5. Computation times have been measured on a 194 MHz MIPS R10000 processor on an SGI Onyx system. The arithmetic code length is shown in percent of the size of the uncompressed slice (262144 bytes). The code lengths and the corresponding compression rates are computed from equation (3.71). We note that the size of the coefficient histogram is not included. However, it is possible to recover the histogram adaptively from the decoded coefficients resulting in even higher compression rates. We have also counted the number of zero coefficients and recorded the range of coefficient values for every transform.

Numerical compression results for a 64^3 brick of the Rayleigh-Taylor data

wavelet	comput. time [sec]	arith. code length [%]	compr. rate	number of zeros [%]	coeff. range min	coeff. range max
ψ_H	0.103	34.5	2.90	41.7	−43	60
ψ_Q	0.177	29.4	3.40	43.3	−89	62
ψ_D	0.222	29.0	3.45	44.3	−95	62
ψ_1	0.151	26.7	3.75	52.6	−83	103
ψ_3	0.246	35.0	2.86	37.0	−87	162
ψ_5	0.308	56.6	1.77	18.8	−512	1177

Table 3.6: Lossless compression results for 64^3 brick of Rayleigh-Taylor data set.

set are provided in Table 3.6. The total number of samples is the same as for the slice, but the transforms are computed in three dimensions. Higher compression rates are obtained, due to the correlation of function values in the third dimension.

3.5.3 Lossy Compression Examples

As an example for lossy compression based on floating-point data, we have chosen the Crater Lake data set, courtesy of U.S. Geological Survey. We have resampled this data set to a regular grid of 320×448 samples, which is about 90 percent of the original data set (159272 samples). For re-sampling we used the function values of the closest original samples We did not use a blending scheme to preserve the structure of the data.

Table 3.7 shows the maximal and L^2 quantization errors in percent of the amplitude for different wavelet transforms at compression rates of ten and 100. The compression rates are obtained by scaling the wavelet coefficients of each individual transform by an individual factor before rounding them to integers. Some of the reconstructed terrain data sets are depicted in Figure 3.31.

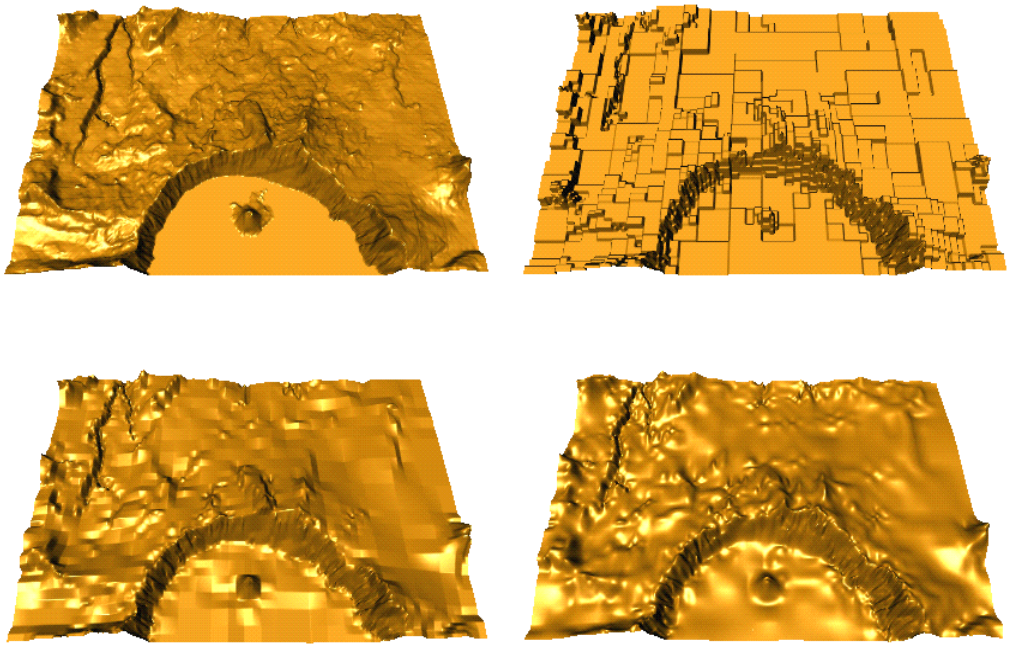

Figure 3.31: Re-sampled Crater Lake data set and reconstructions from a 1:100 compression. Full 320×448 resolution (top left), Haar wavelet ψ_H (top right), linear B-spline wavelet ψ_1 (bottom left), and Duchaineau's wavelet ψ_Q (bottom right).

wavelet	compression rate 10		compression rate 100	
	L^∞-error	L^2-error	L^∞-error	L^2-error
ψ_H	3.55	0.89	16.34	4.01
ψ_D	1.53	0.29	12.21	2.49
ψ_Q	1.18	0.24	9.66	2.03
ψ_1	1.41	0.26	7.94	1.67
ψ_3	3.40	0.66	34.63	7.37

Table 3.7: Lossy compression results for Crater Lake.

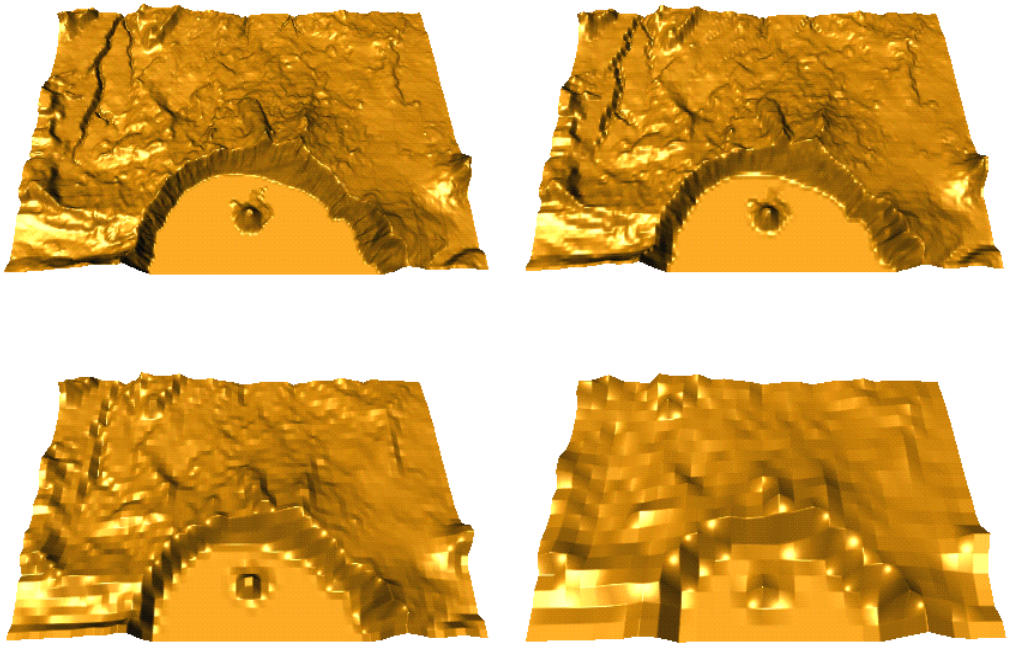

Figure 3.32: Different levels of resolution, $j = 1, 2, 3, 4$, for linear B-spline wavelet ψ_1.

wavelet	level 1, $\frac{1}{4}$ of data		level 2, $\frac{1}{16}$ of data		level 3, $\frac{1}{64}$ of data	
	L^∞-error	L^2-error	L^∞-error	L^2-error	L^∞-error	L^2-error
ψ_H	11.8	0.47	20.3	0.97	24.7	1.78
ψ_D	9.6	0.21	11.4	0.48	13.4	0.99
ψ_Q	9.6	0.19	11.1	0.42	14.8	0.85
ψ_1	10.4	0.22	15.4	0.49	18.1	1.01
ψ_3	11.0	0.21	18.5	0.22	21.3	1.22
ψ_5	13.8	0.31	40.4	2.26	196.5	18.3

Table 3.8: Fitting errors for Crater Lake using different wavelets at three levels of resolution $j = 1, 2, 3$.

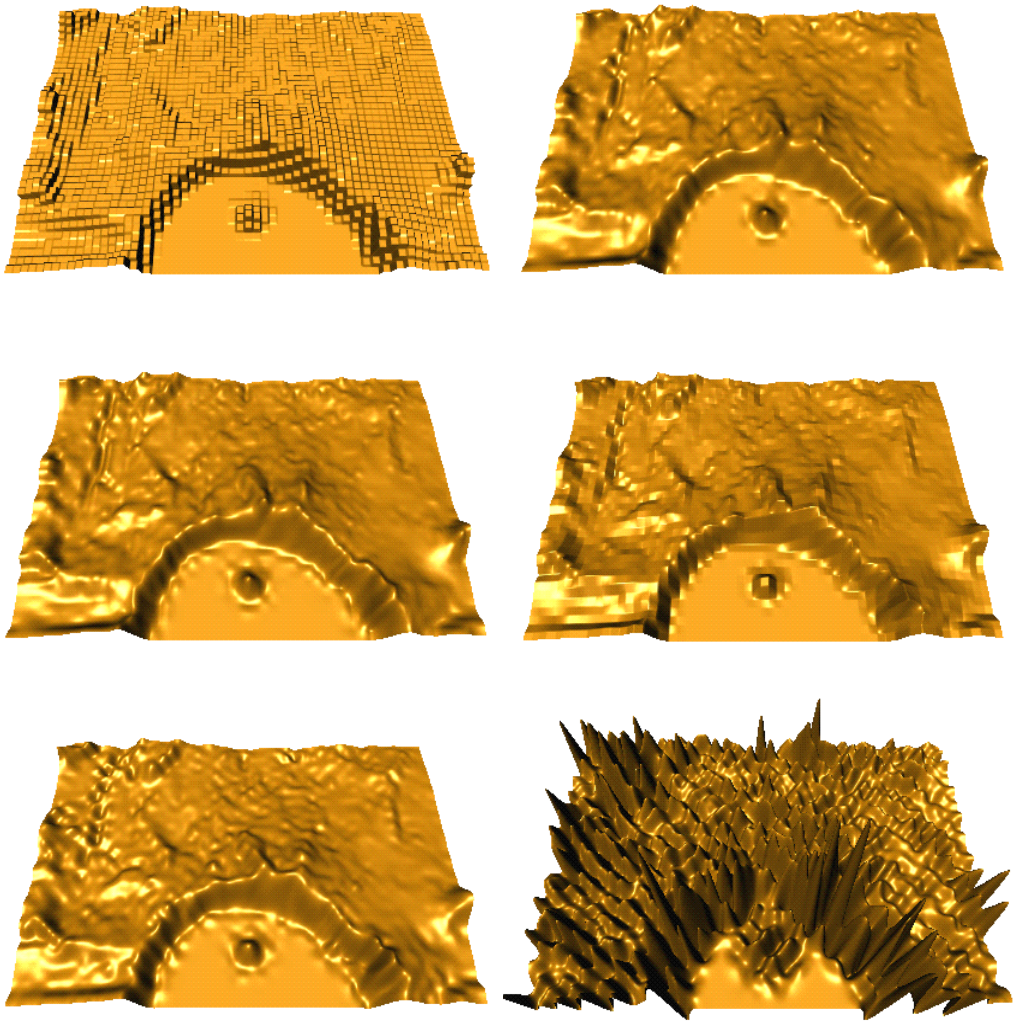

Figure 3.33: Third fitting level ($\frac{1}{64}$ of data) for the wavelets ψ_H, ψ_Q, ψ_D, ψ_1, ψ_3, and ψ_5. The instability of ψ_5 becomes evident.

Another important application of wavelets is the computation of different levels of resolution, obtained by the fitting operator. To visualize the fitting quality for a wavelet transform, we set all coefficients located on fine levels of resolution to zero and use only the subdivision scheme for reconstruction. Different levels for using our linear B-spline wavelet ψ_1 are shown in Figure 3.32. The third fitting level obtained by different wavelet transforms is depicted in Figure 3.33. Here, the instability of the fitting operator for the quintic B-spline wavelet ψ_5 becomes a serious problem. The approximation errors of the fitting schemes for different wavelets at three levels of resolution are compared in Table 3.8.

Chapter 4

Wavelets on Arbitrary Topology

Many scientific data sets are not defined on regular, rectilinear grids. Non-planar geometries, like isosurfaces, shock waves, material boundaries, and computer-aided design (CAD) models cannot be represented as functions defined over Euclidean domains. Even in the case of planar or volumetric data sets, it is often desirable to exploit an underlying grid structure for building a geometric model, rather than re-sampling the data to a uniform grid. Multiresolution models are required to follow the topology or grid structure suggested by the data.

This chapter is concerned with generalizing symmetric lifted B-spline wavelets to surface domains of arbitrary topology, *i.e.*, two-manifolds of arbitrary genus, defined by polygonal base meshes. Base meshes serve as control meshes for coarse-level surface approximations. We apply recursive subdivision schemes, resulting in a smooth surface representation with locally regular (rectilinear) parametrization. Sharp feature lines and boundary curves are processed by different subdivision rules. Detail can be added to or removed from a surface at every level of subdivision. We generalize the four operators,

fitting **F**, compaction-of difference **C**, subdivision **S**, and expansion-of-detail **E**, defined in Chapter 3, to construct a wavelet transform on this mesh hierarchy.

A different application for our generalized wavelet transform is a multiresolution representation defined on tessellations. Starting with a partitioning of an Euclidean domain composed of convex cells (or tiles), we recursively subdivide these cells in a regular fashion. This subdivision hierarchy is then used to represent functions defined on tessellations at multiple levels of resolution. We exploit the tessellations as a natural parametrization.

4.1 Multiresolution Representation of Two-Manifolds

The need for multiresolution surface models of arbitrary topology arises from applications in different fields, such as

- *Computational fluid dynamics (CFD).* For example, isosurfaces in three-dimensional flow fields are surfaces defined by a constant scalar parameter like density, pressure, temperature, and absolute values of velocity and vorticity. For visualization and exploration purposes, parametric surface representations are often preferred, since they explicitly represent surface topology and provide efficient access to the corresponding geometry.

- *Medical imaging.* Surface approximations representing bone or vessel structures are obtained from contouring algorithms applied to volumetric data constructed from computerized tomography (CT) scans. Smooth surface models can be used for visualizing these structures, identifying diseases, and planning surgery.

- *Reverse engineering.* Points sampled from a physical model using, for example, laser range scanners are approximated or interpolated by continuous surface models. This method can be used for designing car or aircraft bodies by building prototypes and converting them into geometric models used for numerical analysis and manufacturing.

A preliminary problem is reconstructing surfaces from some initial data and assigning parametrizations to their geometry. Starting with a coarse parametric surface model, in our case a coarse control (or base) mesh whose polygons are mapped to certain surface regions, we create a hierarchy of finer approximations to the underlying geometry and, if applicable, to functions defined on the geometry. Once our hierarchical surface model is built, it can efficiently be used to display, analyze, and edit local regions at user-defined resolutions. Applications include view-dependent visualization, progressive transmission, CAD, and compression.

4.1.1 Related Work

Many approaches to multiresolution surface modeling exist. We summarize related work on surface reconstruction, triangle meshes and smooth surface modeling, and we review recent developments regarding wavelets defined on arbitrary topology.

In the case of isosurfaces, an initial triangulated surface representation is obtained by standard contouring methods, like *marching cubes*. A more difficult problem is the reconstruction of a surface from a set of unorganized points in three dimensions, which arises in reverse engineering applications where solid objects are sampled from different views using laser-range scanners. Starting with such a set of unorganized points, Hoppe *et al.* [68] construct a signed-distance function based on the three-dimensional Voronoi diagram of these points by using an *Euclidean minimum-spanning tree* (EMST) to propagate a consistent orientation of surface normals. Their distance

function, defined by the signed orthogonal distance from a linear surface approximation in every Voronoi cell, is used to obtain a triangulated isosurface from marching cubes. Curless/Levoy [26] also use signed distance functions for surface reconstruction from range-scanned images. Their algorithm builds an individual signed-distance function for every scanned image and then uses a weighted sum of these distance functions to extract a triangulated surface.

Another Voronoi-based approach to surface reconstruction from unorganized points by triangulating these points directly is described by Amenta *et al.* [1, 2]. Their algorithm uses the medial axis, *i.e.*, the set of points that have equal distance from at least two surface components, to separate distinct surface regions that are close in space. Heckel *et al.* [64, 65] reconstruct surfaces by adaptive clustering based on principal component analysis (PCA). PCA provides a linear surface approximation with a least-squares error estimate for every cluster of unorganized points. From this clustering, a triangulated surface is constructed.

Alpha shapes, introduced by Edelsbrunner/Mücke [47], provide a powerful tool for surface reconstruction by identifying connected surface components. A *simplicial complex* is constructed for a set of unorganized points, containing all simplices, *i.e.*, line segments, triangles and tetrahedra, that fit into spheres of radius α. The optimal radius can interactively be chosen by the user. The final triangulated surface is defined as the outer boundary of the simplicial complex. The idea of identifying triangles of an outer surface boundary by "rolling" a sphere of constant radius along the boundary of a triangulated region, thus augmenting this region, is exploited by Bernardini *et al.* [9], resulting in a highly efficient reconstruction method for densely sampled surfaces.

A reconstruction algorithm for smooth surfaces using B-spline patches is discussed by Eck/Hoppe [45]. Triangular surface meshes are simplified to coarser meshes, conserving surface topology and minimizing geometric distortion. Polygonal base meshes with quadrilateral faces are constructed first.

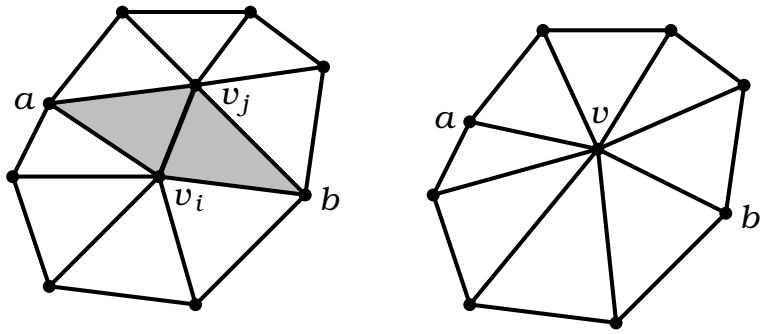

Figure 4.1: Mesh simplification by collapsing edge $v_i v_j$ to a single vertex v, decrementing the valences of vertices a and b.

The faces are then filled with B-spline patches that are fitted to the underlying points using a least-squares approach satisfying C^1-continuity conditions. A similar algorithm by Guo [56] constructs base meshes form alpha shapes and uses a C^2-continuous blending surface representation [54]. Krishnamurthy/Levoy [80] introduce a method for fitting B-spline patches to dense polygon meshes. A surface reconstruction approach based on piecewise algebraic surfaces is described by Bajaj *et al.* [5] and by Bernardini *et al.* [8].

Multiresolution surface representations are important for processing large-scale surface models efficiently. A mesh-simplification method for triangulated surfaces is described by Hoppe [70]. This approach uses edge collapsing, see Figure 4.1, to reduce the number of triangles resulting in multiple levels of resolution. A survey of different mesh simplification methods is provided by Lindstrom/Turk [83, 84]. Recently developed signal processing algorithms for triangulations are described by Guskov *et al.* [58] and by Daubechies *et al.* [33]. Inspired by the wavelet transform, these methods define similar filtering operations on irregular meshes without regular subdivision connectivity. An earlier approach to signal processing on irregular meshes is due to Taubin [127]. A mesh-optimization method is introduced by Rumpf [112]. Interactive multiresolution mesh editing techniques are pre-

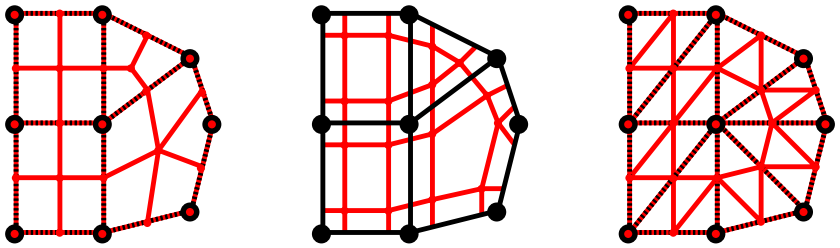

Figure 4.2: Subdivision structures used for Catmull-Clark, Doo-Sabin, and Loop's subdivision surfaces.

sented by Zorin *et al.* [131] and by Kobbelt *et al.* [76].

Surfaces of arbitrary topology can be represented without the need of completely irregular meshes at every level of resolution. Coarse irregular base meshes defining surface topology can be recursively refined in a regular way. The first wavelet constructions for subdivision surfaces were introduced by Lounsbery [90] and by Lounsbery *et al.* [91]. These approaches construct semi-orthogonal wavelets for a hierarchy of scaling functions defined by a subdivision surface scheme. Only in the case of interpolating subdivision, the resulting decomposition filters are finite. Our approach, in contrast, provided in the next section, constructs biorthogonal wavelets with finite filters based on generalized B-spline subdivision.

The first subdivision schemes generating C^1-continuous limit surfaces were described by Catmull/Clark [22] and by Doo/Sabin [40]. Starting with an irregular polygonal control mesh, Catmull-Clark subdivision creates a new control point located at the centroid of every face (polygon) and a new control point for every edge, see Figure 4.2. After the first subdivision step, all faces are quadrilaterals and the only irregularity in the mesh is due to *extraordinary* vertices, *i.e.*, vertices with other than four adjacent edges. Their subdivision scheme generates bicubic B-splines on regular, rectilinear grids. Doo-Sabin subdivision is a generalization of biquadratic B-splines based on a subdivision structure that is dual to the meshes generated by Catmull-Clark

subdivision, *i.e.*, vertices correspond to faces and vice versa. Doo-Sabin subdivision constructs a new control point for every vertex inside each polygon, see Figure 4.2. The initial control points are erased. A smooth subdivision scheme generalizing box splines [15] based on triangular subdivision is introduced by Loop [86].

These three different subdivision schemes are local, linear, and stationary, *i.e.*, they compute the new control points from a weighted sum of a local stencil of the original control points where the weights are constants depending on the local connectivity, yet independent of the subdivision level. In the case of Catmull-Clark and Loop's subdivision, the locations of the original control points are modified, as well. Interpolating subdivision schemes do not modify control points arising from coarser subdivision levels. An interpolating scheme generalizing bicubic C^1-continuous splines based on Catmull-Clark mesh connectivity is described by Kobbelt [78]. A smooth interpolating scheme for triangular meshes using Loop's mesh structure is known as *butterfly* subdivision, introduced by Dyn *et al.* [44]. Its behavior at extraordinary points was improved by Zorin *et al.* [132].

All previously described subdivision schemes have in common, that the number of vertices is (approximately) quadrupled in every subdivision step. Subdivision methods with less aggressive refinement are described by Peters/Reif [105] and by Kobbelt [74]. Subdivision surfaces acting as height maps, thus providing surface detail by offsetting a coarse surface approximation are recently presented by Lee *et al.* [82] and by Guskov *et al.* [59].

Compared to other smooth surface constructions, like blending surfaces [54], triangular B-splines [119, 27], constructions based on triangular or quadrilateral *Bézier patches* [88, 60, 104, 67], and *S-patches* [87], subdivision surfaces require much less implementation and provide control meshes at multiple levels of resolution. The corresponding basis functions span a sequence of nested spaces, which is required for the construction of wavelets. Subdivision schemes generating piecewise polynomials can efficiently be eval-

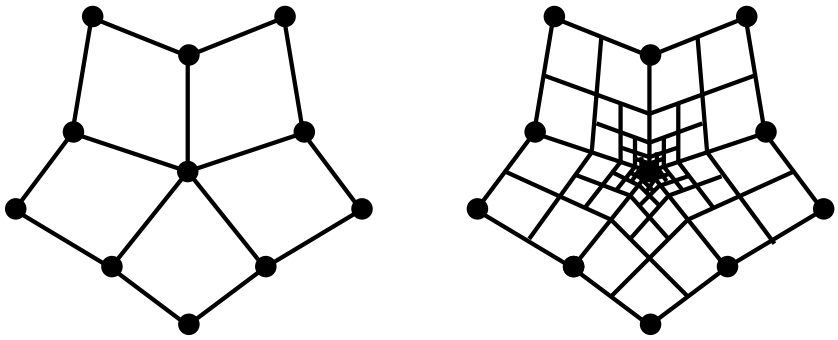

Figure 4.3: Polynomial patches in the neighborhood of an extraordinary point for generalized bilinear subdivision (left) and bicubic subdivision (right).

uated at arbitrary parameter values, even if there are an infinite number of smaller and smaller patches located in the neighborhood of extraordinary points [122], see Figure 4.3. The limit behavior at extraordinary points can be analyzed using the eigenstructure of a local subdivision matrix [106], which was first done by Doo/Sabin [40]. It is feasible to interpolate points and normals and to minimize certain fairness functions when constructing surfaces with subdivision schemes [61].

Inspired by Lounsbery's subdivision-surface wavelet constructions, Schröder/Sweldens [115] developed a variety of wavelets for subdivision schemes embedded in spherical domains. Piecewise constant approaches with nearly orthogonal wavelets were constructed by Nielson et al. [103] and by Bonneau [17, 18]. Subdivision surface wavelets can also be used for representing functions defined on a surface, like texture, opacity, and bump maps. The application of multiresolution surface viewing was described by Certain *et al.* [23].

An important problem arising from the use of subdivision-surface wavelets is the construction of base meshes with smooth parametrizations for the subdivision hierarchy. This problem was addressed by Eck *et al.* [46]. Starting

with a triangulated surface, their approach uses a sequence of *harmonic maps* to construct locally planar surface parametrizations with minimal geometric distortion. A coarse approximating Delaunay triangulation is constructed and the initial surface representation is mapped to the Delaunay triangles. These are recursively refined and the new control points are re-sampled from the initial surface using their local parametrization.

A similar, yet more efficient algorithm for multiresolution adaptive parametrization (MAPS) of surfaces was described by Lee *et al.* [81]. MAPS simplifies a given triangle mesh by recursively removing vertices and re-triangulating. During this procedure, the original mesh is mapped onto the coarser triangulation. The coarsest mesh is regularly refined and re-sampled from the original surface using a parametrization that is smoothed by Loop's subdivision scheme. A *shrink-wrapping* approach for surfaces topological equivalent to a sphere is described by Kobbelt *et al.* [75]. This method approximates surfaces by recursively subdividing, relaxing and projecting the control points of an initial base mesh to a given geometry.

4.1.2 Generalized Lifting Operations

We now generalize the symmetric s-lift and w-lift operations, defined in Chapter 3, to subdivision surfaces defined by irregular base meshes. Our linear, cubic and quintic B-spline wavelet constructions are thus generalized to domains of arbitrary topology for the representation of two-manifolds. The resulting basis functions are composed of tensor-product B-splines in the rectilinear mesh regions and are smooth (C^1-continuous in the bicubic case) at extraordinary points. We introduce an index-free notation to define our generalized lifting operations in a comprehensive way.

Before we define wavelets on arbitrary surfaces, we review our one-dimensional wavelet constructions using a notation without coefficient indices. Considering a decomposition step for a control polygon, depicted in

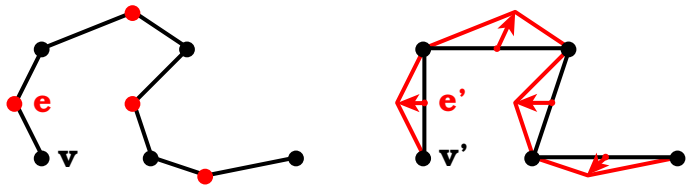

Figure 4.4: Decomposition for a control polygon.

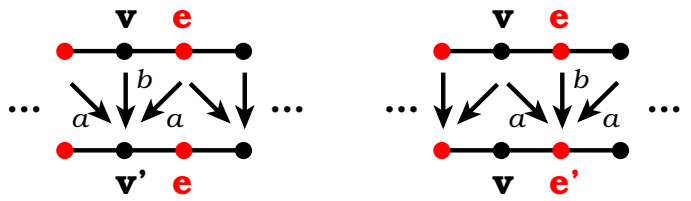

Figure 4.5: S-lift and w-lift operations applied to a control polygon.

Figure 4.4, we denote the points that correspond to vertices in the coarse polygon as **v** points. The points transformed into wavelet coefficients correspond to edges in the coarse polygon and are denoted as **e** points. Decomposition rules for a DWT are defined by two linear operators, a fitting operator $\mathbf{F}$, predicting the control points $\mathbf{v}'$ for a coarser polygon, and a compaction-of-difference operator $\mathbf{C}$, providing the wavelet coefficients $\mathbf{e}'$,

$$\begin{aligned} \mathbf{v}' &= \mathbf{F}(\mathbf{v}, \mathbf{e}) \quad \text{and} \\ \mathbf{e}' &= \mathbf{C}(\mathbf{v}, \mathbf{e}). \end{aligned} \tag{4.1}$$

The corresponding reconstruction rules are due to a subdivision operator $\mathbf{S}$ predicting the shape of a next-finer level control polygon and an expansion operator $\mathbf{E}$ providing the missing details:

$$\begin{pmatrix} \mathbf{v} \\ \mathbf{e} \end{pmatrix} = \mathbf{S}(\mathbf{v}') + \mathbf{E}(\mathbf{e}'). \tag{4.2}$$

In Chapter 3, we defined decomposition and reconstruction rules for symmetric B-spline wavelets in terms of s-lift and w-lift operations, see Figure 4.5. We can re-define these lifting operations using the index-free

operator $\overline{x}_y$ that returns for every point of type y the centroid of all adjacent points of type x, as follows:

$$\textbf{s-lift}(a,b): \qquad \mathbf{v} \leftarrow b\mathbf{v} + 2a\overline{\mathbf{e}}_{\mathbf{v}} \tag{4.3}$$

$$\textbf{w-lift}(a,b): \qquad \mathbf{e} \leftarrow b\mathbf{e} + 2a\overline{\mathbf{v}}_{\mathbf{e}} \tag{4.4}$$

Our one-dimensional decomposition rules for symmetric B-spline wavelets are computed by the following operations:

(i) Linear B-spline wavelet:

$$\begin{array}{c} \text{w-lift}\left(-\frac{1}{2},1\right) \\ \text{s-lift}\left(\frac{1}{4},1\right) \end{array} \tag{4.5}$$

(ii) Cubic B-spline wavelet:

$$\begin{array}{c} \text{s-lift}\left(-\frac{1}{4},1\right) \\ \text{w-lift}\left(-1,1\right) \\ \text{s-lift}\left(\frac{3}{8},2\right) \end{array} \tag{4.6}$$

(iii) Quintic B-spline wavelet:

$$\begin{array}{c} \text{w-lift}\left(-\frac{1}{6},1\right) \\ \text{s-lift}\left(-\frac{9}{16},1\right) \\ \text{w-lift}\left(-\frac{4}{3},1\right) \\ \text{s-lift}\left(\frac{5}{8},4\right) \end{array} \tag{4.7}$$

In the following, we generalize these wavelet transforms to surfaces. Starting with a polygonal mesh, we use the same subdivision structure as for Catmull-Clark surfaces, inserting a new vertex for every face and for every edge. Considering only one subdivision step at a time, we call the initial

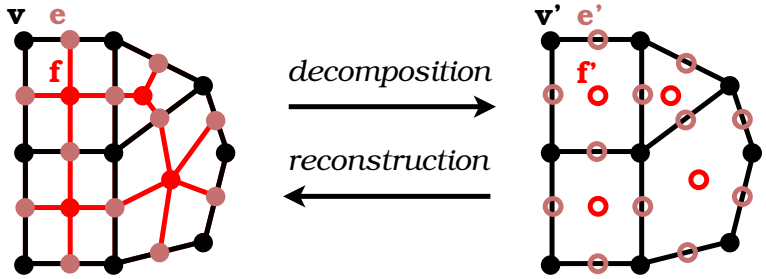

Figure 4.6: Wavelet transform for irregular base mesh.

mesh *submesh* and the mesh resulting from subdivision *supermesh*, since it contains more vertices (about four times as many) as the submesh. We denote the points of the supermesh as $\mathbf{f}$, $\mathbf{e}$, and $\mathbf{v}$, depending on which submesh primitives they correspond to, *i.e.*, faces, edges, and vertices, respectively. A decomposition step maps these points into submesh points $\mathbf{v}'$ and wavelet coefficients $\mathbf{f}'$ and $\mathbf{e}'$, as illustrated in Figure 4.6.

Analogously to the one-dimensional case, decomposition rules for a DWT are defined by a fitting operator $\mathbf{F}$ and by a compaction-of-difference operator $\mathbf{C}$, given by the rules

$$\begin{aligned} \mathbf{v}' &= \mathbf{F}(\mathbf{v},\mathbf{e},\mathbf{f}) \quad \text{and} \\ \begin{pmatrix} \mathbf{e}' \\ \mathbf{f}' \end{pmatrix} &= \mathbf{C}(\mathbf{v},\mathbf{e},\mathbf{f}). \end{aligned} \tag{4.8}$$

Starting with a control mesh at a fine subdivision level with points sampled from a given surface, this decomposition formula is recursively applied to every level transition from fine to coarse, resulting in valid control points for a base mesh and in wavelet coefficients for every level transition.

The corresponding reconstruction rules for the inverse DWT are defined

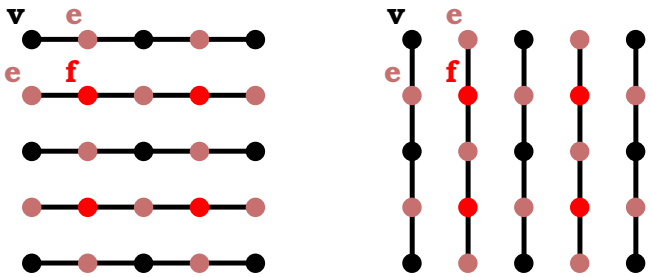

Figure 4.7: A one-dimensional decomposition step is applied to all rows and then to all columns of a rectilinear grid.

by a *subdivision* operator $\mathbf{S}$ and by an *expansion-of-difference* operator $\mathbf{E}$:

$$\begin{pmatrix} \mathbf{v} \\ \mathbf{e} \\ \mathbf{f} \end{pmatrix} = \mathbf{S}(\mathbf{v}') + \mathbf{E}(\mathbf{e}', \mathbf{f}'). \tag{4.9}$$

Reconstruction can be recursively applied to a base mesh with wavelet coefficients. When the finest level of resolution is reached, a smooth limit surface is obtained by applying the subdivision operator $\mathbf{S}$ *ad infinitum*. Analogously, every intermediate control mesh represents a smooth limit surface that is obtained by assuming zero wavelet coefficients on all finer levels, *i.e.*, by using only the subdivision scheme.

On rectilinear grids, a one-dimensional decomposition step can be applied to the rows and columns, independently, as illustrated in Figure 4.7. This approach defines scaling functions as tensor products of one-dimensional scaling functions. The wavelets corresponding to $\mathbf{f}$ points are tensor products of one-dimensional wavelets, and the wavelets located at $\mathbf{e}$ points are tensor products of a wavelet (along the edge) and a scaling function (crossing the edge). For a decomposition step defined by a sequence of lifting operations, the resulting transform remains the same (except for floating-point errors), when every individual lifting operation is applied to the rows and columns, successively.

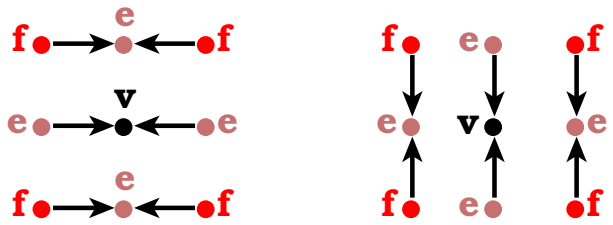

Figure 4.8: S-lift operation applied to rows and then to columns.

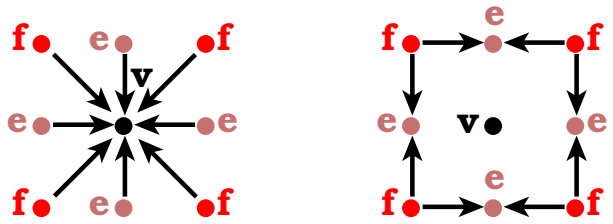

Figure 4.9: S-lift operation computed in different order.

Instead of applying a symmetric lifting operation to the rows and columns of a regular grid, as illustrated in Figure 4.8, we can update the control points of different types individually, see Figure 4.9. Therefore, we need to adjust the weights for the point updates such that the result is the same. The two-dimensional s-lift and w-lift operations can be defined in index-free notation. We use the operator $\overline{x}_y$ returning for every point of type y the centroid of all adjacent points of type x. If there are no adjacent points of this type, like for $\overline{\mathbf{f}}_\mathbf{v}$, then the centroid of the closest stencil of points of type x is returned. The symmetric lifting operations can be defined as

$$\begin{aligned} \textbf{s-lift}(a,b): \qquad \mathbf{v} &\leftarrow b^2\mathbf{v} + 4a^2\overline{\mathbf{f}}_\mathbf{v} + 4ab\overline{\mathbf{e}}_\mathbf{v} \\ \mathbf{e} &\leftarrow b\mathbf{e} + 2a\overline{\mathbf{f}}_\mathbf{e} \end{aligned} \tag{4.10}$$

$$\begin{aligned} \textbf{w-lift}(a,b): \qquad \mathbf{f} &\leftarrow b^2\mathbf{f} + 4a^2\overline{\mathbf{v}}_\mathbf{f} + 4ab\overline{\mathbf{e}}_\mathbf{f} \\ \mathbf{e} &\leftarrow b\mathbf{e} + 2a\overline{\mathbf{v}}_\mathbf{e}. \end{aligned} \tag{4.11}$$

Alternatively, we can change the order of computation, resulting in

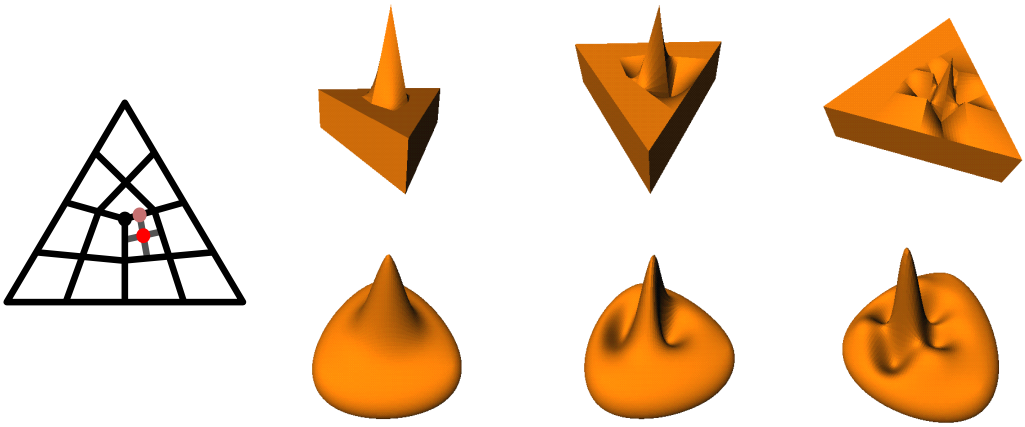

Figure 4.10: Basis functions on top face of a prism near an extraordinary point of valence three. Far left: mesh configuration (top face of prism with control points corresponding to basis functions). Top row: bilinear construction; bottom row: bicubic construction; From left to right: scaling function, **e** wavelet, and **f** wavelet.

slightly different rules, defined as

$$\begin{aligned} \textbf{s-lift}(a,b): \qquad \mathbf{e} &\leftarrow b\mathbf{e} + 2a\overline{\mathbf{f}}_{\mathbf{e}} \\ \mathbf{v} &\leftarrow b^2\mathbf{v} - 4a^2\overline{\mathbf{f}}_{\mathbf{v}} + 4a\overline{\mathbf{e}}_{\mathbf{v}} \end{aligned} \tag{4.12}$$

$$\begin{aligned} \textbf{w-lift}(a,b): \qquad \mathbf{e} &\leftarrow b\mathbf{e} + 2a\overline{\mathbf{v}}_{\mathbf{e}} \\ \mathbf{f} &\leftarrow b^2\mathbf{f} - 4a^2\overline{\mathbf{v}}_{\mathbf{f}} + 4a\overline{\mathbf{e}}_{\mathbf{f}}. \end{aligned} \tag{4.13}$$

We defined these two-dimensional s-lift and w-lift operations so that they are applicable to extraordinary points. Due to the averaging operator $\overline{x}_y$, the total weight added to a control point is independent of its valence. It is possible to use different rules at extraordinary points. We note that the **f** points updated by a w-lift operation are always ordinary points, except for the first subdivision level of an irregular mesh.

We can apply the tools for generalizing wavelet decomposition and reconstruction schemes based on s-lift and w-lift operations to arbitrary polygonal

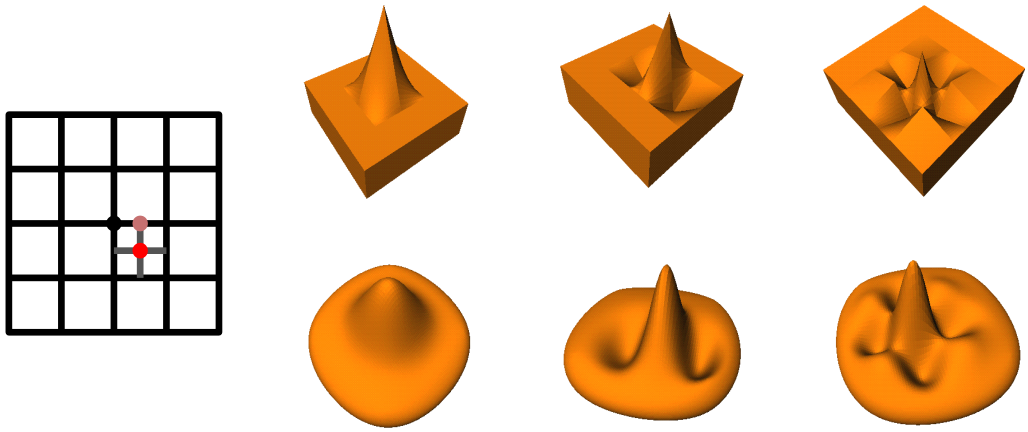

Figure 4.11: Piecewise bilinear and bicubic basis functions on top face of a cube.

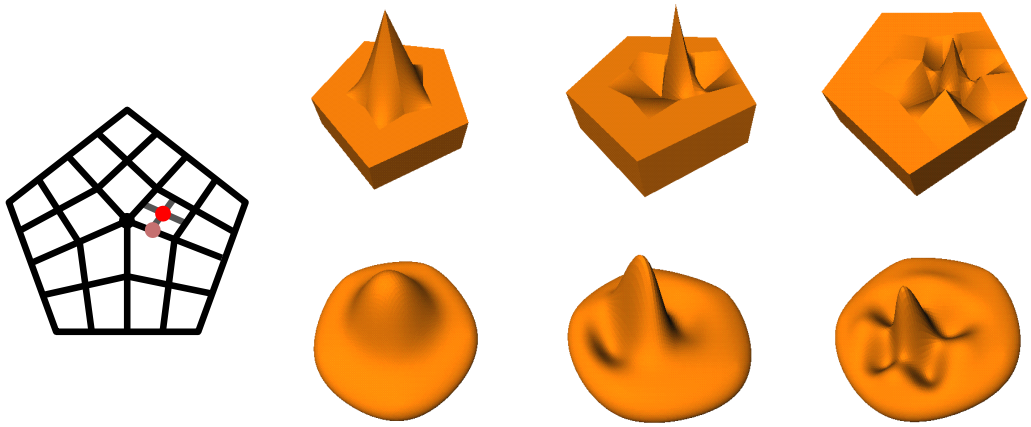

Figure 4.12: Generalized bilinear and bicubic basis functions on top face near an extraordinary point of valence five.

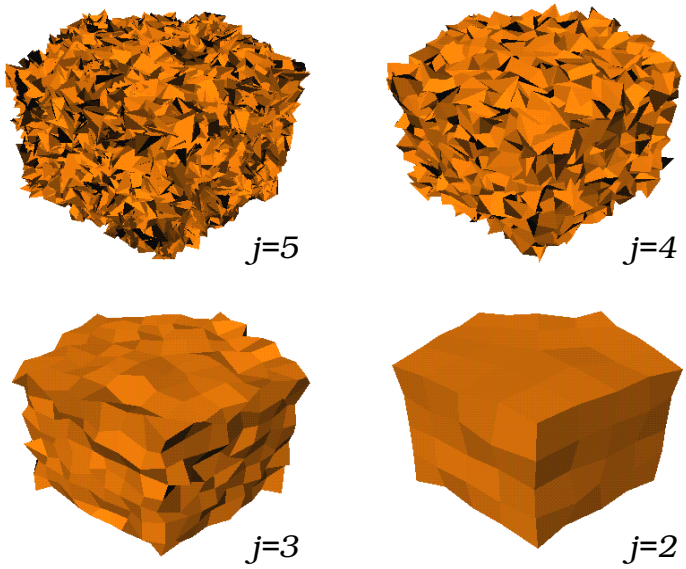

Figure 4.13: Shape with random noise added at subdivision level $j = 5$ and reconstructions for levels $j = 4, 3, 2$ predicted by generalized bilinear fitting.

base meshes. To provide an example, we formulate the decomposition rules for our generalized bilinear B-spline wavelet transform using equations (4.11) and (4.12) and combining the $\mathbf{e}$ updates for both lifting operations:

$$\begin{aligned} 1.\quad & \mathbf{f}' \leftarrow \mathbf{f} + \overline{\mathbf{v}}_{\mathbf{f}} - 2\overline{\mathbf{e}}_{\mathbf{f}} \\ 2.\quad & \mathbf{e}' \leftarrow \mathbf{e} - \overline{\mathbf{v}}_{\mathbf{e}} + \tfrac{1}{2}\overline{\mathbf{f}'}_{\mathbf{e}} \\ 3.\quad & \mathbf{v}' \leftarrow \mathbf{v} - \tfrac{1}{4}\overline{\mathbf{f}'}_{\mathbf{v}} + \overline{\mathbf{e}'}_{\mathbf{v}}. \end{aligned} \tag{4.14}$$

The corresponding reconstruction formula is obtained by inverting these three update operations in reverse order

$$\begin{aligned} 1.\quad & \mathbf{v} \leftarrow \mathbf{v}' + \tfrac{1}{4}\overline{\mathbf{f}'}_{\mathbf{v}} - \overline{\mathbf{e}'}_{\mathbf{v}} \\ 2.\quad & \mathbf{e} \leftarrow \mathbf{e}' + \overline{\mathbf{v}}_{\mathbf{e}} - \tfrac{1}{2}\overline{\mathbf{f}'}_{\mathbf{e}} \\ 3.\quad & \mathbf{f} \leftarrow \mathbf{f}' - \overline{\mathbf{v}}_{\mathbf{f}} + 2\overline{\mathbf{e}}_{\mathbf{f}}. \end{aligned} \tag{4.15}$$

Rules for generalized bicubic and biquintic wavelet transforms are con-

structed analogously. Any use of the two variants for each s-lift and w-lift operations results in an equivalent wavelet transform. Analogously, it is feasible to start with generalizing a reconstruction scheme. Differences in the order of computations may have an impact when integer arithmetic is used. Examples for the constructed basis functions for the generalized bilinear and bicubic wavelet transforms are depicted in Figures 4.10–4.12. The fitting capabilities of the generalized bilinear wavelet transform are demonstrated in Figure 4.13.

Integer arithmetic, as discussed in Chapter 3, can be used for the individual operations, providing a lossless compression algorithm for meshes with control points represented by integer or finite-precision coordinates. Provided that the lifting parameter b ia a non-zero integer for every lifting operation, we can implement equations (4.10) and (4.11) in integer arithmetic:

$$\begin{aligned} \textbf{integer s-lift}(a,b): \qquad \mathbf{v} &\leftarrow b^2\mathbf{v} + [4a^2\overline{\mathbf{f}}_{\mathbf{v}} + 4ab\overline{\mathbf{e}}_{\mathbf{v}}] \\ \mathbf{e} &\leftarrow b\mathbf{e} + [2a\overline{\mathbf{f}}_{\mathbf{e}}], \end{aligned} \tag{4.16}$$

$$\begin{aligned} \textbf{integer w-lift}(a,b): \qquad \mathbf{f} &\leftarrow b^2\mathbf{f} + [4a^2\overline{\mathbf{v}}_{\mathbf{f}} + 4ab\overline{\mathbf{e}}_{\mathbf{f}}] \\ \mathbf{e} &\leftarrow b\mathbf{e} + [2a\overline{\mathbf{v}}_{\mathbf{e}}], \end{aligned} \tag{4.17}$$

where the operator $[\,\cdot\,]$ rounds real numbers to integers. The inverse of these integer lifting operations is defined as

$$\begin{aligned} \textbf{inverse int. s-lift}(a,b): \qquad \mathbf{e} &\leftarrow \tfrac{1}{b}\left(\mathbf{e} - [2a\overline{\mathbf{f}}_{\mathbf{e}}]\right) \\ \mathbf{v} &\leftarrow \tfrac{1}{b^2}\left(\mathbf{v} - [4a^2\overline{\mathbf{f}}_{\mathbf{v}} + 4ab\overline{\mathbf{e}}_{\mathbf{v}}]\right), \end{aligned} \tag{4.18}$$

$$\begin{aligned} \textbf{inverse int. w-lift}(a,b): \qquad \mathbf{e} &\leftarrow \tfrac{1}{b}\left(\mathbf{e} - [2a\overline{\mathbf{v}}_{\mathbf{e}}]\right) \\ \mathbf{f} &\leftarrow \tfrac{1}{b^2}\left(\mathbf{f} - [4a^2\overline{\mathbf{v}}_{\mathbf{f}} + 4ab\overline{\mathbf{e}}_{\mathbf{f}}]\right). \end{aligned} \tag{4.19}$$

4.1.3 Sharp Features and Surface Boundaries

Subdivision surfaces with sharp feature lines and boundaries were presented by DeRose *et al.* [39] and by Zorin *et al.* [14]. The idea is to label certain

edges in a base mesh as features and to apply different subdivision rules to control points located on these edges. Sederberg *et al.* [118] have generalized non-uniform rational B-splines (NURBS) to subdivision of irregular meshes by assigning *knot-spacings* to all edges. Their approach provides a lot of flexibility to surface construction, since the knot-spacings can be used to create any possible blend between a sharp feature and smooth surface. However, their approach requires more implementation and is less intuitive tnan uniform subdivision. Semi-sharp features can also be modeled with uniform schemes by applying the modified subdivision rules to an edge labeled as feature line only for a fixed number of subdivision steps and then using the smooth subdivision scheme for all finer levels [39].

Incorporating modified subdivision rules for features and surface boundaries into a wavelet transform requires to change all four operators $\mathbf{S}$, $\mathbf{E}$, $\mathbf{F}$, and $\mathbf{C}$ in a consistent way. We start with modifying the subdivision operator $\mathbf{S}$ such that the one-dimensional B-spline subdivision rules are applied along sharp edges. The remaining three operators for the wavelet transform are then automatically adapted by the resulting lifting operations.

The one-dimensional cubic reconstruction rules are defined in Chapter 3 as a sequence of lifting operations:

$$\text{s-lift}\left(-\tfrac{3}{16},\tfrac{1}{2}\right);\ \text{w-lift}\,(1,1);\ \text{s-lift}\left(\tfrac{1}{4},1\right).$$

Using index-free notation, the reconstruction formula becomes

$$\begin{aligned} &1.\quad \mathbf{v}' \leftarrow \tfrac{1}{2}\mathbf{v}'' - \tfrac{3}{8}\overline{\mathbf{e}'}_{\mathbf{v}} \\ &2.\quad \mathbf{e} \leftarrow \mathbf{e}' + 2\overline{\mathbf{v}'}_{\mathbf{e}} \\ &3.\quad \mathbf{v} \leftarrow \mathbf{v}' + \tfrac{1}{2}\overline{\mathbf{e}}_{\mathbf{v}}, \end{aligned} \tag{4.20}$$

where $\mathbf{v}''$ represent the initial control points and $\mathbf{e}'$ are the wavelet coefficients representing difference vectors. The control points at a finer level, resulting from reconstruction, are denoted as $\mathbf{v}$ and $\mathbf{e}$.

When modifying the subdivision operator $\mathbf{S}$, to introduce sharp vertices

with C^0 continuity, for example, we need to understand whether the intermediate coefficients $\mathbf{v}'$ represent points or vectors. We associate a weight of one with every control point and a weight of zero with every vector, since vectors (such as wavelet coefficients) represent differences between points. The introduction of features becomes simpler, when all intermediate coefficients represent either points or vectors. Therefore, we scale $\mathbf{v}'$ in equation (4.20) such that it has weight one, resulting in the following reconstruction rules:

$$\begin{aligned} &1. \quad \mathbf{v}' \leftarrow \mathbf{v}'' - \tfrac{3}{4}\overline{\mathbf{e}'}_{\mathbf{v}} \\ &2. \quad \mathbf{e} \leftarrow \mathbf{e}' + \overline{\mathbf{v}'}_{\mathbf{e}} \\ &3. \quad \mathbf{v} \leftarrow \tfrac{1}{2}\mathbf{v}' + \tfrac{1}{2}\overline{\mathbf{e}}_{\mathbf{v}}. \end{aligned} \tag{4.21}$$

This reconstruction scheme is equivalent to equation (4.20), except that the intermediate coefficients $\mathbf{v}'$ represent points. For every sharp vertex, the third update operation in equation (4.21) is ignored, leaving the corresponding control point at its initial position during the subdivision process, as long as no surface detail is added. Sharp vertices can be used to represent cusps on a smooth surface or corners of a surface boundary.

We derive the reconstruction rules for generalized bicubic B-spline wavelets from equation (4.21), which is equivalent to the sequence of lifting operations

$$\text{s-lift}\left(-\tfrac{3}{8}, 1\right);\ \text{w-lift}\left(\tfrac{1}{2}, 1\right);\ \text{s-lift}\left(\tfrac{1}{4}, \tfrac{1}{2}\right).$$

Inserting equations (4.12) and (4.13) provides the generalized bicubic reconstruction scheme, given by these rules:

$$\begin{aligned} &1. \quad \mathbf{e}'' \leftarrow \mathbf{e}''' - \tfrac{3}{4}\overline{\mathbf{f}'}_{\mathbf{e}} \\ &2. \quad \mathbf{v}' \leftarrow \mathbf{v}'' - \tfrac{9}{16}\overline{\mathbf{f}'}_{\mathbf{v}} - \tfrac{3}{2}\overline{\mathbf{e}''}_{\mathbf{v}} \\ &3. \quad \mathbf{e}' \leftarrow \mathbf{e}'' + \overline{\mathbf{v}'}_{\mathbf{e}} \\ &4. \quad \mathbf{f} \leftarrow \mathbf{f}' - \overline{\mathbf{v}'}_{\mathbf{f}} + 2\overline{\mathbf{e}'}_{\mathbf{f}} \\ &5. \quad \mathbf{e} \leftarrow \tfrac{1}{2}\mathbf{e}' + \tfrac{1}{2}\overline{\mathbf{f}}_{\mathbf{e}} \\ &6. \quad \mathbf{v} \leftarrow \tfrac{1}{4}\mathbf{v}' - \tfrac{1}{4}\overline{\mathbf{f}}_{\mathbf{v}} + \overline{\mathbf{e}}_{\mathbf{v}}, \end{aligned} \tag{4.22}$$

where $\mathbf{v}''$ represent initial control points and $\mathbf{e}'''$ and $\mathbf{f}'$ represent wavelet coefficients. The intermediate coefficients $\mathbf{v}'$ and $\mathbf{e}'$ represent points and $\mathbf{e}''$ represents vectors.

This reconstruction formula can be modified to represent sharp edges and vertices correctly. The subdivision operator $\mathbf{S}$ applies the one-dimensional subdivision rules for all $\mathbf{v}$ and $\mathbf{e}$ vertices belonging to sharp edges and it does not modify sharp vertices. To guarantee "pleasant" behavior of the slope near a sharp vertex, edges incident to this vertex are temporarily treated as sharp edges. The following modifications, with respect to equation (4.22), are necessary to represent sharp features:

- For any $\mathbf{e}$ vertex located on a sharp edge or belonging to an edge with a sharp vertex, the first and fifth rules are ignored, *i.e.*,

$$1.\quad \mathbf{e}'' \leftarrow \mathbf{e}''' \quad \text{and} \quad 5.\quad \mathbf{e} \leftarrow \mathbf{e}'.$$

- For any sharp $\mathbf{v}$ vertex, the second and sixth rules are ignored, *i.e.*,

$$2.\quad \mathbf{v}' \leftarrow \mathbf{v}'' \quad \text{and} \quad 6.\quad \mathbf{v} \leftarrow \mathbf{v}'.$$

- For any $\mathbf{v}$ vertex that has incident sharp edges and that is not sharp itself, the second and sixth rules are replaced by

$$2.\quad \mathbf{v}' \leftarrow \mathbf{v}'' - \tfrac{3}{4}\overline{\mathbf{e}''}_{\mathbf{v}} \quad \text{and} \quad 6.\quad \mathbf{v} \leftarrow \tfrac{1}{2}\mathbf{v}' + \tfrac{1}{2}\overline{\mathbf{e}}_{\mathbf{v}}.$$

  For these modified rules, the averages $\overline{\mathbf{e}''}_{\mathbf{v}}$ and $\overline{\mathbf{e}}_{\mathbf{v}}$ are computed from only those adjacent $\mathbf{e}$ vertices that are located on sharp edges.

The decomposition rules for our bicubic wavelet transform are obtained from this reconstruction formula, applying the inverse of every update operation in reverse order, with the same exceptions for vertices located on sharp features. Examples for basis functions with features are depicted in Figure 4.14.

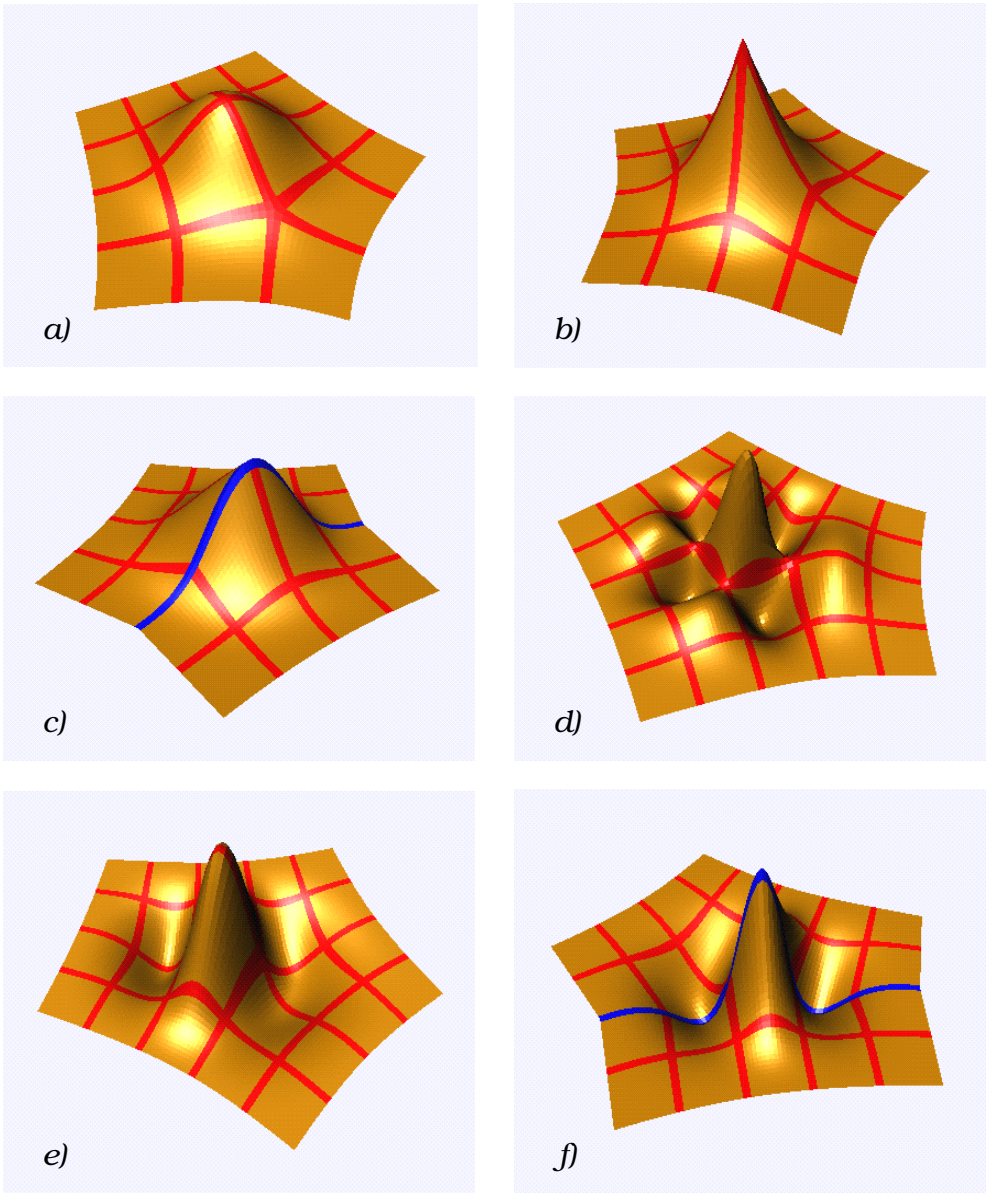

Figure 4.14: Basis fuctions for generalized bicubic B-spline wavelet transform. a) scaling function, b) scaling function located at a sharp vertex (cusp), c) scaling function located on a sharp edge (crease), d) wavelet located on a face, e) wavelet located on an edge, and f) wavelet located on a sharp edge (crease).

4.1.4 Subdivision and Fitting Properties

The four linear operators **S**, **E**, **F**, and **C** for our bicubic wavelet construction correspond to non-square matrices that are multiplied with corresponding sets of control points or wavelet coefficients. Considering that our wavelet transform has an inverse, it holds that **FS** and **CE** are identity matrices. Since the subdivision operator **S** can reproduce every bicubic polynomial on a regular (rectilinear) control mesh, the fitting operator **F** has bicubic precision, as well. This implies that all wavelet coefficients are zero when representing a bicubic polynomial. We note that **F** does not interpolate certain surface points, but that it closely approximates the finer levels of resolution. Interpolation constraints often introduce unwanted "wiggles" to curves and surfaces.

A property of the subdivision operator **S** is that it produces limit surfaces with tangent-plane continuity at extraordinary points. This can be shown by exploring a square sub-matrix of **S** transforming a local set of control points around an extraordinary point into another set of points with the same local mesh topology. Eigenanalysis of this matrix characterizes the limit behavior at the extraordinary point, *i.e.*, the surface limit and its normal can be computed analytically based on the eigenvectors. Our subdivision scheme falls into a class of generalized Catmull-Clark schemes for which the eigenvalues and eigenvectors of a local subdivision matrix were evaluated by Ball/Storry [6]. A proof of tangent-plane continuity was given by Peters/Reif [106].

Our subdivision scheme generates polynomial patches satisfying C^2-continuity constraints in all regular mesh regions. Except for extraordinary points, all surface regions become regular after a sufficient number of subdivisions. Around every extraordinary point there is an infinite number of smaller and smaller patches. However, it is possible to compute the limit-surface efficiently at arbitrary parameter values based on eigenanalysis of subdivision matrices [122].

4.1.5 Isosurface Fitting

Using the wavelet construction described previously, we can efficiently compute detail coefficients at multiple levels of surface resolution when a base mesh and control points on the finest subdivision level are given. In order for the transform to apply to an arbitrary input surface, the surface must be re-mapped to one with subdivision connectivity. In this section we describe an efficient algorithm generating base meshes for isosurfaces and recursively *shrink-wrapping* subdivided meshes towards the underlying isosurface geometries. This algorithm was developed and implemented by Duchaineau and is presented in a recent publication [13]. We use this method to construct a subdivision hierarchy for large-scale isosurfaces obtained from an extremely high-resolution turbulent-mixing hydrodynamics simulation by Mirin *et al.* [98].

Starting with an initial triangulated isosurface representation, we construct a base mesh by performing a variant on edge-collapse simplification according to Hoppe [70], followed by an additional pass removing some edges while keeping the vertices fixed. This class of simplification method works on triangulations (manifolds with boundary), and preserves the genus of the surface, resulting in polygonal meshes. We constrain the simplification to produce polygons with three-, four- and five-sided faces, and vertices of valences from three to eight. These constraints result in higher quality mappings improving compression efficiency, since very low or high degree vertices and faces may cause highly skewed or uneven parametrizations in the subsequent fitting process.

Our variant on edge-collapse simplification uses a fast priority queue implemented using bucket sorting, where within each bucket of priorities the legal edge collapses are listed in first-in-first-out order. Collapses are performed on the highest-priority legal edge, iterating until no legal collapses are possible or a target vertex count is met. We mark an edge as legal to collapse if the rules from [70] and certain additional constraints hold. We

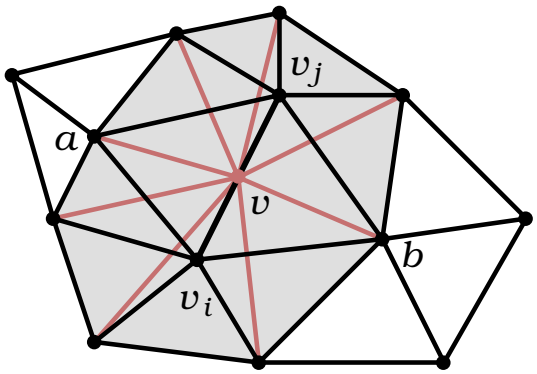

Figure 4.15: Collapsing edge $v_i v_j$ modifies the triangulation in the shaded region. The error estimates for all edges shown are updated.

ensure that valences on any new vertex are in the range $\{3, \ldots, 8\}$ (valence two is allowed for boundary vertices). The new vertex formed by the edge collapse is located at the edge center if neither edge vertex belongs to a surface boundary. Otherwise, the new vertex is located on the boundary. The general configuration is shown in Figure 4.15.

Collapsing edge $v_i v_j$ decrements the valences of vertices a and b belonging to incident triangles as shown in Figure 4.15, and replaces v_i and v_j by a single vertex v of valence $N_i + N_j - 4$, where N_i and N_j are the valences of v_i and v_j, respectively. We check the valences that a, b and v would have after collapse using these formulae to enforce the desired constraints.

The error function used in our mesh-collapse algorithm is simpler than Hoppe's and does not require evaluating shortest distances of a dense sampling of points to the finest mesh every time an edge is updated. It was demonstrated by Lindstrom/Turk [83] that memoryless simplification can provide results at least as good as the methods of Hoppe and others but without expensive metrics computed with respect to the original fine mesh. The priority computation we have developed has some similarity in spirit, but has been re-formulated in response to experience attempting to optimize the subsequent fitting process and resulting mappings. We believe it is

important to test the volume-preserving aspect of the Lindstrom-Turk edge collapses in future work, since this has the potential to reduce some artifacts encountered in especially aggressive reductions (*e.g.*, > 50 times reductions for the turbulence contours we examine).

The priority for an edge collapse is computed as follows: Let c_i be the vectors obtained by taking cross products of consecutive vectors associated with the ring of edges adjacent to the new vertex after collapse. Then $n_i = c_i/\|c_i\|$ are the unit normals to the triangles forming a ring around the new vertex. The unit average normal is $n/\|n\|$, where $n = 1/k \sum_{i=0}^{k-1} n_i$. The error introduced by an edge collapse is estimated by

$$\Delta E_0 \;=\; \max_{i=0,\ldots,k-1} \left| \frac{d}{2} \cdot n_i \right|, \tag{4.23}$$

where $d = v_j - v_i$. To allow priority distinctions in "skewed/tangled" planar neighborhoods, we clamp ΔE_0 to a small positive value by $\Delta E_1 = \max\{\Delta E_0, 10^{-8}\}$. Errors in higher-curvature or tangled neighborhoods are emphasized by a weighting factor, providing the final delta in error energy as

$$\Delta E \;=\; \left(2 - \frac{\min c_i \cdot n}{\max \|c_i\|}\right)^{16} \Delta E_1. \tag{4.24}$$

We keep an estimate of accumulated error for every edge neighborhood, E_{acc}. The original edges are initialized with an accumulated error of zero. The total error for an edge is defined as

$$E \;=\; E_{\text{acc}} + \Delta E \tag{4.25}$$

When an edge is collapsed, the accumulated error for any edge adjacent to the new vertex is set to the maximum of its previous value E_{acc} and the values E for the five old edges destroyed by the collapse. The priority of a collapse is given by $\log(1/E)$. This value is discretized, for example, to about 10^5 buckets within a maximum expected range of $[\log 10^{-10}, \log 10^{10}]$ defining bucket indices.

Upon collapse, a neighborhood of nearby edges must have their legality markings and priorities updated. These edges are: (a) those remaining from the two triangles that were collapsed to edges and (b) the ring of edges formed between consecutive outer endpoints of the edges in (a). The complete stencil of edge updates is shown in Figure 4.15.

To improve the vertex and face valences of the base mesh, we delete some edges that satisfy certain constraints, in a priority-queue order. An edge is eligible for deletion if its incident vertices both have valences at least four, and if the resulting merged face has no more than five sides. If the unit normals of the faces on either side of the edge have a dot product less that a specified threshold, for example .5, then we also make the edge ineligible for removal. The priority for removal is formulated to be higher when the new face has four sides, when the two faces being combined have similar normals, and when high-valence vertices (valence > 4) are on the ends. Removal priority is lower when the new face has more than four sides, when the two combining faces have disparate normals, and when the edge's vertices have valence four.

Given the initial base mesh from the edge collapse and removal procedures, a refinement fitting procedure is the final step in converting the contour surface to have subdivision-surface connectivity, a fair parametrization, and a close approximation to the original unstructured geometry. Our method is inspired by the *shrink-wrapping* algorithm by Kobbelt *et al.* [75], which models an equilibrium between *attracting forces* pulling control points towards a surface and *relaxing forces* minimizing parametric distortion. We iterate the attraction/relaxation phases a few times at a given resolution, then refine using Catmull-Clark subdivision, repeating until a desired accuracy or resolution is attained. Relaxation is provided by a simple Laplacian averaging procedure, where every vertex is replaced by the average of its old position and the centroid of its neighbor vertices. Boundary vertices can be relaxed only along the boundary. The remainder of this section describes the

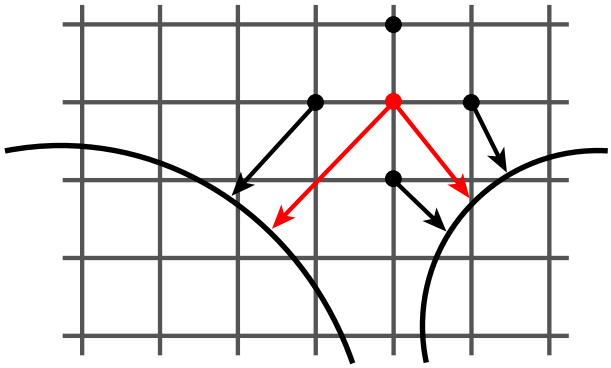

Figure 4.16: Updating closest isosurface points for constructing the signed-distance function.

attraction method.

For attraction, vertices are moved to the actual isosurface along a line defined by the unit average normal of the faces adjacent to the vertex in the current shrink-wrap mesh. The location chosen along this line is determined by use of a signed-distance field for the original contour surface. We choose the normal direction of the current mesh to ensure that samples spread evenly over the surface, especially for high-curvature features. The even spread is facilitated by the mesh relaxation procedure. The signed-distance field is used to help locate the best attraction point because it is a reliable indicator of which way to move and how far, and it can help disambiguate between near isosurface locations by selecting the one facing a direction that most agrees with the mesh normal. The scalar field itself, while available at no additional space or time cost, is generally less reliable due to the possibility of steep or zero gradients. We move to the nearest isosurface along the mesh-normal line that has a contour normal facing in the same direction (the dot product of the two normals is positive). If the distance to this location is greater that a specified threshold, then the point remains at its current position until further iterations/refinements provide a sensible target location.

We briefly outline the algorithm we use for computing the signed-distance

field for the contour. If the scalar field is defined on a regular hexahedral grid, we use the same grid for the signed-distance function. The sign for our distance function can be obtained from the underlying scalar field while estimating the distance to the isosurface involves more work. Our algorithm creates a breadth-first queue of "updated" nodes in the grid, initialized to include the grid nodes for cells containing isosurface, using the isolevel, scalar value and gradient to estimate these initial distances. Each queue entry contains the node index and coordinates of the closest surface point found so far for that node. The first entry on the queue is removed, and all its neighbors are checked to see whether they need to be updated. A neighbor is updated if the removed node's closest point is closer that its closest point, see Figure 4.16. Updating involves replacing the coordinates of the closest point, placing the neighbor at the end of the queue, and storing the new distance in the distance field for the neighbor's node. The queue processing continues until the queue is empty. Typically, every node gets updated only a few times, resulting in an expected linear-time computation of the signed-distance field.

We note that an isosurface of a signed-distance function will have slight differences from the original extracted isosurface, hence the fitting process will converge to a slightly different surface than may be desired. This can be optionally corrected after fitting, by moving the vertices in the scalar-gradient direction to the exact isosurface. This is possible after the fitting process because the points are within a fraction of a cell width from the exact surface and the scalar field is reliable when in that proximity.

4.1.6 Numerical Examples

To demonstrate the performance of our algorithm, we have extracted an isosurface from a block of a high-resolution turbulent-mixing hydrodynamics simulation [98], converted it into our surface representation and displayed

No. of coefficients	Percent of full resolution	L^2-error [%]
237490	20.0	7.6
118728	10.0	20.5
59364	5.0	41.7
19527	1.6	71.3

Table 4.1: Approximation errors in percent of edge length of volume cell for reconstructions from subsets of coefficients.

different levels of resolution. Starting with a block of $256 \times 256 \times 384$ samples, we have constructed an isosurface mesh composed of 976321 vertices. This mesh has been simplified to a base mesh with only 19527 vertices. We have used three subdivision steps for the shrink-wrapping process and obtained a fine-resolution mesh composed of 1187277 vertices, which corresponds to the total number of control points and wavelet coefficients. We obtained computation times of 12 minutes for base mesh generation, about one minute for the shrink-wrapping step, and 30 seconds for computing the wavelet transform on a 250 MHz MIPS R10000 processor.

Assuming that the control points of the shrink-wrapped mesh interpolate the isosurface, we can estimate an approximation error for a mesh reconstructed from only a subset of coefficients by using differences between control points at finest resolution. Error estimates are shown in Table 4.1. The errors are computed in percent of the edge length of one volume cell. The main diagonal of the entire block is about 528 edge lengths. The coarsest resolution possible can be obtained by reconstruction from the base mesh, which corresponds to 1.6 percent of the full resolution. Every wavelet coefficient is a vector-valued quantity with three components.

Figures 4.17–4.20 show the base mesh with interpolating control points and different levels of resolution from local points of view. All figures are rendered at a mesh resolution corresponding to three subdivision levels (same resolution as obtained from shrink-wrapping), using flat shading.

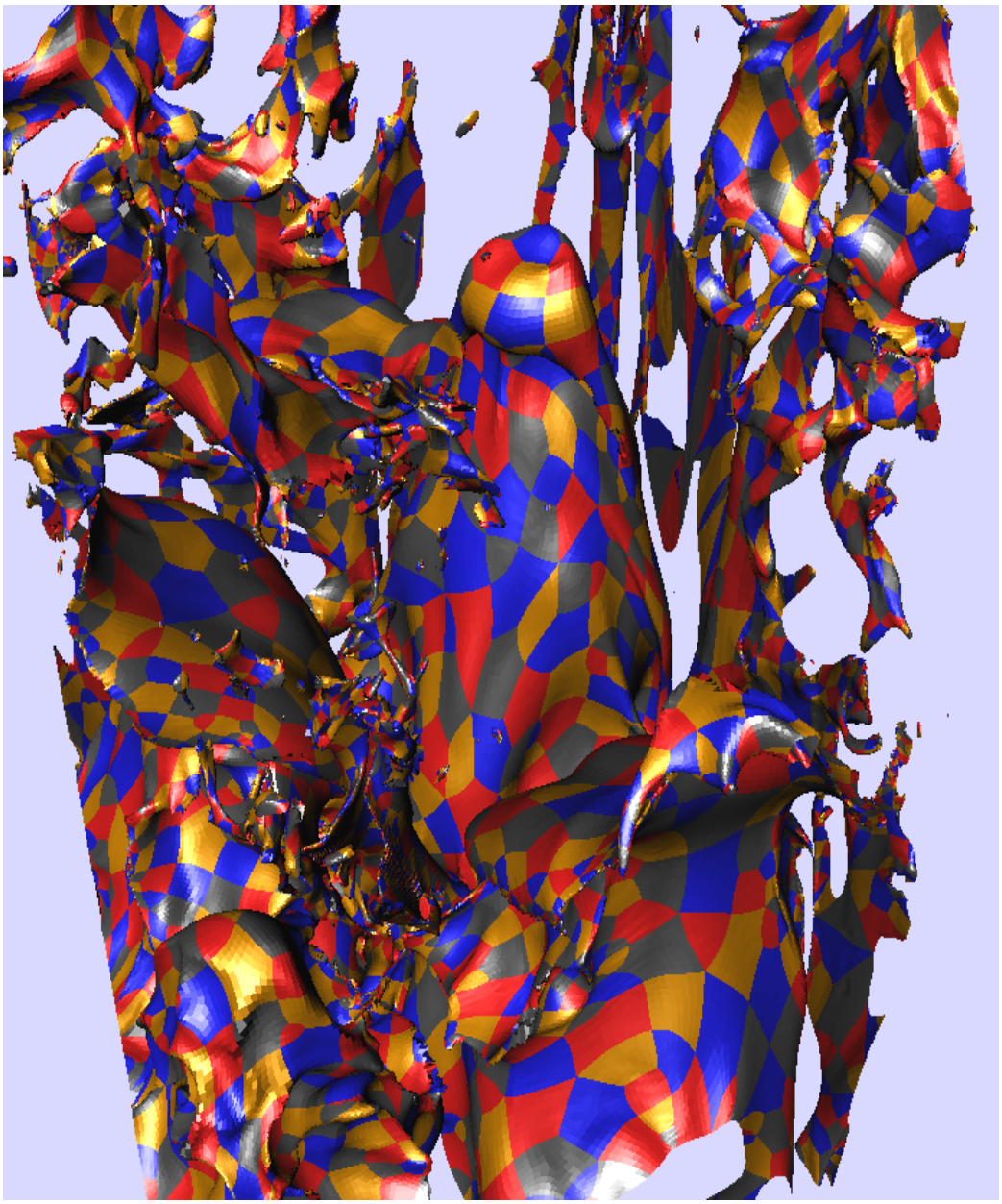

Figure 4.17: Isosurface at full resolution (1187277 control points) with patches corresponding to base-mesh polygons shown in four different colors.

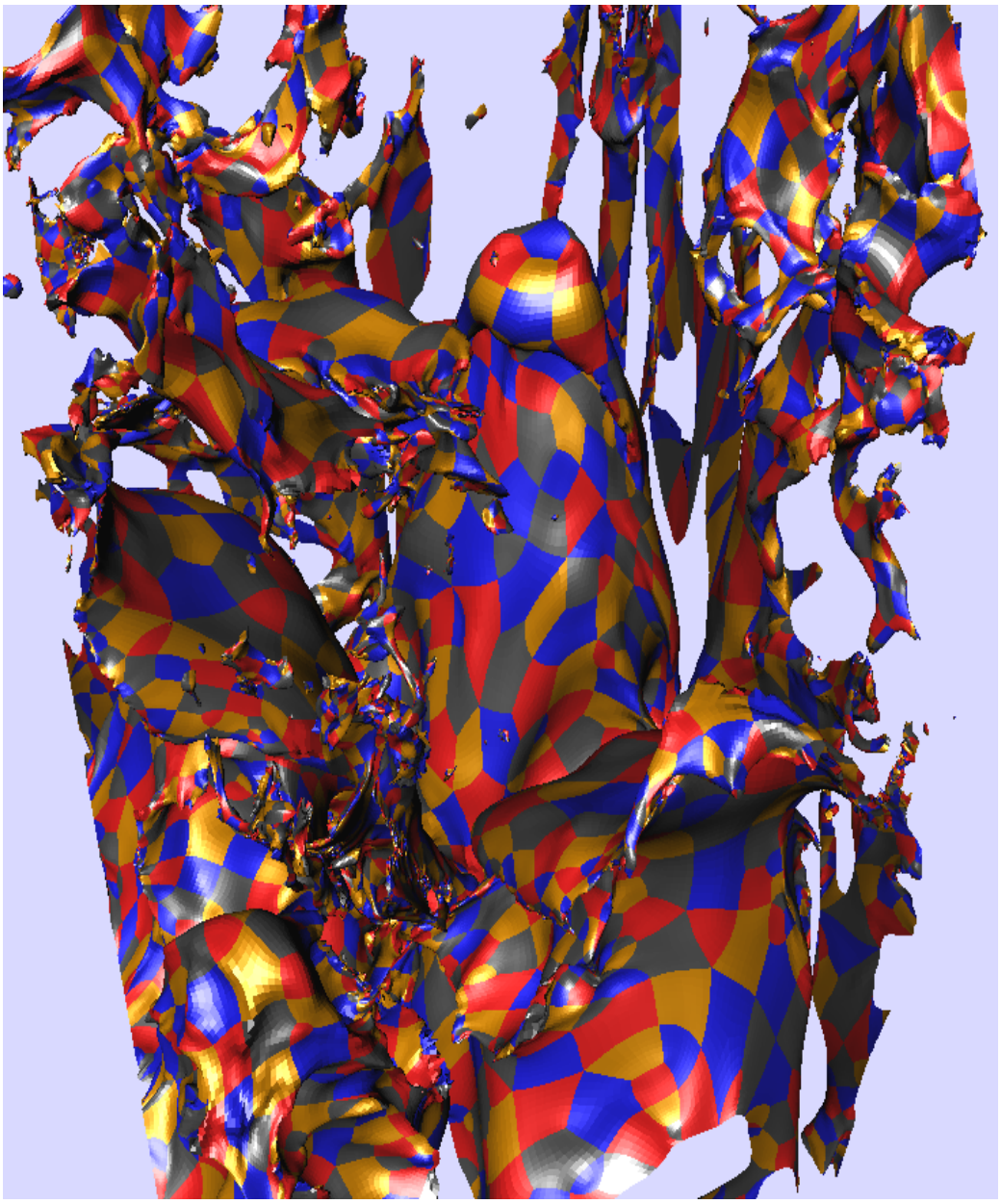

Figure 4.18: Isosurface reconstructed from 5 percent of wavelet coefficients, selected by thresholding.

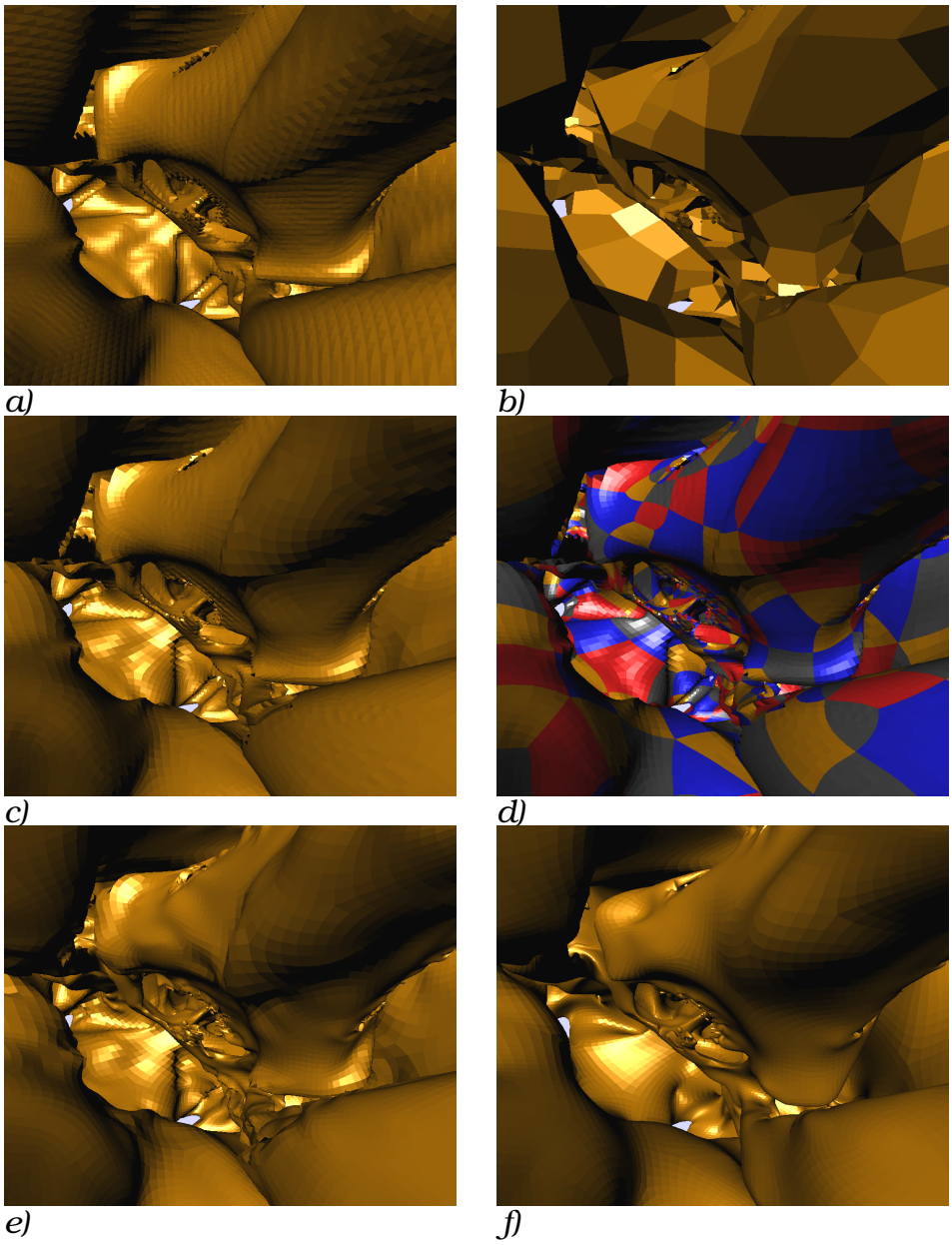

Figure 4.19: Local view of isosurface. a) initial triangulation, b) base mesh, c) full resolution, d) full resolution with colored patches, e) reconstruction from 5 percent of coefficients, and f) reconstruction from base mesh (1.6 percent).

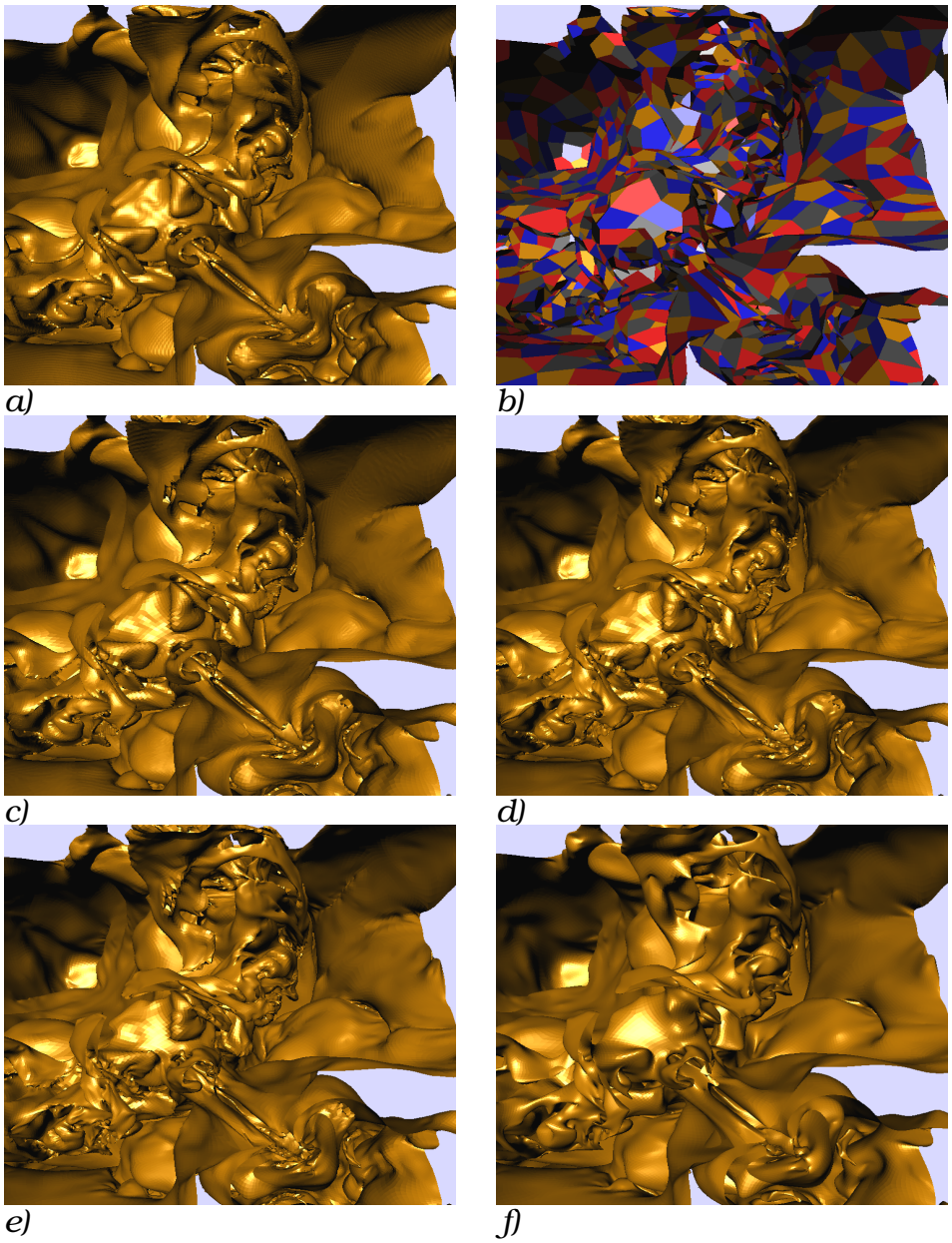

Figure 4.20: Bottom view of isosurface. a) initial triangulation, b) base mesh, c) full resolution, d) reconstruction from 10 percent of coefficients, e) 5 percent, and f) 1.6 percent.

4.2 Wavelets on Tessellations

Besides multiresolution representation of arbitrary two-manifolds, an interesting problem is to represent functions defined on planar tessellations. Both approaches are based on local parametrizations defined by polygonal meshes. In the case of two-manifolds, this parametrization is not part of the geometry. In the case of planar tessellations, however, two coordinates of a geometric model are parameters and the model itself is a graph surface. The planar mesh defining a parametrization and its associated function values are not necessarily refined by the same subdivision rules. Our primary application is image compression, applied to sets of polygonal image regions.

It was shown by Stam that subdivision surfaces can be evaluated analytically at arbitrary parameter values [122]. Thus, they can be used as a tool for constructing continuous basis functions on irregular domains like tessellations. Most subdivision schemes that offer tangent plane continuity, however, assume that the parametric domain is deformed by the same subdivision rules. For evaluation of the corresponding basis functions at global parameter values, this deformation needs to be inverted, which, in general, cannot be done in a closed form. Concerning simpler subdivision schemes that generate C^0-continuous surfaces, like bilinear splines, it is straight-forward to construct global closed-form parametrizations.

We use bilinear subdivision to define parametrizations for tessellations consisting of convex polygonal regions (tiles). The shape of these tiles is preserved during the bilinear subdivision process applied to a tessellation. The tiles would be deformed when using, for example, bicubic subdivision. Due to this piecewise bilinear parametrization, smooth subdivision schemes applied to the associated function values may produce creases along the edges of an initial tessellation. Thus, we use our generalized bilinear wavelet transform to represent functions defined on tessellations.

4.2.1 Parametrization

We define a parametrization for tessellations composed of convex polygons using bilinear subdivision. Such a tessellation is defined by sets of vertices V, edges E, and convex, non-overlapping polygons (faces) F that completely cover a given domain I. It is assumed that the *valence* of every vertex is at least three, except for boundary vertices. The edges are line segments connecting two vertices that encompass all convex combinations of these, except for the vertices themselves. The faces are the remaining open regions enclosed by vertices and edges. It is assumed that every angle between two incident edges enclosing a face is strictly less than 180 degrees, *i.e.*, all faces must be convex and *T-nodes* (or hanging nodes) are not allowed.

We consider an initial tessellation

$$T_0 \;=\; \{V_0, E_0, F_0\}. \tag{4.26}$$

We define recursive refinement rules for a sequence of tessellations $T_1, T_2, \ldots$ The set V_{i+1} is composed of V_i, the midpoints of all edges in E_i, and the centroids of all faces in F_i. The set E_{i+1} is composed of four edges incident to every midpoint of an edge in E_i connecting this midpoint to the two adjacent vertices in V_i and the centroids of the two adjacent faces in F_i. The midpoints of boundary edges have only three incident edges, since they have only one adjacent face in F_i. The set F_{i+1} contains the remaining open surface regions in the plane. All polygons in F_{i+1} are convex quadrilaterals, provided that all polygons in F_i are convex, see Figure 4.21.

We define a local parametrization for the quadrilaterals in F_1. This parametrization can be used for all subsequent levels of subdivision, since the quadrilaterals are uniformly subdivided. For every face $f_k \in F_1$, defined by points $\mathbf{p}_{k,00}, \mathbf{p}_{k,10}, \mathbf{p}_{k,11}, \mathbf{p}_{k,01} \in V_1$ (counterclockwise order), we define local coordinates $u, v \in [0, 1]$ so that every point $\mathbf{p}$ in the closure of f_k can be

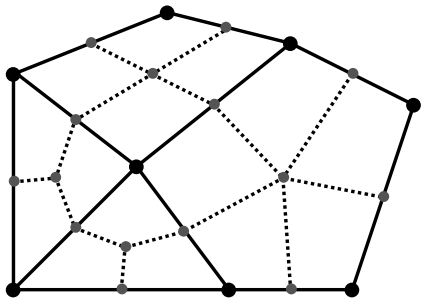

Figure 4.21: Subdivision process applied to a tessellation.

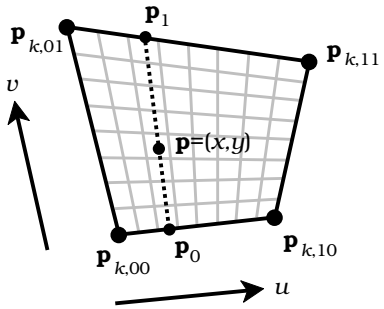

Figure 4.22: Local parameters u and v for point $\mathbf{p}$ (bilinear interpolation).

written as the result of bilinear interpolation:

$$\begin{aligned}\mathbf{p} \;&= (1-u)(1-v)\mathbf{p}_{k,00} \;+\; u(1-v)\mathbf{p}_{k,10} \\ &+\; (1-u)v\mathbf{p}_{k,01} \;+\; uv\mathbf{p}_{k,11}.\end{aligned} \tag{4.27}$$

It is more difficult to compute the local coordinates u and v for a point $\mathbf{p} = (x, y)$ in the domain I. We compute these local coordinates in this way: first, the face index k needs to be determined (k is not uniquely defined when $\mathbf{p}$ lies on an edge or coincides with a vertex); second, we compute u from the z-component of

$$\begin{aligned}(\mathbf{p} - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_0) \;&=\; 0, \quad \text{where} \\ \mathbf{p}_0 \;&=\; (1-u)\mathbf{p}_{k,00} \;+\; u\mathbf{p}_{k,10} \quad \text{and} \\ \mathbf{p}_1 \;&=\; (1-u)\mathbf{p}_{k,01} \;+\; u\mathbf{p}_{k,11}.\end{aligned} \tag{4.28}$$

Equation (4.28) is a quadratic equation that has a unique solution $u \in [0,1]$, provided that the quadrilateral f_k is convex. Once u is determined, v can be computed from

$$\mathbf{p} \;=\; (1-v)\mathbf{p}_0 + v\mathbf{p}_1. \tag{4.29}$$

This process is illustrated in Figure 4.22. Computing the local parameters u and v from the global ones is expensive, due to the evaluation of a square root for solving the quadratic equation. We therefore avoid using global parameters in our algorithm. For re-sampling a function represented by samples on a grid obtained from recursive subdivision of a tessellation T_j, we suggest to use graphics hardware to render the quadrilaterals of T_1 and to use texture representing all finer subdivision levels.

4.2.2 Wavelet Transform

The input for our wavelet transform is a bilinearly blended function $f(I)$ defined by samples located at the vertices of a subdivided tessellation. The range of this function is an arbitrary vector space, whose dimension depends on the application. For example, the range of a function representing a RGB image has three dimensions. For rendering applications, additional coordinates for opacity and z-buffering would have to be added. In the case of a radiosity application, the domain I represents all surface components in a scene and f describes the local radiance emanating from them. Since our wavelet transform is applied independently, yet in the same way to all coordinates, the number of dimensions does not have an impact.

Given a hierarchy of tessellations, a wavelet transform can be defined based on the mesh hierarchy. Initially, a function is represented by a fine tessellation T_j and by a sample of the function $f(I)$ at every vertex in V_j. These samples are denoted as $\mathbf{v}$, $\mathbf{e}$, and $\mathbf{f}$ points, depending on the types of their associated vertices in V_j that correspond to vertices, edges and faces, respectively, in the next coarser tessellation T_{j-1}. A decomposition step

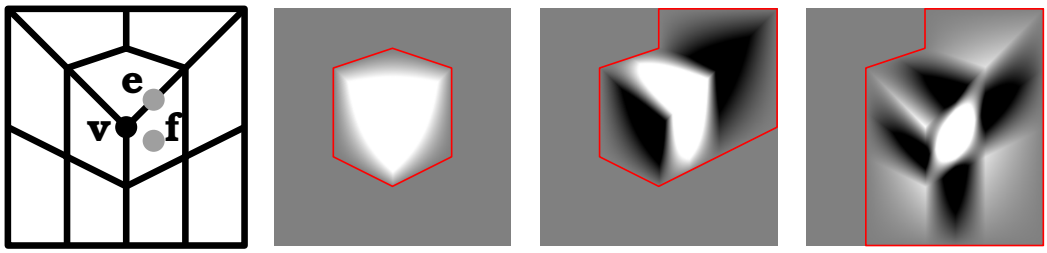

Figure 4.23: Basis functions near an *extraordinary vertex* of valence three. Scaling function (top right), **e** wavelet (bottom left), and **f** wavelet (bottom right). Dark regions correspond to negative and bright regions to positive function values.

transforms $\mathbf{v}$ points into control points $\mathbf{v}'$ of the next coarser level, $j-1$, and the **f** and **e** points become wavelet coefficients representing the missing detail. Using only the $\mathbf{v}'$ points as samples associated with the vertices in V_{j-1}, decomposition is recursively applied to all levels until base level $j = 0$ is reached. We use the decomposition and reconstruction rules for our generalized bilinear B-spline wavelet transform, as defined by equations (4.14) and (4.15). Basis functions defined on a tessellation are depicted in Figure 4.23.

To represent boundary edges properly and to model discontinuities along certain edges within a tessellation – which requires a double set of coefficients along these discontinuities – all vertices located on boundary edges are transformed by the one-dimensional decomposition and reconstruction rules, instead. Corner vertices of the tessellation are not modified by the subdivision rules.

The computation time for one decomposition step is linear in the number of transformed vertices. Since only one quarter of these vertices is transformed again on the next coarser level, the computation time for the wavelet transform, starting with n samples, is $O(n + \frac{1}{4}n + \frac{1}{16}n + \cdots) = O(n)$. Applications for our wavelet transform are outlined in the next section.

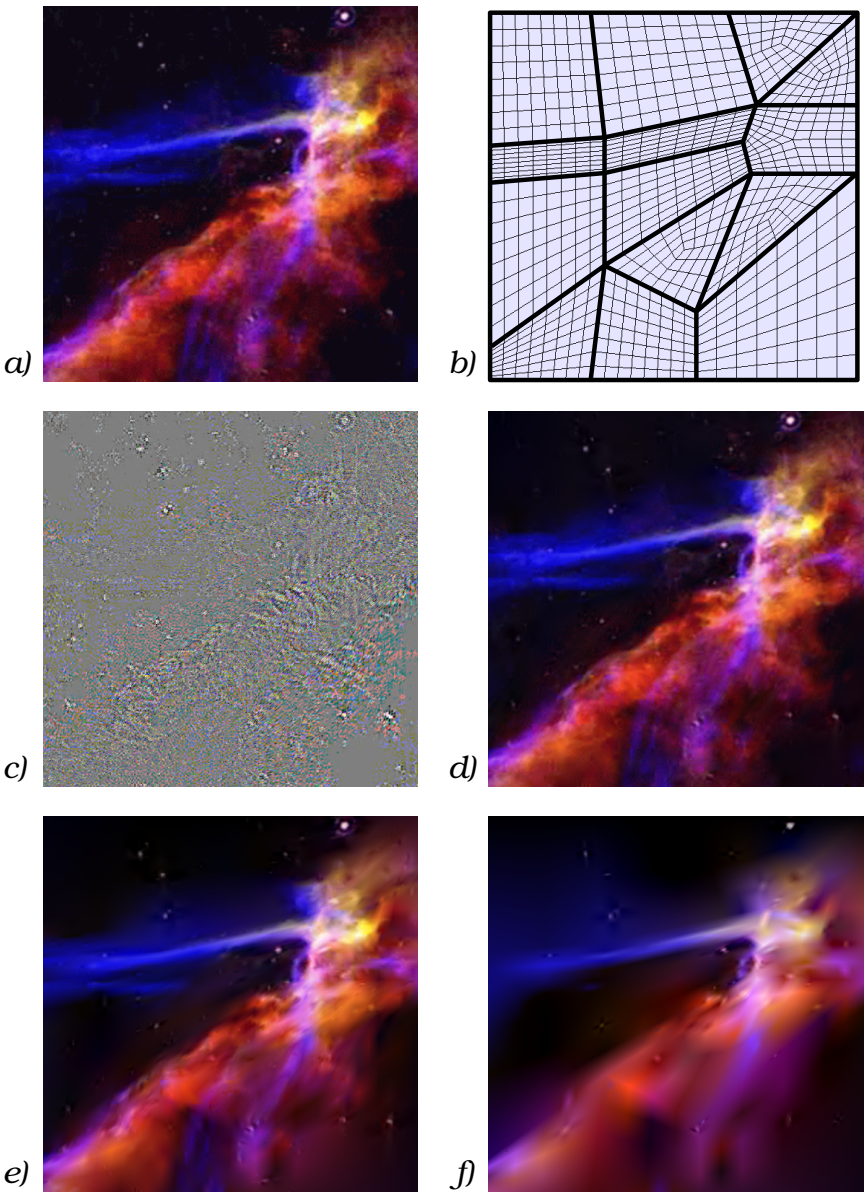

Figure 4.24: "Cygnus Loop" image. a) original, re-sampled image on tessellation T_7 (217921 samples), b) tessellation T_0 and three levels of subdivision, c) bilinear wavelet transform, d) reconstruction from 10 percent of wavelet coefficients, e) reconstruction from 1 percent, and f) reconstruction from 0.1 percent.

4.2.3 Results and Applications

To demonstrate that our wavelet transform works correctly, we transformed the "Cygnus Loop" Hubble telescope image, courtesy of NASA. The image was re-sampled using a tessellation T_0 composed of 20 vertices and 14 faces. The finest tessellation T_7 has 217921 vertices, which is about 3.5 times the number of pixels of the original image (249×251). Figure 4.24 shows the re-sampled image, the tessellation, our wavelet transform, and reconstructions using 10, 1, and 0.1 percent of the wavelet coefficients. Since the coefficients are RGB values, we use the Euclidean length for thresholding. For progressive transmission purposes, the coefficients can be sorted by decreasing absolute values in expected linear time by using a hash table. For lossless compression of a re-sampled image, the wavelet transform can be implemented in integer arithmetic. The integer-valued coefficients have low expected modulus and are compressed by arithmetic coding [99]. Some of the most important applications of our wavelet transform are:

- *Video compression.* Video data can be compressed in a straight-forward way by applying a trivariate tensor product wavelet transform. Correlation in time, however, can be exploited much better when applying a wavelet transform defined on a grid that is deformed or moves with the objects. Our wavelet transform has therefore a great potential to improve compression rates for moving images.

- *Image Morphing.* Morphing algorithms deform objects and compute the intermediate image regions by blending two images. This blending operation can be improved by performing it in the range of our wavelet transform. This treatment would blend the individual frequency bands rather than pixel values resulting in a much more realistic image. Our wavelet transform can be applied to polygonal regions that are deformed simultaneously with the blending process.

- *Radiosity.* Occlusion of objects causes radiance functions to be discontinuous on smooth surfaces [85]. These discontinuities define tessellations on the surface regions that provide a natural parametrization for the radiance function. Since our wavelets have two vanishing moments (in local parameters, except at extraordinary points), they form an ideal basis for efficient and stable integration of radiosity kernels.

- *Scattered Data Approximation.* Starting with a tessellation of selected scattered data points, our efficient wavelet transform can be used for further regular refinement and finer-detail approximation. Our wavelet basis functions are piecewise bilinear in local parameters and can be computed analytically for fitting purposes. To construct initial tessellations at certain levels of resolution, we can use, for example, the hierarchical Voronoi diagram, defined in Chapter 2.

4.3 Conclusions and Future Work

We have presented a technique for multiresolution analysis and representation of arbitrary two-manifolds, of functions defined on two-manifolds and on planar tessellations. The corresponding decomposition and reconstruction filters have finite width filters and are efficiently computed based on local lifting operations. Our surface representation is piecewise polynomial, C^0-continuous in the bilinear case and C^2-continuous in the bicubic case (except at extraordinary points, where C^1 continuity is satisfied).

Our wavelet transforms have a wide range of applications that still need to be explored. Future work will be directed at constructing wavelets and subdivision surfaces on planar tessellations with C^1 continuity and local parametrizations that can be evaluated in a closed form. We also plan to use our lifted wavelet construction for domains defined by three-dimensional lattices of arbitrary topology. Subdivision schemes similar to Catmull-Clark

volumes [92] can be complemented by a wavelet construction.

Many other research issues related to subdivision surfaces and wavelets remain, for example: How should one change a uniform parametrization to a chordal one to improve data approximation? Knot intervals can be associated with edges in irregular base meshes, analogously to the knot vectors for non-uniform B-splines [118]. It might be feasible to construct wavelets for non-uniform subdivision schemes with knot intervals as additional modeling parameters.

Our Wavelets on two-manifolds can be used as an underlying data representation for sculpting or solid modeling operations. Since feature lines are represented by using modified subdivision rules, all kinds of solids can be defined by a single subdivision surface without the need of intersecting and trimming different patches. Combined with the efficiency and compactness of wavelet representations, subdivision schemes become a powerful tool for multiresolution surface editing in CAGD.

Appendix

Least-squares Fitting

In the following, we briefly review least-squares approximation, see [16]. Given a set of n points $\mathbf{p}_i$ with associated function values f_i and m basis functions $B_j(\mathbf{x})$, least-squares approximation determines a coefficient vector $\mathbf{c} = (c_1 \cdots c_m)^T$ for an approximating function

$$f(\mathbf{x}) \;=\; \sum_{j=1}^{m} c_j \; B_j(\mathbf{x}) \tag{A.30}$$

such that the residual

$$r(\mathbf{c}) \;=\; \sum_{i=1}^{n} \Big(f(\mathbf{p}_i) - f_i \Big)^2, \quad n \geq m, \tag{A.31}$$

is minimized. A set of necessary conditions for the minimum is given by the equations

$$\frac{\partial r}{\partial c_j} \;=\; 0, \quad j = 1, \ldots, m. \tag{A.32}$$

By inserting (A.30) into (A.31) one obtains a linear system of equations, given by

$$\sum_{k=1}^{n} \sum_{i=1}^{m} c_i \; b_{ik} \; b_{jk} \;=\; \sum_{k=1}^{n} f_k \; b_{jk}, \quad b_{ij} = B_i(\mathbf{x}_j), \tag{A.33}$$

which can be written in matrix form as

$$BB^T\mathbf{c} \;=\; B\mathbf{f}.$$

This system can be solved in $O(n)$ time, since the number of basis functions is fixed. However, it may happen that the matrix BB^T is singular, e.g., when nearly all points are colinear. In the case of a singular system, a reduced set of basis functions may solve the problem. If the system is non-singular, the global minimum for the residual is found.

Notations for L^2 and l^2

The Hilbert space $L^2(\mathbb{R})$ consists of all functions $f : \mathbb{R} \to \mathbb{C}$ for that the following norm converges [51]:

$$\|f\|_{L^2} \;=\; \sqrt{\langle f, f\rangle_{L^2}}, \quad \text{where}$$

$$\langle f, g\rangle_{L^2} = \int_{\mathbb{R}} f(x)\,\overline{g(x)}\,dx$$

defines the inner product for $L^2(\mathbb{R})$. (The bar denotes complex conjugation.) Two functions f and g are equal in $L^2(\mathbb{R})$, if and only if

$$\|f - g\|_{L^2} \;=\; 0.$$

Hence, two functions that differ at most for a discrete set of values are still considered equal in $L^2(\mathbb{R})$. Two functions f and g are *orthogonal*, *i.e.*, $f \perp g$, if and only if

$$\langle f, g\rangle_{L^2} \;=\; 0.$$

Hilbert spaces $L^2(\mathbb{R}^n)$ for functions $f : \mathbb{R}^n \to \mathbb{C}$ are defined analogously by using the above inner product modified by integration over the domain

$\mathbb{R}^n$. An inner product can alternatively be defined as an integral with a non-negative weight function, e.g.,

$$\langle f, g \rangle_{L^2(\mathbb{R}, w(x)dx)} \;=\; \int_{\mathbb{R}} f(x)\; \overline{g(x)}\; w(x)\; dx, \quad w(x) > 0\; \forall x \in \mathbb{R}.$$

In the discrete case, one can define corresponding Hilbert spaces for sequences $f : \mathbb{Z} \to \mathbb{C}$, e.g., l^2 has the inner product

$$\langle f, g \rangle_{l^2} \;=\; \sum_{i \in \mathbb{Z}} f_i\; \overline{g_i}.$$

A linear, surjective map $T : H_1 \to H_2$ between two Hilbert spaces is called *isometry*, if it satisfies the identity

$$\|f\|_{H_1} \;=\; \|T(f)\|_{H_2}.$$

Any isometry has an inverse $T^{-1} = T^*$, with the property

$$\langle f, T^*(g) \rangle_{H_1} \;=\; \langle T(f), g \rangle_{H_2}.$$

Sponsors and Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. In particular, the author was recently funded through a Student Employee Graduate Research Fellowship at Lawrence Livermore National Laboratory. This work was also supported by the National Science Foundation under contracts ACI 9624034 and ACI 9983641 (CAREER Awards), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of ALSTOM Schilling Robotics, Chevron, Silicon Graphics, Inc. and ST Microelectronics, Inc.

Bibliography

[1] N. Amenta, M. Bern, and M. Kamvysselis, *A new Voronoi-based surface reconstruction algorithm*, Computer Graphics, Proceedings of Siggraph '98, ACM, 1998, pp. 415–421.

[2] N. Amenta and M. Bern, *Surface reconstruction by Voronoi filtering*, Discrete & Computational Geometry, Vol. 22, No. 4, Springer-Verlag, Dec. 1999, pp. 481–504.

[3] R. Aravind, G.L. Cash, J.P. Worth, *On implementing the JPEG still-picture compression algorithm*, Proceedings of the SPIE - The International Society for Optical Engineering, Vol. 1199, part 2, 1989, pp. 799–808.

[4] E.F. d'Azevedo, *Optimal triangular mesh generation by coordinate transformation*, SIAM Journal on Scientific and Statistical Computing, Vol. 12, No. 4, 1991, pp. 755–786.

[5] C.L. Bajaj, F. Bernardini, and G. Xu, *Reconstructing surfaces and functions on surfaces from unorganized three-dimensional data*, Algorithmica, Vol. 19, No. 1–2, Springer-Verlag, Sept.-Oct. 1997, pp. 243–61.

[6] A.A. Ball, D.J.T. Storry, *Conditions for tangent plane continuity over recursively generated B-spline surfaces*, ACM Transactions on Graphics, Vol. 7, No. 2, April 1988, pp. 83–102.

[7] J.C. Barnes, B. Hamann, and K.I. Joy, *An edge-preserving, data-dependent triangulation scheme for hierarchical rendering*, H. Hagen, G. Nielson, and F. Post, eds., Proceedings of Scientific Visualization–Dagstuhl '97, IEEE, 1999, pp. 1-10.

[8] F. Bernardini, C.L. Bajaj, J. Chen, and D.R. Schikore, *Automatic reconstruction of 3D CAD models from digital scans*, International Journal of Computational Geometry & Applications, Vol. 9, No. 4–5, World Scientific, Aug.-Oct. 1999, pp. 327–369.

[9] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, *The ball-pivoting algorithm for surface reconstruction*, manuscript, to be published.

[10] M. Bertram, J.C. Barnes, B. Hamann, K.I. Joy, H. Pottmann, and D. Wushour, *Piecewise optimal triangulation for the approximation of scattered data in the plane*, Computer-Aided Geometric Design, Elsevier, to appear, 2000.

[11] M. Bertram, M.A. Duchaineau, B. Hamann, and K.I. Joy, *Generalized B-spline subdivision surface wavelets and lossless compression*, Computer-Aided Geometric Design, Elsevier, submitted, 2000.

[12] M. Bertram, M.A. Duchaineau, B. Hamann, and K.I. Joy, *Wavelets on planar tesselations*, Proceedings of the International Conference on Imaging Science, Systems, and Technology (CISST 2000), Las Vegas, Nevada, CSREA Press, 2000, pp. 619–625.

[13] M. Bertram, M.A. Duchaineau, B. Hamann, and K.I. Joy, *Bicubic lifted subdivision-surface wavelets for large-scale isosurface representation and visualization*, Proceedings of Visualization 2000, Salt Lake City, Utah, IEEE, Oct. 2000, to appear.

[14] H. Biermann, A. Levin, and D. Zorin, *Piecewise smooth subdivision surfaces with normal control*, Computer Graphics, Proceedings of Siggraph 2000, ACM, to appear, 2000.

[15] W. Boehm, *Triangular spline algorithms*, Computer Aided Geometric Design, Vol. 2, 1985, pp. 61–67.

[16] W. Boehm and H. Prautzsch, *Geometric Concepts for Geometric Design*, A.K. Peters, Ltd., Wellesley, Massachusetts, 1994.

[17] G.-P. Bonneau, *Multiresolution analysis on irregular surface meshes*, IEEE Transactions on Visualization and Computer Graphics (TVCG), Vol. 4, No. 4, IEEE, Oct.-Dec. 1998, pp. 365–378.

[18] G.-P. Bonneau, *Optimal triangular Haar bases for spherical data*, Proceedings of Visualization '99, IEEE, 1999, pp. 279–284 & 534.

[19] J.L. Brown, *Vertex based data dependent triangulations*, Computer Aided Geometric Design, Vol. 8, No. 3, 1991. pp. 239–251.

[20] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, *Wavelet transforms that map integers to integers*, Applied and Computational Harmonic Analysis, Vol. 5, No. 3, Academic Press, July 1998, pp. 332–369.

[21] J.M. Carnicer, W. Dahmen, and J.M. Pena, *Local decomposition of refinable spaces and wavelets*, Applied and Computational Harmonic Analysis, Vol. 3, No. 2, Academic Press, April 1996, pp. 127–153.

[22] E. Catmull and J. Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes*, Computer Aided Design, Vol. 10, No. 6, 1978, pp. 350–355.

[23] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle, *Interactive multiresolution surface viewing*, Computer Graphics, Proceedings of Siggraph '96, ACM, 1996, pp. 91–97.

[24] P. Christensen, E. Stollnitz, D. Salesin, T. DeRose, *Wavelet radiance*, G. Sakas, P. Shirley, S. Müller, eds., Proceedings of the Fifth Eurographics Workshop on Photorealistic Rendering Techniques, Springer-Verlag, Berlin, Germany, 1995, pp. 295–309 & 432.

[25] C.K. Chui, *An Introduction to Wavelets*, Academic Press, San Diego, California, 1992.

[26] B. Curless and M. Levoy, *A volumetric method for building complex models from range images*, Computer Graphics, Proceedings of Siggraph '96, ACM, 1996, pp. 303–312.

[27] W. Dahmen, C.A. Micchelli, and H.-P. Seidel, Blossoming begets B-spline bases built better by B-patches, Mathematics of Computation, Vol. 59, No. 199, American Mathematical Society, 1992, pp. 97–115.

[28] W. Dahmen, *Decomposition of refinable spaces and applications to operator equations*, Numerical Algorithms, Vol. 5, 1993, pp. 229–245.

[29] W. Dahmen, *Wavelet and multiscale methods for operator equations*, Acta Numerica, Vol.6, Cambridge University Press, 1997, pp. 55–228.

[30] W. Dahmen, *Wavelet methods for PDEs – some recent developments*, IGPM report No. 183, RWTH Aachen, Germany, Dec. 1999.

[31] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992.

[32] I. Daubechies and W. Sweldens, *Factoring wavelet transforms into lifting steps*, Journal of Fourier Analysis and Applications, Vol. 4, No. 3, CRC Press Inc., 1998, pp. 245–267.

[33] I. Daubechies, I. Guskov, P. Schröder, and W. Sweldens, *Wavelets on irregular point sets*, to be published.
`http://cm.bell-labs.com/who/wim/index.html`

[34] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin, Germany, 1997.

[35] C. de Boor, *On calculating with B-splines*, Journal of Approximation Theory, Vol. 6, No. 1, 1972, pp. 50–62.

[36] L. De Floriani, B. Falcidieno, and C. Pienovi, *A Delaunay-based method for surface approximation*, Proceedings of Eurographics '83, Amsterdam, Netherlands, 1983, pp. 333–350, 401.

[37] L. De Floriani, B. Falcidieno, G. Nagy, and C. Pienovi, *A hierarchical structure for surface approximation*, Computers & Graphics, Vol. 8, No. 2, 1984. pp. 183–193.

[38] L. De Floriani and E. Puppo, *Constrained Delaunay triangulation for multiresolution surface description*, Proceesings Ninth IEEE International Conference on Pattern Recognition, IEEE, 1988, pp. 566–569.

[39] T. DeRose, M. Kass, and T. Truong, *Subdivision surfaces in character animation*, Computer Graphics, Proceedings of SIGGRAPH '98, ACM, 1998, pp. 85–94.

[40] D. Doo and M. Sabin, *Behaviour of recursive division surfaces near extraordinary points*, Computer Aided Design, Vol. 10, No. 6, 1978, pp. 356–360.

[41] M. Duchaineau, *Dyadic splines*, Ph.D. thesis, Department of Computer Science, University of California, Davis, 1996.
`http://graphics.cs.ucdavis.edu/~duchaine/dyadic.html`

[42] M. Duchaineau, M. Wolinsky, D.E. Sigeti, M.C. Miller, C. Aldrich, and M.B. Mineev-Weinstein, *ROAMing terrain: real-time optimally adapting meshes*, Proceedings of Visualization '97, IEEE, 1997, pp. 81–88, 585.

[43] N. Dyn, D. Levin, and S. Rippa, *Algorithms for the construction of data dependent triangulations*, J.C. Mason, and M.G. Cox, eds., Algorithms for Approximation II, Chapman and Hall, New York, 1990, pp. 185–192.

[44] N. Dyn, D. Levin, and J.A. Gregory, *A butterfly subdivision scheme for surface interpolation with tension control*, ACM Transactions on Graphics, Vol. 9, No. 2, April 1990, pp. 160–169.

[45] M. Eck and H. Hoppe, *Automatic reconstruction of B-spline surfaces of arbitrary topological type*, Computer Graphics, Proceedings of Siggraph '96, ACM, 1996, pp. 325–334.

[46] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, *Multiresolution analysis of arbitrary meshes*, Computer Graphics, Proceedings of Siggraph 95, ACM, 1995, pp. 173–182.

[47] H. Edelsbrunner and E.P. Mücke, *Three-dimensional alpha shapes*, ACM Transactions on Graphics, Vol. 13, No. 1, 1994. pp. 43–72.

[48] G. Farin, *Surfaces over Dirichlet tessellations*, Computer Aided Geometric Design, Vol. 7, No. 1–4, 1990, pp 281–292.

[49] G. Farin, *Curves and Surfaces for CAGD*, Fourth edition, Academic press, San Diego, California, 1997.

[50] A. Finkelstein and D.H. Salesin, *Multiresolution curves*, Computer Graphics, Proceedings of Siggraph '94, ACM, 1994, pp. 261–268.

[51] O. Forster, *Analysis 3* (in German), Viehweg Verlag, Braunschweig, Germany, 1984.

[52] T.S. Gieng, B. Hamann, K.I. Joy, G. Schussman, and I.J. Trotts, *Constructing hierarchies for triangle meshes*, IEEE Transactions on Visualization and Computer Graphics (TVCG), Vol. 4, No. 2, 1998, pp. 145–161.

[53] S.J. Gortler, P. Schröder, M.F. Cohen, P. Hanrahan, *Wavelet radiosity* Computer Graphics, Proceedings of Siggraph '93, ACM, 1993, pp. 221–230.

[54] C.M. Grimm and J.F. Hughes, *Modeling surfaces of arbitrary topology using manifolds*, Computer Graphics, Proceedings of Siggraph '95, ACM, 1995, pp. 359–368.

[55] M.H. Gross, R. Gatti, and O. Staadt, *Fast multiresolution surface meshing*, Proceedings of Visualization '95, IEEE, 1995, pp 135–42, 446.

[56] B. Guo, *Surface reconstruction: from points to splines*, Computer Aided Design, Vol. 29, No. 4, Elsevier, April 1997, pp. 269–277.

[57] B. Guo, J. Menoo, and B. Willette, *Surface reconstruction using alpha shapes*, Computer Graphics Forum, Vol. 16, No. 4, Blackwell Publishers for Eurographics Association, Oct. 1997, pp. 177–190.

[58] I. Guskov, W. Sweldens, and P. Schröder, *Multiresolution signal processing for meshes*, Computer Graphics, Proceedings of Siggraph '99, ACM, 1999, pp. 325–334.

[59] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder, *Normal meshes*, Computer Graphics, Proceedings of Siggraph 2000, ACM, 2000, to appear.

[60] H. Hagen, G. Nielson, and Y. Nakajima, *Surface design using triangular patches*, Computer Aided Geometric Design, Vol. 13, No. 9, 1996, pp. 895–904.

[61] M. Halstead, M. Kass, and T. DeRose, *Efficient, fair interpolation using Catmull-Clark surfaces*, Computer Graphics, Proceedings of Siggraph '93, ACM, 1993, pp. 35–44.

[62] B. Hamann and B. Jordan, *Triangulations from repeated bisection*, M. Daehlen, T. Lyche and L.L. Schumaker, eds., Mathematical Methods

for Curves and Surfaces II, Vanderbilt University Press, Nashville, Tennessee, 1998, pp. 229–236.

[63] P.S. Heckbert, M. Garland, *Multiresolution modeling for fast rendering*, Proceedings Graphics Interface '94, Canadian Information Processing Society, 1994, pp. 43–50, 246.

[64] B. Heckel, A.E. Uva, B. Hamann, and K.I. Joy, *Surface reconstruction using adaptive clustering methods*, IEEE Transactions on Visualization and Computer Graphics, submitted, 2000.

[65] B. Heckel, A.E. Uva, and B. Hamann, *Clustering-based generation of hierarchical surface models*, Wittenbrink, C.M. and Varshney, A., eds., Late Breaking Hot Topics Proceedings, Visualization '98, IEEE, 1998, pp. 41–44.

[66] M.L. Hilton, B.D. Jawerth, A. Sengupta, *Compressing still and moving images with wavelets*, Multimedia Systems, Vol. 2, Springer, 1994, pp. 218–227.

[67] K. Hölling and H. Mögerle, *G-splines*, Computer Aided Geometric Design, Vol. 7, No. 1–4, Elsevier, 1990, pp. 197–207.

[68] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface reconstruction from unorganized points*, Computer Graphics, Proceedings of Siggraph '92, ACM, 1992, pp. 71–78.

[69] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Mesh optimization*, Computer Graphics, Proceedings of Siggraph '93, ACM, 1993, pp. 19–26.

[70] H. Hoppe, *Progressive meshes*, Computer Graphics, Proceedings of Siggraph '96, ACM, 1996, pp. 99–108

[71] H. Hoppe, *Smooth view-dependent level-of-detail control and its application to terrain rendering*, Proceedings of Visualization '98, IEEE, 1998, pp. 35–42 & 516.

[72] B. Jawerth, W. Sweldens, *An overview of wavelet based multiresolution analysis*, Technical report, Industrial Mathematics Initiative, University of South Carolina, Department of Mathematics, 1992.

[73] R. Klein, D. Cohen-Or, T. Huttner, *Incremental view-dependent multiresolution triangulation of terrain*, Proceedings of the Fifth Pacific Conference on Computer Graphics and Applications, Journal of Visualization and Computer Animation, Vol. 9, No. 3, Wiley, 1998, pp. 129–143.

[74] L. Kobbelt, *Sqrt(3)-subdivision* Computer Graphics, Proceedings of Siggraph 2000, ACM, to appear, 2000.

[75] L.P. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel, *A shrink wrapping approach to remeshing polygonal surfaces*, Proceedings of Eurographics '99, Computer Graphics Forum, Vol. 18, Blackwell Publishers, 1999, pp. 119–129.

[76] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, *Interactive multiresolution modeling on arbitrary meshes*, Computer Graphics, Proceedings of Siggraph 98, ACM, 1998, pp. 105–114.

[77] L. Kobbelt and P. Schröder, *A multiresolution framework for variational subdivision*, ACM Transactions on Graphics, Vol. 17, No. 4, Oct. 1998, pp. 209–237.

[78] L. Kobbelt, *Interpolatory subdivision on open quadrilateral nets with arbitrary topology*, Proceedings of Eurographics '96, Computer Graphics Forum Vol. 15, Blackwell Publishers, 1996, pp. 409–420.

[79] O. Kreylos, B. Hamann, *On simulated annealing and the construction of linear spline approximations for scattered data*, Proceedings of Joint Eurographics-IEEE TCCG Symposium on Visualization, Springer-Verlag, Vienna, Austria, 1999, pp. 189–198.

[80] V. Krishnamurthy and M. Levoy, *Fitting smooth surfaces to dense polygon meshes*, Computer Graphics, Proceedings of Siggraph '96, ACM, 1996, pp. 313–324.

[81] A.W.F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, *MAPS: multiresolution adaptive parameterization of surfaces*, Computer Graphics, Proceedings of SIGGRAPH '98, ACM, 1998, pp. 95–104.

[82] A. Lee, H. Moreton, and H. Hoppe, *Displaced subdivision surfaces*, Computer Graphics, Proceedings of SIGGRAPH 2000, ACM, 2000, to appear.

[83] P. Lindstrom, G. Turk, *Fast and memory efficient polygonal simplification*, Proceedings of Visualization '98, IEEE, 1998, pp. 279–286 & 544.

[84] P. Lindstrom, G. Turk, *Evaluation of memoryless simplification*, IEEE Transactions on Visualization and Computer Graphics (TVCG), Vol. 5, No. 2, 1999.

[85] D. Lischinski, F. Tampieri, and D.P. Greenberg, *Combining hierarchical radiosity and discontinuity meshing*, Computer Graphics, Proceedings of Siggraph '93, ACM, 1993, pp. 199-208.

[86] C.T. Loop, *Smooth subdivision surfaces based on triangles*, M.S. thesis, Department of Mathematics, University of Utah, 1987.

[87] C. Loop and T. DeRose, *Generalized B-spline surfaces of arbitrary topology*, Computer Graphics, Proceedings of Siggraph '90, ACM, 1990. pp. 347–356.

[88] C. Loop, *A G^1 triangular spline surface of arbitrary topological type*, Computer Aided Geometric Design, Vol. 11, No. 3, Elsevier, 1994, pp. 303–330.

[89] A.K. Louis, P. Maaß, A. Rieder, *Wavelets: Theorie und Anwendungen* (in German), B.G. Teubner, Stuttgart, Germany, 1994.

[90] J.M. Lounsbery, *Multiresolution analysis for surfaces of arbitrary topological type*, Ph.D. thesis, Department of Mathematics, University of Washington, 1994.

[91] M. Lounsbery, T. DeRose, and J. Warren, *Multiresolution analysis for surfaces of arbitrary topological type*, ACM Transactions on Graphics, Vol. 16, No. 1, ACM, Jan. 1997, pp. 34–73.

[92] R. MacCracken and K.I. Joy, *Free-form deformations with lattices of arbitrary topology*, Computer Graphics, Proceedings of SIGGRAPH '96, ACM, 1996, pp. 181–188.

[93] S.G. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 7, July 1989. pp.674–93.

[94] B.F.J. Manly, *Multivariate Statistical Methods, A Primer*, second edition, Chapman & Hall, New York, 1994.

[95] M. Margaliot and C. Gotsman, *Piecewise-linear surface approximation from noisy scattered samples*, Proceedings of Visualization '94, IEEE, 1994, pp. 61–68.

[96] A. Maus, *Delaunay triangulation and convex hull of n points in expected linear time*, BIT, Vol. 24, No. 2, pp. 151–163, 1984.

[97] Y. Meyer, *Wavelets, Algorithms & Applications*, Society for Industrial and Applied Mathematics (SIAM), Phiadelphia, 1993.

[98] A.A. Mirin, R.H. Cohen, B.C. Curtis, W.P. Dannevik, A.M. Dimits, M.A. Duchaineau, D.E. Eliason, D.R. Schikore, S.E. Anderson, D.H. Porter, P.R. Woodward, L.J. Shieh, and S.W. White, *Very high resolution simulation of compressible turbulence on the IBM-SP system*, Proceedings of Supercomputing 99 Conference, Portland, Oregon, November 1999.

[99] A. Moffat, R.M. Neal, and I.H. Witten, *Arithmetic coding revisited*, ACM Transactions on Information Systems, Vol. 16, No. 3, July 1998, pp. 256–294.

[100] E. Nadler, *Piecewise linear best l_2 approximation on triangles*, C.K. Chui, L.L. Schumaker, and J.D. Ward, eds., Approximation Theory V, Academic Press, 1986, pp 499–502.

[101] J. v. Neumann, *Mathematische Grundlagen der Quantenmechanik* (in German), Grundlehren der mathematischen Wissenschaften, Band 38, Springer-Verlag, Heidelberg, Germany, 1968.

[102] G.M. Nielson and T.A. Foley, *A survey of applications of an affine invariant norm*, T. Lyche and L.L. Schumaker, eds., Mathematical Methods in Computer Aided Geometric Design, Academic Press, 1989, pp. 445–467.

[103] G.M. Nielson, I.-H. Jung, and J. Sung, *Haar wavelets over triangular domains with applications to multiresolution models for flow over a sphere*, Proceedings of Visualization '97, IEEE, 1997, pp. 143–150.

[104] J. Peters, *Curvature continuous spline surfaces over irregular meshes*, Computer Aided Geometric Design, Vol. 13, No. 2, Elsevier, 1996, pp. 101–131.

[105] J. Peters and U. Reif, *The simplest subdivision scheme for smoothing polyhedra*, ACM Transactions on Graphics, Vol. 16, No. 4, 1997, pp. 420–431.

[106] J. Peters and U. Reif, *Analysis of algorithms generalizing B-spline subdivision*, SIAM Journal on Numerical Analysis, Vol. 13, No. 2, April 1998, pp. 728–748.

[107] H. Pottmann, R. Krasauskas, B. Hamann, K.I. Joy, and W. Seibold, *On piecewise linear approximation of quadratic functions*, Journal of Geometry and Graphics, Vol. 4, No. 1, Heldermann Verlag, 2000, pp. 9–31.

[108] E. Quak and L.L. Schumaker, *Least squares fitting by linear splines on data dependent triangulations*, P.J. Laurent, A. Le Méhauté, and L.L. Schumaker, eds, Curves and Surfaces, Academic Press, 1991, pp. 387–390.

[109] E. Quak and N. Weyrich, Decomposition and reconstruction algorithms for spline wavelets on a bounded interval, technical report, CAT report 294, Center for Approximation Theory, Texas A&M University, April 1993.

[110] U.A. Reif, *A unified approach to subdivision algorithms near extraordinary vertices*, Computer-Aided Geometric Design, Vol. 12, No. 2, Elsevier, March 1995, pp. 153–174.

[111] S. Rippa, *Long and thin triangles can be good for linear interpolation*, SIAM Journal on Numerical Analysis, 1992, Vol. 29, No. 1, pp. 257–270.

[112] M. Rumpf, *A variational approach to optimal meshes*, Numerische Mathematik, Vol. 72, No. 4, Springer-Verlag, 1996, pp. 523–540.

[113] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, *Decimation of triangle meshes*, Computer Graphics, Proceedings of Siggraph '92, ACM, 1992, pp. 65–70.

[114] P. Schröder and P. Hanrahan, *Wavelet methods for radiance computations*, G. Sakas, P. Shirley, S. Müller, eds., Proceedings of the Fifth Eu-

rographics Workshop on Photorealistic Rendering Techniques, Springer-Verlag, Berlin, Germany, 1995, pp. 310–326 & 433.

[115] P. Schröder and W. Sweldens, *Spherical wavelets: efficiently representing functions on the sphere*, Computer Graphics, Proceedings of Siggraph '95, ACM, 1995, pp. 161–172.

[116] L.L. Schumaker, *Computing optimal triangulations using simulated annealing*, Computer Aided Geometric Design, Vol. 10, No. 3–4, 1993, pp. 329–345.

[117] S.E. Schussman, M. Bertram, B. Hamann and K.I. Joy, *Hierarchical data representations based on planar Voronoi diagrams*, R. van Liere, I. Hermann, and W. Ribarsky, eds., Proceedings of VisSym '00, Joint Eurographics and IEEE TCVG Conference on Visualization, Amsterdam, Netherlands, May 2000.

[118] T.W. Sederberg, D. Sewell, and M. Sabin, *Non-uniform recursive subdivision surfaces*, Computer Graphics, Proceedings of Siggraph '98, ACM, 1998, pp. 287–394.

[119] H.-P. Seidel, *Representing piecewise polynomials as linear combinations of multivariate B-splines*, T. Lyche and L.L. Schumaker (eds), Curves and Surfaces, Academic Press, New York, 1992, pp. 559–566.

[120] J.M. Shapiro, *Embedded image coding using zerotrees of wavelet coefficients*, IEEE Transactions on Signal Processing, Vol. 41, No. 12, Dec. 1993.

[121] R. Sibson, *locally equiangular triangulation.* The Computer Journal, Vol. 21, No. 2, 1992, pp. 65–70.

[122] J. Stam, *Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values*, Computer Graphics, Proceedings of Siggraph '98, ACM, 1998, pp. 395–404.

[123] E.J. Stollnitz, T.D. DeRose, D.H. Salesin, *Wavelets for Computer Graphics–Theory and Applications*, Morgan Kaufmann Publishers Inc., San Francisco, California, 1996.

[124] W. Sweldens: *Wavelets, signal compression and image processing*, Siggraph Course Notes, ACM, 1994.

[125] W. Sweldens, *The lifting scheme: a custom-design construction of biorthogonal wavelets*, Applied and Computational Harmonic Analysis, Vol. 3, No. 2, 1996, pp. 186–200.

[126] H. Tao, R.J. Moorehead, *Progressive transmission of scientific data using biorthogonal wavelet transform*, Proceedings of Visualization '94, IEEE, 1994, pp. 93–99.

[127] G. Taubin, *A signal processing approach to fair surface design*, Computer Graphics, Proceedings of Siggraph 95, ACM, 1995, pp. 351–358.

[128] O.V. Vasilyev, D.A. Yuen, S. Paolucci, *Solving PDEs using wavelets*, Computers in Physics, Vol. 11, No. 5, AIP, 1997, pp. 429–435.

[129] I.H. Witten, R.M. Neal, and J.G. Cleary, *Arithmetic coding for data compression*, Communications of the ACM, Vol. 30, No. 6, June 1987, pp. 520–540.

[130] J.C. Xia, A. Varshney, *Dynamic view-dependent simplification for polygonal models*, Proceedings of Visualization '96, IEEE, 1996, pp. 327–334 & 498.

[131] D. Zorin, P. Schröder, and W. Sweldens, *Interactive multiresolution mesh editing*, Computer Graphics, Proceedings of Siggraph '97, ACM, 1997, pp. 259–268.

[132] D. Zorin, P. Schröder, and W. Sweldens, *Interpolating subdivision for meshes with arbitrary topology*, Computer Graphics, Proceedings of Siggraph '96, ACM, 1996, pp. 189–192.