Wavelet-Based Multiresolution Surface Approximation from Height Fields

Sang-Mook Lee

Dissertation Submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial Fulfillment of the Requirements of the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

February 1, 2002
Blacksburg, Virginia

Keywords: Surface Approximation, Height Field, Wavelet, Multiresolution Analysis, Templates, Triangulation, Level-of-Detail.

Wavelet-Based Multiresolution Surface Approximation from Height Fields

Sang-Mook Lee

Abstract

A height field is a set of height distance values sampled at a finite set of sample points in a two-dimensional parameter domain. A height field usually contains a lot of redundant information, much of which can be removed without a substantial degradation of its quality. A common approach to reducing the size of a height field representation is to use a piecewise polygonal surface approximation. This consists of a mesh of polygons that approximates the surfaces of the original data at a desired level of accuracy. Polygonal surface approximation of height fields has numerous applications in the fields of computer graphics and computer vision.

Triangular mesh approximations are a popular means of representing three-dimensional surfaces, and multiresolution analysis (MRA) is often used to obtain compact representations of dense input data, as well as to allow surface approximations at varying spatial resolution. Multiresolution approaches, particularly those moving from coarse to fine resolutions, can often improve the computational efficiency of mesh generation as well as can provide easy control of level of details for approximations.

This dissertation concerns the use of wavelet-based MRA methods to produce a triangular-mesh surface approximation from a single height field dataset. The goal of this study is to obtain a fast surface approximation for a set of height data, using a small number of approximating elements to satisfy a given error criterion. Typically, surface approximation techniques attempt to balance error of fit, number of approximating elements, and speed of computation. A novel aspect of this approach is the direct evaluation of wavelet coefficients to assess surface shape characteristics within each triangular element at a given scale. Our approach hierarchically subdivides and refines triangles as the resolution level increases.

Acknowledgments

I would first like to thank my advisor, Dr. A. Lynn Abbott, for his invaluable guidance, patience and inspiration during my entire studying at Virginia Tech. Without his supports, this dissertation would not have been completed. I am also indebted to him for providing financial support during the study.

I would like to express my sincere appreciation to Dr. Schmoldt for his assistance including financial support through the years.

I have been honored to have Dr. Peter M. Athanas, Dr. Richard W. Conners, Dr. Daniel L. Schmoldt, and Dr. Hugh F. VanLandingham as my committee. Special thanks go to my previous teacher, Dr. Dong-Seok Jung at Inha University, for his support and encouragement.

I would like to thank my friends in Blacksburg for their support and warmhearted friendship. My special thanks go to Dr. Jae-Hong Park for his friendship, support, and valuable discussions. I also thanks to Erol Sarigul and Anbumani Subramanian for their support and friendship.

I could not thank enough my parents and parents-in-law for their love and faith in me. Thanks also go to my brother, sisters, sister-in-law, and brother-in-law for their constant encouragement and assistance to my family.

Most of all, my dearest thanks must go to my beloved wife, Kap-Sang, for her support, patience, understanding, and endless love throughout my study in Blacksburg. Finally, I want to express my love to my daughter, Seo-Hyun who always makes me have courage and hope.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A height field is a set of height distance values sampled at a finite set of sample points in a two-dimensional parameter domain. A height field usually contains a lot of information, much of which is redundant and can be removed without a substantial degradation of its quality. A common approach to reducing the size of a height field representation is to use a piecewise polygonal surface approximation. This consists of a mesh of polygons that approximates the surfaces of the original data at a desired level of accuracy. Polygonal surface approximation of height fields has numerous applications in the fields of computer graphics and computer vision.

Computer graphics applications have used polygonal surface models for both simulation and display, while computer vision applications often utilize surface models focused on such tasks as surface reconstruction, pose estimation, and 3D object recognition. Accordingly, development of a fast and accurate approximation method for height fields has long been pursued in both arenas. In addition, many applications require the polygonal surface models to have a capability of level-of-detail control by which the surface models can be represented in various degrees of fidelity to the original data. The main focus of this dissertation is to design a fast algorithm that automatically constructs surface approximations from height fields acquired with an arbitrary input device. In order to achieve high speed as well as the level-of-detail control, this study exploits a wavelet-based approach.

Polygonal surface approximation from input data is a useful preprocessing step for further high level processing. In computer vision, the input data are mostly *range images* acquired with a variety of 3D (three-dimensional) sensing techniques such as *n*-camera stereo [Pulli et al. 98], structured light [Hu and Stockman 89], and time-of-flight laser range-finders [Journet and Bazin 00]. Other sources are *terrain data* acquired from satellite photographs in remote sensing [Pottier et al. 99], and *isosurfaces*, which are extracted from volume data with the "marching cubes" algorithm [Nielson and Hamann 91] in scientific visualization.

With effective approximation techniques, those measured datasets could be reduced in size considerably, and utilized in computer graphics and geometric computer-aided design [Chen and Medioni 92, Curless and Levoy 96, Ramamootrhi and Arvo 99]. Also, high level processing such as analysis of surface characteristics [Sacchi et al. 99], pose estimation [Linnainmaa et al. 88, Chen and Stockman 96, Ji et al. 98], and 3D object recognition [Stark and Bowyer 96, Johnson and Hebert 98] in computer vision, can benefit from such models. Accordingly, a fast polygonal surface approximation method can be very useful for computer vision applications.

In addition, multiresolution analysis (MRA) is a popular trend in 3D data analysis, offering the possibility of compactly representing the input data as well as allowing representation at variable resolution. Wavelets are naturally suited to extracting information at a desired level of resolution and they have very attractive features from a computational point of view [Stollnitz et al. 95a, b, Starck et al. 98]. Therefore, wavelet-based multiresolution analysis has broad range of applications; these include data compression [Lewis and Knowles 92, Shapiro 93], object recognition in 2D [Tieng and Boles 97] and 3D [Paulik and Wang 98], tracking [Hongwei and Zhongkang 96], 3D surface recovery [Yaou and Chang 94, Maeda et al. 97], multiresolution description of meshes in computer graphics and computer-aided geometric modeling [Gortler and Cohen 95, Takahashi et al. 97], as well as surface approximation [Gross et el. 95, Yu and Ra 99, Paster and Rodriguez 99]. Most MRA techniques in computer graphics have used a wavelet-based framework for the decomposition and reconstruction of polygonal meshes [Eck et al. 95, Schroeder and Sweldens 95, Lounsbery et al. 97, Madhuram et al. 97, Mielson et al. 97, Staadt et al. 97, Valette et al. 99, Bonneau 98, Bertram et al. 00]. However, most of earlier studies have not fully exploited *direct decomposition* methods, which directly decompose input data based on wavelet transform. Only the applications for 3D surface recovery have utilized the direct decomposition methods.

1.2 Problem Statement and Goal

The goal of this study is to develop a fast and effective algorithm that can generate a fast triangular-mesh surface approximation for an arbitrary set of height data. The algorithm must balance error of fit, number of approximating elements, and speed of computation. Thus, the surface approximation problem can be formulated as a minimization problem of an energy function

$$\epsilon = \alpha E_{fit}(\Gamma) + \beta E_{size}(\Gamma) + \lambda E_{speed}\,, \tag{1.1}$$

where Γ represents a set of approximating elements, and α, β, and λ are constants. The first term corresponds to fitting error between the approximation and the input data. The second term penalizes meshes that contain large numbers of approximation elements. The speed of the algorithm is formulated in the third term. It is not a function of Γ, because the speed of the algorithm depends on the algorithm itself.

1.3 Proposed Approach

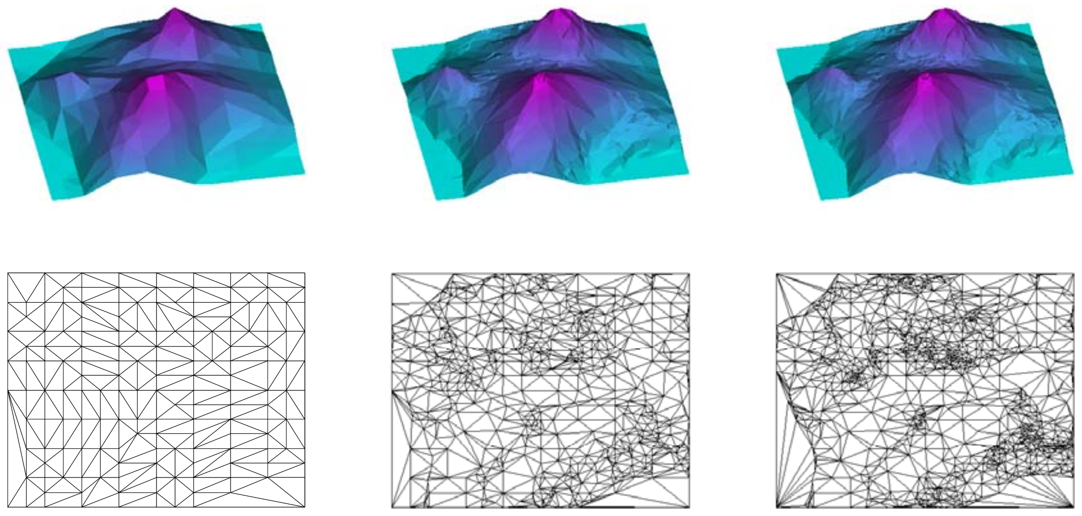
Many other researchers have considered the problem of surface approximation. Most previous work can be categorized as either refinement or decimation approaches. A refinement approach starts with a low-quality approximation and builds more and more accurate ones. The opposite case is decimation, a fine-to-coarse approach, starting from an exact fit to the data and discarding some details to create less and less accurate approximations.

This study focuses on the construction of a fast refinement (coarse-to-fine) method for surface approximation that directly utilizes wavelet information from input data. Instead of exploiting the entire dataset to approximate surfaces, which is quite computationally expensive and cannot easily deal with isolated regions of interest, the new algorithm will utilize wavelet decomposition in order to obtain a fast refinement scheme at various resolution levels. The high level outline of the proposed approach is as follows:

1. Decompose a given height field with a wavelet transform.
2. Construct an initial triangulation using predefined templates at the coarsest level.
3. Repeat until the finest level is reached or the approximation meets a user defined error criterion:
   (a) Select a candidate region.
   (b) Refine the triangles in the candidate region.
   (c) Reduce redundancy and regularize triangular mesh.

While most refinement algorithms have a structure similar to this, the proposed approach has some unique features such as multiscale decomposition of the height field and predefined templates. These features speed up the processing time while maintaining good fitness of the approximations. In addition, the proposed algorithm employs a redundancy removal to reduce the size penalty in the energy function (1.1).

As a result of the proposed approximation method, consider the surface approximations shown in Figure 1.1. A geographic terrain dataset is used in this example, and it contains values from a planar grid of size 256×256. This is an example of a digital elevation model (DEM). In order to obtain approximations, the two-dimensional domain is tessellated with a triangular mesh and then vertices of 3D triangular faces are placed at the height values. Accordingly, the approximated surfaces are piecewise planar patches connecting each triangular region through their common edges. Every 3D vertex of the resulting triangulation is taken directly from the initial dataset.

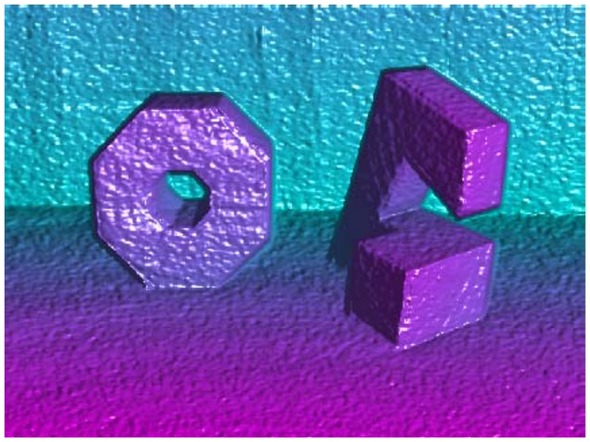


(a) (b) (c)

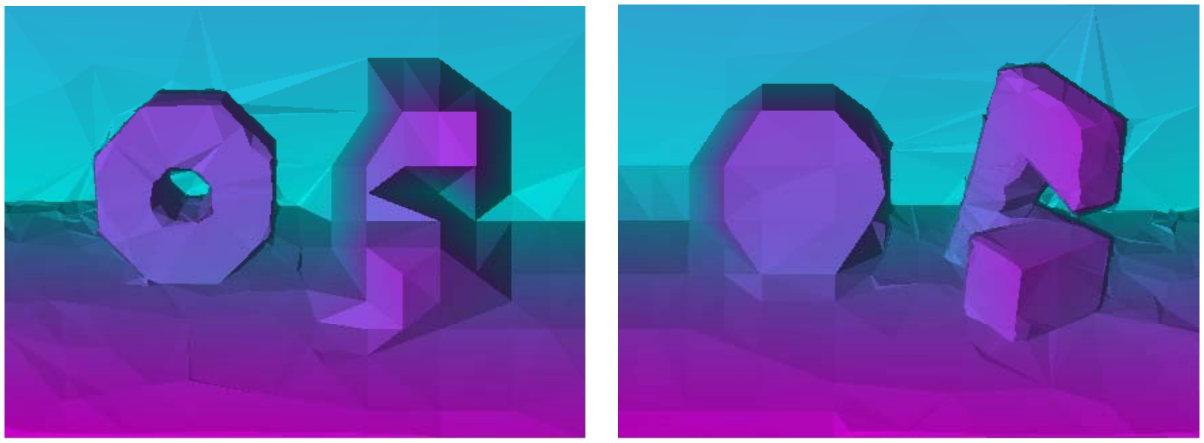
Figure 1.1 Surface approximations with different resolution levels. Wavelet decomposition of input data provides useful information for the surface approximations at various resolution levels. Rendered outputs are displayed in the first row and the corresponding triangular meshes are present in the second row. (a) The coarsest surface approximation. (b) An approximation at an intermediate resolution level. (c) The finest approximation. Only 1.8% of the input data points are used for this approximation.

The three example approximations are rendered with a Lambertian reflection model and displayed in the first row of Figure 1.1. Corresponding triangular meshes are present in the second row. The coarsest approximation (a), constructed from evaluating wavelet coefficients at the highest decomposition level, shows only the overall shape of the input data, and most details are not recovered at this resolution level. In some situations, the approximation at this resolution level is enough, while finer approximations are required for other situations. An intermediate approximation (b) represents many details of the terrain, and its triangular mesh contains tiny triangles. The finest approximation (c) has a much denser triangular mesh in areas of high variability, especially around the peaks of mountains and along the coastline. But even for the finest approximation, only 1.8% of the input data points have been used, and this represents a significant level of data compression.

The proposed approach is capable to have local control of level of detail by using a selective refinement technique that is readily accomplished with the wavelet transform. The most common approach for selective refinement in computer graphics applications is to use a view dependent refinement, by which the entire scene is represented with different levels of detail depending on viewpoint and other parameters such as lighting [Xia and Varshney 96, Hoppe 97, Hoppe 98]. Although there are numerous publications on the selective refinement for terrain datasets and other graphical objects, no such an effort has been performed for range images. In computer vision applications, the selective refinement of range datasets provides both fully reconstructed objects and roughly approximated objects in a single dataset. Figure 1.2 shows results for one case of selective refinement. One of two objects in a single dataset is fully refined while the other remains in a coarse approximation. Selective refinement may be a very attractive property for most computer vision applications.



(a)



(b)

Figure 1.2 Two examples of selective refinement. The proposed approach uses a selective refinement technique that is readily accomplished with the wavelet decomposition of input data. (a) Original dataset. (b) Selective refinement is capable of refining isolated regions of interest.

1.4 Contributions

Specific contributions of this study include the following:

- *A new fast algorithm for approximation of a dense height field using multiresolution analysis with wavelets*. The algorithm has potential application in the areas of computer graphics, computer-aided design, and computer vision. Detailed explanation of the algorithm appears in chapter 3 and chapter 4.

- *A new wavelet-based approach for shape analysis at different levels of detail*. If the 3D shape of a target object is known, it is possible to use the information to approximate the object. This dissertation has developed novel methods for initial mesh construction, and for a refinement scheme based on shape information inferred from the wavelet detail coefficients. This is described in section 3.5 and section 4.4.

- *A set of triangulation templates that can be used for fast triangular tessellation of the 2D domain, based on shape information of the input data*. These are used for fast triangulation at the coarsest resolution level. Section 3.5 will describe how to construct the templates.

- *A shape-preserving refinement algorithm for triangular meshes based on shape analysis of input height fields*. The proposed refinement algorithm for triangular meshes utilizes the shape information implicated in the wavelet detail coefficients so that approximated surfaces can represent the overall shape of the input height fields even at low-resolution levels. Section 4.4 includes details of the algorithm.

- *A method of selective triangulation*. The proposed surface approximation method can construct selective triangulations in which the entire dataset is represented with different levels of detail according to regions of interest. Figure 1.2 shows an example of the selective triangulation. This is described in section 5.6.

1.5 Prospective Applications

Multiresolution surface approximation with wavelets has a wide range of applications. These include:

- *Fast height field approximation*. A height field is a set of data sampled at points in a planar domain. Terrain data, a common type of height field, is used in many applications, including flight simulators, ground vehicle simulator, video games, and in computer graphics for entertainment. Computer vision uses height fields to represent range data acquired by stereo and laser range finders. In most of these applications, it is desirable to have a fast approximation method as well as an efficient data structure for representing and displaying the height field. Typically, the raw sample data is highly redundant. A fast approximation that preserves visual accuracy to a reasonable extent is desired. Using a multiresolution surface approximation with wavelets, it is possible to approximate the height field quickly, while eliminating redundancies.

- *Reverse engineering*. The approximated surfaces with triangular meshes can be used as geometric models that facilitate computer-aided design and subsequent numerical analysis and manufacturing.

- *Level-of-detail (LOD) control in various applications*. Like other MRA techniques, multiresolution surface approximation can control the level of detail of the surface approximation.

- *Analysis of surface characteristics at different resolution levels*. Since the approximation is done by successively refining the coarsest level approximation, it is possible to track surface characteristics at each resolution level. This scheme provides a fast and useful way to find important or critical characteristics on surfaces.

- *3D object recognition*. The proposed surface approximation method can be used in a 3D object recognition system to do coarse-to-fine matching

- *Fast 3D object display on remote terminal.* Transmission of a complex 3D object, to a remote terminal, especially linked with a wireless network, is slow. Thus, it is important to reduce the number of data elements being displayed. An attractive way for displaying such an object over a network is to begin with a low-resolution version that can be quickly rendered, and then progressively improve the display by transmitting detail information.

1.6 Overview of Material

The rest of this dissertation is organized as follows. Chapter 2 presents a brief literature review and the background of some concepts of surface approximation related to the research in this area. This chapter includes fundamental concepts and definition of height field and multiresolution surface approximation with triangular mesh, and introduces wavelet-based multiresolution analysis and some of local operators over triangular meshes. Error metrics and measurement methods are also covered in this chapter. Chapter 3 presents an overview of two-dimensional wavelet decomposition and brief concepts of graph theory. Based on these two ingredients, the chapter then presents a set of shape-preserving triangulation templates that are designed to reflect the underlying shapes of input height fields. The proposed surface approximation method uses these templates to create an initial triangular mesh. Chapter 4 explains the main algorithm for the wavelet-based iterative local refinement method that consists of three sub-procedures: candidate selection, refinement, and mesh enhancement. Section 4.3 through 4.5 will cover these sub-procedures. At the end of this chapter, the detailed version of the approximation method is summarized along with a flow diagram. In addition, all parameters used in the proposed approximation algorithm are summarized. In Chapter 5, time complexity and memory usage of the algorithm are analyzed before the extended experiments are performed. The experiments include the multiresolution surface approximation, performance on noisy datasets, selective triangulation, and empirical running time, showing that the proposed algorithm is feasible for fast surface approximation. Concluding remarks and future directions of this work are presented in Chapter 6.

Chapter 2

Background and Related Work

This chapter provides an overview of background material used throughout the rest of this dissertation. It starts with several terminologies for surface approximation based on triangular mesh construction, and continues to discuss two different approaches for triangular mesh construction. Then, this chapter introduces multiresolution analysis (MRA) techniques and addresses feasibility of using wavelet transform for fast and successive approximation of surface from various types of height fields. Also, related works are introduced in this chapter.

2.1 Height Field and Triangulation

A *height field* is a set of distance data over a planar domain. A planar domain is a two-dimensional *parameter domain*, $\Omega \subset \mathbb{R}^2$, in which height data are sampled. Therefore, it can be said that a height field is a set of height data sampled at a finite set of sample points $\mathbb{V} \subset \Omega$, that is;

$$\mathbb{V} = \{v_i \mid v_i = (x_i, y_i) \in \Omega, i = 1, \cdots, N\}. \tag{2.1}$$

N represents the number of sample points. Commonly, the height fields are sampled at a regular grid, so that sample points are regularly distributed in the parameter domain. In this dissertation, the set of sample points can be denoted by

$$\mathbb{V} = \left\{ v_{ij} \mid v_{ij} = (x_i, y_j) \in \Omega, i = 1, \cdots, N_1, j = 1, \cdots, N_2 \right\}, \tag{2.2}$$

where N_1 and N_2 represent the number of ticks for each axis in the parameter domain. For convenience in representing and referring to the sample points, however, the former notation is used throughout the rest of the dissertation. With this representation of sample points, a height field is formulated by

$$f(x_i, y_i) = z_i, \quad i = 1, \cdots, N, \tag{2.3}$$

where the z_i are the height values.

2.1.1 Approximation of Height Fields

A common type of height field is terrain data acquired from satellite photographs in remote sensing [Pottier et al. 99]. Terrain data is used in many applications, including flight simulators, ground vehicle simulators, and computer graphics for background scenes. In computer vision, the height fields are mostly range images acquired with a variety of 3D sensing techniques such as *n*-camera stereo acquisition system [Abbott and Ahuja 90, Ahuja and Abbott 93, Pulli et al. 98], structured light [Hu and Stockman 89], and time-of-flight measurement [Journet and Bazin 00]. Range images can be used not only for model generation in computer graphics and computer-aided geometric design [Chen and Medioni 92, Curless and M. Levoy 96, Ramamootrhi and Arvo 99], but also for surface characteristics analysis [Sacchi et al. 99], pose estimation [Linnainmaa et al. 88, Chen and Stockman 96, Ji et al. 98], and 3D object recognition [Stark and Bowyer 96, Johnson and Hebert 98] in computer vision. In most of these applications, it is unnecessary and wasteful to process all sampled data. Consequently, an efficient data reduction and approximation technique is desired.

Techniques for data reduction and approximation should efficiently reduce the number of sample points and then approximate the height field with the remaining sample points with desired fidelity for each application. Typically in computer graphics, height fields are represented by a mesh of polygons and rendered with graphics hardware that uses polygons as a fundamental building block for object description. For flight simulator applications, graphics hardware should render those polygons in real time (30 frames per second) to display realistic terrain information. However, since there are more than 2 million triangles in a 1024×1024 grid of terrain data, even recent graphics hardware, which can display roughly 31 million triangles per second, does not meet real time requirements when it handles whole raw sets of sample points. Most applications in computer vision do not consider whole data sets, but focus on important features like edges and surface characteristics such as surface normal direction and curvature. Fortunately, a detailed polygonal representation is highly redundant in most applications, so that its use is unnecessary and wasteful. That is, large planar regions are well approximated by large polygons while small polygons are preserved to represent rough regions with reasonable accuracy.

2.1.2 Piecewise Planar Approximation with Triangulation

The previous section has discussed the need for height field approximation. This section will define some terminology used in the rest of the dissertation. Since a triangular mesh is the most popular representation of an object, a primary objective of this dissertation is to build a triangular mesh for any type of height field.

A *triangle* is a planar patch determined by three non-collinear points connected with straight line segments, and can be represented by

$$T_i = (v_{i_1}, v_{i_2}, v_{i_3}), \quad 1 \le i_1 \ne i_2 \ne i_3 \le N , \tag{2.4}$$

where the *vertices* v_{i_1}, v_{i_2}, and v_{i_3} are non-collinear sample points, and i_1, i_2, and i_3 are indices for those sample points. The vertices determine line segments for the triangle, called *edges*, and are denoted by

$$E_i = (v_{i_1}, v_{i_2}), \quad 1 \le i_1 \ne i_2 \le N . \tag{2.5}$$

Now, we can define a triangulation of the finite parameter domain, $\Omega \subset \mathbb{R}^2$ introduced in Section 2.1. Given a proper subset of sample points $\hat{\mathbb{V}} \subset \mathbb{V}$, a *triangulation*, $\Gamma_{\hat{\mathbb{V}}}$ of Ω is a set of triangles if and only if the following conditions hold:

(i) $\hat{\mathbb{V}}$ is the set of all vertices of triangles in $\Gamma_{\hat{\mathbb{V}}}$;

(ii) $E_i \cap E_j$ is either $\varnothing$ or a vertex if $i \neq j$;

(iii) $\Omega = \bigcup_{i=1}^{t} T_i$, where t is a number of triangles in given triangulation;

(iv) $T_i \cap T_j$ is either $\varnothing$, a common edge, or a vertex if $i \neq j$.

Therefore, a triangulation $\Gamma_{\hat{\mathbb{V}}}$ is defined as;

$$\Gamma_{\hat{\mathbb{V}}} = \{T_i \mid i = 1, \cdots, t\}. \tag{2.6}$$

Also, it is convenient to define an operator that counts a number of elements in a set. Operator $|\cdot|$ on a set means the number of elements in the set. Thus, $t = |\Gamma_{\hat{\mathbb{V}}}|$ is the number of triangles in the triangulation $\Gamma_{\hat{\mathbb{V}}}$, and $\hat{N} = |\hat{\mathbb{V}}|$ is the number of sample points in $\hat{\mathbb{V}}$.

With a triangulation of the parameter domain $\Omega \subset \mathbb{R}^2$, height fields are approximated by piecewise planar patches in $\mathbb{R}^3$ by lifting vertices to the height values and connecting each triangular region through the common edges. Let f_{T_i} denote an approximation by a triangle T_i. Then, for $i = 1, 2, \cdots, N$, f_{T_i} corresponds to a planar patch that satisfies $P_i(x, y) = a_i x + b_i y + c_i$ determined by three vertices of the triangle T_i and defined only within the interior of the triangle:

$$f_{T_i}(x, y) = \begin{cases} P_i(x, y) & (x, y) \in \Omega(T_i) \\ 0 & otherwise \end{cases}, \tag{2.7}$$

$$P_i(x_i, y_i) = z_i, \qquad i = i_1, i_2, \text{and } i_3, \tag{2.8}$$

where $\Omega(T_i)$ represents the interior of T_i and $i = i_1, i_2, \text{and } i_3$ indicates the indices of the vertices of T_i.

For the case of noisy datasets, an improvement over (2.8) is given by

$$P_i(x_i, y_i) = g(z_i), \quad i = i_1, i_2, \text{and } i_3 \tag{2.9}$$

is commonly used for the surface approximation problems. Here, the function $g(\bullet)$ will provide approximate value for each vertex. Throughout this dissertation, it is assumed that $g(z_i)$ has already performed. Then, given a triangulation $\Gamma_{\hat{\mathbb{V}}}$, a piecewise planar approximation $f_{\Gamma_{\hat{\mathbb{V}}}}$ of height fields is given by

$$f_{\Gamma_{\hat{\mathbb{V}}}} = \sum_{i=1}^{t} f_{T_i} \,. \tag{2.10}$$

Since an approximation always contains errors, it is necessary to consider an error metric, denoted by $\left\| f - f_{\Gamma_{\hat{\mathbb{V}}}} \right\|$, to measure the approximation quality. The error metric quantifies differences between an acquired height field and its approximation. Of course, the primary objective of this dissertation is to construct an efficient triangulation method such that the approximation retains a good quality of fit while keeping the number of triangles as small as possible. In-depth discussion on several error metrics will be provided in later section.

Figure 2.1 illustrates the concepts introduced up to now. A height field and regular sample points, at which height values are sampled, are depicted in Figure 2.1 (a) and (b), respectively. Triangulation of all sample points, as shown in Figure 2.1 (c), is not an efficient means of object representation in most cases. By eliminating redundant points to obtain a proper subset of sample points $\hat{\mathbb{V}}$, we can construct a triangulation and its piecewise planar approximation (Figure 2.1 (d)).

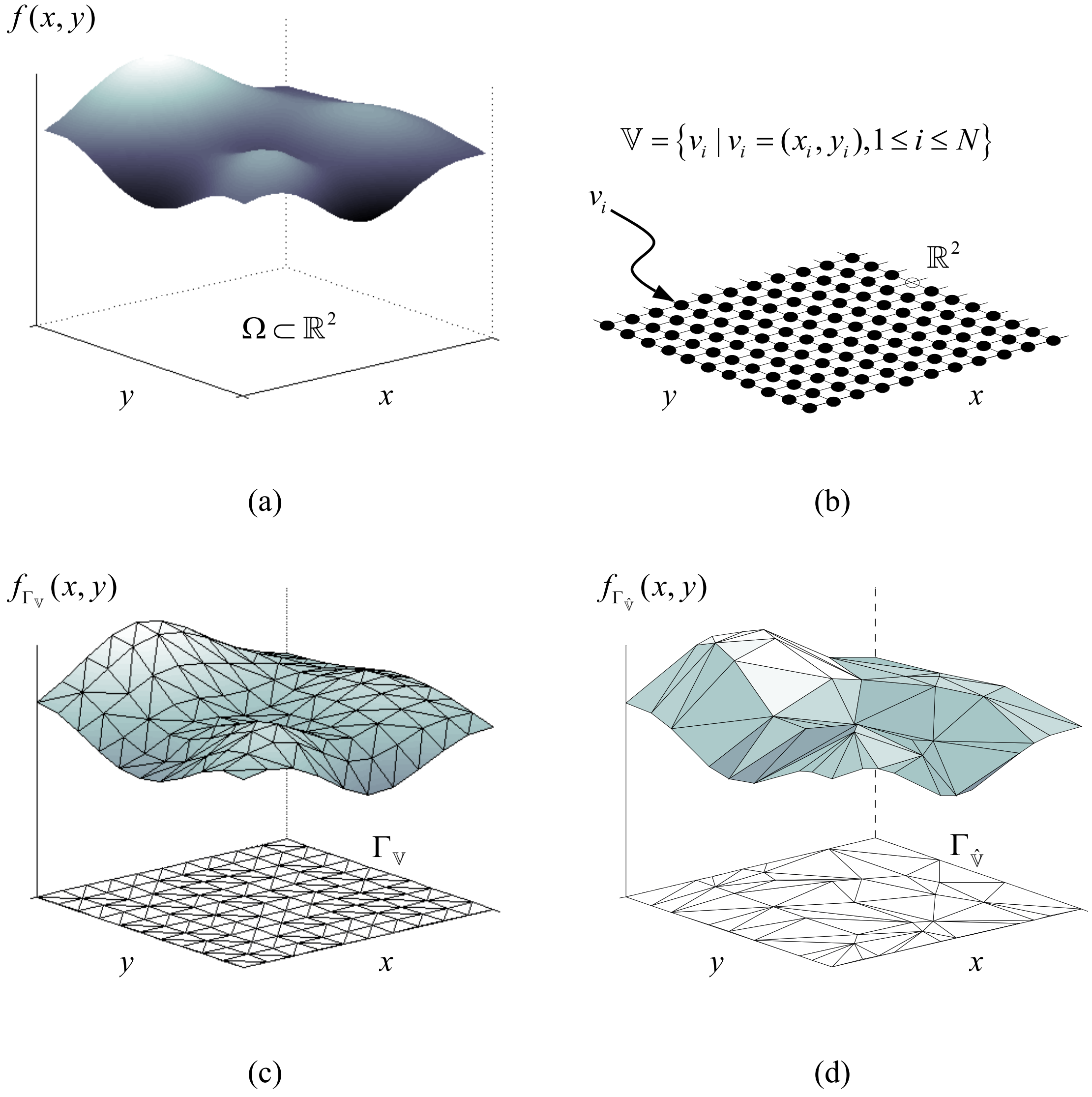

$f(x, y)$
$\Omega \subset \mathbb{R}^2$
y
x
(a)
$\mathbb{V} = \{v_i \mid v_i = (x_i, y_i), 1 \le i \le N\}$
v_i
$\mathbb{R}^2$
y
x
(b)
$f_{\Gamma_{\mathbb{V}}}(x, y)$
$\Gamma_{\mathbb{V}}$
y
x
(c)
$f_{\Gamma_{\hat{\mathbb{V}}}}(x, y)$
$\Gamma_{\hat{\mathbb{V}}}$
y
x
(d)

Figure 2.1 Piecewise planar approximation with triangular meshes. (a) A height field f defined over $\mathbb{R}^2$. (b) The sampling grid $\mathbb{V}$, contained within the parametric domain Ω. (c) Dense triangulation $\Gamma_{\mathbb{V}}$ using all points from the grid $\mathbb{V}$. (d) A triangulation $\Gamma_{\hat{\mathbb{V}}}$ using a smaller vertex set $\hat{\mathbb{V}}$ and its height field approximation $f_{\Gamma_{\hat{\mathbb{V}}}}$.

2.1.3 Common Approaches to Triangulation

To achieve the previously described triangulation, extensive research has been done so far. Many algorithms can be categorized as either refinement or decimation approaches. A refinement approach, also called a coarse-to-fine approach, starts with a coarse approximation and builds up more and more accurate ones. Decimation is a fine-to-coarse approach, starting from exact fit to the sampled data and discarding some details to create approximations with less and less accuracy. There are several methods to refine triangles; greedy insertion [De Floriani et al. 85], feature-based method [Scarlatos and Pavlidis 92], and hierarchical subdivision [De Floriani et al. 84]. The greedy insertion method finds a point at a time that reduces approximation error most, and then inserts it as a new vertex in the triangulation, producing more accurate approximation. Although this method results in good quality, it is slow since it has to search for candidate points to be inserted. Many variations on the greedy insertion algorithm have been explored over the years to make it fast and accurate [Silva et al. 95, Garland and Heckbert 95]. In comparison, the feature-based method first finds some important points using various feature detection techniques such as edge detector or Laplacian filter, and then uses these features to determine vertices or edges of triangulation. This method is quite popular in cartographic application, but not in most others.

Known as an adaptive form of pyramids, hierarchical subdivision methods recursively subdivide a triangle to smaller sub-triangles to construct finer and finer approximation of the surface. Hierarchical subdivision methods are generally fast, simple, and they facilitate multiresolution modeling, but yield poorer quality than greedy insertion methods.

2.1.4 Delaunay Triangulation vs. Data-dependent Triangulation

For height fields and parametric surfaces, there is a natural 2-D parameterization of the surface. Basic triangulation methods are described in a 2-D domain, or in a 3-D domain where height z is a function of x and y. In general, the topology of the triangulation can be chosen using only the x–y projections of the input points, or it can be chosen using the heights of the input points as well. The latter approach is called data-dependent triangulation [Dyn et al. 90].

The most popular triangulation method that does not use height values is *Delaunay triangulation*, which is defined as the dual of the *Voronoi diagram* of a given point set. Given *N* points on the plane, each point p_i associates with a *Voronoi polygon* that is defined by

$$\kappa(p_i) = \bigcap_{i \neq j} H(p_i, p_j), \tag{2.11}$$

where $H(p_i, p_j)$ is a half-plane consisting of a set of points in $\mathbb{R}^2$ that are closer to p_i than to p_j. Accordingly, a Voronoi polygon is the intersection of $N-1$ half-planes, and is a convex polygonal region having no more than $N-1$ sides. These *N* regions partition the plane into a convex set which known as the Voronoi diagram. Figure 2.2 (a) shows an example of Voronoi diagram. Each of the original *N* points belongs to a unique Voronoi polygon, and the line segments represent the set of all points that have more than one nearest neighbor. Its dual graph, formed by connecting each pair of points whose Voronoi polygons share an edge, produces a planar triangulation of the point set known as the Delaunay triangulation (Figure 2.2 (b)) [Preparata and Shamos 85].

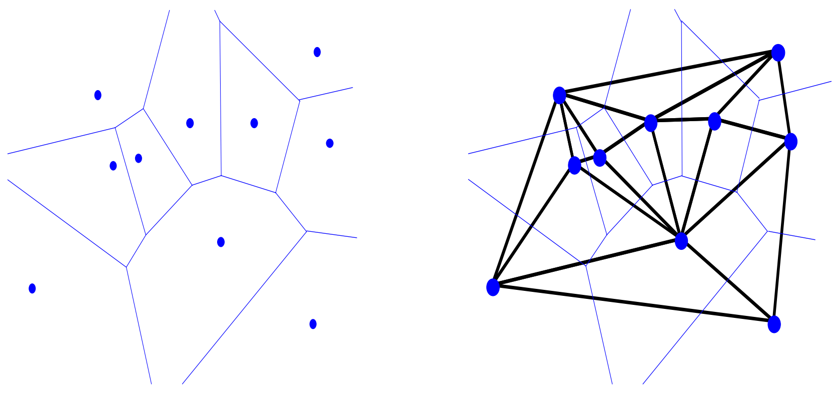
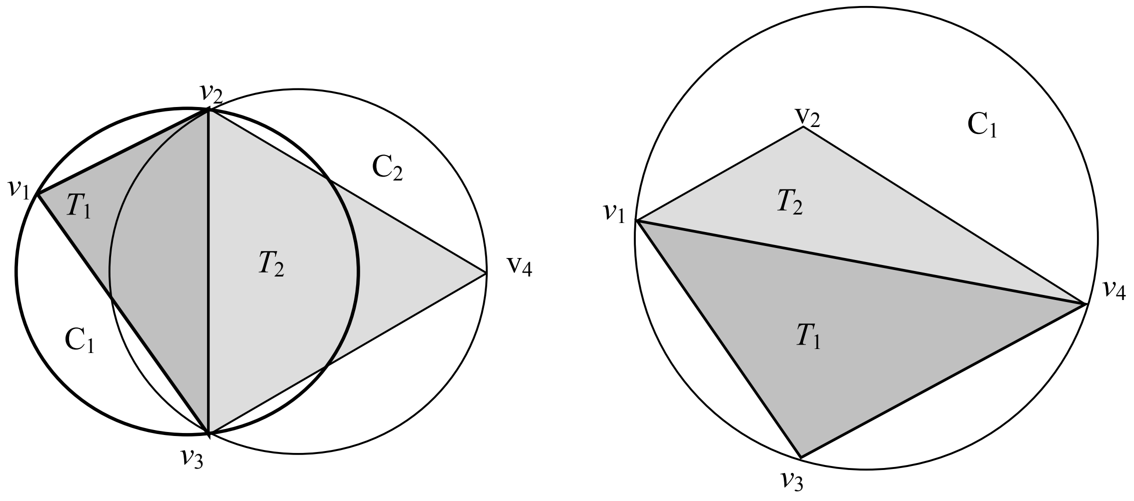


Figure 2.2 Example Voronoi diagram and Delaunay triangulation. (a) Voronoi diagram for the points shown. (b) Corresponding Delaunay triangulation.

In a Delaunay triangulation, the circumscribing circle (circumcircle) of each triangle contains no vertices in its interior [Preparata and Shamos 85]. Thus, the Delaunay triangulation for a set of discrete points corresponds to the triangulation that maximizes the minimum angle for all triangles, among all triangulations of a given point set [Edelsbrunner 87]. This helps to minimize the occurrence of very thin sliver triangles. Delaunay triangulations of *m* points can either be computed by using divide-and-conquer, or by incrementally inserting vertices one at a time and updating the triangulation after each insertion [Guibas and Stolfi 85]. The former approach has cost $O(m \log m)$, while the latter incremental method has worst case cost of $O(m^2)$. Typical costs for the incremental approach are much better than $O(m^2)$, however.

Sometimes Delaunay triangulations are not optimal in terms of approximation error, and for many applications maximization of the minimum angle is not the appropriate goal. Triangulation methods that attempt to optimize the approximation of z or other data associated with the triangulation are called data-dependent triangulation methods. Figure 2.3 shows the difference between Delaunay and non-Delaunay (data-dependent) triangulations. To decide which triangulation to choose, the error between both possible triangulations and the surface is computed, and then the triangulation with the smallest error is chosen. This can be computationally expensive for complex surfaces. Several researchers have shown that slivers can be good when the surface being approximated is highly curved in one direction. Such slivers would not be generated by Delaunay triangulation, which avoids slivers by tending to choose "fat" triangles. Although data-dependent triangulations generally yield better approximation results, Delaunay triangulations are widely used because they are fast and no search for the error is required.



(a) Delaunay triangulation. (b) Non-Delaunay triangulation

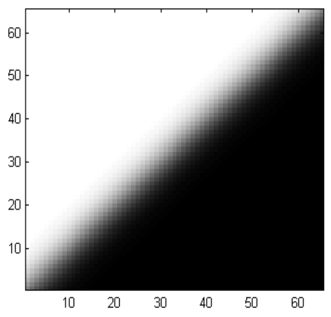
Figure 2.3 Triangulation of 4 vertices. (a) Delaunay triangulation. The circumscribing circle (circumcircle) of each triangle contains no vertices in its interior. Circumcircle (C_1) of triangle T_1 does not contain vertex v_4. (b) Non-Delaunay triangulation. Circumcircle (C_1) of triangle T_1 contains vertex v_2.

There is an example in Figure 2.4 that shows differences between Delaunay triangulation and data-dependent triangulation. The function used in this example is

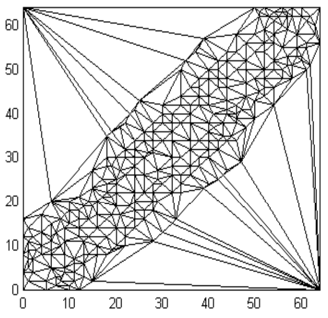
$$F(x,y) = 500 \times \frac{\tanh(9y-9x)+1}{9}, \quad x,y \in [0,1.5], \tag{2.12}$$

which is used as an example in [Dyn et al. 90]. The test function has a sharp rise, running diagonally across the square. For the test data set, 64×64 samples are taken at the evenly spaced sampling grid. Figure 2.4 (a) shows the original function value in gray scale image. For the experiment, the proposed triangulation method (presented in Chapter 4) has produced both Delaunay and data-dependent triangulations. The refinement process has continued until every triangle meets the approximation error tolerance ε_0, which has been set to 0.1. The Delaunay triangulation yields 296 vertices and 573 faces (Figure 2.4(b)), while the data-dependent triangulation results in 82 vertices and 144 faces ((c)). Hence, data-dependent triangulation produces much less triangles than Delaunay method, complying with the same error tolerance. Notice that the differences of triangle shapes in both triangulation. The data-dependent triangulation produces thin and long triangles along the diagonal direction, whereas more regular shapes are found in the Delaunay triangulation. Other noticeable results are found in the interpolated gray scale images ((e) and (f)). The interior of each triangle is filled with the bilinear interpolation of gray values at the vertices. The gray scale image generated by data-dependent triangulation qualitatively shows better results.

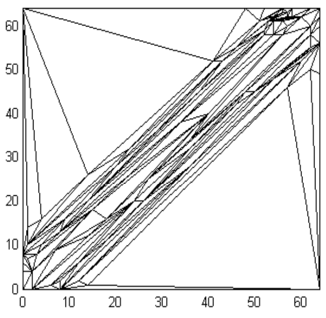
Another Delaunay triangulation, shown in Figure 2.4 (d), is created to have 144 faces, the same number as the one in the data-dependent triangulation. The resulting gray scale image has poor quality ((g)), and also the approximation error tolerance is not satisfied.



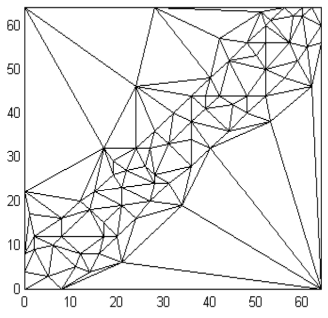
(a)



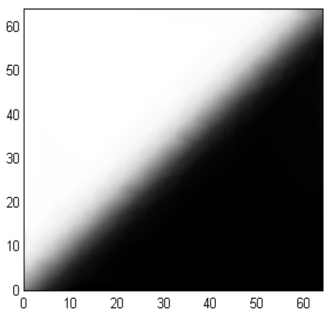
(b)



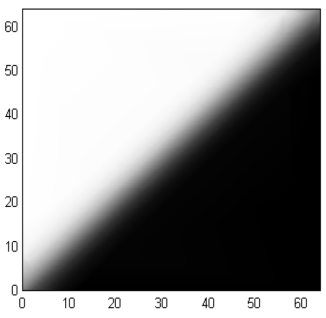
(c)



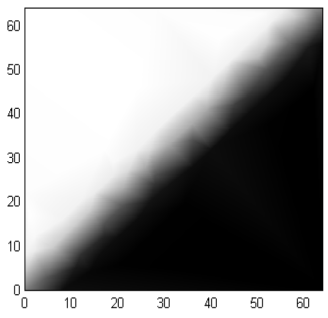
(d)



(e)



(f)



(g)

Figure 2.4 Delaunay and data-dependent triangulations. (a) The test function displayed as a gray scale image. (b) and (c) show the Delaunay (573 faces) and the data-dependent (144 faces) triangulations when the error tolerance is $\varepsilon_0 = 0.1$. (d) Another Delaunay triangulation with 144 faces. Interpolated gray scale images are in (e), (f) and (g).

2.2 Wavelet-Based Multiresolution Analysis

2.2.1 Overview

For over a decade, wavelets have been a popular tool in such fields as a function approximation, signal processing, and physics. In 1989, Mallat developed a rigorous theory of wavelet based multiresolution analysis [Mallat 89], although related pyramidal techniques for image analysis [Burt and Adelson 83] precede his work. In the brief time since their development, wavelets have found use in signal analysis [Mallat 89, Rioul and Vetterli 91], image processing [DeVore et al. 92, Mallat and Hwang 92, Mallat 96], physics [Arneodo et al. 94], and numerical analysis [Beylkin et al. 91]. More recently, wavelets have been applied to a wide range of computer graphics problems, including curve modeling [Finkelstein and Salesin 94], image editing [Berman et al. 94], optimization for surface interpolation [Pentland 92], and animation control [Liu et al. 94]. Good tutorials on the use of wavelets in computer graphics are presented in [Stollnitz et al. 95a, b, Schroeder 96].

There are several reasons for the wide use of wavelets. The foremost is the ability of wavelets to represent functions and datasets compactly at multilevel of resolutions. Multiresolution algorithms modify the resolution of datasets to reduce the computational complexity by beginning at low-resolution and adaptively increasing the resolution to gather the necessary details. In addition, multiresolution representation of geometric models allows the use of sufficient precision for a given portion of interest while at the same time allowing coarse representation in regions outside the interest area, which is known as level-of-detail (LOD) control in computer graphics applications.

Meanwhile, the multiresolution techniques have been used to detect local intensity variation in gray scale images. Sudden change, or edges, of input data are often the most important features for many applications such as pattern recognition, surface characteristic analysis, and pose estimation. Even in surface approximation problems, the knowledge of edge locations plays an important role in improving the appearance of surfaces. The multiscale Canny edge detector [Canny 86] has been used in computer vision to extract edge information from input images. In addition, Mallat and Zhong in [Mallat and Zhong 92] show that the multiscale Canny edge detector can be reformulated by modulus maxima detection in a two-dimensional wavelet transform. The following subsections briefly introduce wavelet-based multiresolution analysis and multiscale edge detection. These two concepts are fundamental ingredients in constructing the proposed hierarchical multiresolution surface approximation technique.

2.2.2 Framework of Multiresolution Analysis with Wavelets

Multiresolution analysis as formulated by Mallat [Mallat 89] provides a convenient framework by which to develop analysis and synthesis filters. According to his framework, a linear function space V^J can be decomposed into a chain of nested linear function spaces $V^0 \subset V^1 \subset V^2 \subset V^3 \subset \cdots \subset V^J$. A space V^j contains functions of resolution j, with the detail increasing as j increases, and is obtained by eliminating the orthogonal complement from a higher space V^{j+1}. Thus, we define the orthogonal complement space W^j as

$$W^j := \left\{ f \in V^{j+1} \mid \langle f, g \rangle = 0,\ \forall g \in V^j \right\}, \tag{2.13}$$

where $\langle \bullet,\bullet \rangle$ represents an inner product of linear functions. The space W^j fills the difference between V^{j+1} and V^j, and the following notation is often used;

$$V^{j+1} = V^j \oplus W^j, \tag{2.14}$$

where $\oplus$ indicates the orthogonal summation. It follows from this definition that the linear function space V^J can be represented by the nested orthogonal summation

$$V^J = W^{J-1} \oplus W^{J-2} \oplus \cdots \oplus W^1 \oplus W^0 \oplus V^0. \tag{2.15}$$

The basis functions of V^j are called *scaling functions* at the resolution level j and denoted by $\phi_n^j(x) = \phi(2^j x - n)$, $n = 1,2,3,\cdots$. In other words, a linear function space is generated by translations and dilations of a single refinable function $\phi(x)$. In a similar fashion, the *wavelets* $\psi_n^j(x)$ $n = 1,2,3,\cdots$ are ordinarily defined to be functions that span the orthogonal complement space W^j. Like the scaling functions, the wavelets $\psi_n^j(x)$ are also generated by dilations and translations of a single function $\psi(x)$, with the additional property that they are defined to be orthogonal to the $\phi_n^j(x)$. More precisely, every basis function $\psi_n^j(x)$ of W^j is orthogonal to every basis function of V^j under the chosen inner product.

A function $f^J \in V^J$ can be decomposed into $f^{J-1} \in V^{J-1}$ and $g^{J-1} \in W^{J-1}$ because any function $f^{j+1} \in V^{j+1}$ can be written uniquely as an orthogonal decomposition

$$f^{j+1} = f^j + h^j . \tag{2.16}$$

In this way, the decomposition of f^J at successive resolution levels produces $h^{J-1}, h^{J-2}, \cdots, h^0$, and f^0. Conversely, f^J can be reconstructed by

$$f^J = f^0 + \sum_{j=0}^{J-1} h^j . \tag{2.17}$$

2.2.3 Multiscale Edge Detection

As the proposed method in this dissertation utilizes multiresolution information of edges contained in the input image, it is helpful to review a multiresolution (or multiscale) edge detector. The Canny edge detector is widely used in computer vision to locate sharp intensity changes and to find object boundaries in an image. The Canny edge detector classifies a pixel as an edge if the gradient magnitude of the pixel is larger than those of pixels at both its sides in the direction of maximum intensity change. Consequently, a typical implementation of the Canny edge detector includes following steps:

(1) smoothing the image with an appropriate Gaussian filter to reduce image details,

(2) calculation of gradient magnitude and gradient direction at each pixel,

(3) locating local maximum point of gradient magnitude along the gradient direction, and

(4) thinning and edge-linking.

Here, a Gaussian filter is used as a convolution kernel to smooth the image, and the width of the Gaussian filter selects the *scale*, or level of detail in the result.

Mallat and Zhong in [Mallat and Zhong 92] show that the multiscale Canny edge detector can be reformulated by modulus maxima detection in a two-dimensional wavelet transform when wavelets are constructed by partial derivatives of the convolution kernel.

Denote the convolution kernel with $\theta(x,y)$. Two wavelets are

$$\psi^1(x,y) = \frac{\partial\theta(x,y)}{\partial x},\ \psi^2(x,y) = \frac{\partial\theta(x,y)}{\partial y}, \tag{2.18}$$

and their dilated versions are denoted by

$$\psi_m^k(x,y) = \frac{1}{2^m}\psi^k\left(\frac{x}{2^m},\frac{y}{2^m}\right) \quad \text{for} \quad 1 \le k \le 2. \tag{2.19}$$

Then, Mallat and Zhong prove that the two detail coefficients of the wavelet transform are proportional to the partial derivatives of input data smoothed by a dilated version of convolution kernel, that is,

$$\begin{pmatrix} d_m^1(u,v) \\ d_m^2(u,v) \end{pmatrix} = -2^m \begin{pmatrix} \frac{\partial}{\partial u}(f * \tilde{\theta}_m)(u,v) \\ \frac{\partial}{\partial v}(f * \tilde{\theta}_m)(u,v) \end{pmatrix}, \tag{2.20}$$

where f is input image, and $\tilde{\theta}_m$ is the dilated version of convolution kernel denoted by

$$\tilde{\theta}_m(x,y) = \frac{1}{2^m}\theta\left(\frac{-x}{2^m},\frac{-y}{2^m}\right). \tag{2.21}$$

In other words, the magnitude of the gradient vector of f smoothed by $\tilde{\theta}_m$,

$$\sqrt{\left|\frac{\partial}{\partial u}(f * \tilde{\theta}_m)(u,v)\right|^2 + \left|\frac{\partial}{\partial v}(f * \tilde{\theta}_m)(u,v)\right|^2} \tag{2.22}$$

is proportional to the magnitude of the wavelet transform coefficients, which called *modulus* and defined by

$$M\left((f*\tilde{\theta}_m)(u,v)\right)=\sqrt{\left|d_m^1(u,v)\right|^2+\left|d_m^2(u,v)\right|^2}\ . \tag{2.23}$$

In addition, the direction of the gradient vector is estimated by an angle

$$A\left((f*\tilde{\theta}_m)(u,v)\right)=\arctan\left(\frac{d_m^2(u,v)}{d_m^1(u,v)}\right). \tag{2.24}$$

The first column of Figure 2.5 shows reconstructed images from wavelet approximation coefficients at decomposition level $2\leq m\leq 4$. The images correspond to those that smoothed by the convolution kernel in Canny's edge detector. The second column displays the modulus $M\left((f*\tilde{\theta}_m)(u,v)\right)$ at each decomposition level. At fine scale (m = 2), $M\left((f*\tilde{\theta}_m)(u,v)\right)$ has a large value at many points, but most of them are removed as the scale increases (e.g. m = 4). However, the smoothing changes the location of edges. Note that Canny's approach [Canny 87] defines the edges at the scale 2^m as points (u_0,v_0) where $M\left((f*\tilde{\theta}_m)(u,v)\right)$ is locally maximum in the one-dimensional neighborhood that is along the direction given by $A\left((f*\tilde{\theta}_m)(u,v)\right)$. Accordingly, the modulus at a large scale gives the coarse information about the location of edges in input images, whereas it contains detail information as the scale becomes finer.

Since the gradient vector points in the direction of maximum change of the intensity surface, the angle $A\left((f*\tilde{\theta}_m)(u,v)\right)$ is orthogonal to the tangent of the edge curve of $(f*\tilde{\theta}_m)(u,v)$. The third column of Figure 2.5 shows the gradient vectors, which are depicted by arrows. It is easily noticed that the arrows direct across the image edges.

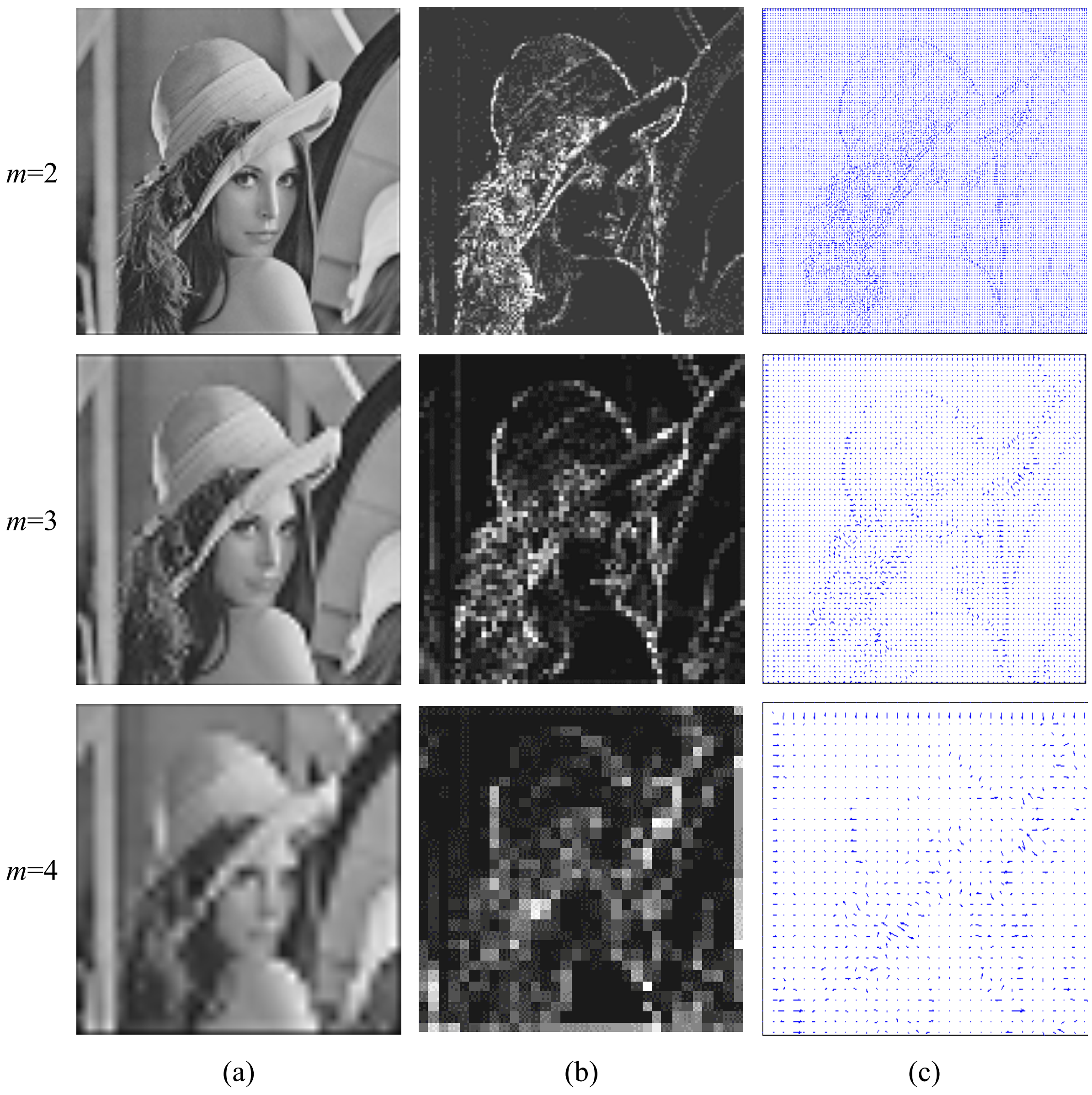


Figure 2.5 Multiscale edge detection. (a) Smoothed images. (b) $M\left((f*\tilde{\theta}_m)(u,v)\right)$. (c) Gradient vector.

2.3 Multiresolution Surface Representation

Single-resolution polygonal models, which are composed of a fixed set of vertices and a fixed set of faces, can only provide a single fixed resolution representation of an object. However, some applications want to use multiresolution models by which the resolution of the representation varies according to the need of applications or the pursuit of the time cost reduction. For example, in a perspective view of large terrain dataset, large polygons are used to represent objects in the distance, while much finer polygons are required for displaying closer objects. This is one prominent property of multiresolution representation, known as level of detail (LOD) control. In computer vision, most applications would like to know an overall shape of an object before they focus on detail analysis. With this approach, it is possible to reduce processing time in object recognition, pose estimation, and object querying. As another example, when a polygonal model must be transmitted over the network, first a server sends out only an initial coarse representation, and then transmits the refinement information in sequence to make a client system build a final detail representation. This is much cheaper in time cost than transmitting the final model at once explicitly.

This section will provide a terminology of multiresolution surface representation for later use in this dissertation, and summarize previous works on wavelet usage for multiresolution surface representation.

2.3.1 Terminology for Multiresolution Surface Representation

Consider a two-dimensional data set f taken at a set of sample points $\mathbb{V}_0 \subset \mathbb{R}^2$. A hierarchical mesh is obtained by constructing an initial mesh $\hat{\Gamma}_M \triangleq \Gamma_{\hat{\mathbb{V}}_M}$ at the coarsest level, M, with a set of sample points $\hat{\mathbb{V}}_M$, and by refining the initial mesh to finer meshes $\hat{\Gamma}_{M-1}$, $\hat{\Gamma}_{M-2}, \dots,$ $\hat{\Gamma}_0$, with sets $\hat{\mathbb{V}}_{M-1}, \hat{\mathbb{V}}_{M-2}, \cdots, \hat{\mathbb{V}}_0$, respectively. The original data set is approximated over the entire domain by a piecewise planar function $f_{\hat{\Gamma}_m}$ that interpolates all data values at points of $\hat{\mathbb{V}}_m$, $0 \le m \le M$. Typically, the number of points in $\hat{\mathbb{V}}_m$ increases as m decreases, and normally $\hat{\mathbb{V}}_0 \ne \mathbb{V}_0$. This implies that $\left|\hat{\Gamma}_m\right| < \left|\hat{\Gamma}_{m-1}\right|$ for $0 < m \le M$, and $\left|\hat{\Gamma}_0\right| \ne \left|\Gamma_0\right|$, where $\left|\bullet\right|$ represents mesh size as defined in section 2.1.2.

In most applications, given an error metric $\varepsilon_m = \| f - f_{\hat{\Gamma}_m} \|$, the main goal of mesh generation is to minimize $\left|\hat{\Gamma}_0\right|$ subject to the error ε_0 being kept below a given error criterion. However, the minimization of mesh size for a given accuracy is an NP-hard problem and heuristics are needed for practical implementation.

2.3.2 Wavelets in Surface Approximation

While many researchers [Hoppe et al. 93, Soucy and Laurendeau 96, Garland and Heckbert 97, Lounsbery et al. 97, Yemez and Schmitt 99, Morris and Kanade 00] have worked to achieve an efficient multiresolution representation of polygonal model of an object, wavelets have been widely explored for multiresolution analysis purposes in surface approximation.

Most related research [DeRose et al. 94, Eck et al. 95, Madhuram et al. 97, Mielson et al. 97, Staadt et al. 97, Bonneau 98, Valette et al. 99] has applied multiresolution analysis to the decomposition of triangulated surfaces. Their primary concerns are to edit surface at different level of detail, to transmit and reconstruct surfaces progressively, and to save storage. Among them, Eck et al. have constructed a parameterization of an arbitrary initial mesh Γ_0 over a simple domain. Then, 4-to-1 subdivisions shown in Figure 2.6 are applied over the simple domain until the difference between Γ_0 and an approximation $\hat{\Gamma}_0$ is bounded by a specified remeshing tolerance. In this way, $\hat{\Gamma}_0$ has subdivision connectivity and can be used in the multiresolution analysis. In addition, Bonneau has constructed a wavelet-like transform that can be used on meshes with arbitrary connectivity. His work is based on the concept of multiresolution analysis over non-nested spaces, which are generated by the so-called BlaC-wavelets, a combination of the Haar function with the linear B-Spline function [Bonneau et al. 96].

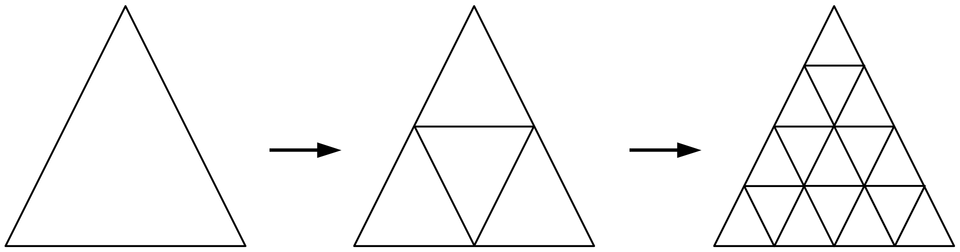


Figure 2.6 4-to-1 subdivision connectivity.

Some researchers [Yaou and Chang 94, Maeda et al. 97] use multiresolution analysis techniques to recover surface representation from acquired data set, in which they typically use spline wavelets so that they can directly reconstruct the surface with the spline [Dreger et al. 98]. Only a few efforts have been made for surface approximation by applying multiresolution analysis to input data directly [Gross et el. 95, Yu and Ra 99]. Gross et al. have used wavelet decomposition directly on the input height field to remove unimportant vertices in quad-tree structures, while Yu and Ra focus on the edge-preserving scheme. Both utilize the wavelet coefficients directly to extract useful features that might be in the input dataset. However, they focused on constructing fast algorithms, not on minimizing number of triangles.

2.4 Summary of Previous Work

Table 2.1 summarizes surface approximation methods that are introduced in previous sections. Surface approximation methods are categorized largely as either refinement or decimation approaches. Most refinement and decimation approaches are based on error measurement between approximations and datasets, which provides good fits for the approximations. Therefore, those methods must calculate the approximation error every time they insert or delete a triangle in order to determine the next insertion or deletion. The result is high computational cost.

Table 2.1 Summary of previous work

	Decimation	Refinement	Wavelet Method
Main criterion	Approximation error	Approximation error	Wavelet coefficients
Complexity	Slow	Slow	Fast
Goodness of fit	Maintain good shape	High likelihood of losing important details	Dependent on thresholds
Number of triangles	Can be small	Can be many	Lots of redundancy
Progressive display	Possible with hierarchical data structure	Possible with hierarchical data structure	Impossible
Robustness to noise	Not good	Dependant on error metric	Noise can spread out to neighboring triangles

Generally, decimation approaches maintain overall shapes of input datasets, because they start from an exact fit of a dataset, and therefore know which triangles can be eliminated without substantial degradation in quality. Accordingly, they may achieve a near optimal solution in terms of the number of triangles for a given error criterion. In contrast, there is a high probability of losing important details in refinement approaches as they start from a coarse approximation and construct a finer approximation. Without an exhaustive search for an appropriate refinement scheme, the refinement approaches tend to generate more triangles than decimation approaches. Both the decimation and the refinement approaches can be used for progressive display on remote terminals.

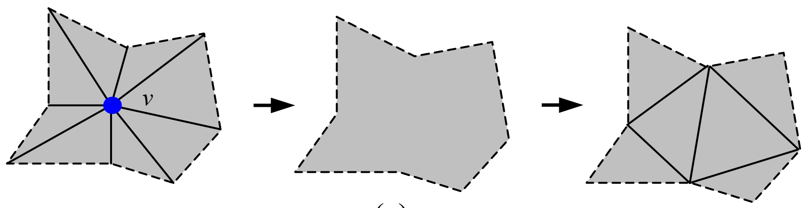
Another category of surface approximation methods is a wavelet-based approach, in which wavelet decomposition is performed directly on the input height field in order to extract useful features from input datasets. The primary goal of the wavelet method is constructing a fast algorithm, but not in minimizing the number of triangles. It is possible to control the goodness of fit for approximations by selecting different thresholds for wavelet coefficients. A major drawback of the approach is that progressive display is not feasible for this approximation method. For noisy datasets, the decimation approaches are the weakest among the three categories since there is no mechanism for noise reduction, while the refinement approaches can smooth the noise level in their approximations by using suitable error metrics. In the wavelet methods, noise can be spread from neighbor to neighbor when the dataset is decomposed with wavelets having a wide support.

2.5 Surface Simplification

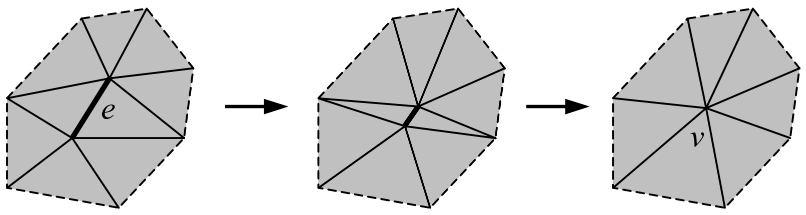
As mentioned in section 2.1.3, many surface simplification methods can be categorized as either refinement or decimation approaches. The proposed method in this dissertation is a refinement approach based on wavelet coefficient analysis. Even though the new method is designed to construct a triangulation that has fewer triangles than other methods, it is still possible to produce tiny triangles around the points at which have high gradient value in input dataset. Those redundant tiny triangles should be removed in order to obtain a compact triangulation. Therefore, this section will introduce some mesh simplification methods that have been used in surface approximation literature.

2.5.1 Local Operators over Triangular Meshes

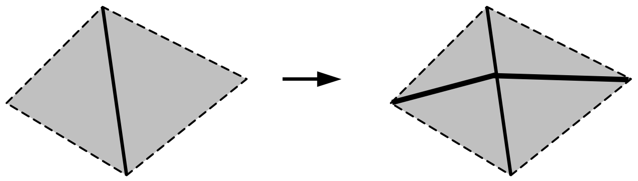
Local operators used in surface simplification attempt to reduce mesh complexity while preserving underlying topology. There are several such local operators: vertex removal, edge collapse, vertex clustering [Rossignac and Borrel 93, Garland and Heckbert 95, Ronfard and Rossignac 96], and face removal [Hamann 94]. On the other hand, some operators increase the mesh complexity in order to produce better approximation. Edge split operator and subdivision templates are included in this category. In addition, there are other operators that do not reduce mesh complexity but improves the fitting of the simplification to the original surface creating well-shaped triangles. These are called fitting operators, and edge swap and vertex displacement operators are in this category. Among them, vertex removal, edge collapse, edge split and edge swap operators are used in this dissertation and will be briefly introduced here. Figure 2.7 illustrates those local operators.



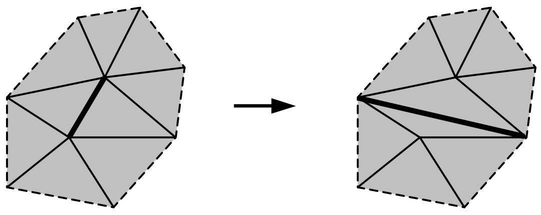
(a)



(b)



(c)



(d)

Figure 2.7 Local operators over triangular meshes. (a) Vertex removal. The shaded triangles share vertex *v*. Removing *v* and its associated edges results in a polygon bounded by the dotted lines. The polygon is re-triangulated with constrained-Delaunay triangulation. (b) Edge collapse. An edge *e* collapses to a vertex *v* degenerating two faces. (c) Edge split bisects the common edge of two adjacent triangles yielding four smaller triangles. (d) Edge swap does not affect the mesh complexity, but it is used to reduce approximation error or to produce well-shaped triangles.

A *vertex removal* operator is used in [Schroeder et al. 92, Cohen et al. 96, Soucy and Laurendeau 96] and in most terrain simplification methods. The vertex v and its *t* incident triangles are removed. The resulting holes are re-triangulated using *t*-2 triangles as shown in Figure 2.7(a). A common strategy for selecting a vertex to be removed is based on curvature estimation at the vertex. Vertices with low curvature values are removed first.

While a vertex removal operator considers vertices as input parameter, an *edge collapse* operator takes edges to be collapsed as input. The selected edge collapses to a vertex reducing the number of faces by two, the number of edges by three, and the number of vertices by one ((b)). Edge selection is commonly based on error estimation. This operator is used in [Ronfard and Rossignac 96, Algorri and Schmitt 96, Hoppe 96, Hoppe 97]. *Edge split* operator splits common edge of two adjacent triangles making four smaller triangles [Hoppe et al. 93]. This operator is used in order to increase mesh complexity by which the approximation could by finer. In this dissertation, the edge split operator is a main tool for the proposed bottom-up refinement method. The *edge swap* operator ((d)) is used in [Hoppe 96, Ciampalini et al. 97] to minimize the error produced by previously applied operators, such as vertex removal and edge collapse.

2.5.2 Mesh Optimization

An optimization approach for surface simplification is presented in [Hoppe et al. 93]. To minimize an energy function, the optimization method selects one of three local operators: edge collapse, edge swap, and edge split. The energy function is defined as the sum of three components:

$$E(\Gamma_{\hat{\mathbb{V}}}) = E_{error}(\Gamma_{\hat{\mathbb{V}}}) + E_{size}(\Gamma_{\hat{\mathbb{V}}}) + E_{spring}(\Gamma_{\hat{\mathbb{V}}})\,. \tag{2.25}$$

$E_{error}(\Gamma_{\hat{\mathbb{V}}})$ measures fidelity to the input mesh as the sum of squared distances of the points X from the current mesh, where X is a set of points over the original mesh computed by uniform sampling. $E_{size}(\Gamma_{\hat{\mathbb{V}}})$ penalizes meshes with a large number of vertices so that the optimization encourages vertex removal if $E_{error}(\Gamma_{\hat{\mathbb{V}}})$ increases slightly. $E_{spring}(\Gamma_{\hat{\mathbb{V}}})$, measured as the sum of edge length, is designed as a regularizing term that penalties long edges.

They start with an initial mesh generated from sampled data set, and evaluate the energy function at each step for the three local operators. Local operator that yields largest reduction of the energy is chosen. The main advantage of this approach is the high quality of the simplified representation, but long processing times are required.

2.6 Evaluations of Surface Approximations

2.6.1 Overview

As stated earlier, the main goal of surface approximation is to extract surface patches to approximate a given dataset. Those surface patches should represent the original dataset with high accuracy. In most cases, geometric error metrics are used to measure the accuracy of the approximation. However, in some applications such as computer graphics, the overall appearance of the approximation should be the same as the original objects. Accordingly, we have to define an *approximation error*, which gives a quantitative measure of approximation geometrically and visually. Thus, the defining an approximation error is quite application dependent. Since most applications use geometric error between the approximation and the original dataset, this section will mostly discuss the geometric error metrics. Then, a brief introduction to the similarity measure of appearance, mostly used in computer graphics applications [Heckbert and Garland 94, Hughes et al. 1996, Bajaj and Schikore 1996, Garland and Heckbert 98], will follow.

In addition to geometric accuracy and similarity of appearance, it would be desirable in most applications to reduce the number of elementary building blocks, as far as the approximation satisfies a given error tolerance. As mentioned in previous sections, this dissertation is devoted to the development of a novel triangulation method so that the elementary building block considered here is a triangle. Hence, fewer triangles render graphic objects faster and reduce time complexity in computer vision application. Minimizing the number of triangles in a triangulation scheme is an NP-hard problem and is not appropriate in practical use. Although Hoppe et al. have formulated an energy function penalizing large mesh sizes [Hoppe et al. 93], their optimization scheme has been limited to minimization in local areas.

2.6.2 Measures of Approximation Error

In surface approximation of height fields, we can consider height distance, normal distance, and Hausdorff distance as useful error measures. Among them, Hausdorff distance is most commonly used but is time consuming. Height distance is the simplest error measure, and it measures the height difference between a point P on a triangular face and its corresponding point Q on the height field. This simple height distance is not an appropriate measure when it is measured at a point with high gradient value. To overcome this drawback, normal distance has been regarded as a good alternative in many applications. Figure 2.8 shows the difference between the height distance and normal distance in case of 1D signal. The original dataset $f(x)$ is approximated with a piecewise linear function $\hat{f}(x)$.

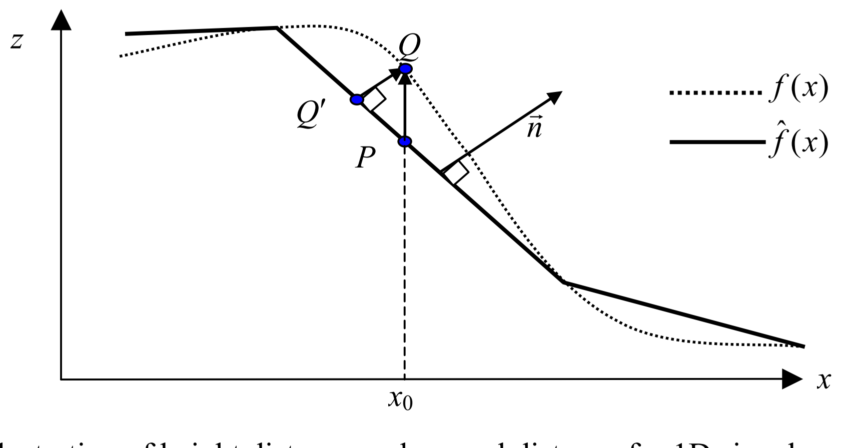


Figure 2.8 Illustration of height distance and normal distance for 1D signal.

Then, the height distance between $f(x)$ and $\hat{f}(x)$ at x_0 is simply the absolute value of the difference of two functions,

$$d(f,\hat{f})_{x_0} = \left| f(x_0) - \hat{f}(x_0) \right| = \left| Q - P \right|, \tag{2.26}$$

while the normal distance is defined by

$$\bar{d}(f,\hat{f})_{x_0} = \left| \overrightarrow{QP} \bullet \vec{n} \right| = \left| Q - Q' \right|, \tag{2.27}$$

where $\vec{n}$ is the unit normal vector of the line segment in which the point P is placed.

The normal distance measure also has problems especially where the 3D shapes are approximated. [Seibold and Wyill 98] shows good illustrations for those problems. Therefore, in case of 3D shape approximation, the Hausdorff distance [Preparata and Shamos 85], which is a metric for comparing point sets, has been considered. The Hausdorff distance has been used in diverse applications, including defect detection [Chetverikov and Khenokh 99], gesture recognition [Kahn et al. 96], robot localization [Olson 97], range image analysis [Sim and Dudek 99], and content-based video and database indexing. Hausdorff measures have been used in computer vision nearly exclusively to match binary patterns of contour or edges [Huttenlocher and Rucklidge 93, Rucklidge 96].

The Hausdorff distance [Grünbaum 67] from a set $\mathbf{A}$ to a set $\mathbf{B}$ is defined as

$$h(\mathbf{A} \to \mathbf{B}) = \max_{v \in \mathbf{A}} \min_{w \in \mathbf{B}} \left\| v - w \right\|, \tag{2.28}$$

and the Hausdorff distance between two sets $\mathbf{A}$ and $\mathbf{B}$ is equal to

$$h(\mathbf{A},\mathbf{B}) = \max \left\{ h(\mathbf{A} \to \mathbf{B}), h(\mathbf{B} \to \mathbf{A}) \right\}, \tag{2.29}$$

where $\left\| \bullet \right\|$ denotes the Euclidean vector length operator. Suppose $\mathbf{A} \subset \mathbb{R}^3$ is a set of points on an approximated surface and $\mathbf{B} \subset \mathbb{R}^3$ is a dataset for a height field. Then, if the Hausdorff distance is bounded by ε_0, this means that every point of the approximation is within a Euclidean distance of ε_0 of the original dataset and vice versa. That is, the Hausdorff distance searches for a global minimum distance, which is not practical. So, almost all applications [Klein et al. 96, Soucy and Laurendeau 96, Ciampalini et al. 97, Kobbelt et al. 98] use its localized version, in which search area is restricted to a small corresponding neighborhood of a given point. We can rewrite the definition as a localized version as follows:

$$\hat{h}(\mathbf{A} \to \mathbf{B}) = \max_{v \in \mathbf{A}} \min_{w \in \mathbf{R}(v) \subset \mathbf{B}} \|v - w\|, \tag{2.30}$$

where $\mathbf{R}(v) \subset \mathbf{B}$ is the corresponding neighborhood of a point v. Therefore, the modified overall Hausdorff distance between the original set and the approximation is defined by

$$\hat{h}(\mathbf{A}, \mathbf{B}) = \max\left\{\hat{h}(\mathbf{A} \to \mathbf{B}), \hat{h}(\mathbf{B} \to \mathbf{A})\right\}. \tag{2.31}$$

2.6.3 Error Metrics for Approximation Error

In order to quantify the goodness of fit of an approximation to a given dataset, we have to use an error metric. Most commonly used error metrics are the L_∞ and L_2 norms [Prenter 75]. The L_∞ norm measures the maximum deviation between the original and the approximation and is defined by

$$\left\|f - \hat{f}\right\|_\infty = \max_{(x,y) \in \Omega} \left|d(f, \hat{f})\right|, \tag{2.32}$$

where $d(f, \hat{f})$ represents the height or normal distance measures, as described in the previous section, between the original f and the approximation $\hat{f}$ at a point (x, y) in the parametric domain $\Omega \subset \mathbb{R}^2$. The L_2 norm, defined by

$$\left\|f - \hat{f}\right\|_2 = \sqrt{\sum_{(x,y) \in \Omega} d^2(f, \hat{f})}, \tag{2.33}$$

provides a measure of the average deviation between the original and the approximation. When it is divided by the number of points in a set Ω, $N(\Omega)$, that is,

$$\frac{1}{N(\Omega)} \sqrt{\sum_{(x,y) \in \Omega} d^2(f, \hat{f})} \tag{2.34}$$

is called the root mean square (RMS) error. The Hausdorff distance constructs an error metric by itself, which has properties similar to the L_∞ norm.

It is well known that the L_∞ norm is a strict condition that does not allow even one point to deviate from an error bound, but it is overly sensitive to any noise that might be present in acquiring height fields. In contrast, while the L_2 norm is more generous on noise, it tends to smooth out high frequent variations that are parts of actual dataset. For that reason, [Garland and Heckbert 97] suggested the use of combination of these two error metrics. In the combined error metrics, L_∞ norm acts as a global error bound, and in that error bound, L_2 norm tries to estimate better overall fit tolerating small noises.

2.6.4 Similarity Measure of Appearance

Whereas the distance measures are used to evaluate geometric similarity of approximation to the original dataset, a similarity measure of appearance focuses on how much two objects look the same. Content-based image retrieval system compares the appearance between a query image (RGB or grayscale images) and the database, and in computer graphics applications, the appearance similarity the between the rendered model and the original is an important factor for human perception.

While there are many other similarity measures [Puzicha et al. 97, Rubner and Tomasi 98, Aksoy and Haralick 01] between two images I_1 and I_2, a simple average sum of squared differences of all corresponding pixels defined by

$$d(I_1, I_2) = \frac{1}{n^2} \sum_x \sum_y \left\| I_1(x,y) - I_2(x,y) \right\|^2 \tag{2.35}$$

has been widely used in image compression, tracking, and optical flow estimation. Again, the notation $\|\bullet\|$ represents Euclidean vector length, and we assume that the image sizes are $n \times n$. This simple definition is suitable for local error measurement, and thus makes the error estimation fast.

2.6.5 Quality Control and Topological Validity of the Triangular Meshes

The Delaunay triangulation, which maximizes the minimum angle over all triangulations of the vertices, produces well-shaped meshes. As shown in section 2.1.4, however, the use of Delaunay triangulation alone sometimes yields a poor approximation. In contrast, data-dependent triangulation can produce a good-quality approximation compared to the Delaunay triangulation, but it often produces excessively long and thin triangles. As a result, we may need to control the quality of the triangular mesh by suppressing long and thin triangles, only allowing them if they reduce the approximation error significantly.

We can define an energy function for mesh quality control such as

$$\xi(T_i, T_j) = \alpha \Delta E + \beta \Delta A \,. \tag{2.36}$$

The energy function defined above is a combination of the differences of approximation errors (ΔE) and the difference of minimum angles (ΔA) of two triangulations, which can be obtained by a swap operation in section 2.5.1. An example on the mesh quality control is shown in Figure 2.9. The same dataset as one in section 2.1.4 is used for the example. For $\varepsilon_0 = 0.1$, 149 vertices and 278 faces are generated. As we expected, long and thin slivers, as like ones shown in Figure 2.4, have disappeared in the triangulation.

Meanwhile, the triangular mesh must be topologically valid. The most common cause of invalid topology generation is the swap operation. Figure 2.10 (a) and (b) show this case. When the common edge $E(b,d)$ of two triangles $T_1 = (a,b,d)$ and $T_2 = (b,c,d)$ is swapped to produce a new common edge $E(a,c)$, the new triangles $T_1' = (a,b,c)$ and $T_2' = (a,c,d)$ do not satisfy the condition (iv) in section 2.1.2 to be a proper triangulation. This case can be avoided by checking the convexity of the quadrilateral formed by two triangles T_1 and T_2. Non-convex quadrilaterals generate invalid topology of triangulations when their diagonals are swapped. Another invalid topology is caused by a vertex incompatibility between adjacent triangles as shown in Figure 2.10 (c). In this case, also the condition (iv) for a proper triangulation does not hold, because triangles T_1 and T_2 share a part of common edge $E(a,v)$ and a vertex v.

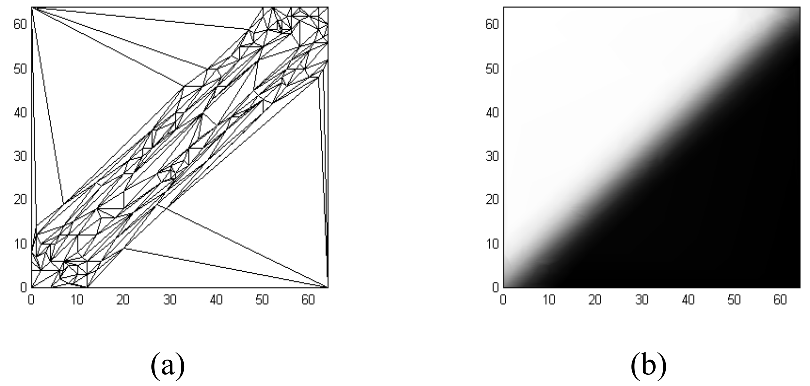


Figure 2.9 An example of mesh quality control. Mesh quality control suppresses long and thin slivers in triangulations (a) with a good quality of the interpolated images (b), but increases the mesh size.

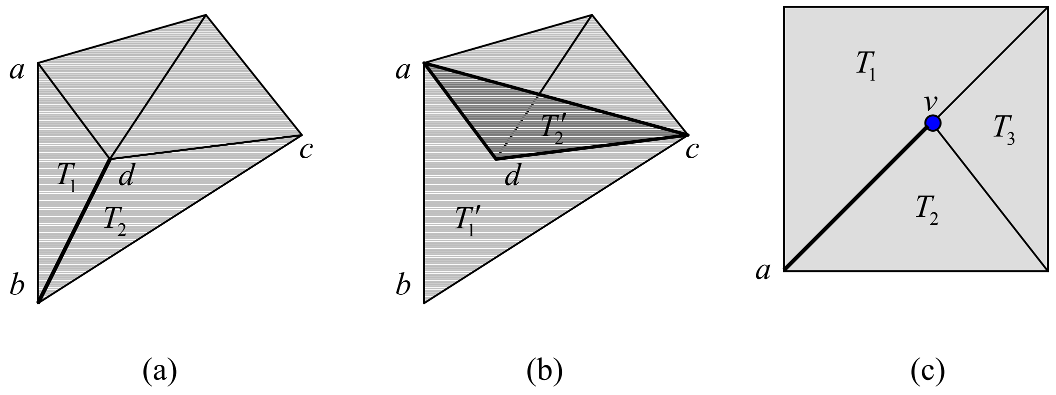


Figure 2.10 Topological invalidity. (a) The quadrilateral *a*-*b*-*c*-d, formed by T_1 and T_2, is not convex. (b) Triangles $T_1' = (a,b,c)$ and $T_2' = (a,c,d)$ do not meet condition (iv) in section 2.1.2 to be a proper triangulation. (c) Vertex incompatibility leads to an invalid topology of triangulations.

Chapter 3

Wavelet Decomposition and Initial Triangulation

In Chapter 2, we have introduced basic theory of multiresolution analysis and important concepts of triangulation. Based on the background, this chapter presents our new initial triangulation method after discussing 2D wavelet decomposition. Any wavelet class can be used in surface approximation application, but since our goal is to construct a fast and efficient algorithm for piecewise planar approximation of height fields, we need to be careful in choosing wavelet classes to satisfy our purpose. Generally, using wavelets with wide support requires long processing time in decomposition and in mesh construction. As a result, we are going to restrict our selection of wavelets to ones that have smaller support as long as they satisfy other requirements such as smoothness and symmetry. The proposed initial triangulation in this dissertation uses predefined templates. This chapter also provides the way in which we select those templates. Our prime concern of selecting the templates is to preserve underlying shape of input datasets at the coarsest level of resolution. To achieve shape-preserving templates, we perform a moderate analysis of wavelet detail coefficients. This chapter also presents cost analysis and experimental results of the initial triangulation.

3.1 Design Goals

Typical surface approximation techniques strive for obtaining a good quality of the approximated surfaces while keeping the number of building blocks small. On the other hand, some real-time applications such as recognition tasks require a fast algorithm for the analysis of surface characteristics. These conflicting design goals of the surface approximation problem can be conceptually formulated as a minimization problem of an energy function

$$\epsilon = E_{fit}(\Gamma_{\hat{\mathbb{V}}}) + E_{size}(\Gamma_{\hat{\mathbb{V}}}) + E_{speed}\,, \tag{3.1}$$

where $\Gamma_{\hat{\mathbb{V}}}$ represents (as described in section 2.1.2), a triangulation of a set of vertices $\hat{\mathbb{V}}$. The first term E_{fit} corresponds to fitting error between the approximation and the input data, and determines quality of the approximation. We have discussed about various error metrics by which the fitting error is measured. L_2 and L_∞ norms of distance measures between the approximate and the input dataset are commonly used. In section 2.6.2, we considered various distance measures that can be applied in our surface approximation task. The Hausdorff distance requires non-negligible time cost even if it is measured with the localized version. As a consequence, we use the normal distance for our distance measure.

The second term E_{size} penalizes large numbers of triangles, since one of our primary goals is to create a *compact* approximation that has as small number of triangles as possible, so long as the fitness criterion is satisfied. Intuitively, we observe that more triangles produce more accurate approximations. That is, a tradeoff exists between the quality and the compactness. Moreover, there is another tradeoff between the quality and the speed of the algorithm that is formulated as E_{speed} in (3.1). The speed is not a function of triangulation itself, but of the number of triangles. Typically, it is very challenging to design such an algorithm that fulfills those criteria: quality, compactness, and speed. To achieve fast performance for real-time applications, the new approach proposed in this dissertation focuses on developing a fast algorithm, compromising quality and compactness in some respects.

3.2 Outline of Algorithm

This section describes our new method for triangular mesh generation, which is a refinement algorithm based on wavelet coefficient analysis. It begins with a coarse polygonal mesh and iteratively adds new vertices or subdivides the meshes to create a finer approximation. The high level outline of the algorithm is as follows:

1. Decompose a given height field with a wavelet transform.
2. Construct an initial triangulation using predefined templates at the coarsest level.
3. Repeat until the finest level is reached or the approximation meets a user defined error criterion:
    (a) Select a candidate region.
    (b) Refine the triangles in the candidate region.
    (c) Mesh enhancement (regularization and reduction).

Most refinement algorithms have a structure similar to this. However, the new approach in this dissertation has some unique features different from basic refinement algorithms. First of all, the new approach is a multiresolution approximation method that utilizes multiscale decomposition of the height field. Secondly, the new approach constructs an initial triangular mesh using predefined templates, which are designed to exploit underlying edge information of the given height field. Next, it refines only candidate regions of each resolution level that contain large variation. This reduces processing time. At a low-resolution level, we may see a significant loss of detail information, but the refinement process will recover the details as the resolution level increases. Finally, the proposed algorithm employs a mesh enhancement step that consists of the regularization and the reduction. The regularization, related to the section 2.6.4, controls the mesh quality with a user defined control parameter, while the reduction devotes to the mesh reduction by removing redundant triangles. The reduction operation is the opposite of refinement and is necessary to generate compact approximations. In step 2, we may need the mesh enhancement step to optimize the triangular mesh in terms of error and the number of triangles at the resolution level.

Figure 3.1 presents an example to illustrate the key concepts of our approach. First, the algorithm generates a multiscale representation of input data (Figure 3.1(a)) using wavelet decomposition. Suppose that we decompose the data up to level $M = 3$. At the coarsest level, we select a rectangular mesh, which covers the entire parametric domain Ω, and denote it with $\hat{\Gamma}_3$. All vertices of the rectangular mesh are located at $(2^M i, 2^M j)$, $i, j = 0, 1, 2, \cdots$ (b). We consider this rectangular mesh as the coarsest level of polygonal representation of a given height field. Then, based on the analysis of wavelet detail coefficients inside each rectangular region, the algorithm triangulates the rectangular mesh using predefined templates to produce $\tilde{\Gamma}_2$, which is an approximation at level $M - 1 = 2$ (c). Using predefined templates, the algorithm generates the initial triangulation $\tilde{\Gamma}_2$, that is, the coarsest approximation very fast. However, it may not be optimal in the sense of error and of the number of elements. Consequently, the step 2 needs a mesh enhancement using the local operators described in section 2.5, by which some triangles are modified through local analysis to generate a more accurate and compact mesh $\hat{\Gamma}_2$. This concludes the processing for level $M - 1 = 2$ (d). To generate the triangular mesh at each subsequent level, step 3 is repeated until the finest triangulation $\hat{\Gamma}_0$ is reached. We use notations $\tilde{\Gamma}_m$ and $\tilde{\Gamma}_m$ to denote an immediate result of refinement and its enhanced version at level *m*, respectively. Therefore, the algorithm proceeds as

$$\hat{\Gamma}_3 \rightarrow \tilde{\Gamma}_2 \rightarrow \hat{\Gamma}_2 \rightarrow \tilde{\Gamma}_1 \rightarrow \hat{\Gamma}_1 \rightarrow \tilde{\Gamma}_0 \rightarrow \hat{\Gamma}_0 \tag{3.2}$$

Figure 3.1 (e) and (f) show the triangulations $\hat{\Gamma}_1$ and $\hat{\Gamma}_0$ and the corresponding approximations.

In summary, the novelty of my new method is twofold: (a) design of triangulation templates, and (b) use of wavelet coefficients to speed up the refinement process. Some researchers [Gross et al. 96, Yu and Ra 99] have developed similar schemes, but theirs are not refinement algorithms. Instead, they remove vertices or edges based on the significance analysis of wavelet detail coefficients. In contrast, my algorithm is purely a refinement method, which is suitable to construct a multiresolution representation, and the wavelet coefficients are used for selecting a proper template in the initial triangulation step and candidate region in refining step. Also, the wavelet coefficients get involved in refinement schemes and redundant removal. The local operators used in the redundant removal are *edge swap, edge collapse* and *vertex removal* explained in section 2.5.1, and they have often been used for polygonal surface simplification [Hoppe et al. 93, Eck et al. 95, Garland and Heckbert 95]. However, our work is the first to utilize wavelet coefficients directly in these steps.

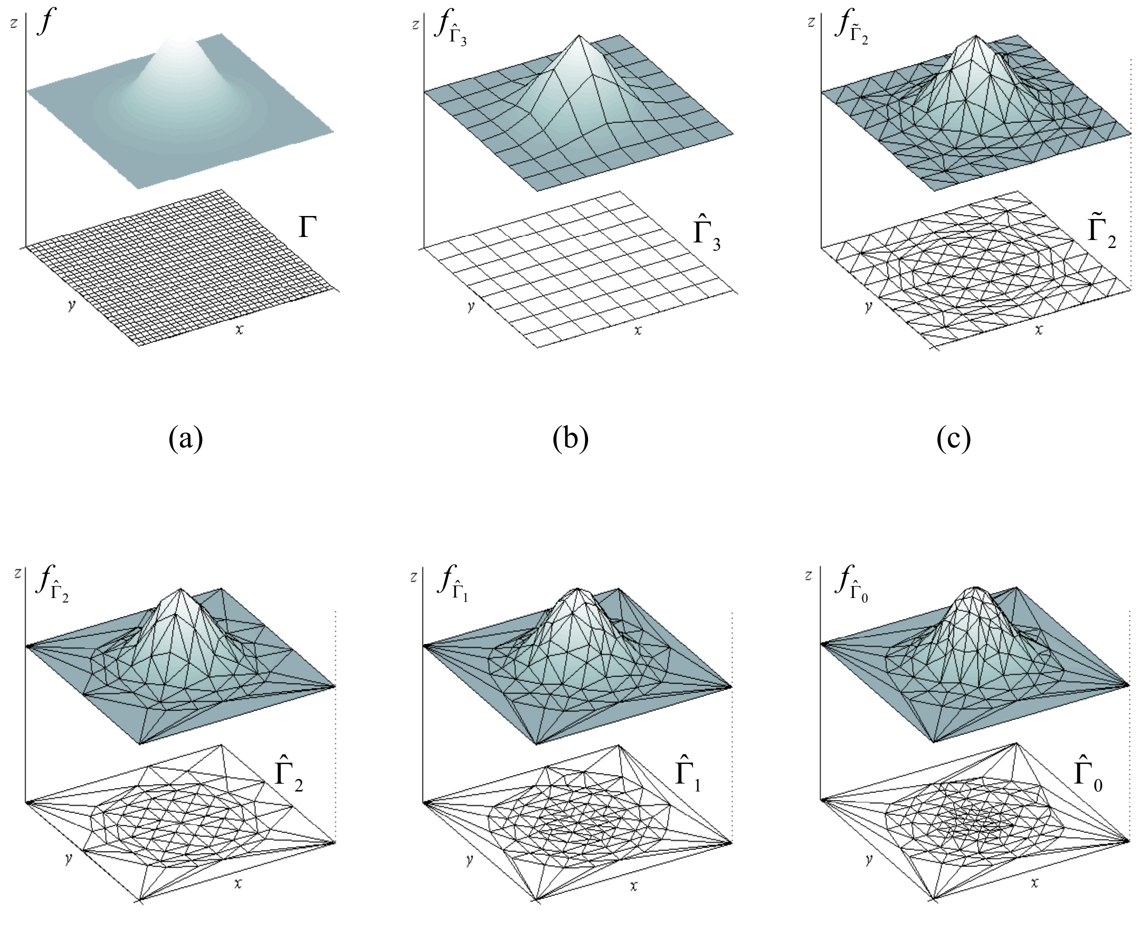


Figure 3.1 Illustration of mesh generation using $M = 3$. Each figure contains a mesh and its data approximation. (a) Input dataset. (b) Rectangular grid $\hat{\Gamma}_3$ for the coarsest level. (c) Initial triangulation $\tilde{\Gamma}_2$. (d) Enhanced result, $\hat{\Gamma}_2$. (e) A subsequent triangulation $\hat{\Gamma}_1$. (f) Finest-level triangulation $\hat{\Gamma}_0$ and its approximation.

3.3 Two-Dimensional Wavelet Decomposition

A common approach in multiresolution analysis is to convert a given dense dataset to a hierarchy of representations, each associated with a different scale. As the first step of our algorithm, wavelet decomposition provides such multiresolution information of input height fields. The chosen wavelets should have some desirable properties: locality, smoothness, and symmetry. In this section, we will discuss the basic theory behind the 2D wavelet decomposition and approximation. The discussion will also address the wavelet selection strategy for our approximation goal. Let *m* = 0 represent the resolution level of the original data. Larger values of *m* correspond to decreasing levels of resolution (or increasing scale), in which fine details are lost but more global features of the data are retained.

3.3.1 2D Wavelet Decomposition and Approximation

In order to decompose two-dimensional (2D) signals, it is common to define a separable wavelet orthonormal basis of $\mathbf{L}^2(\mathbb{R}^2)$, a set of finite energy functions. Let $\phi(x)$ be a scaling function, and let $\psi(x)$ be the corresponding wavelet that constructs an orthonormal basis of $\mathbf{L}^2(\mathbb{R})$. Then, three wavelets are obtained by tensor product,

$$\begin{aligned} \psi^1(x,y) &= \psi(x)\phi(y) \\ \psi^2(x,y) &= \phi(x)\psi(y) \\ \psi^3(x,y) &= \psi(x)\psi(y) \end{aligned} \tag{3.3}$$

and the wavelet family

$$\left\{\psi^1_{m,i,j}(x,y), \psi^2_{m,i,j}(x,y), \psi^3_{m,i,j}(x,y)\right\}_{(i,j)\in\mathbb{Z}^2} \tag{3.4}$$

is a 2D orthonormal basis of a detail space at resolution level *m* [Mallat 98]. Each wavelet $\psi^k_{m,i,j}$ is derived by scaling and shifting the corresponding ψ^k :

$$\psi^k_{m,i,j}(x,y) = 2^{-m}\psi^k(2^{-m}x - i, 2^{-m}y - j) \text{ for } 1 \le k \le 3 . \tag{3.5}$$

Therefore, with a 2D scaling function defined by

$$\phi_{m,i,j}(x,y)=2^{-m}\phi(2^{-m}x-i)\phi(2^{-m}y-j)\,, \tag{3.6}$$

any $f(x,y)\in\mathbf{L}^2(\mathbb{R}^2)$ can be represented by

$$f(x,y)=\sum_{(i,j)} a_{M,i,j}\phi_{M,i,j} + \sum_{m=1}^{M}\sum_{(i,j)}\sum_{k=1}^{3} d^k_{m,i,j}\,\psi^k_{m,i,j}\ , \tag{3.7}$$

where $a_{M,i,j}=\left\langle f,\phi_{M,i,j}\right\rangle$ and $d^k_{m,i,j}=\left\langle f,\psi^k_{m,i,j}\right\rangle$ are called the *approximation coefficients* and *detail coefficients*, respectively. The symbol $\langle\cdot,\cdot\rangle$ denotes an inner product that performs orthogonal projection of the signal onto the basis, and *M* is the highest decomposition level. The first term of (3.6) represents the coarsest approximation of the function $f(x,y)$, and the second term shows that the original function can be recovered by successive addition of detail information at resolution levels $1\le m\le M$. Hence, a detail coefficient with a large magnitude contributes a substantial amount of the corresponding wavelet component to the approximation. Consequently, an approximation of a function *f* can be represented by

$$\hat{f}(x,y)=\sum_{(i,j)} a_{M,i,j}\phi_{M,i,j} + \sum_{m=1}^{M}\sum_{(i,j)}\sum_{k=1}^{3} \chi_m(d^k_{m,i,j})\psi^k_{m,i,j} \tag{3.8}$$

where

$$\chi_m(d^k_{m,i,j})=\begin{cases} d^k_{m,i,j} & \left|d^k_{m,i,j}\right|>\tau_m \\ 0 & otherwise \end{cases} \tag{3.9}$$

and τ_m is a threshold value for detail coefficients at level *m*.

Another interpretation of the wavelet decomposition is that regions containing high spatial frequencies generate detail coefficients of large magnitude. That is, a discontinuity in the horizontal direction tends to cause a large magnitude in $d^1_{m,i,j}$, while a discontinuity in the vertical direction yields a large magnitude in $d^2_{m,i,j}$. Similarly, a region that contains corners or discontinuities that are oriented diagonally tends to produce a large magnitude in $d^3_{m,i,j}$.

3.3.2 Wavelet Vanishing Moments and Support Size

A wavelet with p vanishing moments is orthogonal to polynomials of degree $p-1$. That is,

$$\int_{-\infty}^{+\infty} t^k \psi(t)dt = 0 \qquad \textit{for } \; 0 \le k < p. \tag{3.10}$$

Therefore, all the polynomial signals

$$s(t) = \sum_{0 \le k < p} a_k t^k \tag{3.11}$$

have zero wavelet coefficients. This property ensures the suppression of signals that are polynomials of a degree lower than p. The support size of a wavelet and the number of vanishing moments have a special relation that if a wavelet has p vanishing moments then its support is at least of size $2p-1$. Therefore, when choosing a particular wavelet, we face a trade-off between the number of vanishing moments and the support size. If a signal f has few isolated singularities and is very regular between singularities, we must choose a wavelet with many vanishing moments to produce a large number of small wavelet coefficients $\langle f, \psi_{m,i} \rangle$. If the density of singularities increases, it might be better to decrease the size of its support at the cost of reducing the number of vanishing moments.

3.3.3 Haar Wavelets

In principle, any kind of wavelet can be employed for the decomposition purpose. Among them, the Haar wavelet is first explored, because it is the simplest one and has the shortest support among all orthogonal wavelets, which makes the analysis of decomposition coefficients simpler than other wavelets having larger support. This property will become clear when the region selection issue arises later.

The 1-dimensional Haar wavelet is defined as [Vetterli and Kovacevic 95]

$$\psi(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t < 1 \\ 0 & \textit{otherwise} \end{cases}, \tag{3.12}$$

and the whole set of orthogonal basis functions are obtained by dilation and translation as

$$\psi_{m,i}(t) = 2^{-m/2}\psi(2^{-m}t - i), \quad m,i \in \mathbf{Z}, \tag{3.13}$$

where *m*, *i* are the scale factor and shift factor, respectively. That is, a wavelet function $\psi_{m,i}(t)$ is of length 2^m and shifted by $2^m i$. The corresponding scaling function in the Haar case is

$$\phi(t) = \begin{cases} 1 & 0 \le t < 1 \\ 0 & otherwise \end{cases}, \tag{3.14}$$

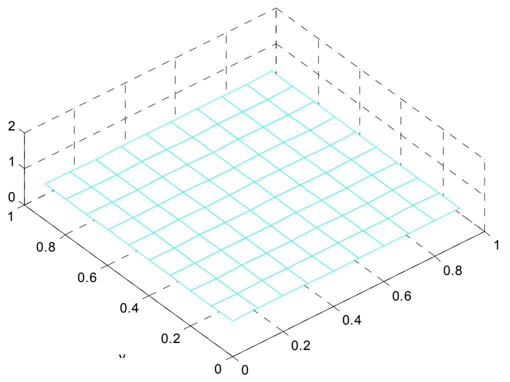
and its dilated and translated version is

$$\phi_{m,i}(t) = 2^{-m/2}\phi(2^{-m}t - i), \quad m,i \in \mathbf{Z}. \tag{3.15}$$

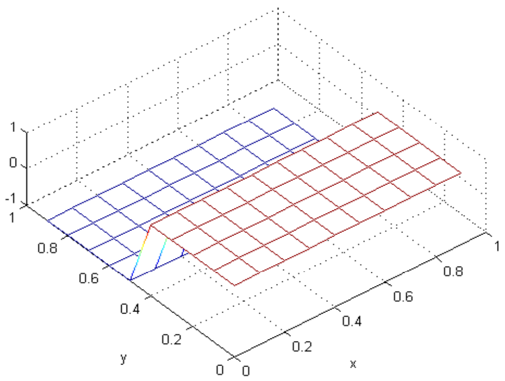
Then, as described in the previous section, the 2D scaling function and wavelet functions for the Haar case are as follows:

$$\begin{aligned} \phi_{m,i,j}(x,y) &= 2^{-m}\phi(2^{-m}x - i)\phi(2^{-m}y - j) \\ \psi^1_{m,i,j}(x,y) &= 2^{-m}\psi(2^{-m}x - i)\phi(2^{-m}y - j) \\ \psi^2_{m,i,j}(x,y) &= 2^{-m}\phi(2^{-m}x - i)\psi(2^{-m}y - j) \\ \psi^3_{m,i,j}(x,y) &= 2^{-m}\psi(2^{-m}x - i)\psi(2^{-m}y - j) \end{aligned} \tag{3.16}$$

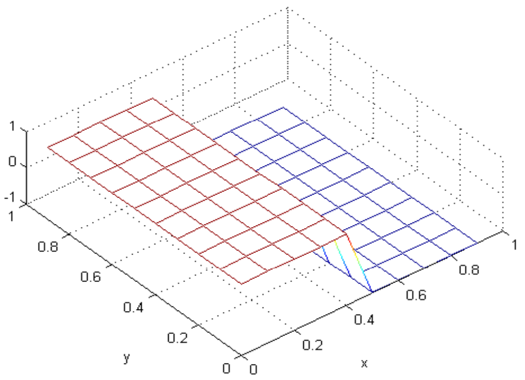
While any wavelet orthogonal basis obeys the decomposition framework described above, the Haar wavelet has a few important features. First, in the Haar case, since the number of coefficients for the scaling and wavelet function is only two, we need only average and difference operations to decompose signals. Figure 3.2 shows the 2D Haar scaling function and wavelets, which are piecewise constant. This piecewise constant feature of the Haar wavelet results in a piecewise constant approximation of smooth function, which is undesirable for attempting to get optimal solution. However, the goal of this study is to find a triangular mesh that covers a parametric domain, with which a given height field is approximated by piecewise planar patches. For that reason, we only want to know which part of the data has high frequency information so that we can do more work only on that part, leaving other parts untouched.



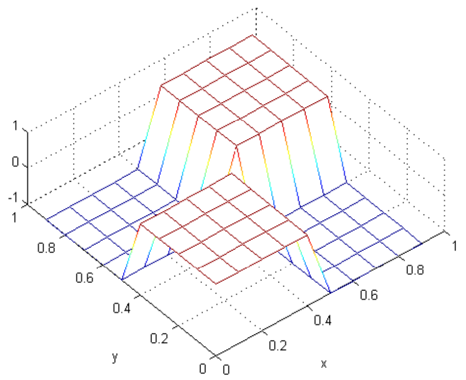
(a) $\phi_{0,0,0}(x,y)$



(b) $\psi^1_{0,0,0}(x,y)$



(c) $\psi^2_{0,0,0}(x,y)$



(d) $\psi^3_{0,0,0}(x,y)$

Figure 3.2 Two dimensional Haar scaling function and wavelets.

For the purpose of data analysis in a local area, the space localization property, related to the support of wavelets, becomes important in our problem. Since the Haar wavelet has the shortest support of length 2, which means it has only 2 non-zero coefficients, its support is exactly in accordance with the sampling grid at a given resolution in the discrete-time wavelet decomposition. In other words, the supports at one scale do not overlap with each other, which means that the Haar wavelet has perfect space localization property. Figure 3.3 shows this property in the 1D case. Wavelets $\psi_{m,i}$ and $\psi_{m,j}$ do not overlap if $i \neq j$ for every resolution level m. This localization property plays an important role in avoiding an ambiguity problem that commonly arises from wavelets having wide support.

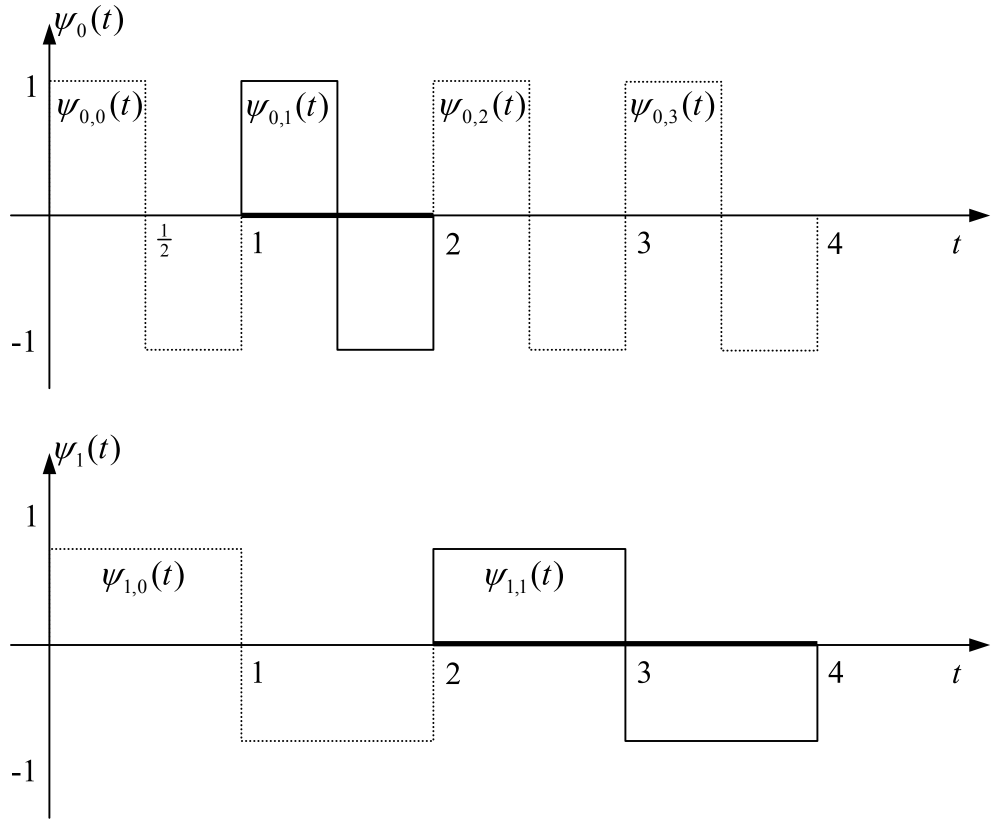


Figure 3.3 Illustration of the perfect space locality of one-dimensional Haar wavelet.

3.3.4 Biorthogonal Wavelets

As noted in the previous section, the locality and the smoothness of wavelets are desirable properties in our application. In addition, symmetric wavelets are preferred in some applications, such as signal processing and computer graphics [Stollnitz et al. 96]. Unfortunately, it is known that the Haar wavelet is the only real-valued wavelet that is compactly supported, symmetric and orthonormal [Daubechies 88]. However, even if the scaling function and the wavelet are not orthogonal, it is possible to perfectly reconstruct the signal from the transform by using biorthogonal wavelet bases. The framework for the biorthogonal bases of compactly supported wavelets is presented in [Cohen et al. 92].

Let $\left\{\phi_n^j\right\}_{n\in\mathbb{Z}}$ and $\left\{\tilde{\phi}_n^j\right\}_{n\in\mathbb{Z}}$ be bases of two approximation spaces V_j and $\tilde{V}_j$, and $\left\{\psi_n^j\right\}_{n\in\mathbb{Z}}$ and $\left\{\tilde{\psi}_n^j\right\}_{n\in\mathbb{Z}}$ be bases of two detail spaces W_j and $\tilde{W}_j$ such that

$$V^j \oplus W^j = V^{j+1} \text{ and } \tilde{V}^j \oplus \tilde{W}^j = \tilde{V}^{j+1}. \tag{3.17}$$

where $\oplus$ represents the direct sum of two vector spaces. In the biorthogonal framework, V_j is not orthogonal to W_j but is to $\tilde{W}_j$ whereas $\tilde{V}_j$ is not orthogonal to $\tilde{W}_j$ but is to W_j. As a consequence, any $f(x)\in\mathbf{L}^2(\mathbb{R})$ can be represented by

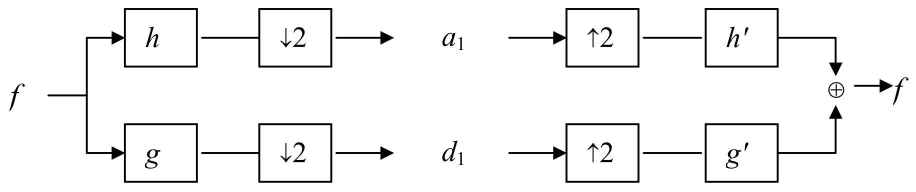
$$f(x)=\sum_{i\in\mathbb{Z}}\left\langle f,\phi_{M,i}\right\rangle\tilde{\phi}_{M,i}+\sum_{m=1}^{M}\sum_{i\in\mathbb{Z}}\left\langle f,\psi_{M,i}\right\rangle\tilde{\psi}_{m,i}\,. \tag{3.18}$$

Decomposition coefficients in a wavelet biorthogonal basis are computed with a filter bank algorithm, in which wavelet decomposition and reconstruction is calculated with discrete convolutions of *conjugate mirror filters* associated with scaling functions and wavelets. The implementation of the filter bank algorithm is depicted in Figure 3.4. Conjugate mirror filters h and $\tilde{h}$ correspond to scaling functions ϕ and $\tilde{\phi}$, respectively, and g and $\tilde{g}$ correspond to wavelets ψ and $\tilde{\psi}$, respectively. Then, the decomposition coefficients are calculated by four convolutions,

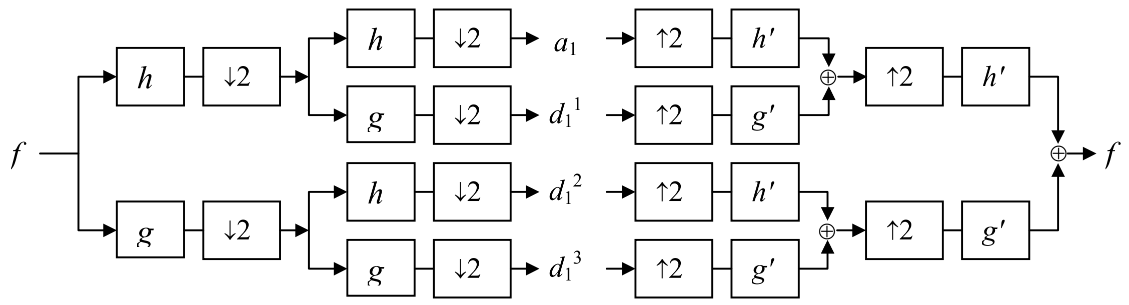
$$\begin{aligned} a_{m+1}[i,j] &= a_m * hh[2i,2j] \\ d_{m+1}^1[i,j] &= a_m * hg[2i,2j] \\ d_{m+1}^2[i,j] &= a_m * gh[2i,2j] \\ d_{m+1}^3[i,j] &= a_m * gg[2i,2j] \end{aligned}, \tag{3.19}$$

where $a_0 = f$.

The most commonly used biorthogonal wavelet is the B-spline wavelets, which was independently developed by [Chui 92] and [Unser et al. 93]. Several different wavelets are depicted in Figure 3.5. The first column of the figure shows scaling functions and the second column shows corresponding wavelets. A linear spline wavelet with 4 vanishing moment,



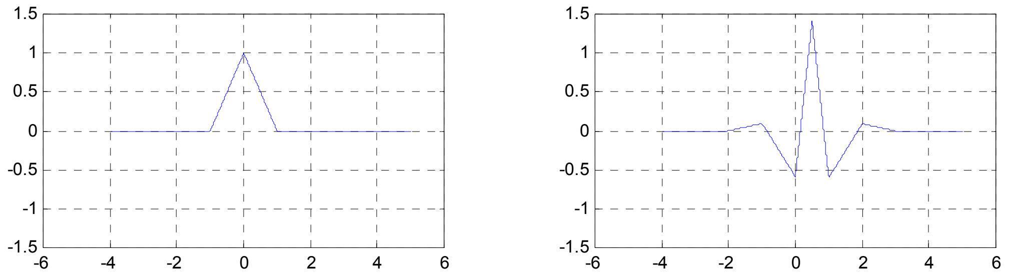
(a)



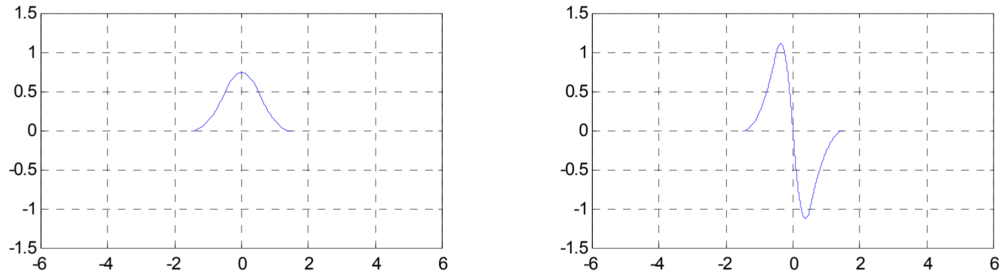
(b)

Figure 3.4 Filter bank algorithms for biorthogonal wavelet transforms. (a) 1D signal and (b) 2D signal.

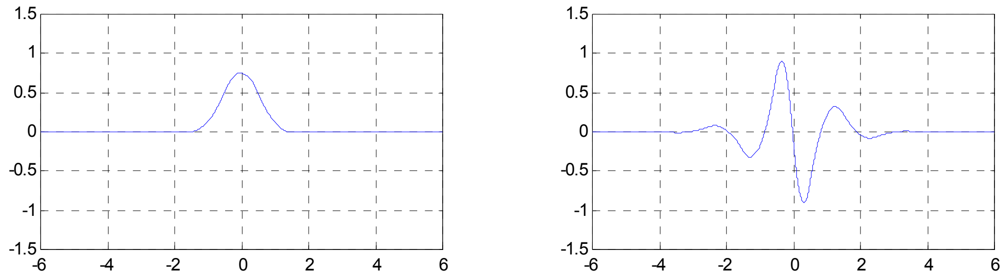
whose scaling function is the 1st B-spline function, is depicted in Figure 3.5(a). The 2nd order B-spline wavelets with different vanishing moments are shown in (b) and (c). The 2nd order B-spline having 1 vanishing moment (Figure 3.5(b)) is known as spline wavelet whose scaling function is a cubic spline and the corresponding wavelet is a quadratic spline. The wavelet shown in (c) has 7 vanishing moment, providing wider support area. A Daubechies wavelet having 7 vanishing moment is depicted in (d). An illustrative example of decomposition of 1D discrete signal is in Figure 3.6. Suppose the input signal contains only a none-zero data at x = 11 (Figure 3.6(a)). In this example, the Daubechies wavelet has widest support and its decomposition yields more approximation and detail coefficients with high magnitude (Figure 3.6(f)), whereas the Haar wavelet produces only one coefficient in each of its approximation and detail spaces (Figure 3.6(b)). Figure 3.7 shows 2D spline wavelets.



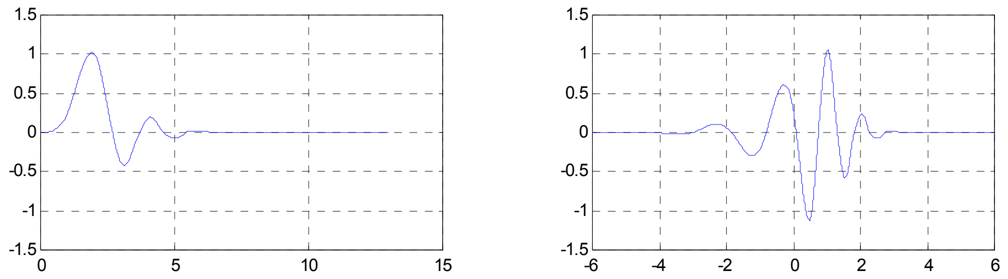
(a)



(b)

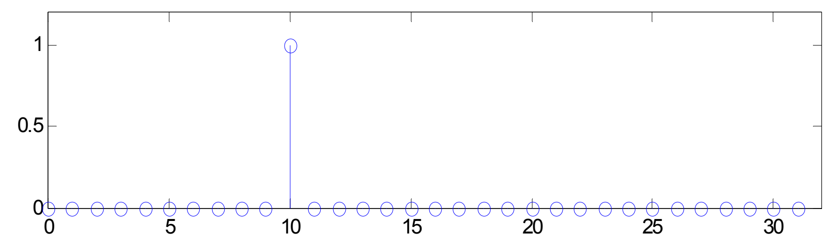


(c)

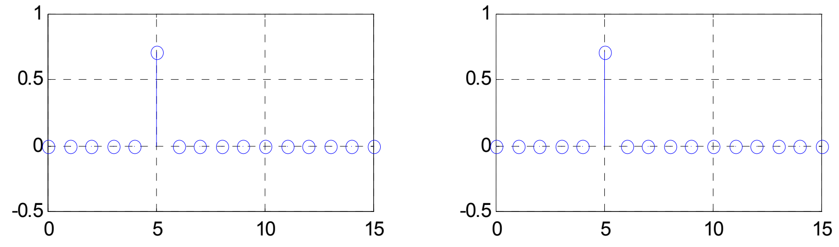


(d)

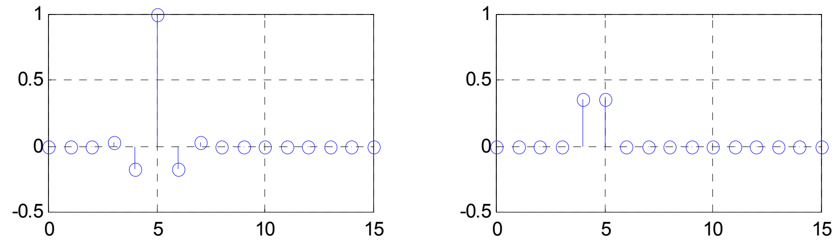
Figure 3.5 Scaling functions and wavelets. (a) Linear B-spline wavelet with vanishing moment of 4. (b) Spline wavelet with vanishing moment of 1. Its scaling function is a cubic spline and the corresponding wavelet is a quadratic spline. (c) 2nd order B-spline wavelet with vanishing moment of 7. Higher vanishing moment provides a wider support. (d) Daubechies wavelet.



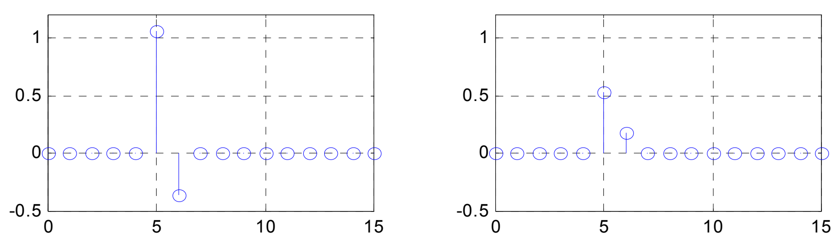
(a)



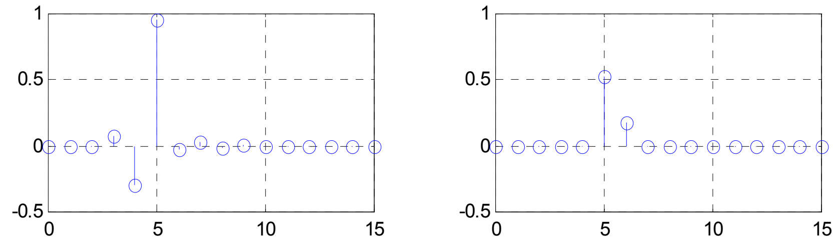
(b)



(c)

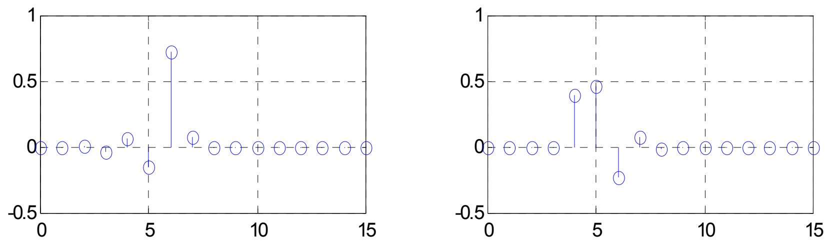


(d)



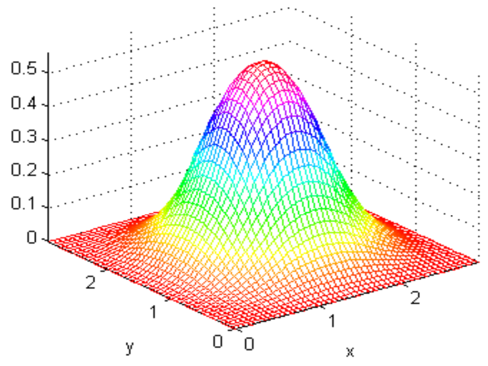
(e)

Figure 3.6 1D signal decomposition. (a) Input signal. (b) Haar. (c) Linear B-Spline. (d) Spline wavelet. (e) 2nd order B-Spline with vanishing moment of 7.

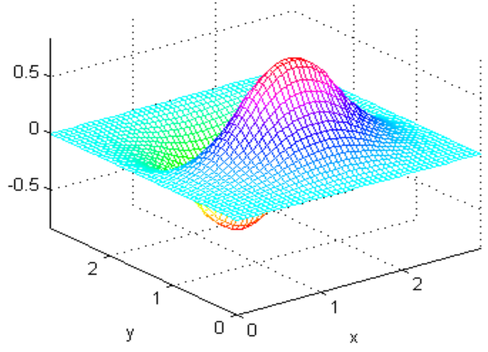


(f)

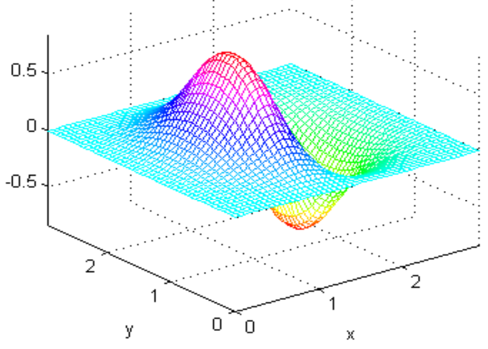
Figure 3.6 (continued) (f) Daubechies wavelet with vanishing moment of 7.



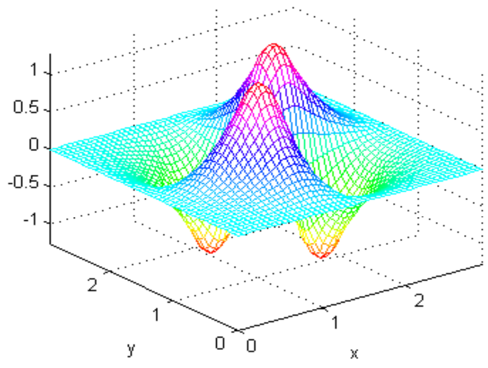
(a) $\phi_{0,0,0}(x,y)$



(b) $\psi^1_{0,0,0}(x,y)$



(c) $\psi^2_{0,0,0}(x,y)$



(d) $\psi^3_{0,0,0}(x,y)$

Figure 3.7 2D Spline wavelet. The scaling function is a 2D cubic spline (a) and the corresponding wavelets are constructed with the tensor product of 1D cubic spline and 1D quadratic spline (b-d).

3.4 Structurally Equivalent Graphs

The new surface approximation algorithm is based on a triangulation of the parametric domain. Since the primary goal of this study is to construct a fast algorithm that accurately approximates a height field by using as few triangles as possible. As our first step, the initial triangulation uses predefined templates. Those templates are viewed as *simple planar graph* that is defined as a set of points in a plane and a set of line segments, each of which joins two points. This section will provide essential definitions used in graph theory and formal description of structural equivalence [Gross and Yellen 99].

A *graph* $G = (V, E)$ is a mathematical structure consisting of two sets V and E. The elements of V are called *vertices* (or *nodes*), and the elements of E are called *edges*. Each edge has a set of one or two vertices associated to it, which are called its *endpoints*. *Adjacent vertices* are two vertices that are joined by an edge, and *adjacent edges* are two edges that have an endpoint in common. A *self-loop* is an edge that joins a single endpoint to itself, and a *multi-edge* is a collection of two or more edges having identical endpoints. A graph is simple if it has neither self-loops nor multi-edges. In addition, a graph is *planar* if there are no edge-crossings. A general graph *G* shown in Figure 3.8(a) contains a self-loop *a*, and multi-edges *c* and *d*. In addition, the edges *h* and *g* cross each other so that the graph *G* is not planar. Figure 3.8(b) shows an example of simple planar graph.

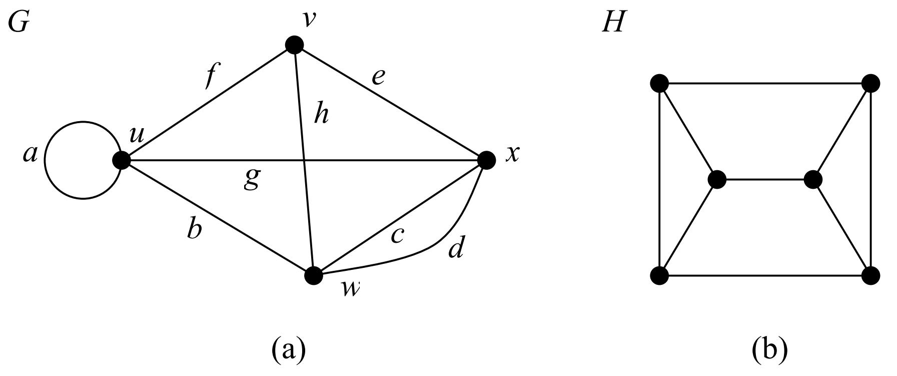


Figure 3.8 A graph. (a) General graph. (b) Simple planar graph.

Since the shape of an edge and its position is space are not part of a graph's specification, each graph has infinitely many spatial representations. Each graph in Figure 3.9 has only three vertices and four edges, and they are labeled differently. However, it is not difficult to recognize that these two drawings represent the same graph, if we realize that there are following bijections on the vertex names and edge names:

$$\begin{array}{c} u \to x \\ v \to z \\ w \to y \end{array} \qquad \begin{array}{c} a \to g \\ b \to h \\ c \to f \\ d \to e \end{array}. \tag{3.20}$$

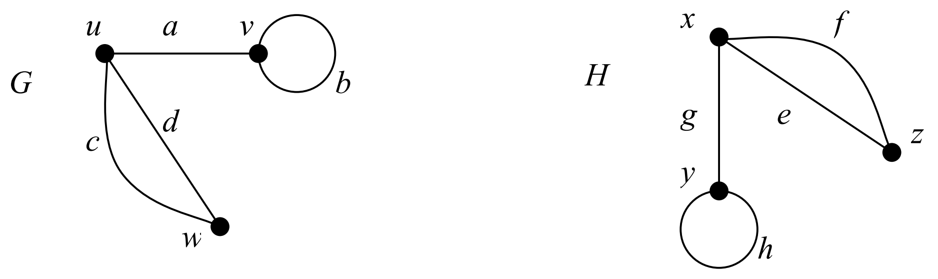


Figure 3.9 Two different drawings of the same graph.

A graph *isomorphism* $g: G \to H$ is a pair of bijections

$$g_V : V_G \to V_H \text{ and } g_E : E_G \to E_H$$

such that for every edge $e \in E_G$, the function g_V maps the endpoints of e to the endpoints of the edge $g_E(e)$. A composite function g of two isomorphisms g_1 and g_2 denoted by

$$g = g_1 \circ g_2 = g_1(g_2) \tag{3.21}$$

is also an isomorphism. Two graphs *G* and *H* are said to be *isomorphic* if there exists an isomorphism from *G* to *H*. Furthermore, for two simple graphs *G* and *H*, if there is a vertex bijection $g: V_G \to V_H$ such that $g(x)$ is adjacent to $g(y)$ if and only if *x* is adjacent to *y* for all $x, y \in V_G$, then the two graphs are isomorphic. Therefore, a vertex bijection can be an isomorphism for simple graphs.

3.5 Construction of Shape-Preserving Templates

Initial triangulation is an important step in our triangulation scheme, because it provides the initial subdivision that affects all subsequent steps. Nevertheless, this step is performed very fast via 47 predefined triangulation templates. Most of the templates are generated by isomorphic transformations of 4 basic templates and 9 fundamental forms. This section shows how the basic templates and the fundamental forms were selected.

3.5.1 Wavelet Detail Coefficient and Discontinuity

In our application, the set V_M consists of points at $\{(i2^M, j2^M) | \, i, j = 0, 1, 2, \ldots\}$, representing a regular rectangular grid in the 2D plane. Accordingly, the proposed method considers the regular rectangular grid as a mesh at the coarsest level. By the dilation property of the wavelet decomposition, each wavelet detail coefficient $d^k_{M,i,j}$ primarily represents the information contained in each rectangular patch with opposite corners given by $(i2^M, j2^M)$ and $((i+1)2^M, (j+1)2^M)$. Hence if $\left|d^1_{M,i,j}\right| = \max\limits_{1\le k\le 3}\left(\left|d^k_{M,i,j}\right|\right)$ and $\left|d^1_{M,i,j}\right| > \tau_M$, then there is a depth discontinuity mainly in the horizontal direction within the corresponding rectangular patch, and the detail coefficient $d^1_{M,i,j}$ is called *dominant coefficient* in the rectangular patch.

Figure 3.10 illustrates the selection of the dominant coefficients and corresponding templates. The dark region represents an object on the background (light region). The coefficients in each square in Figure 3.10(a) indicate dominant coefficients inside each corresponding square. The square on left top corner does not have a dominant coefficient because the region is relatively flat. The rectangular patch on right top corner contains vertical discontinuity, and thus its dominant coefficient is $d^1_{M,i,j}$. Our algorithm attempts to preserve this discontinuity by selecting a triangular subdivision so that a vertically oriented edge is inserted within the rectangular patch. Similarly, if a vertically oriented depth discontinuity is present (bottom left corner in (a)), then a different triangulation is chosen so that a horizontal edge is introduced. Figure 3.10(b) shows the resulting triangulation.

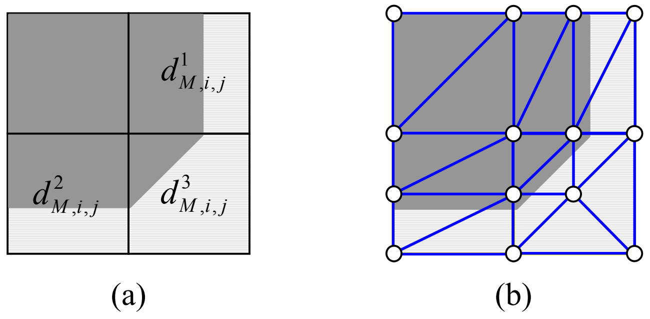


Figure 3.10 Illustration of the concept of initial triangulation. (a) Dominant coefficients. (b) Triangulation with the predefined templates.

3.5.2 Basic Templates for Initial Triangulation

The analysis of wavelet detail coefficients described in the previous section has implied the use of at least four different templates: one for each dominant coefficient. Figure 3.11 shows the construction of the four basic templates that we have chosen to subdivide a given rectangular patch. The first template (Figure 3.11(a)) denoted π_f is the default case; it is intended for relatively flat (planar) portions of the input data. A specific diagonal edge is chosen for the template π_f, and the same diagonal direction will be used for the vertical and horizontal templates. The diagonal direction may reflect the directional property of the discontinuities enclosed in the square region. The templates having opposite diagonal direction will be discussed in a later section.

To explain how the templates are constructed, let $d^1_{M,i,j}$ be the dominant coefficient for a square region, representing the fact that the square region contains vertical discontinuities. Like the feature-based triangulation methods [Scarlatos and Pavlidis 92], our algorithm places a vertical edge in the middle of the square (Figure 3.11(b)), facilitating better approximation of discontinuities in the horizontal direction. This requires two additional vertices in the vertex set $\mathbb{V}_M$. Among four different ways to triangulate two rectangles, the one shown in the right of (b) is chosen as our vertical template. Similarly, the horizontal and diagonal templates are constructed as shown in Figure 3.11(c) and (d). They introduce respectively horizontal and diagonal edges to a given square.

It is interesting to note that vertical template π_v and horizontal template π_h are isomorphic if we use an isomorphism $g_d : \pi_v \to \pi_h$ that is defined by

$$g_d = g_v \circ g_r \tag{3.22}$$

where g_r and g_v are also isomorphism that represent rotation and vertical flip of vertices. Therefore, the two templates have a relation

$$g_d(\pi_v) = \pi_h \text{ or } g_d(\pi_h) = \pi_v . \tag{3.23}$$

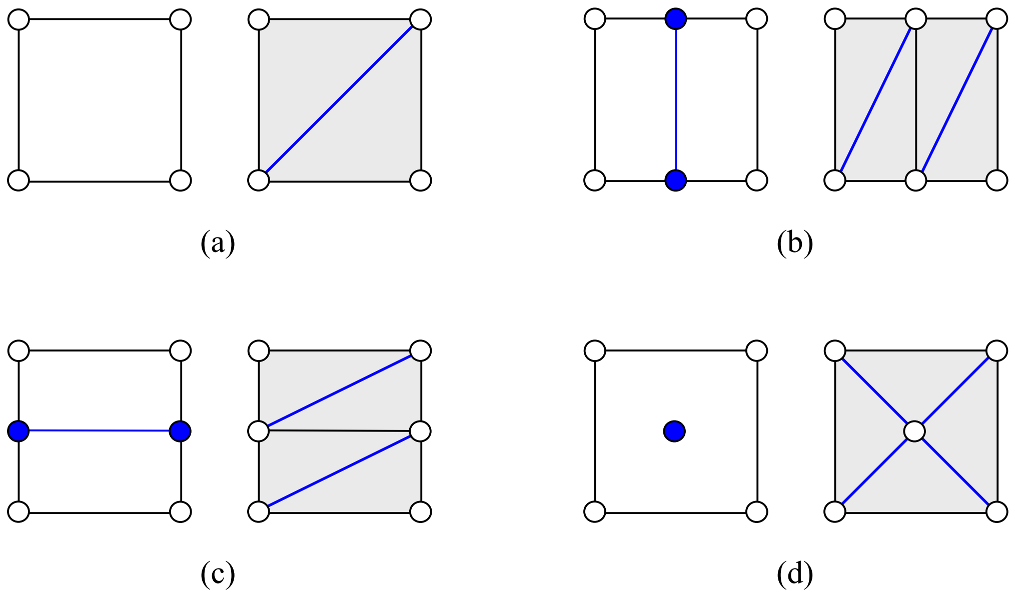


Figure 3.11 Construction of basic templates: (a) flat (π_f), (b) vertical (π_v), horizontal (π_h), and (d) diagonal (π_d).

3.5.3 Directional Property of Discontinuities and Dual Templates

We interpret the horizontal and the vertical wavelet coefficients as approximations of two partial derivatives of the input data at a given scale. This is especially appropriate for the case of the Haar wavelet, which is the most common case and is the one used here. That is,

$$d_m^1 \approx \frac{\partial s^{m-1}}{\partial x},\ d_m^2 \approx \frac{\partial s^{m-1}}{\partial y}, \tag{3.24}$$

and s^{m-1} is the average information at level m-1. Thus, the orientation of the discontinuity can be estimated by

$$\theta^m(x,y) \approx \tan^{-1}\left(d_m^2 / d_m^1\right). \tag{3.25}$$

Figure 3.12 illustrates the concept. Interior of each rectangular regions in (a) and (b) contains depth discontinuities in two different directions. The light regions correspond to large depth values and the dark regions to small depth values. Suppose that we decompose the rectangular region with the 2D Haar wavelets (Figure 3.2). Positive and negative symbols in the rectangular regions represent the shape of the Haar wavelets. For the first case (a), the wavelet decomposition gives

$$d_{m,i,j}^1 > 0 \text{ and } d_{m,i,j}^2 > 0,$$

but for the second case (b),

$$d_{m,i,j}^1 < 0 \text{ and } d_{m,i,j}^2 > 0.$$

Let us assume that the dominant coefficient for the region is $|d_{m,i,j}^2| = \max_{1\le k\le 3}(|d_{m,i,j}^k|)$ and $|d_{m,i,j}^2| > \tau_m$. Then, the basic template π_h can be used for the first case, and its triangulation is shown in (c). On the other hand, for the second case a better approximation may be produced by selecting a *dual* template of π_h, which is obtained by a vertical flip of π_h and denoted with $\hat{\pi}_h$.

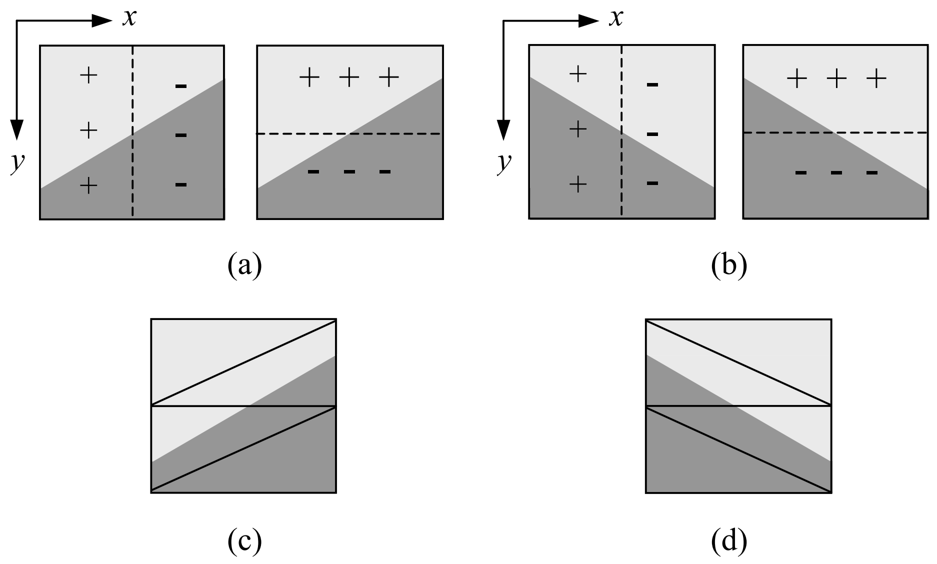


Figure 3.12 Illustration of directional property of discontinuities. (a) Discontinuities enclosed in the rectangular region produce Haar wavelet coefficients $d^1_{m,i,j} > 0$ and $d^2_{m,i,j} > 0$. (b) The opposite direction gives $d^1_{m,i,j} < 0$ and $d^2_{m,i,j} > 0$. (c) Basic template π_h is selected for this region. (d) A dual template of π_h may produce a better approximation for the opposite direction of discontinuities.

With similar means, duals for vertical and flat templates can be constructed and depicted in Figure 3.13. The diagonal template is symmetric, and thus it is the same with its dual, i.e. $\pi_d = \widehat{\pi}_d$. Also, the dual and basic templates have a special relation. Each dual template can be generated from an isomorphic transform g_v, which has been defined as vertical flip in the previous section. Therefore, the following relations are established:

$$\begin{aligned} g_v(\pi_f) &= \widehat{\pi}_f \\ g_v(\pi_v) &= \widehat{\pi}_v \\ g_v(\pi_h) &= \widehat{\pi}_h \end{aligned} \quad . \tag{3.26}$$

Note that the dual templates $\widehat{\pi}_v$ and $\widehat{\pi}_h$ are isomorphic with the isomorphism g_d defined in previous section.

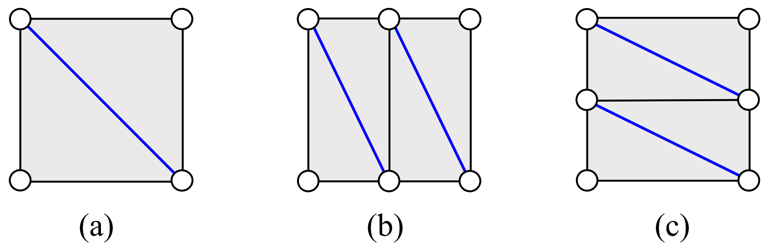


Figure 3.13 Dual templates. (a) $\hat{\pi}_f$. (b) $\hat{\pi}_v$. (c) $\hat{\pi}_h$.

3.5.4 Vertex Compatibility and Fundamental Forms of Variants

Ideally, 4 basic templates and their duals in the previous sections could be chosen independently for each rectangle of $\hat{\Gamma}_M$. However, this would necessarily not result in a proper triangulation, as illustrated in Figure 3.14(a). If basic templates π_h and π_v are placed side by side as shown, then vertex *v* from template π_h is not matched with a vertex from π_v, yielding an invalid mesh (section 2.6.4). For this reason, we augment the set of basic templates with others that retain *vertex compatibility* between adjacent rectangles at level *M*-1. New templates are called *variants*. For the example in Figure 3.14(a), two variants for the vertical templates are possible to maintain vertex compatibility. Note that we are preserving the directions of diagonal edges. The variant shown in (b) produces more regular triangular meshes while the one in (c) yields better approximation by preserving the estimated orientation of discontinuities.

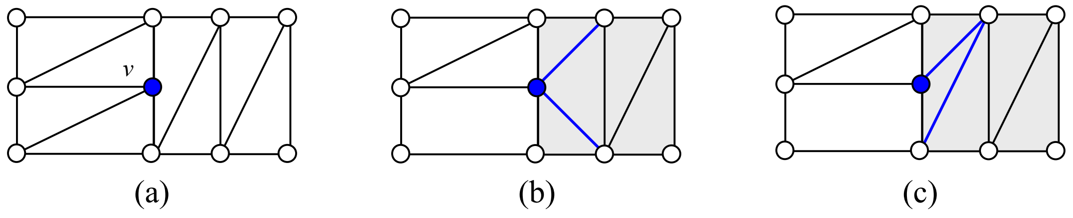


Figure 3.14 Vertex compatibility. Basic templates lying side by side may be incompatible (a), but this can be corrected by a small change to one of the two templates at the right (b) or (c).

Thus, the construction of variants utilizes template information at adjacent rectangles. Figure 3.15 shows the adjacent rectangles involved in the variant construction. The shaded regions are the rectangles to be checked to decide which variants are used for the rectangle in center. For example, if the center has to use π_f, four sides of the center are checked for the vertex compatibility: when the rectangle on top or bottom uses π_v, or when the rectangle on left or right uses π_h, the template π_f has to be altered to its one of variants. Similarly, four adjacent rectangles are considered for π_d to determine a variant to be used ((d)), and two adjacent rectangles are involved for π_v and π_h as shown in (b) and (c), respectively.

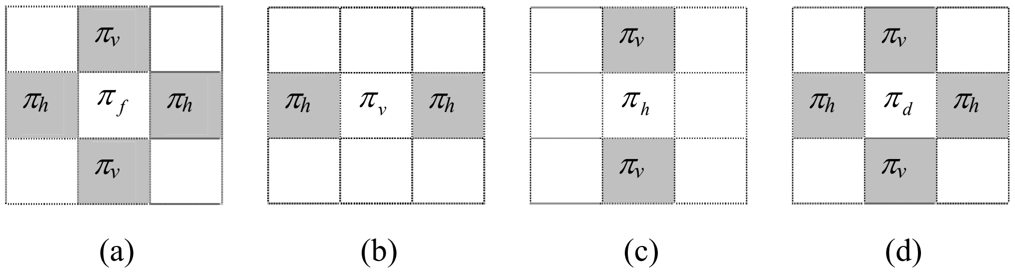


Figure 3.15 Adjacent rectangles for vertex compatibility. (a) π_f. (b) π_v. (c) π_h. (d) π_d.

The number of combinations that selects *n* from *m* adjacent rectangles is given by ${}_m\mathrm{C}_n$, where *m* represents the number of shaded rectangles for each template in Figure 3.15. Thus, the total number of variants for a template is

$$N = \sum_{i=1}^{m} {}_m\mathrm{C}_i \tag{3.27}$$

The templates π_f and π_d have

$$\sum_{i=1}^{4} {}_4\mathrm{C}_i = 15$$

variants each. For π_v and π_h, 3 variants are possible for each. Accordingly, a total of 36 variants can be constructed.

However, most of the possible variants can be induced from several fundamental forms, depicted in Figure 3.16. The construction of the form π_1 is illustrated in Figure 3.14 and avoids vertex incompatibility on the left side of vertical template π_v. The form π_2 is the case that both sides of π_v require vertex compatibility (see Figure 3.15(b)). The forms π_3 and π_4 establish the vertex compatibilities in cases that the template π_f has vertex incompatibilities on its one or two adjacent sides. The form from π_5 to π_9 corresponds to diagonal template π_d.

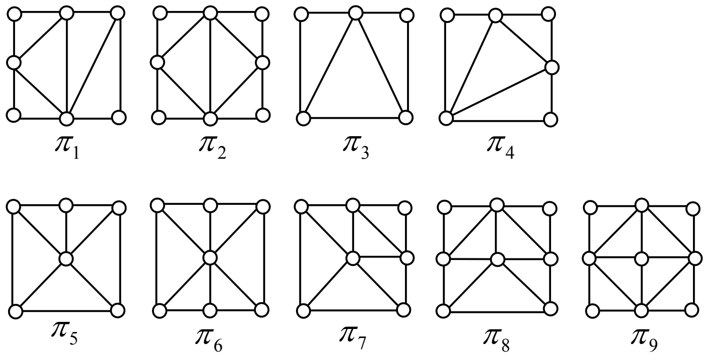


Figure 3.16 Fundamental forms for variant construction.

The fundamental forms are designed to ensure vertex compatibility as well as to make smooth edge connection. An example of the smooth connectivity of edges is illustrated in Figure 3.17. Suppose that we have dominant wavelet coefficient such as shown in Figure 3.17(a). Then, the corresponding basic templates are assigned to each rectangle ((b)), and the adjacent rectangles are evaluated to determine variants. In this example, the template π_f requires a variant because its adjacent rectangles contain π_v on top and π_h on right side (refer Figure 3.15(a)). Figure 3.18 shows 6 possible variants for this case. We have chosen the first variant in Figure 3.18 as our one of fundamental forms for π_f, because the triangular mesh created with the form is most regular and connects two vertices at which incompatibility occurs. Therefore, the use of the chosen variant results in a triangulation that is locally regular and has a smooth edge connection. The resulting triangulation is shown in

Figure 17(d). An edge in the fundamental form π_4 connects vertical edge in π_v and horizontal edge in π_h.

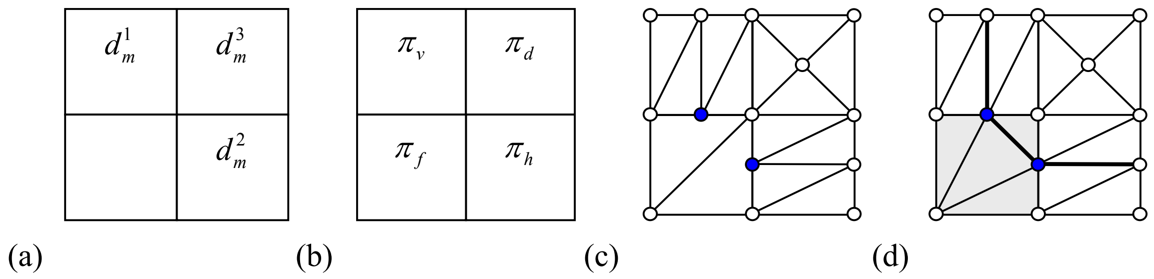


Figure 3.1 Edge connection capability of fundamental form π_4. Instead of using π_f in the square region in left lower corner, the use of the form π_4 not only resolves the vertex compatibility problem but also makes a smooth connection between the vertical edge in upper region and the horizontal edge in right region. In addition, the form produces a locally regular mesh.

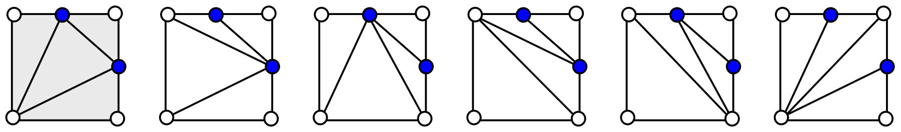


Figure 3.2 Six possible variants for π_f when there are vertex incompatibilities at two locations.

3.5.5 Complete set of template

In previous sections, we have discussed the construction of basic templates and their duals, and have chosen the fundamental forms for variants. The basic templates and their duals are isomorphic with an isomorphism g_v defined as a vertical flip operation. Moreover, all variants needed for vertex compatibility can be generated form the fundamental forms, if we have appropriate isomorphism.

Therefore, it is convenient to define an isomorphism so that every template can be induced from the basic templates and the fundamental forms. Let g_h, g_v, and g_r be isomorphism that respectively perform horizontal flip, vertical flip, and rotation of vertices and edges. Then, a composite function

$$g_{hvr} = g_h \circ g_v \circ g_r \tag{3.28}$$

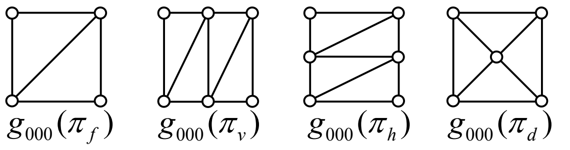
is also an isomorphism that executes a rotation, vertical flip, and horizontal flip. Generally, the isomorphism g_{hvr} is not commutative:

$$g_{hvr} \neq g_{hrv} \neq g_{rvh} \neq g_{vhr} \neq g_{rvh} \neq g_{rhv}\,. \tag{3.29}$$

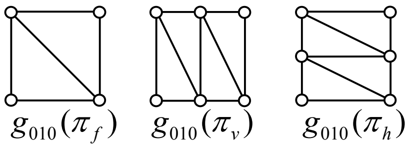
In addition, we use a notation g_{ijk} with $i, j, k = \{0,1\}$, to denote

$$\begin{array}{lll} g_{000} = \mathbf{I} & g_{111} = g_{hvr} & \\ g_{100} = g_h & g_{010} = g_v & g_{001} = g_r \ , \\ g_{110} = g_{hv} & g_{101} = g_{hr} & g_{011} = g_{vr} \end{array} \tag{3.30}$$

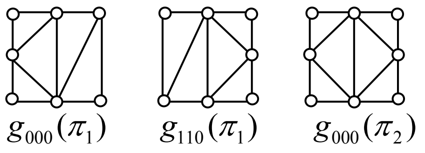
where $\mathbf{I}$ represents an identity operation such that $\mathbf{I}(\pi) = \pi$. We call them generating functions, and the complete set of templates is collected in Figure 3.19.



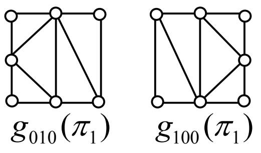
(a) Basic templates.



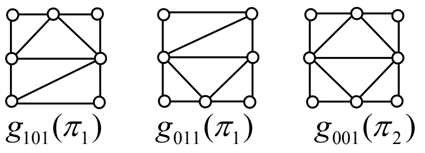
(b) Duals for basic templates.



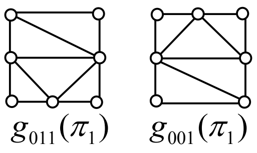
(c) Vertical variants.



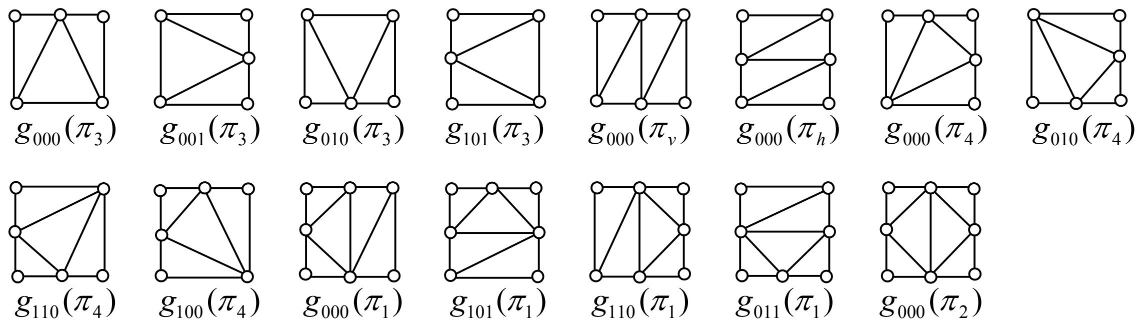
(d) Duals for vertical variants.



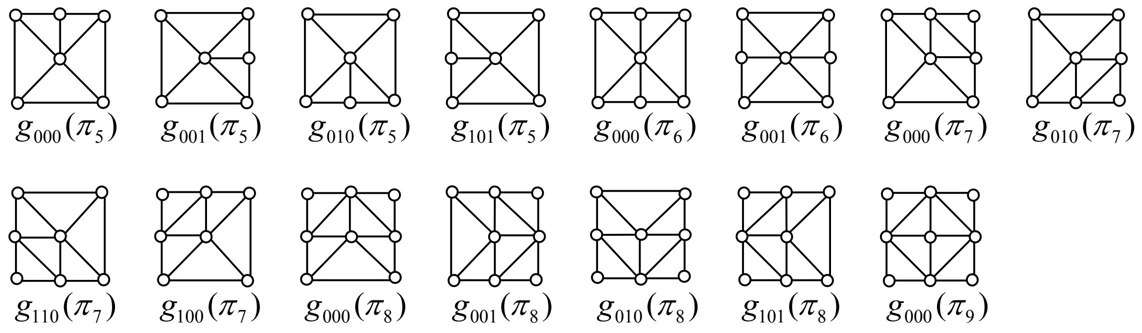
(e) Horizontal variants.



(f) Duals for horizontal variants.



(g) Variants for flat templates.



(h) Variants for diagonal templates.

Figure 3.19 Complete template set and generating functions.

3.6 Initial Triangulation

In previous sections, we have discussed 2D wavelets and their properties, and designed templates for the initial triangulation. Our primary concern is to construct a fast and efficient algorithm for triangular surface approximation from any height field. By means of efficiency, the algorithm should produce as small a number of triangles as possible while it satisfies a given error tolerance. The proposed triangulation templates are suitable for this purpose because the algorithm simply evaluates wavelet coefficients and assigns templates for rectangular regions. This section will devote to the quantitative and qualitative evaluation of our initial triangulation.

3.6.1 Assessing Cost of Wavelet Transform

The wavelet transform of a finite image of $N \times N$ pixels is equivalent to computing four convolutions in Equation (3.16). If we use biorthogonal wavelets, the conjugate mirror filters associated with those wavelets are FIR (finite impulse response) filters and the convolutions can be done with multiplication and addition. When the conjugate mirror filters have size K, the filter bank algorithm in Figure 3.4 requires $2K2^{-2(m-1)}N^2$ multiplications and additions to decompose an image up a resolution level m [Mallat 98]. This is directly verified when we consider that the size of the approximation coefficient a_m at decomposition level m is $2^{-m}N \times 2^{-m}N$. Therefore, the number of convolutions required for the decomposition of a_m is $2^{-2m+3}N^2$. Suppose that $N = 512$ and the decomposition level $M = 5$. Then, the total number of convolutions is

$$2N^2 + 2^{-1}N^2 + 2^{-3}N^2 + 2^{-5}N^2 + 2^{-7}N^2 = 698{,}368 .$$

The latest floating point DSP processors in the market perform about 1,000,000 convolutions per second when the FIR filter has 16 coefficients, and more than 2,700,000 convolutions per second with 8 coefficients [TI 00]. In order to obtain locality of the wavelet transform, typical filter length of biorthogonal wavelets used in the decomposition is less than or equal to 8. Hence, the wavelet transform can be done very fast compared to the rest of our triangulation, which is discussed in later chapters.

3.6.2 Tessellation

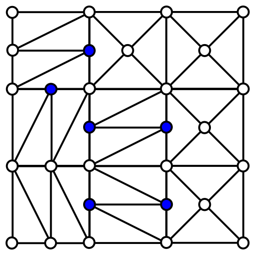
After the wavelet transform, the algorithm evaluates the detail coefficients as discussed in section 3.5.1 to tessellate the square regions with appropriate templates. As mentioned earlier, each wavelet detail coefficient $d^k_{M,i,j}$ captures the information contained mostly in a square given by $[2^M i, 2^M (i+1)) \times [2^M j, 2^M (j+1))$. Accordingly, assessing which templates are needed in each square is straightforward. In order to get a proper triangulation that satisfies the vertex compatibility of the mesh, however, we need to take surrounding information into account as we did in section 3.5.4.

To make the examination easier, we generate a table that contains template labels. An example of such a table is shown in Figure 3.20 (a). The labels `F`, `H`, `V`, and `D` represent that templates to be used in corresponding squares are π_f, π_h, π_v, and π_d, respectively. The lower case letters `f`, `h`, `v`, and `d` indicate the use of duals. Suppose that we are going to tessellate the shaded region in Figure 3.20(a). The use of basic templates and their duals may result in vertex incompatibilities as shown in (b). To avoid these vertex incompatibilities, several variants are introduced as shown in (c).

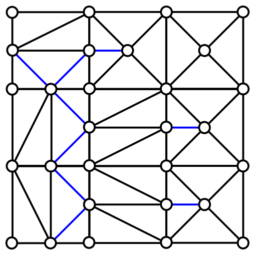
The table size is quite small. For instance, when decomposition level $M = 5$ and image size $N = 512$, the table is of size $2^{-M} N \times 2^{-M} N = 16 \times 16$. As a result, the proposed initial triangulation method is really fast because as explained it uses predefined templates, as well as the manipulation cost is low because of the use of such a small size 2D array.

F	F	F	F	f	f	f	f
F	F	H	H	h	h	f	f
F	H	D	D	D	D	v	f
F	V	H	D	D	H	v	f
f	v	h	D	D	H	V	F
f	v	D	D	D	D	V	F
f	f	h	H	H	H	F	F
f	f	f	F	F	F	F	F

(a)



(b)



(c)

Figure 3.20 Initial triangulation. (a) Table of labeling. (b) Triangulation using only basic templates and their duals. Vertex incompatibilities occur at the vertices depicted with filled circles. (c) Triangulation based on vertex compatibility.

3.6.3 Analysis of Approximation Error

Some experiments are shown in this section to demonstrate the performance of the proposed initial triangulation method. In these experiments, we have used two different sets of synthetic data. Synthetic data was used because it facilitates an assessment of accuracy, and because the use of known, regular shapes provides quick visual assessment of the resulting mesh.

As a first data set, we used three basic quadratic surfaces: elliptic, parabolic, and hyperbolic. Many surface approximation methods [Faugeras et al. 83, Bolle and Vemuri 91, Miller and Goldman 95, Seibold and Joy 99] use the quadratic form as a fundamental building block to approximate underlying surface shape in a given data set. General quadratic surface in $\mathbb{R}^2$ is defined by

$$q(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F\ . \tag{3.31}$$

By introducing coordinate change by translation and rotation, the quadratic equation becomes

$$q'(x,y) = A'x^2 + C'y^2\ . \tag{3.32}$$

Now we can easily decide the geometric type of the quadratic by the following rule [Thomas and Finney 95]:

$$A'C' = \begin{cases} > 0 & \textit{elliptic} \\ = 0 & \textit{parabolic} \\ < 0 & \textit{hyperbolic} \end{cases}\ . \tag{3.33}$$

Our quadratic surfaces used in this experiment are

$$\begin{aligned} f^1(x,y) &= -5x^2 - 10y^2 \\ f^2(x,y) &= 5x^2 \\ f^3(x,y) &= 5x^2 - 10y^2 \end{aligned}\ .$$

Figure 3.21 shows the resulting initial triangulations of the three quadratic surfaces. In this experiment, we generate 128×128 sample points and decompose them up to level $M = 4$ with the Haar wavelet, yielding an 8×8 labeling table. The first column of the figure shows the three quadratic surfaces. The resulting initial triangulations are in the second column. The dotted curves in the triangulation indicate level contour of each surface. Note that the edges of the triangulations tend to follow the contour curves, that is, our new initial triangulation method tries to put edges along with the contour curves of given surfaces. The last column shows approximations of each surface with the corresponding triangulations. The visual quality of the approximations is excellent for the three quadratic surfaces.

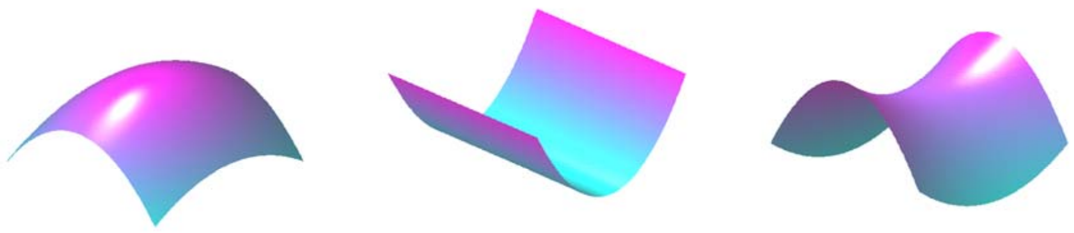
Approximation errors for the quadratic surfaces are measured by normal distance with the L_∞ norm (see sections 2.6.1 and 2.6.2). The error is calculated for each planar surface bounded by each triangular patch, and summarized in Table 3.1. Here, the errors between surface f and its piecewise planar approximation $f_{\tilde{\Gamma}_{M-1}}$ is calculated by

$$\begin{aligned} E_{\max} &= \max_{\tilde{\Gamma}_{M-1}} \max_{(x,y)\in\Omega(T_i)} d(f, f_{\tilde{\Gamma}_{M-1}}(T_i)) \\ E_{total} &= \sum_{i=1}^{|\tilde{\Gamma}_{M-1}|} \max_{(x,y)\in\Omega(T_i)} d(f, f_{\tilde{\Gamma}_{M-1}}(T_i)) \quad , \\ E_{mean} &= \frac{1}{|\tilde{\Gamma}_{M-1}|} E_{total} \end{aligned} \tag{3.34}$$

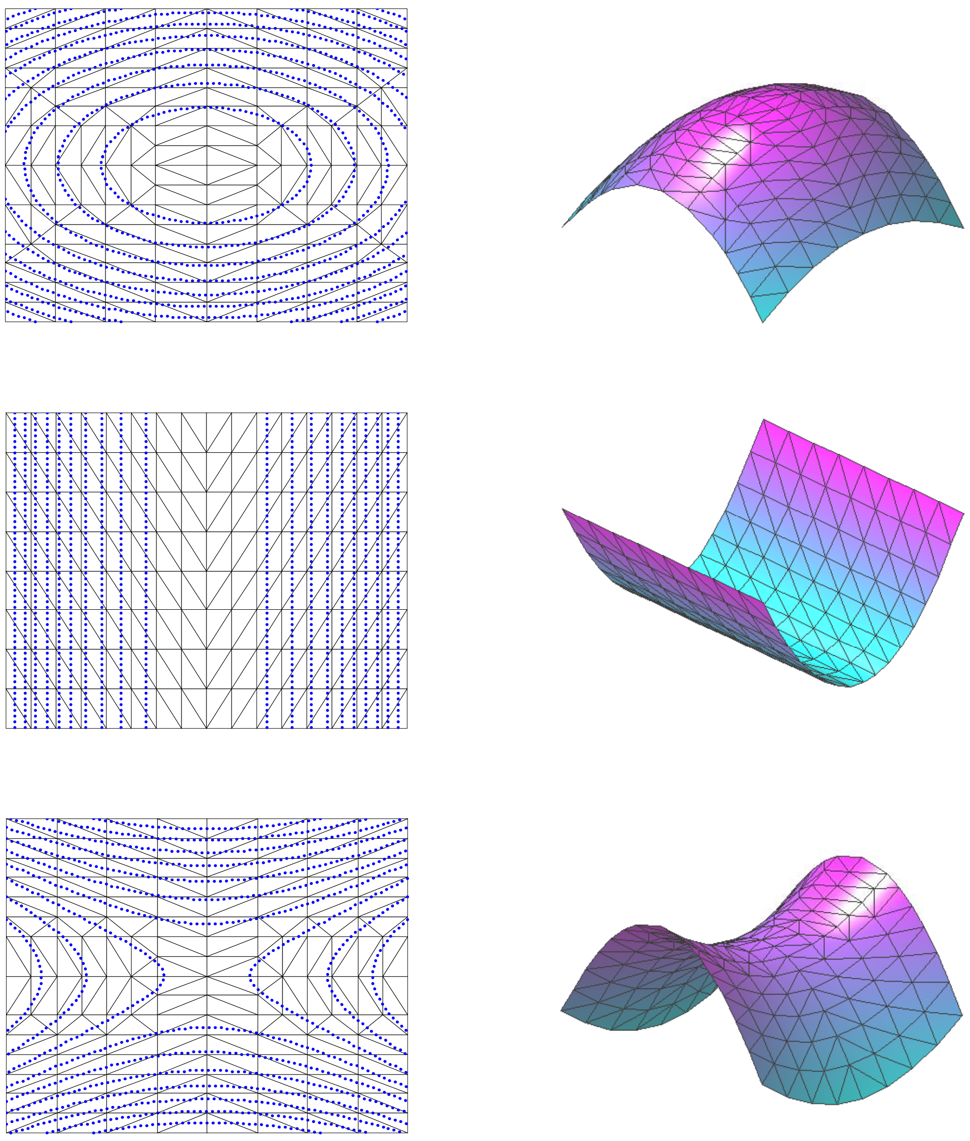
where $T_i \in \tilde{\Gamma}_{M-1}$ is a triangle in the initial triangulation $\tilde{\Gamma}_{M-1}$, and $\Omega(T_i)$ indicates the parametric domain corresponding T_i. The parabolic, the simplest surface, has the smallest error among three quadratic surfaces, while the elliptic surface has largest but still reasonably low error. When the error tolerance for each triangle is set to $\varepsilon_0 = 1$, it can be said that these initial triangulations result in quite good approximations. The parabolic surface already satisfies the error tolerance, and the others do not need much refinement to meet the tolerance.

Table 3.1 Approximation errors for quadratic surfaces

	Elliptic $f^1_{\tilde{\Gamma}_{M-1}}$	Parabolic $f^2_{\tilde{\Gamma}_{M-1}}$	Hyperbolic $f^3_{\tilde{\Gamma}_{M-1}}$
$\lvert\tilde{\mathbb{V}}_{M-1}\rvert$	159	153	159
$\lvert\tilde{\Gamma}_{M-1}\rvert$	276	256	272
E_{max}	1.392	0.199	1.177
E_{total}	172.14	39.98	111.895
E_{mean}	0.624	0.156	0.405



(a)



(b)

Figure 3.21 Initial triangulation of quadratic surfaces. The dashed curves represent constant-value contours of the quadratic surfaces. The triangle edges in initial triangulations are well aligned with the level contours.

The second synthetic data set consists of two simple images shown in Figure 3.22(a) and (b). Image f^4 contains smooth changes through the diagonal, and image f^5 has discontinuities along the boundary of circle. The images are of size 128×128 and have pixel values varying 0 to 110. They are decomposed up to level $M = 4$ with Haar wavelet, yielding 8×8 labeling tables and corresponding initial triangulations $\tilde{\Gamma}_{M-1}$. Figure 3.23 (a) and (e) show the initial triangulations, and the approximated images are shown in Figure 3.23(b) and (f). The gray values within each triangle are obtained by bilinear interpolation in the vertex gray values. Figure 3.23(c) and (g) show enhanced meshes by applying local operations as described in section 2.5.1. The local operations enhance the initial triangulation $\tilde{\Gamma}_{M-1}$ to $\hat{\Gamma}_{M-1}$, resulting in triangulations that contain fewer triangles and lower E_{total}. The numerical results for the approximations are summarized in Table 3.2. The E_{mean} for $f^5_{\hat{\Gamma}_{M-1}}$ is greater than for $f^5_{\tilde{\Gamma}_{M-1}}$ because of the drastic reduction of the number of triangles from $f^5_{\tilde{\Gamma}_{M-1}}$ to $f^5_{\hat{\Gamma}_{M-1}}$.

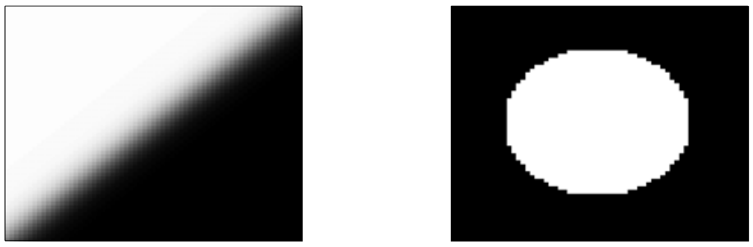
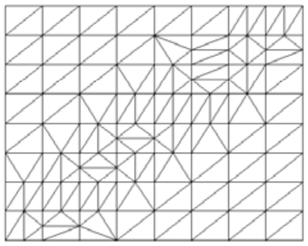
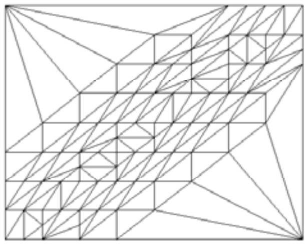

(a) f^4 (b) f^5

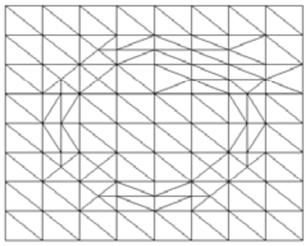
Figure 3.22 Second datasets for initial triangulation. (a) Image f^4 contains smooth changes through the diagonal, and (b) image f^5 has discontinuities along the boundary of circle.

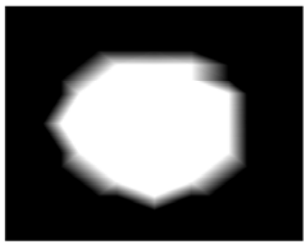

(a) $\tilde{\Gamma}_{M-1}$

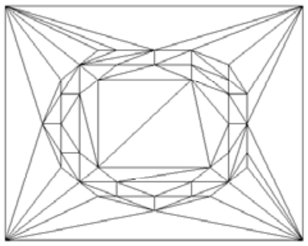
(b) $f^{4}_{\tilde{\Gamma}_{M-1}}$


(c) $\hat{\Gamma}_{M-1}$

(d) $f^{4}_{\hat{\Gamma}_{M-1}}$


(e) $\tilde{\Gamma}_{M-1}$


(f) $f^{5}_{\tilde{\Gamma}_{M-1}}$


(g) $\hat{\Gamma}_{M-1}$

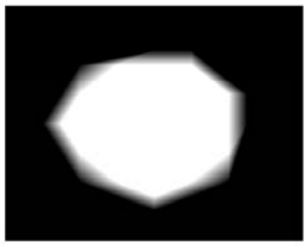

(h) $f^{5}_{\hat{\Gamma}_{M-1}}$

Figure 3.23 Initial triangulations for gray-scale images of the previous figure using Haar wavelet.

Table 3.2 Approximation errors for gray-scale images – Haar wavelet.

	$f^4_{\tilde{\Gamma}_{M-1}}$	$f^4_{\hat{\Gamma}_{M-1}}$	$f^5_{\tilde{\Gamma}_{M-1}}$	$f^5_{\hat{\Gamma}_{M-1}}$
$\left\|\tilde{\mathbb{V}}_{M-1}\right\|$	119	89	106	43
$\left\|\tilde{\Gamma}_{M-1}\right\|$	200	158	178	80
E_{max}	1.53	1.062	4.984	3.53
E_{total}	98.022	38.034	101.38	66.585
E_{mean}	0.49	0.241	0.57	0.832

Figure 3.24 and Table 3.3 show the approximation results using the 1st order B-spline wavelet. Since the B-spline wavelet has wider support than Haar wavelet, discontinuities in image affect wavelet detail coefficients in broad region. It is called a noise spreading effect. The noise spreading effect in wavelets having wider support is more noticeable if the discontinuities are sharp and have large level difference.

Table 3.3 Approximation errors for gray-scale images – B-spline wavelet.

	$f^4_{\tilde{\Gamma}_{M-1}}$	$f^4_{\hat{\Gamma}_{M-1}}$	$f^5_{\tilde{\Gamma}_{M-1}}$	$f^5_{\hat{\Gamma}_{M-1}}$
$\left\|\tilde{\mathbb{V}}_{M-1}\right\|$	118	90	133	49
$\left\|\tilde{\Gamma}_{M-1}\right\|$	198	158	230	92
E_{max}	2.853	1.062	4.984	3.53
E_{total}	149.827	37.544	137.168	79.112
E_{mean}	0.757	0.238	0.596	0.86

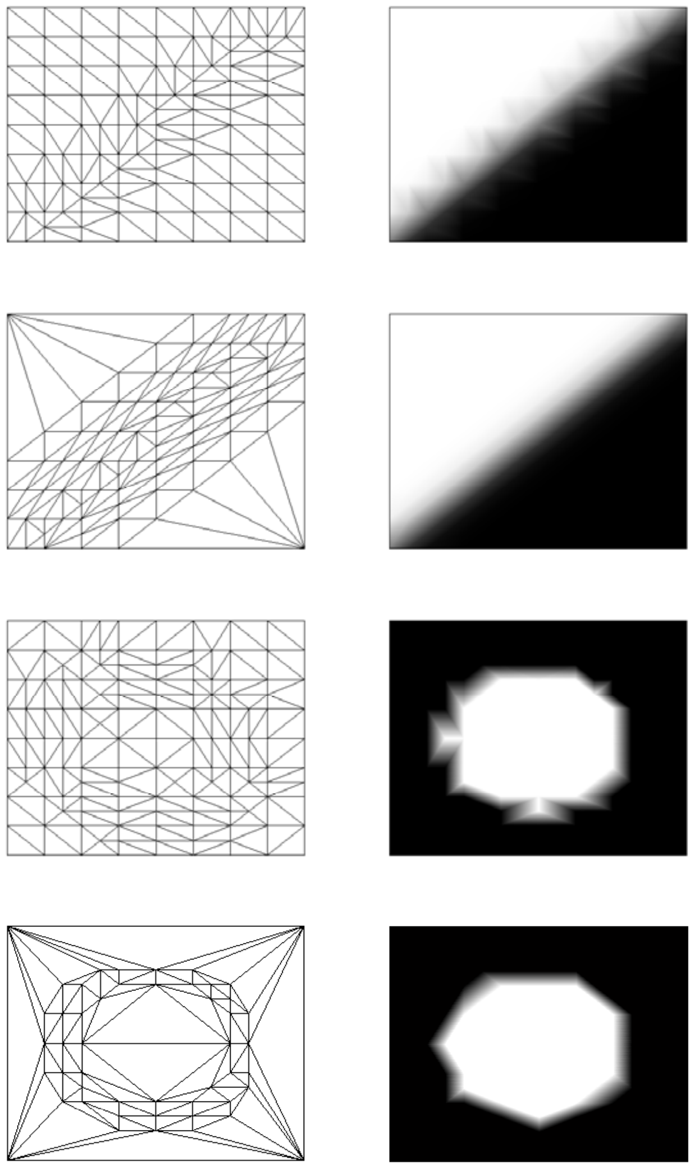


Figure 3.24 Initial triangulations for gray-scale images using 1st order B-spline wavelet.

Chapter 4

Wavelet-Based Iterative Local Refinement

As the completion of the initial triangulation, this chapter will continue the rest of the algorithm introduced in section 3.2. Wavelet detail coefficients have played an important role in constructing templates and initial triangular mesh in previous chapter. These coefficients are used again in selecting a candidate region for further refinement. These coefficients are also used in shape-preserving edge split that is proposed in this dissertation. Since the refinement process may result in a highly irregular mesh or produce redundant triangles, it is required to regularize the mesh and remove the redundant triangles. The regularization relies on the edge swap operator, while the removal uses eigenanalysis of covariance matrix of unit normal vectors for a local area. The edge split and eigenanalysis are closely related to differential geometry on surface, so this chapter starts with a brief introduction to the subject.

4.1 Differential Geometry on Surface

A comprehensive introduction to differential geometry is beyond the scope of this work. [O'Neill 66] provides reasonably clear and modern mathematical description of differential geometry and [Gray 98] provides with good examples and illustrations. An excellent overview on this topic is found in [Besl and Jain 86] with some computational techniques.

4.1.1 Surfaces in Euclidean Space $\mathbf{E}^3$

Formally, a *surface* in $\mathbf{E}^3$ is a subset $\boldsymbol{M}$ of $\mathbf{E}^3$ such that for each point $\mathbf{p}$ of $\boldsymbol{M}$ there exists a proper *patch* $\mathbf{x}$ in $\boldsymbol{M}$ whose image $\mathbf{x}(D)$ contains a neighborhood of $\mathbf{p}$ in $\boldsymbol{M}$. Here, patch is a one-to-one mapping of an open set D of $\mathbf{E}^2$ into $\mathbf{E}^3$. Figure 4.1 illustrates the definition.

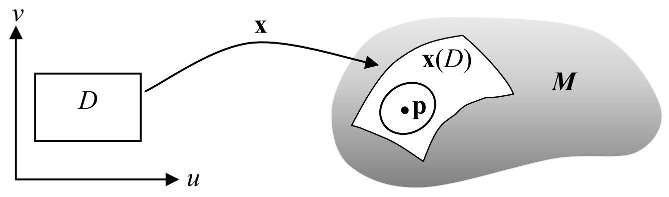


Figure 4.1 Surface in $\mathbf{E}^3$.

Therefore, given a closed differentiable manifold surface $\boldsymbol{M}$ which has been divided into a set of patches, a given surface patch is defined by the mapping

$$\mathbf{x} = \mathbf{x}(u,v) = \left[x_1(u,v) \quad x_2(u,v) \quad x_3(u,v)\right]^{\mathrm{T}} \tag{4.1}$$

where the functions x_i are of class C^2. Consider small neighborhood of the surface at a point $\mathbf{p} = \mathbf{x}(u_0, v_0)$. Then, the partial derivatives of the patch function $\mathbf{x}$ at $\mathbf{p}$

$$\begin{aligned} \mathbf{x}_u(u_0,v_0) &= \begin{bmatrix} \dfrac{\partial x_1}{\partial u} & \dfrac{\partial x_2}{\partial u} & \dfrac{\partial x_3}{\partial u} \end{bmatrix}^{\mathrm{T}} \\ \mathbf{x}_v(u_0,v_0) &= \begin{bmatrix} \dfrac{\partial x_1}{\partial v} & \dfrac{\partial x_2}{\partial v} & \dfrac{\partial x_3}{\partial v} \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{4.2}$$

span the tangent plane of the surface $\boldsymbol{M}$ at the point, provided we make the standard assumption that $\mathbf{x}_u \times \mathbf{x}_v \neq 0$ (Figure 4.2(a)). Consequently, the unit surface normal at $\mathbf{p}$ is written by

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|} = \frac{1}{\|\mathbf{x}_u \times \mathbf{x}_v\|} \begin{vmatrix} U_1 & U_2 & U_3 \\ \dfrac{\partial x_1}{\partial u} & \dfrac{\partial x_2}{\partial u} & \dfrac{\partial x_3}{\partial u} \\ \dfrac{\partial x_1}{\partial v} & \dfrac{\partial x_2}{\partial v} & \dfrac{\partial x_3}{\partial v} \end{vmatrix} \tag{4.3}$$

where U_1, U_2 and U_3 constitute the natural frame field on $\mathbf{E}^3$ (Figure 4.2(b)).

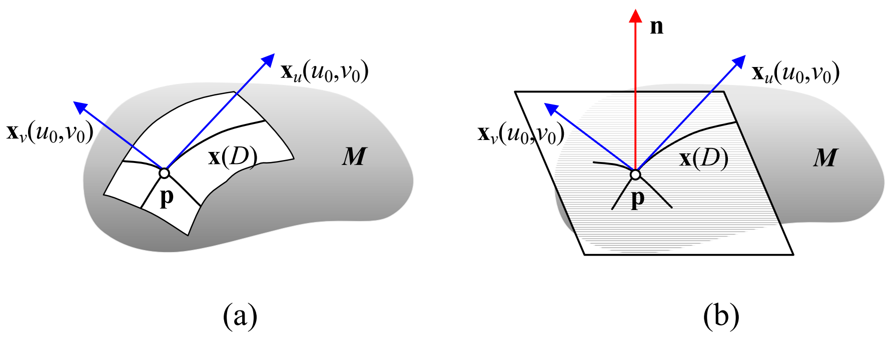


Figure 4.2 Differentiable surface. (a) Tangent vectors. (b) Tangent plane and normal vector.

Height fields defined in section 2.1 can be expressed as functions on D fomulated as

$$\mathbf{x}(u,v) = \begin{bmatrix} u & v & f(u,v) \end{bmatrix}^{\mathrm{T}}. \tag{4.4}$$

This is known as *Monge patch* whose normal vector is

$$\mathbf{n} = \mathbf{x}_u \times \mathbf{x}_v = \begin{vmatrix} U_1 & U_2 & U_3 \\ \frac{\partial u}{\partial u} & \frac{\partial v}{\partial u} & \frac{\partial f}{\partial u} \\ \frac{\partial u}{\partial v} & \frac{\partial v}{\partial v} & \frac{\partial f}{\partial v} \end{vmatrix} = \begin{vmatrix} U_1 & U_2 & U_3 \\ 1 & 0 & \frac{\partial f}{\partial u} \\ 0 & 1 & \frac{\partial f}{\partial v} \end{vmatrix} = \begin{bmatrix} -\frac{\partial f}{\partial u} & -\frac{\partial f}{\partial v} & 1 \end{bmatrix}^{\mathrm{T}}. \tag{4.5}$$

For a Monge patch, normal vector field is completely determined by gradient vector field $\mathbf{g} = \begin{bmatrix} \frac{\partial f}{\partial u} & \frac{\partial f}{\partial v} \end{bmatrix}^{\mathrm{T}}$.

4.1.2 The First and Second Fundamental Forms

The *first fundamental form* is just the dot product of a tangent vector with itself and measures the squared length of a tangent vector in a given direction. Since any tangent vector $\mathbf{t}$ can be expressed as a linear combination of the two first order partial derivatives $\mathbf{x}_u$ and $\mathbf{x}_v$ of $\mathbf{x}(u,v)$, i.e., $\mathbf{t} = \mathbf{x}_u \delta u + \mathbf{x}_v \delta v$, a direction vector $\mathbf{w} = \begin{bmatrix} \delta u & \delta v \end{bmatrix}^{\mathrm{T}}$ provides a convenient representation for tangent vectors. This is frequently expressed in differential form as $d\mathbf{t} = \mathbf{x}_u du + \mathbf{x}_v dv$. Then, the first fundamental form is

$$\begin{aligned} I(\mathbf{w}) &= \mathbf{w}\bullet\mathbf{w} = (u'\mathbf{x}_u + v'\mathbf{x}_v)\bullet(u'\mathbf{x}_u + v'\mathbf{x}_v) \\ &= \mathbf{x}_u\bullet\mathbf{x}_u(u')^2 + 2\mathbf{x}_u\bullet\mathbf{x}_v u'v' + \mathbf{x}_v\bullet\mathbf{x}_v(v')^2 \\ &= \begin{bmatrix} u' & v' \end{bmatrix} \begin{bmatrix} \mathbf{x}_u\bullet\mathbf{x}_u & \mathbf{x}_u\bullet\mathbf{x}_v \\ \mathbf{x}_u\bullet\mathbf{x}_v & \mathbf{x}_v\bullet\mathbf{x}_v \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} \\ &= \mathbf{w}^{\mathrm{T}} G\mathbf{w} \end{aligned} \qquad . \qquad (4.6)$$

Here, $u' = du$ and $v' = dv$. The last equation shows that $I(\mathbf{w})$ is given by the quadratic form.

The *second fundamental form* $II(\mathbf{w})$ measures the change in the normal vector $\mathbf{n}$ in the direction $\mathbf{w}$ and is expressed as $-d\mathbf{N}(\mathbf{w})\bullet\mathbf{w}$. $d\mathbf{N}(\mathbf{w})$ is a linear map that maps a tangent vector to a vector which expresses the change of the normal in the direction of the tangent. Then, the quadratic form of $II(\mathbf{w})$ is

$$\begin{aligned} II(\mathbf{w}) &= -d\mathbf{n}(\mathbf{w})\bullet\mathbf{w} = -(u'\mathbf{n}_u + v'\mathbf{n}_v)\bullet(u'\mathbf{x}_u + v'\mathbf{x}_v) \\ &= -\mathbf{n}_u\bullet\mathbf{x}_u(u')^2 - \mathbf{n}_u\bullet\mathbf{x}_v u'v' - \mathbf{n}_v\bullet\mathbf{x}_u u'v' - \mathbf{n}_v\bullet\mathbf{x}_v(v')^2 \\ &= (\mathbf{n}\bullet\mathbf{x}_{uu})(u')^2 + 2(\mathbf{n}\bullet\mathbf{x}_{uv})u'v' + (\mathbf{n}\bullet\mathbf{x}_{vv})(v')^2 \\ &= \begin{bmatrix} u' & v' \end{bmatrix} \begin{bmatrix} \mathbf{n}\bullet\mathbf{x}_{uu} & \mathbf{n}\bullet\mathbf{x}_{uv} \\ \mathbf{n}\bullet\mathbf{x}_{uv} & \mathbf{n}\bullet\mathbf{x}_{vv} \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} \\ &= \mathbf{w}^{\mathrm{T}} B\mathbf{w} \end{aligned} \qquad (4.7)$$

4.1.3 Surface Curvature

The normal curvature κ_n in the direction $\mathbf{w}$ is

$$\kappa_n = \frac{II(\mathbf{w})}{I(\mathbf{w})} = \frac{\mathbf{w}^{\mathrm{T}} B\mathbf{w}}{\mathbf{w}^{\mathrm{T}} G\mathbf{w}}. \qquad (4.8)$$

Let **p** be a point of $\boldsymbol{M} \subset \mathbf{E}^3$. The maximum and minimum values of the normal curvature of $\boldsymbol{M}$ at **p** are called the principal curvatures of $\boldsymbol{M}$ at **p**, and are denoted by κ_1 and κ_2. The directions in which these extreme values occur are called principal directions of ***M*** at **p**. The principal directions and curvatures are also the eigenvectors and eigenvalues, respectively, of the Weingarten map $G^{-1}B$ [Carmo 76]. Given these principal curvatures, we can define the Gaussian curvature $K = \kappa_1\kappa_2$ and the mean curvature $H = \frac{1}{2}(\kappa_1 + \kappa_2)$. Note that there is a sign ambiguity present in the curvatures κ_1 and κ_2. If the surface normal is flipped, the signs of both κ_1 and κ_2 will change. However, the Gaussian curvature is independent of the choice of the surface normal, because $K = \kappa_1\kappa_2$ is unaffected.

A given point on the surface can be classified according its principal curvatures. A point **p** of $\boldsymbol{M} \subset \mathbf{E}^3$ is umbilic provided the normal curvature κ_n is constant in all directions. For example, every point on a sphere is umbilic. A special case occurs at a point **p** called a planar point, where both principal curvatures vanish. For non-umbilic points at which $\kappa_1 \neq \kappa_2$, there are exactly two principal directions, and these are orthogonal. Figure 4.3 illustrates the three resulting categories.

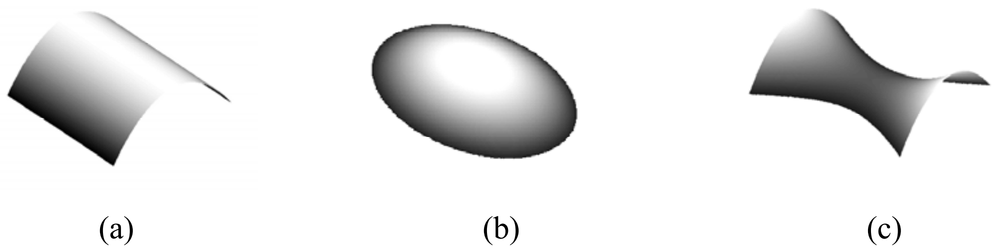


Figure 4.3 Classifications for non-umbilic points. (a) Parabolic ($K = 0$). (b) Elliptic ($K > 0$) (c) Hyperbolic ($K < 0$).

4.2 Outline of Wavelet-Based Iterative Local Refinement

4.2.1 Outline

The proposed method for triangular mesh generation uses a refinement approach, which is a coarse-to-fine mesh construction scheme. A typical procedure for a refinement approach is to build progressively toward a more accurate approximation by increasing the number of triangles. For economy of representation, however, new triangles should be introduced only where warranted by the underlying data. In contrast to most refinement methods, our approach makes use of wavelet detail coefficients directly for finding and splitting triangles to be subdivided. These triangles are subdivided through a local operator that we call *edge splitting* to produce smaller triangles. The resulting mesh is then enhanced through *edge swapping*, *edge collapsing*, and *vertex removal* local operators, which have been introduced in section 2.5.1. The high level outline of the refinement procedure, which is introduced in section 3.2, is replicated here for convenience:

3) Repeat until the finest level is reached:
   (a) Select candidates.
   (b) Refine candidates.
   (c) Perform mesh enhancement (regularization and reduction).

One intuitive idea for reducing the time cost in refinement algorithms is to carefully select candidate regions that need to be refined. Most approaches, however, measure or estimate approximation error to determine those regions, which are computationally intensive. In our triangulation method, wavelet coefficients at several decomposition levels are obtained by the filter bank algorithm and can be used as an indicator to select those candidates for each resolution level. As shown in wavelet-based approximation (section 3.3.1), detail coefficients of large magnitude add significant amount of corresponding wavelets to the coarsest approximation (equation (3.7)). In polygonal approximation of height fields, we consider a planar surface as an approximation at one resolution level, and also consider the refinement as an addition of detail. As a result, triangles that contain wavelet detail coefficients with large magnitude are candidates for our refinement method.

When the candidates are selected, the next step is to subdivide them with our new shape preserving edge split method, which will be described in section 4.4. Consequently, the resulting mesh becomes finer and corresponds to an approximation at a higher resolution level. For our refinement scheme, we employ only one local operator called edge splitting (see section 2.5.1) to refine the current triangular mesh. The edge split operator alone can produce slivers, which is not desirable in many applications. For that reason, we use an edge swap operator to increase the regularity of the resulting mesh. Finally, as a last step, the mesh enhancement procedure removes redundant triangles possibly unnecessarily generated by the split operation. This final step is worthwhile to produce a smaller number of triangles for a given error bound, if it is done without much time cost. The refinement process is repeated until either the finest level or user specified error is reached.

4.2.2 Iterative Approach

The edge swap operator together with the edge split operation are sufficient to construct all subdivision templates commonly used in the triangular mesh subdivision scheme. Figure 4.4 demonstrates the sufficiency. The shaded triangles are the templates used in most subdivision schemes [Scarlatos and Pavlidis 92]. Therefore, successive use of edge splitting and edge swapping is equivalent to the face subdivision scheme. Precisely, two or three splits with appropriate swap operations are enough to produce the subdivision templates.

As a consequence, the proposed algorithm iterates refinement and regularization steps, which uses split and swap operators, respectively, in order to perform mesh approximation. Experimental results discussed in Chapter 5 will show that three iterations are often enough to produce accurate approximation satisfying a user-defined error bound, while twice is still good for computer most vision applications.

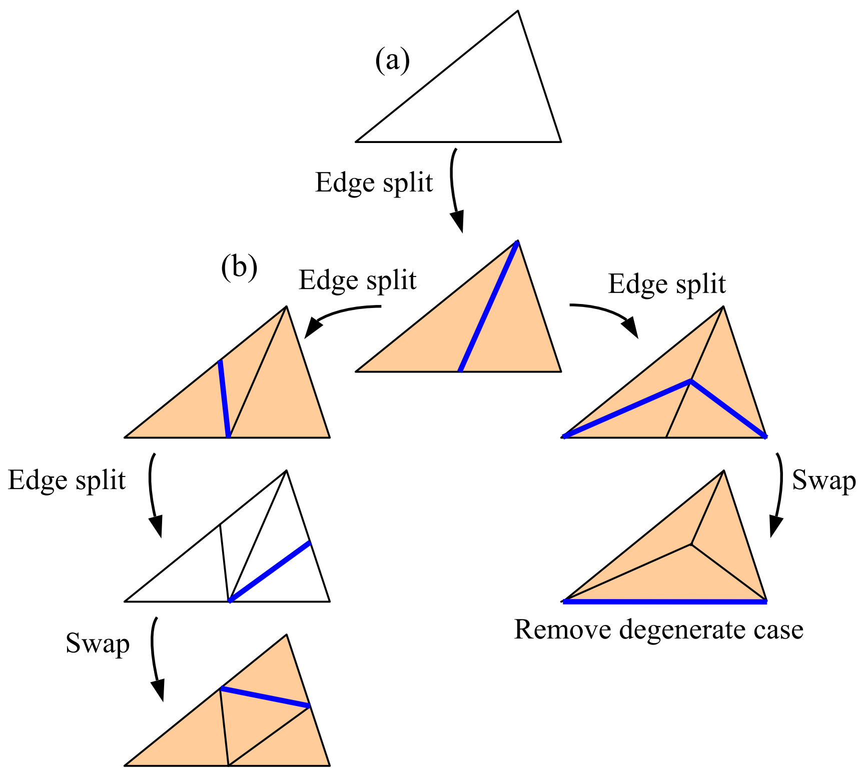


Figure 4.4 Demonstration of sufficiency. (a) A given triangle to be subdivided. (b) Successive application of edge split and edge swap construct all subdivision templates (shaded triangles) commonly used in the mesh refinement literature.

4.3 Region Selection for Refinement

4.3.1 Neighborhood

[Bonneau and Gerussi 98] have defined a small neighborhood of a vertex for the purpose of localization of their mesh decomposition/reconstruction process. We are going to use a similar idea, but in a slightly different way. First, let v denote a vertex of interest. Then, the i-neighborhood of the vertex $\mathcal{N}_i(v)$ is defined by:

$$\begin{aligned}
&\mathcal{N}_0(v) = \{T_i \mid v \in T_i\} \\
&\mathcal{N}_1(v) = \{T_i \mid T_i \cap T_j = E_j, T_j \in \mathcal{N}_0(v) \; and \; T_i \notin \mathcal{N}_0(v)\} \, . \\
&\mathcal{N}_2(v) = \{T_i \mid T_i \cap T_j = v_j, T_j \in \mathcal{N}_0(v)\}
\end{aligned} \tag{4.9}$$

A 0-neighborhood is a set of triangles that share a given vertex v. The corresponding 1-neighborhood consists of triangles whose union with this 0-neighborhood is an edge E_j and that are not members of the 0-neighborhood. Finally, a 2-neighborhood is made of triangles whose union with the 0-neighborhood is only a vertex $v_j \neq v$. We call it *vertex-based neighborhood* (Figure 4.5(a)). By a similar way, we can define a *face-based neighborhood* (Figure 4.5(b)) of a triangular face of interest T such as:

$$\begin{aligned}
&\mathcal{N}_0(T) = \{T\} \\
&\mathcal{N}_1(T) = \{T_i \mid T_i \cap T = E_i\} \\
&\mathcal{N}_2(T) = \{T_i \mid T_i \cap T = v_k\}
\end{aligned} \tag{4.10}$$

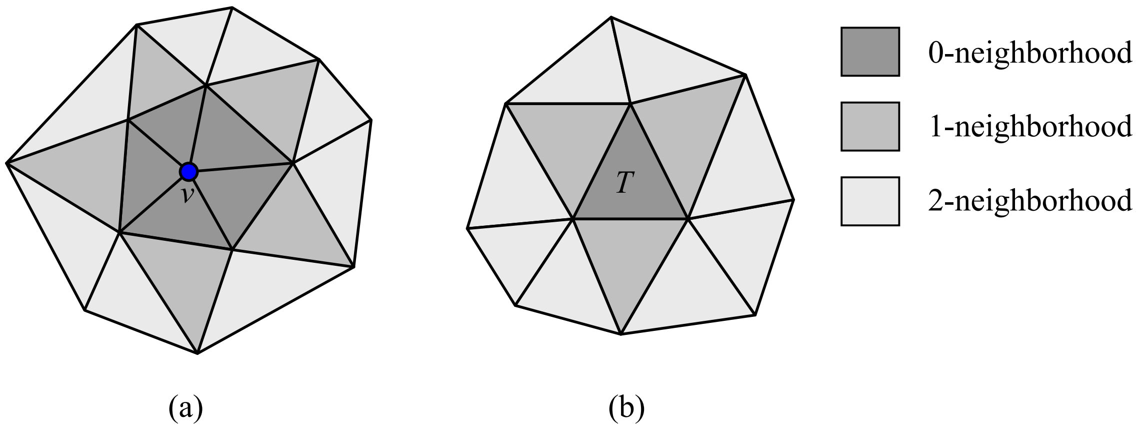


Figure 4.5 Neighborhoods. (a) Vertex-based neighborhood. (b) Face-based neighborhood. $\mathcal{N}_0(v)$ consists of all triangles that directly connected to the vertex v, and $\mathcal{N}_0(T)$ is the triangle T itself.

In addition, we define an *active region* that consists of a union of 0-neighborhoods: $\bigcup_{i=1}^{|\mathbb{V}|} \mathcal{N}_0(v_i)$ for the vertex-based neighborhood or $\bigcup_{i=1}^{|\Gamma|} \mathcal{N}_0(T_i)$ for the face-based neighborhood. A union of 1-neighborhoods $\bigcup_i \mathcal{N}_1(\bullet)$ is defined as a *dependant region*. Triangles in the active and dependant regions take parts in the refinement process, and the triangles in an active region must be refined in the refinement process. Triangles in the 2-neighborhoods are only involved in *post-refinement processing* that regularizes refined mesh using edge swap operator.

Generally, a face-based neighborhood contains fewer triangles than a corresponding vertex-based neighborhood. This implies that the use of vertex-based neighborhood takes more triangles into account in the next refinement step, increasing the possibility to produce more accurate approximation than using face-based neighborhood. However, like the support of wavelets, wider support means more processing time because the refinement process should consider more triangles. That is, the combination of vertex-based neighborhood and wide support wavelets could be a bad choice in terms of efficiency of speed. Likewise, the use of face-based neighborhood and narrow support wavelets together could result in inaccurate approximation especially in the vicinity of discontinuities of input data set.

4.3.2 Criterion and Selection Scheme

As discussed for the signal approximation in section 3.3.1, the new region selection scheme needs appropriate criteria to select candidate triangles to be refined. This can be accomplished using signal energy $\mathrm{E}(f)$ that is defined by the L^2 norm:

$$\mathrm{E}(f) = \|f\|^2 = \iint |f(x,y)|^2 \, dxdy \tag{4.11}$$

Parseval's theorem says that any representation of a signal transformed by orthonormal bases has the same energy. Unfortunately, this statement does not hold for biorthogonal wavelet bases, but squared sum of wavelet coefficients is a measure of the fraction of energy provided by corresponding wavelet functions. Therefore, the squared sum of detail coefficients

$$e_{m,i,j} = \left|d^1_{m,i,j}\right|^2 + \left|d^2_{m,i,j}\right|^2 + \left|d^3_{m,i,j}\right|^2 \tag{4.12}$$

can be used as a decision criterion for the region selection scheme, by which the algorithm tries to determine how much *detail energy* is in the corresponding square $[2^m i, 2^m(i+1)) \times [2^m j, 2^m(j+1))$. That is, if $e_{m,i,j}$ is greater than a threshold δ_d, the corresponding square has non-negligible amount of detail information and must be refined such as in initial triangulation.

Now, it becomes an issue how to choose locations of detail coefficients for region selection. In the previous section, we defined two different types of neighborhood: the vertex-based and face-based neighborhoods. These neighborhoods use different means to determine coefficient locations for the candidate region selection. Let (i, j) be the location

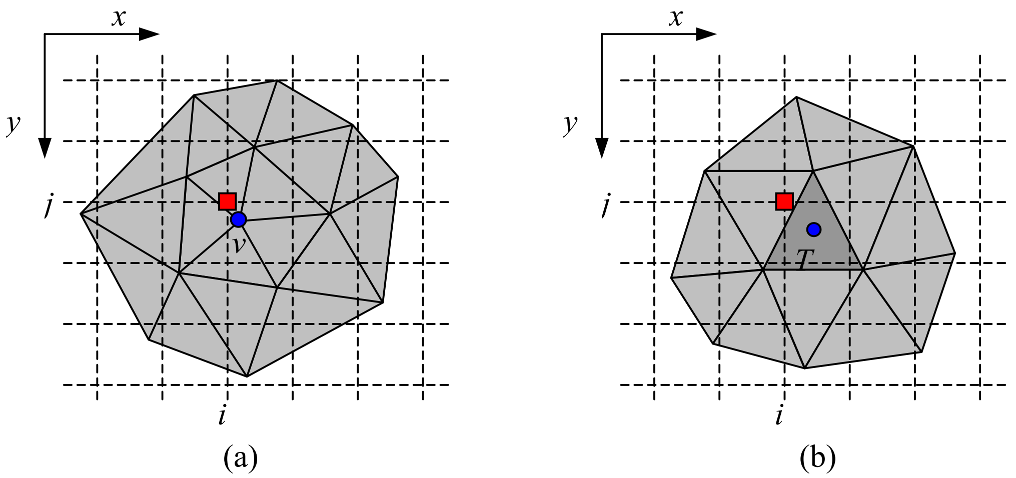


Figure 4.6 Region selection. (a) Vertex-based neighborhood: A vertex of interest v is determined as a part of candidate region if the detail coefficients at nearest neighbor (i, j) of v have enough energy. (b) Face-based neighborhood: Triangle T is a candidate when detail coefficients at $(i, j) = (\lfloor b_x \rfloor, \lfloor b_y \rfloor)$ have enough energy. The small circle shows the barycenter of triangle T. Left top corner is the origin in this coordinate system.

for calculating detail energy. Then, for a vertex-based neighborhood, detail coefficients at nearest neighbor point of interested vertex v are evaluated for the decision criterion. The small square box in Figure 4.6(a) indicates the location (i, j). Likewise, for a face-based neighborhood, the location is determined by

$$(i, j) = (\lfloor b_x \rfloor, \lfloor b_y \rfloor) \tag{4.13}$$

where (b_x, b_y) represents the barycenter (center of gravity) of a triangle T (Figure 4.6(b)), and $\lfloor \bullet \rfloor$ is a floor function.

4.3.3 Examples

This section shows experimental results of region selection scheme. Two previously used gray-scale images (Figure 3.22) are decomposed up to level $M = 4$ with three different wavelets (Haar, spline wavelet, and 2nd order B-spline having 7 vanishing moment), and then triangulated with the proposed initial triangulation method. The first column in Figure 4.7 shows initial triangulation using Haar wavelet. Results of using spline wavelet and 2nd order B-spline wavelet are in the second and the third columns, respectively. The first two rows correspond to the image f^4 in which has smooth change along the diagonal, and next two rows to the image f^5 in which has discontinuity along the boundary of a circle. For the image f^4, changing the wavelets does not significantly vary the size of the initial triangulation, nor of the selected region as shown in Figure 4.7(a-f). Their numerical results are summarized in Table 4.1(a). The sizes of meshes for the spline wavelet and the 2nd order B-spline wavelet are the same, and only slightly larger than the Haar case. Selected regions are also approximately the same for the spline wavelet and the 2nd order B-spline wavelet. Especially, the number of triangles in *i*-neighborhood $\left|\bigcup \mathcal{N}_i(T_k)\right|$ is the same for the spline wavelet and the 2nd order B-spline wavelet. These numbers are also slightly larger than the Haar case. As expected, the vertex-based neighborhood (d-f) is larger than the face-based neighborhood (a-c).

In the case of discontinuities such as in image f^5, the differences due to different wavelets become apparent. Figure 4.7(g-i) show that wider-support wavelets produce more triangles and a wider active region $\bigcup \mathcal{N}_0(T_k)$ for refinement. Generally, a large level difference in discontinuities may affect the wavelet coefficients located at distant place. This effect is more common in wide-support wavelets as shown in Figure 3.6. Difference of selected regions is depicted in Figure 4.7(g-i) for the face-based neighborhood, and in (j-l) for the vertex-based neighborhood. Again, the face-based neighborhood is smaller than the vertex-based neighborhood (Table 4.1(b)).

Region selection also depends on the threshold δ_d for detail energy. It is natural that lower threshold selects larger region and produces accurate approximation. Therefore, the threshold δ_d is an important factor by which determines the quality of generated mesh. Effects on mesh quality due to different δ_d will be discussed in the next chapter.

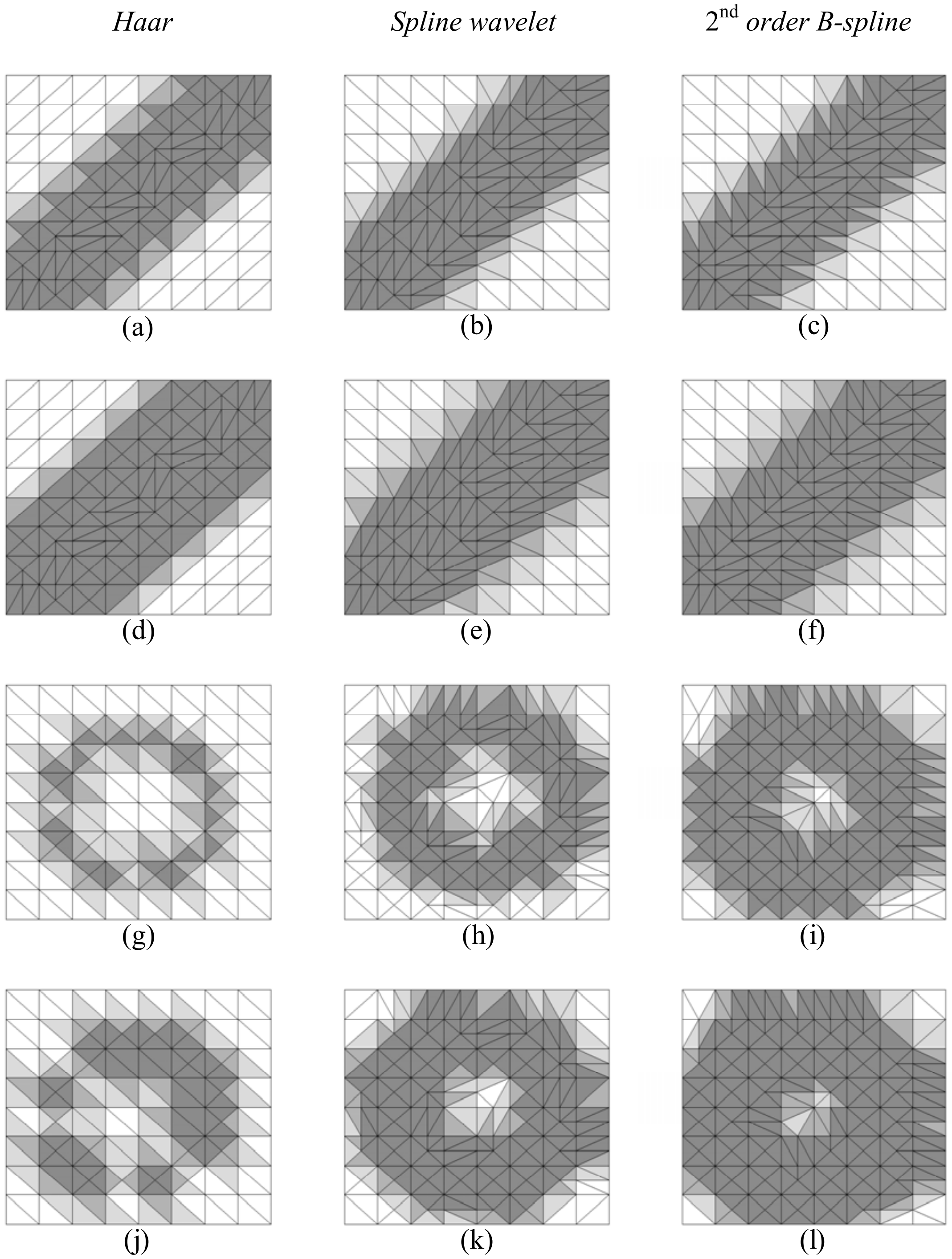


Figure 4.7 Examples for region selection. Haar, spline wavelet and 2nd order B-spline wavelet are used in first, second, and third column, respectively. Generally, a face-based neighborhood (a-c, g-i) has smaller selected region for the next refinement step than a vertex-based neighborhood (d-f, j-l).

Table 4.1 Numerical results of region selection for (a) f^4 and (b) f^5.

(a)

	Wavelets	Haar	Spline wavelet	2nd B-spline
	$\lvert\tilde{\Gamma}_{M-1}\rvert$	210	232	232
Face-Based Neighborhood	$\lvert\bigcup\mathcal{N}_0(T_k)\rvert$	138	150	150
	$\lvert\bigcup\mathcal{N}_1(T_k)\rvert$	22	22	22
	$\lvert\bigcup\mathcal{N}_2(T_k)\rvert$	10	20	20
Vertex-Based Neighbrohood	$\lvert\bigcup\mathcal{N}_0(v_k)\rvert$	160	170	172
	$\lvert\bigcup\mathcal{N}_1(v_k)\rvert$	10	20	20
	$\lvert\bigcup\mathcal{N}_2(v_k)\rvert$	8	18	16

(b)

	Wavelets	Haar	Spline wavelet	2nd B-spline
	$\lvert\tilde{\Gamma}_{M-1}\rvert$	168	258	262
Face-Based Neighborhood	$\lvert\bigcup\mathcal{N}_0(T_k)\rvert$	24	118	169
	$\lvert\bigcup\mathcal{N}_1(T_k)\rvert$	42	47	48
	$\lvert\bigcup\mathcal{N}_2(T_k)\rvert$	39	38	27
Vertex-Based Neighborhood	$\lvert\bigcup\mathcal{N}_0(v_k)\rvert$	55	170	213
	$\lvert\bigcup\mathcal{N}_1(v_k)\rvert$	34	39	28
	$\lvert\bigcup\mathcal{N}_2(v_k)\rvert$	38	28	12

4.4 Shape-Preserving Edge Split

A candidate region for refinement is a set of triangles in an "active region" in which detail energy is above a given threshold. High detail energy implies that there can exist variations in height field around the location at which the detail energy is taken. Accordingly, it is reasonable to subdivide the triangles in an active region so that the refined triangles fit the data more precisely. The subdivision method used in this dissertation is based on edge split local operator described in section 2.5.1. The edge split is simple to implement, and can produce all subdivision templates along with the use of edge swap, as demonstrated in Figure 4.4. This section will describe a strategy to select adjacent triangle pairs to be subdivided, and a method for locating a split position on their common edge.

4.4.1 Gradient Consistent Pairing

The proposed algorithm searches for *valid pairs*, which consist of pairs of adjacent triangles that have consistent discontinuities through their common edge. Wavelet detail coefficients can be used, again, to determine valid pairs. For instance, triangles T_1 and T_2 in Figure 4.8(a) are adjacent with a common edge $E(a,b)$ and they are likely to be associated with discontinuities in horizontal direction, that is, the dominant wavelet coefficients for both triangles are $|d_{m,i,j}^{1}| = \max_{1 \le k \le 3}(|d_{m,i,j}^{k}|)$. Accordingly, the triangles T_1 and T_2 can be a *valid pair*. The wavelet coefficients associated with those triangles are determined by the same way as the face-based neighborhood selection. On the other hand, the triangles T_2 and T_3 contain discontinuities in horizontal and vertical directions, respectively; the discontinuity is not consistent across the common edge $E(b,c)$, and T_2 and T_3 are not considered as a valid pair. Consequently, splitting $E(a,b)$ (a) creates a better approximation than splitting $E(b,c)$ (b).

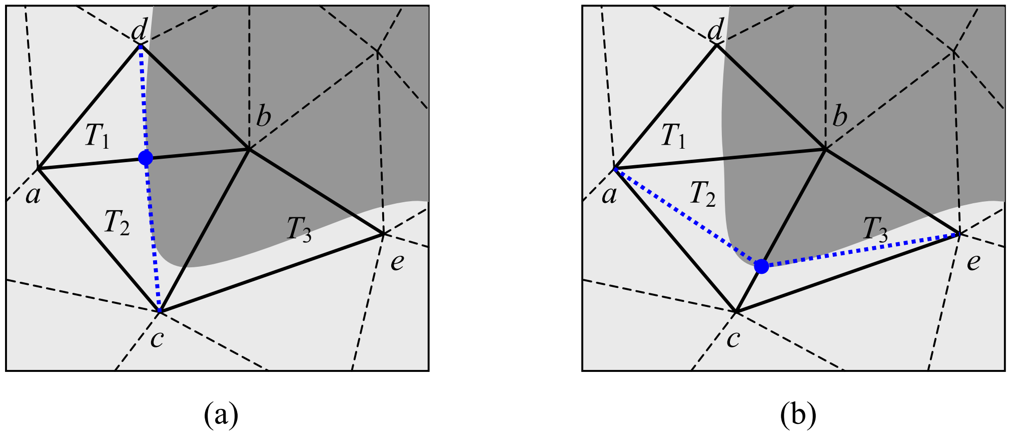


Figure 4.8 Example of a valid pair of triangles. (a) Triangles T_1 and T_2 contain discontinuities in horizontal direction that are consistent across the common edge $E(a,b)$. (b) Triangles T_2 and T_3 contain in consistent discontinuities, and are not considered as a valid pair. Consequently, we expect a better approximation by splitting $E(a,b)$ than $E(b,c)$.

The pairing method proposed in this dissertation can be viewed as a *gradient consistent* pairing. As discussed in section 2.2.3, wavelet coefficients can be used to estimate the gradient vectors of an input image at different scale, if the wavelets are expressed by

$$\psi^1(x,y)=\frac{\partial\theta(x,y)}{\partial x},\ \psi^2(x,y)=\frac{\partial\theta(x,y)}{\partial y} \tag{4.14}$$

where $\theta(x,y)$ is a convolution kernel. Since the gradient vector by definition is normal to the edge (here, the word 'edge' refers to discontinuities in image, not sides of triangles) in the image, the consistency in gradient vector across the two adjacent triangles implies a connected image edge.

The multiscale Canny edge detector uses the Gaussian function as the convolution kernel [Canny 87], and the Gaussian function satisfies the derivative condition (4.14) so that it can be employed in wavelet transform framework for multiscale signal analysis. Another convolution kernel that satisfies the condition (4.14) is the spline wavelet introduced in Figure 3.5(b). The spline wavelet has been widely used in signal processing and volume rendering [Mallat and Zhong 92, Guo 95]. Therefore, if the scale function and the wavelets satisfy the condition in (4.14), we can use the following relation

$$\begin{pmatrix} d_m^1(x,y) \\ d_m^2(x,y) \end{pmatrix} = -2^m \begin{pmatrix} \dfrac{\partial}{\partial u}(f * \tilde{\theta}_m)(x,y) \\ \dfrac{\partial}{\partial v}(f * \tilde{\theta}_m)(x,y) \end{pmatrix} \tag{4.15}$$

The same assumption has been made in the dual template construction for the proposed initial triangulation method (see section 3.5.3). With this assumption, the pairing method described here is equivalent to the gradient consistent pairing as illustrated in Figure 4.9. The figure shows two triangles T_1 and T_2 that contain discontinuities. Gradient vectors $\mathbf{g}_1$ and $\mathbf{g}_2$ on both sides of the common edge $E = (v_a, v_b)$ in Figure 4.9 are both downword because of the image discontinuities in vertical direction. Consequently, the pairing strategy can be fomulated by

$$\frac{1}{\|\mathbf{g}_1\|\|\mathbf{c}\|}|\mathbf{g}_1 \bullet \mathbf{c}| > \delta_p \text{ and } \frac{1}{\|\mathbf{g}_2\|\|\mathbf{c}\|}|\mathbf{g}_2 \bullet \mathbf{c}| > \delta_p, \tag{4.16}$$

$$0.5 < \delta_p < 1$$

where $\mathbf{c} = \overrightarrow{v_a v_b}$ is a vector representation of the common edge, and δ_p is a threshold for pairing criteria. By the above criteria, two gradient vectors that have $2\delta_p$ deviation can be paired with an appropriate common edge. The only restriction for this pairing method is that quadrilaterals made of two contiguous triangles should be either strict convex or non-strict convex (Figure 4.10). This restriction is imposed in order not to generate slivers.

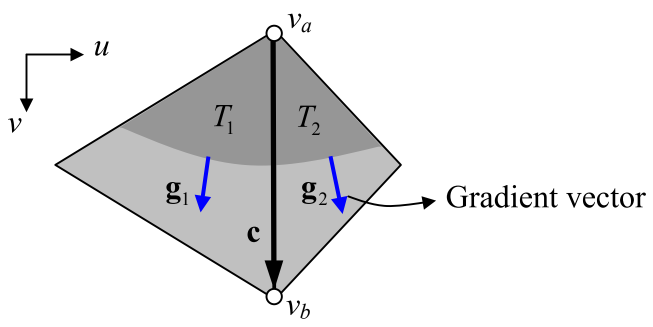


Figure 4.9 Gradient consistencies. Gradient vectors $\mathbf{g}_1$ and $\mathbf{g}_2$ interior of triangles T_1 and T_2 point almost the same direction, since the image discontinuities on both triangles are consistent.

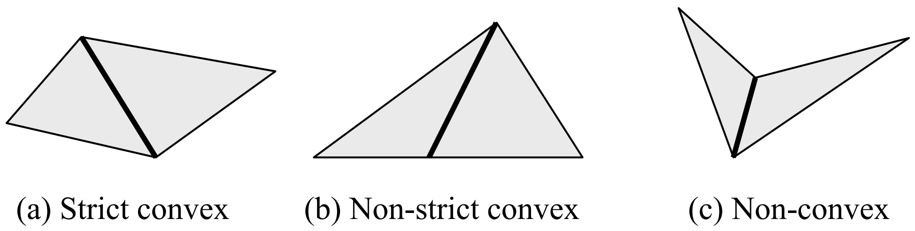


Figure 4.10 Restriction in the pairing procedure. Splitting non-convex quadrilaterals may generate slivers.

Indeed, the proposed pairing method may leave some triangles in $\mathcal{N}_0(\bullet)$ unpaired. Those leftovers are paired with triangles in $\mathcal{N}_1(\bullet)$ and the pairs are still considered as valid pairs. In addition, if an edge of an unpaired triangle T is part of the boundary and it fulfills the pairing criteria (4.16) with an *imaginary triangle* T_{img}, which is artificially formed outside the boundary and assigned with the same gradient vector with the triangle T, then the unpaired triangle T is paired with the imaginary triangle T_{img} (Figure 4.11). These pairs are categorized a *quasi-valid pair*. Despite previous tests for pairing, it is possible that there are still leftovers, which are classified as *unpaired*.

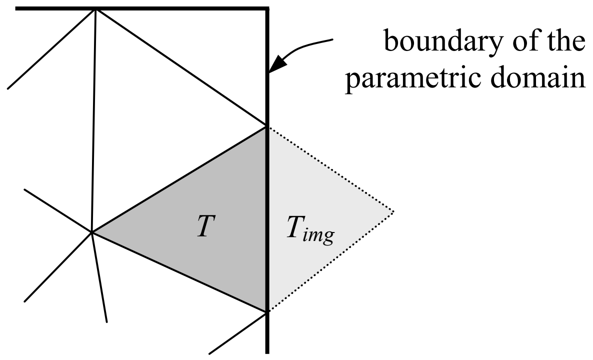


Figure 4.11 A quasi-valid pair. The unpaired triangle T is paired with the imaginary triangle T_{img} when it satisfies the pairing criteria (4.16) with the edge on the boundary of the parametric domain. The imaginary triangle is artificially formed outside the boundary and assigned with the same gradient vector with the triangle T.

4.4.2 Split Point

After the pairs are constructed, the next step is to find a split point on the common edge of each pair. As mentioned earlier, the common edges in *quasi-valid pairs* are bisected at the midpoint. Likewise, it is possible to select a midpoint of the common edge as a split for *valid pairs*, for the sake of efficiency of speed. However, more sophisticated selection could produce a better approximation with smaller number of triangles. Again, the assumption made by the equation (4.15) is used here.

In 2D parametric space, the gradient vector $\mathbf{g} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}^{\mathrm{T}} = \frac{1}{2^{-m}} \begin{bmatrix} -d_m^1 & -d_m^2 \end{bmatrix}^{\mathrm{T}}$ completely determines the directions of image discontinuities, which can be represented by 2D vectors defined by

$$\mathbf{g}^{\perp} = \begin{bmatrix} -\frac{\partial f}{\partial y} & \frac{\partial f}{\partial x} \end{bmatrix}^{\mathrm{T}}. \tag{4.17}$$

With the assumption made in equation (4.15), the directional information at scale m becomes

$$\mathbf{g}^{\perp} = \frac{1}{2^{-m}} \begin{bmatrix} d_m^2 & -d_m^1 \end{bmatrix}^{\mathrm{T}}. \tag{4.18}$$

Then, we can obtain a line equation $l(x, y)$ that goes in the direction of $\mathbf{g}^{\perp}$ from the barycentric point $\mathbf{c}_t = (c_x, c_y)$ of the triangle. The constant factor $\frac{1}{2^{-m}}$ can be omitted without generality. Thus, the line can be expressed by

$$\begin{aligned} &l(x, y) = d_m^1 x + d_m^2 y + r = 0, \\ &r = -(d_m^1 c_x + d_m^2 c_y) \end{aligned} , \tag{4.19}$$

and will pass through somewhere on the common edge. The algorithm finds an intersection point between $l(x, y)$ and the common edge. If we let the implicit line equation of the common edge be $Ax + By + C = 0$, then the intersection point $v_i = (x_i, y_i)$ can be calculated as follow:

$$v_i = (x_i, y_i) = \frac{1}{D}\left(-rB + d_m^2 C, \quad rA - d_m^1 C\right), \tag{4.20}$$

where $D = d_m^1 B - d_m^2 A$, and $D \neq 0$ is guaranteed with the criterion (4.16).

Since each triangle of a pair has an intersection point on the common edge, there are two intersection points. Accordingly, the algorithm sets a midpoint of those two intersection points as the final split point. Figure 4.12 illustrates the procedure. The split point on planar mesh $\mathbf{v}_i = (x_i, y_i, z(x_i, y_i))$ is obtained by lifting the previously calculated 2D split point up to data point such as shown in Figure 4.13.

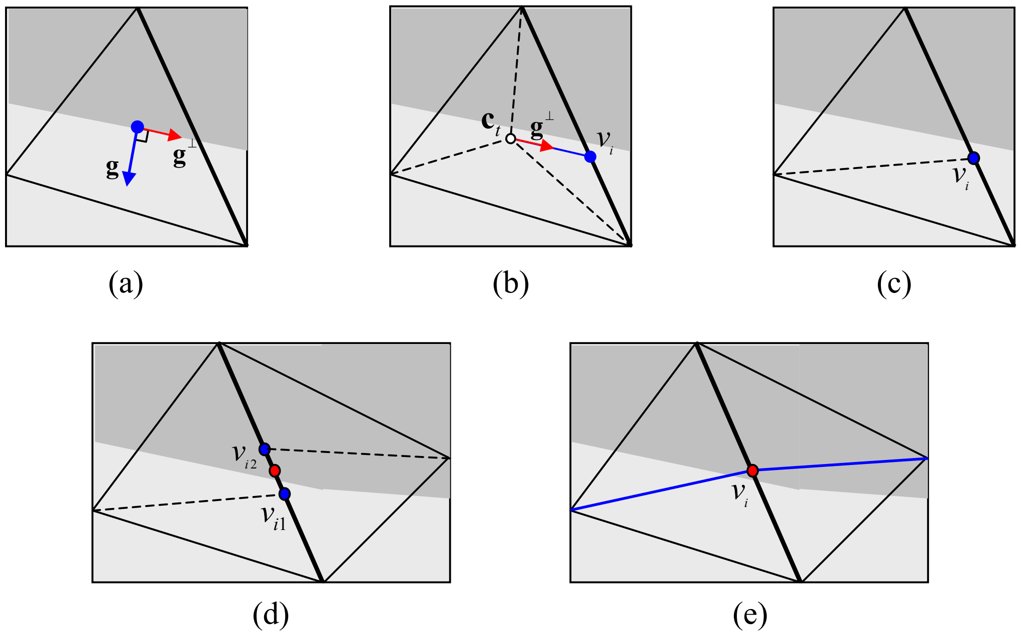


Figure 4.12 Split point in 2D parametric space. (a) Gradient vector $\mathbf{g}$ is estimated with the wavelet coefficient, and a vector $\mathbf{g}^\perp$ represents the direction of the image discontinuities. (b) An intersection point v_i between $l(x,y)$ and the common edge is calculated. $l(x,y)$ is a line that goes in the direction of $\mathbf{g}^\perp$ from the barycentric point $\mathbf{c}_t = (c_x, c_y)$ of the triangle. (c) The point v_i on the common edge can be used as a split point. (d) There exist two intersection points v_{i1} and v_{i2} on the common edge for a pair. (e) The final split point v_i is set to a midpoint of the two intersection points.

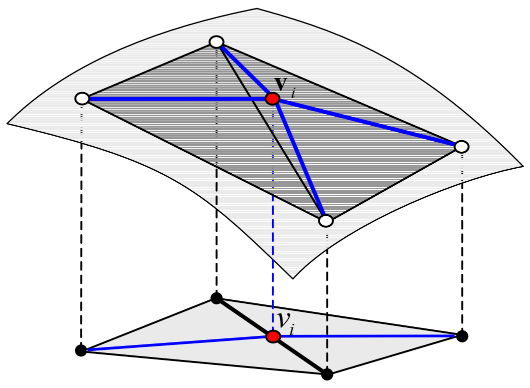


Figure 4.13 Split point on a mesh. The split point on planar mesh $\mathbf{v}_i = (x_i, y_i, z(x_i, y_i))$ is obtained by lifting the 2D split point v_i up to a data point.

4.4.3 Vertex Movement and Unpaired Triangles

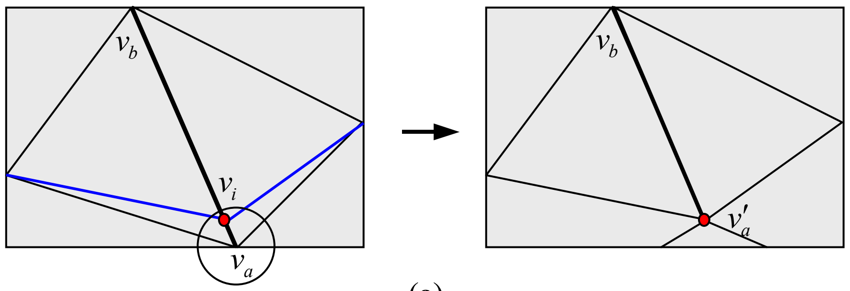
There exists a possibility that a split point is very close to one of the vertices of the common edge, by which the split operation produces slivers. For this case, the algorithm uses a threshold δ_v such that the Euclidean distance is tested between the split point and one of the vertices of common edge. If the distance is less than the threshold, the resulting triangles by the edge split operation could likely be slivers.

$$\left|v_i - v_k\right| < \delta_v, \quad k = a \ or \ b, \tag{4.21}$$

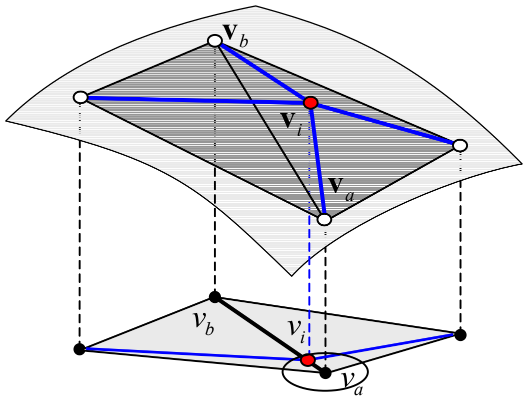
where, v_a and v_b represent two vertices associated with a common edge. In this situation, the algorithm moves one of vertices of the common edge to the split point so that the actual split does not occur but overall fitness can be improved by adding no triangles.

Figure 4.14(a) illustrates the vertex movement in the 2D parametric space. Since $\left|v_a - v_i\right| < \delta_v$ for this example, the vertex v_a moves to the split point v_i and no split occur. However, movement in 2D parametric space does not imply the movement in 3D surface. As shown in Figure 4.14(b), even if the intersection point satisfies $\left|v_a - v_i\right| < \delta_v$ in 2D parametric domain, the lifted points $\mathbf{v}_a$ and $\mathbf{v}_i$ may be far apart: $\left|\mathbf{v}_a - \mathbf{v}_i\right| > \delta_v$. Height field approximation tests the 3D distance for the assessment of the vertex movement, while the distance in 2D parametric space is used for grayscale image approximation.

There is no need to calculate two intersection points for quasi-valid pairs. Since the common edge in a quasi-valid pair is on the boundary of the parametric domain, the split point is determined by an intersection point by the triangle in active region. Finally, the unpaired triangles are subdivided with ternary template (Figure 4.15), and the barycenter of each is used as an interior point for the ternary subdivision.



(a)



(b)

Figure 4.14 Vertex movement. (a) For grayscale images, the distances are measured in 2D parametric domain. (b) In case of height field approximation, the 3D distances are measured to determine the vertex movement.

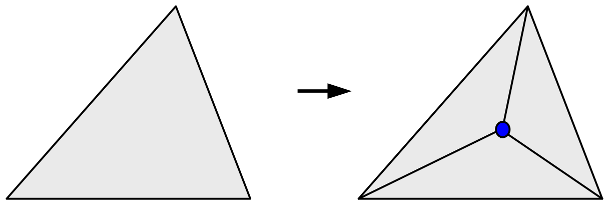


Figure 4.15 Ternary subdivision.

4.5 Mesh Enhancement

The edge split operation alone often produces very narrow needle shaped triangles. Sometimes, these *slivers* are inevitable and represent good fits to the input data. However, in most cases, eliminating slivers does not reduce the quality of the approximation appreciably, but does reduce the size of the mesh representation. A well-known approach to eliminating slivers without reducing the size of the mesh is called *edge swapping*. An edge swap operator is implemented by altering diagonals of quadrilateral formed with two adjacent triangular faces, as illustrated in Figure 2.6(d). While the edge swap operator regularizes the mesh, the proposed algorithm employs *vertex removal* and *edge collapse* operators to eliminate redundant triangles that could occur as a result of the refinement process. These two operators reduce the mesh size as described in Section 2.5.1.

4.5.1 Regularization

A well-known triangulation method called Delaunay triangulation generates a triangular mesh that maximizes the minimum angle of all triangles, among all triangulations of a given point set. But the Delaunay method could produce poor triangulation for height fields and even for grayscale images as argued in section 2.1.4. In contrast, data-dependent triangulation results in slivers. Even if the slivers produce a good fit for input data, it is desirable for some applications to reduce the number of slivers.

Therefore, this dissertation has suggested the use of an energy function for mesh quality control described in section 2.6.4. Specifically, the energy function are defined by

$$\xi(T_i,T_j) = \alpha\Re_A + (1-\alpha)\Re_E$$
$$\Re_A = \frac{\min\measuredangle(t_i,t_j)}{\min\measuredangle(T_i,T_j)}, \quad \Re_E = \frac{E(T_i)+E(T_j)+1}{E(t_i)+E(t_j)+1} \tag{4.22}$$

where $\Re_A$ and $\Re_E$ respectively represent alteration ratios of minimum angles and approximation errors of two different triangulations. Given two adjacent triangles T_i and T_j, the new triangulation, obtained by a swap operation, has a new pair of triangles t_i and t_j. Hence, $\Re_A$ is an increasing ratio of the minimum angles by changing the triangulation (T_i, T_j) to the new triangulation (t_i, t_j), and $\Re_E$ has a value greater than 1 when the swap operation reduces the approximation error.

The parameter α, named as *mesh regularity factor*, controls the regularity of triangular meshes. When $\alpha = 1$, the resulting triangulation is Delaunay, while $\alpha = 0$ produces pure data-dependent triangulation. Thus, for $0 < \alpha < 1$, the function allows slivers in some degree if the error reduction is prominent and the triangles are not excessively thin. The results are compared in sections 2.1.4 and 2.6.4. Edge swapping is applied to the quadrilaterals that are *strict convex*, which means that no three vertices are collinear. Of course, the candidate region for edge swap operation is a set of triangles in the union of *i*-neighborhood $\bigcup_{i=0}^{2} \mathcal{N}_i(\bullet)$. The newly generated triangles by the refinement process are automatically assigned to 0-neighborhood.

4.5.2 Vertex Removal

The vertex removal operator, employed in order to reduce the mesh size, removes a vertex v if all triangles in $\mathcal{N}_0(v)$ face in almost same direction and satisfy an error criterion. The reduction of mesh size may not be necessary at early stages of the refinement process and can be postponed until the last refinement is completed at the finest resolution level. However, since the refinement process may generate redundant triangles even at early stages of the algorithm, it can be computationally efficient to eliminate those redundant triangles as early as possible. Accordingly, the proposed algorithm employs vertex removal operation as well as edge collapse operation discussed in the next section.

As the first step of the vertex removal, all the surface normals of the faces around a vertex v, that is, triangles in $\mathcal{N}_0(v)$ are mapped onto a unit sphere. This mapping is called Gauss mapping and defined by $G : n\left(\mathcal{N}_0(v)\right) \to \Sigma$, where $n\left(\mathcal{N}_0(v)\right)$ is a unit normal vector field related to $\mathcal{N}_0(v)$ and Σ is a unit sphere. That is, the Gauss mapping of unit normal vector $\mathbf{n} = n_1U_1 + n_2U_2 + n_3U_3$ is a point $G(\mathbf{n}) = (n_1, n_2, n_3)$ on the unit sphere Σ. Accordingly, if the points on the unit sphere are not spread away from a center point, the vertex v can be removed. Therefore, the criterion for selecting candidate vertex for removal is formulated by

$$\mathbf{n}_0 = \frac{1}{\left|\mathcal{N}_0(v)\right|}\sum_{i=1}^{k}\mathbf{n}_i \tag{4.23}$$

$$\theta = \max_{1\le i\le k}\left(\cos^{-1}\left(\mathbf{n}_i \bullet \mathbf{n}_0\right)\right) < \theta_{th} \tag{4.24}$$

where $\mathbf{n}_i$ represents unit normal vector of i^{th} triangle of $\mathcal{N}_0(v)$, $\mathbf{n}_0$ is an average unit normal of the $\mathcal{N}_0(v)$, and $\left|\mathcal{N}_0(v)\right| = k$ means the size of 0-neighborhood of vertex v. That is, for the vertex to be removed, the maximum angle deviation between $\mathbf{n}_0$ and $\mathbf{n}_i$ should be less than a threshold θ_{th}. Instead of using (4.23), a faster way to evaluate the criterion is to use

$$\min_{1\le i\le k}\left(\mathbf{n}_i \bullet \mathbf{n}_0\right) < \cos(\theta_{th})\,. \tag{4.25}$$

Figure 4.16 shows the Gauss mapping and the criterion. When the neighborhood of a vertex v $\mathcal{N}_0(v)$ forms a relatively level surface (a), the unit normal vectors are mapped within a small neighborhood on the unit sphere (b). Then, the decision criterion measures the deviation of the unit normal vectors (c).

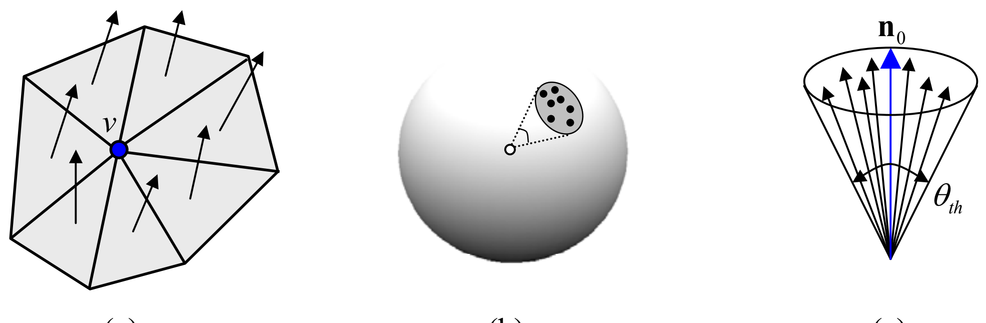


Figure 4.16 Gauss mapping and vertex selection criterion for removal. (a) Unit normal vectors for $\mathcal{N}_0(v)$. (b) Gauss map. (c) Selection criterion.

Another way to select a candidate vertex is to compute eigenvalues of the covariance matrix of the unit normal vectors, which defined by:

$$C_1 = \frac{1}{|\mathcal{N}_0(v)|} \sum_{i=1}^{k} (\mathbf{n}_i - \mathbf{n}_0)(\mathbf{n}_i - \mathbf{n}_0)^{\mathrm{T}} \tag{4.26}$$

This covariance matrix has been used to estimate surface geometry from range images [Liang and Todhunter 90, Berkmann and Caelli 94]. The eigenvalues and eigenvectors of C_1 determine three orthogonal vectors, two of which define a tangent plane of the unit sphere at $G(\mathbf{n}_0)$. For example, if the tangent plane is determined by two eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$, the corresponding eigenvalues λ_1 and λ_2 are utilized as a criterion for vertex selection.

A computational way to calculate the tangent plane at $G(\mathbf{n}_0)$ is to use a parametric form of the sphere, in which Euclidean coordinates on a sphere are defined by:

$$\mathbf{x}(u,v) = (r\cos v \cos u,\ r\cos v \sin u,\ r\sin v)\,. \tag{4.27}$$

That is, a point $\mathbf{x}(u,v)$ is determined with longitude $u\ (-\pi < u < \pi)$ and latitude $v\ (-\pi/2 < v < \pi/2)$. Then, the tangent vectors at (u,v) are

$$\begin{aligned} \mathbf{x}_u(u,v) &= r(-\cos v \sin u,\ \cos v \cos u,\ 0) \\ \mathbf{x}_v(u,v) &= r(-\sin v \cos u,\ -\sin v \sin u,\ \cos v) \end{aligned} \tag{4.28}$$

Given a Euclidean coordinate $G(\mathbf{n}_0) = (g_x, g_y, g_z)$ on a sphere, longitude and latitude are calculated with

$$v_0 = \cos^{-1}\left(\frac{g_z}{r}\right), \qquad u_0 = \cos^{-1}\left(\frac{g_x}{r\sin v}\right). \tag{4.29}$$

Given both tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$, they are obtained by the estimation (4.25) or by the computational method (4.27), orthonormal projections of $\mathbf{n}_i - \mathbf{n}_0$ onto this tangent plane produce scattered points on 2D plane (Figure 4.17).

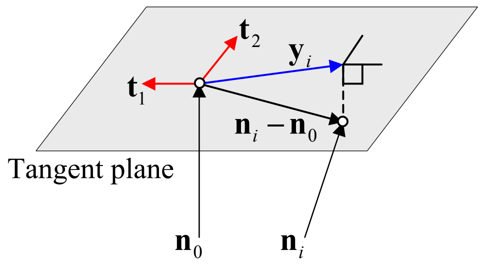


Figure 4. 17 Tangent plane and orthonormal projections of $\mathbf{n}_i - \mathbf{n}_0$.

Then, eigenvalues and eigenvectors of a new two-dimensional covariance matrix [Berkmann and Caelli 94]

$$C_2 = \frac{1}{|\mathcal{N}_0(v)|}\sum_{i=1}^{k}(\mathbf{y}_i - \mathbf{y}_0)(\mathbf{y}_i - \mathbf{y}_0)^{\mathrm{T}}$$
$$\mathbf{y}_i = \begin{bmatrix} (\mathbf{n}_i - \mathbf{n}_0)\bullet\mathbf{t}_1 \\ (\mathbf{n}_i - \mathbf{n}_0)\bullet\mathbf{t}_2 \end{bmatrix} \tag{4.30}$$
$$\mathbf{y}_0 = \frac{1}{|\mathcal{N}_0(v)|}\sum_{i=1}^{k}\mathbf{y}_i$$

provide a shape information of an ellipse on the tangent plane. The eigenvectors are the principal axes of the resulting ellipse and the eigenvalues represent amount of extent of the principal axes. Accordingly, the vertex selection criterion is equivalent to the thresholding the eigenvalues.

Although the use of covariance matrixes tends to provide an accurate estimation of eigenvalues, there are some drawbacks in using those matrixes for the vertex selection of the proposed algorithm that aims to a fast generation of polygonal surface representation. In the case of using C_1, it is obviously difficult to identify the tangent vectors by looking at the eigenvalues, because they may be similar in magnitude. Meanwhile, the use of C_2 requires more computing time, because we need to calculate tangent vectors and project the surface normals to the tangent plane. For the moment, the required information for vertex selection is a measure of variation of normal vectors in $\mathcal{N}_0(v)$ along the surface of unit sphere not in the direction of average normal vector $\mathbf{n}_0$. Therefore, the use of an autocorrelation matrix $\mathbf{R}(v)$ defined as

$$\mathbf{R}(v) = \frac{1}{|\mathcal{N}_0(v)|} \sum_{i=1}^{k} \mathbf{n}_i \mathbf{n}_i^{\mathrm{T}} \tag{4.31}$$

satisfies our requirement. The reason this definition is useful is that it considers $\mathbf{n}_0 = 0$ so that the deviation of the normal vectors from the origin is relatively large compared to the variation along the surface of the unit sphere. Thus, it is reasonable to associate the largest eigenvalue with this deviation, by means of which the selection criterion is to evaluate the two smallest eigenvalues to see if they are less than a threshold. In order words, if the two smallest eigenvalues are λ_1 and λ_2 and the criteria

$$\lambda_i < \delta_\lambda \;\; for\, i = 1, 2 \tag{4.32}$$

is satisfied, then the vertex v can be removed. The threshold value δ_λ is another important factor to determine the mesh quality. If it is high, the approximation results in a sparse triangulation that is small in size. In contrast, a dense triangular mesh is generated with low δ_λ.

The underlying assumption leading to this conclusion is that the approximated surface by the triangles in $\mathcal{N}_0(v)$ is locally smooth around the vertex v. Nevertheless, the criterion works for the case where a discontinuity is present, as the first two largest eigenvalues are excessively large.

4.5.3 Edge Collapse

Another mesh reduction operator, edge collapsing has been used for mesh simplification preserving the topology of the mesh [Ronfard and Rossignac 96, Algorri and Schmitt 96, Hoppe 96, Hoppe 97]. While most edge-collapse operations rely on error estimation to select candidate edges to be collapsed, this dissertation utilizes the autocorrelation matrix defined in equation (4.31). Since, by its nature, the vertex removal operator tends to remove vertices on relatively flat or smooth area on data, it cannot remove vertices on discontinuities where the criteria (4.32) fail. Even if the criteria are not satisfied, however, there is a possibility to reduce the mesh size.

Figure 4.18 illustrates the possible edge collapse. The vertices v_1 and v_2 in Figure 4.18(a) cannot be removed since the criteria (4.32) fails at these vertices, but if one of the smallest eigenvalues is less than the threshold δ_λ for each vertex v_1 and v_2, that is,

$$\lambda_1 < \delta_\lambda < \lambda_2 \qquad \textit{for } v_1 \text{ and } v_2, \tag{4.33}$$

then the two vertices may be collapsed to a vertex. A criterion for the edge collapsing needs to compare eigenvectors corresponding to the eigenvalues λ_2 of each autocorrelation matrix at vertices v_1 and v_2. Let $\mathbf{e}_{21}$ and $\mathbf{e}_{22}$ be eigenvectors corresponding to the eigenvalues λ_2 of autocorrelation matrices $\mathbf{R}(v_1)$ and $\mathbf{R}(v_2)$, respectively. Then, a criterion

$$\left|\mathbf{e}_{21} \bullet \mathbf{e}_{22}\right| > \delta_c \tag{4.34}$$

identifies two 0-neighborhoods $\mathcal{N}_0(v_1)$ and $\mathcal{N}_0(v_2)$ that have almost the same geometric orientation (Figure 4.18(b)), and the edge $\overline{v_1 v_2}$ can be collapsed into a vertex v without a significant quality loss. The same geometric orientation means that the Gauss maps of the surface normal vectors have the same distribution as illustrated in Figure 4.19. For the sake of the efficiency of speed, we select a midpoint of the edge $\overline{v_1 v_2}$ as a collapsing point. Of course, the mesh validity in section 2.6.4 should be tested before the collapse. An edge-collapse operation removes two faces, an edge, and a vertex (Figure 4.18(c)).

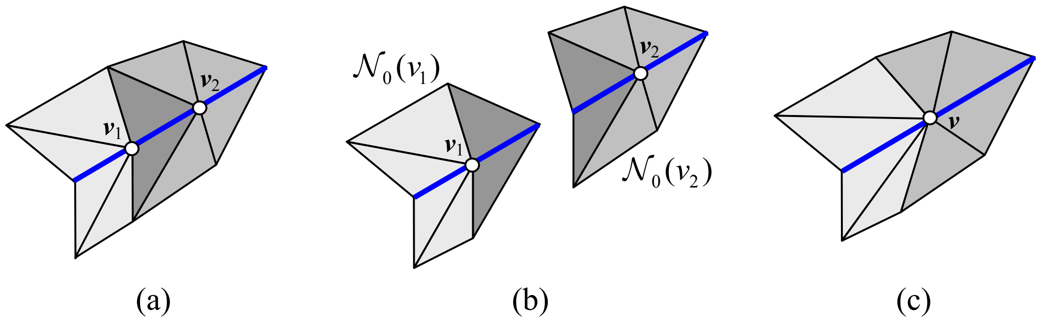


Figure 4.18 Edge collapse. (a) Candidate edge. Although the criteria (4.32) fails at vertices v_1 and v_2, the vertices may be collapsed into a vertex when the criteria (4.33) and (4.34) are satisfied. (b) 0-neighborhoods $\mathcal{N}_0(v_1)$ and $\mathcal{N}_0(v_2)$ have almost the same geometric orientation. (c) Edge collapse operation selects a midpoint of the edge $\overline{v_1v_2}$ as a collapsing point.

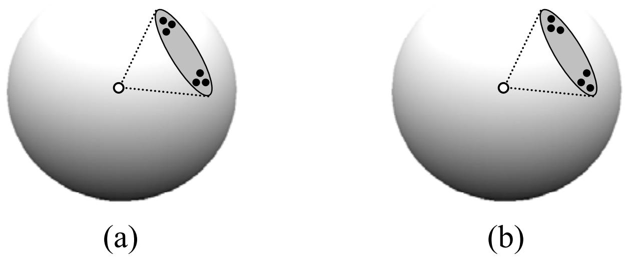


Figure 4.19 The same geometric orientation. (a) Gauss map of surface normal vectors in $\mathcal{N}_0(v_1)$. (b) Gauss map of surface normal vectors in $\mathcal{N}_0(v_2)$. Gauss maps of the surface normal vectors have the same distribution on the unit sphere.

4.6 Summary of Algorithm

In section 3.2, the high-level outline of the surface approximation algorithm was described. Details of each step have been described throughout Chapter 3 and Chapter 4. Now, a more detailed description is given:

1. Decompose a given height field with a wavelet transform (sec. 3.3).
2. Perform initial triangulation with templates (see section 3.5 for template construction).
   (a) Evaluate wavelet detail coefficients at the coarsest resolution level M (sec. 3.6.2).
   (b) Assign labels for each square region (sec. 3.6.2).
   (c) Tessellate square regions with templates to construct $\tilde{\Gamma}_{M-1}$ (sec. 3.6.2).
   (d) Enhance $\tilde{\Gamma}_{M-1}$ to form $\hat{\Gamma}_{M-1}$ using local operators (sec 4.5).
3. Set $i = M-2$ and set the number of iterations $1 \le k \le 3$. Repeat until the finest level ($i = 0$) is reached or the user specified error criterion is satisfied:
   (a) Iterate the following *k* times.
      i. Calculate detail energy to select candidates (sec. 4.3.2)
      ii. Construct neighborhood (sec. 4.3.1).
      iii. Perform gradient consistent pairing (sec. 4.4.1).
      iv. Find split point for shape-preserving edge split producing a finer mesh $\tilde{\Gamma}_i$ (sec. 4.4.2 and sec. 4.4.3).
      v. Regularize $\tilde{\Gamma}_i$ with edge swap operator (sec. 4.5.1).
   (b) Remove redundant vertices with vertex removal (sec. 4.5.2) and edge collapse (sec. 4.5.3) operators yielding an enhancement $\hat{\Gamma}_i$.
   (c) If *i* = 0, then stop. Else decrease *i* by one.

Table 4.2 shows the relation between the above procedure and detail coefficients and resolution levels. The procedure in second column uses wavelet detail coefficients at the decomposition level at its left-hand side to produce triangulations at various resolutions. Figure 4.20 shows the flow diagram of the proposed algorithm, and the parameters used in the proposed algorithm are summarized in Table 4.3.

Table 4.2 Relation between detail coefficients and resolution levels

Decomposition Level	Procedure	Resolution Level
M	Step 1 and 2	*M*-1
M-1	Step 3(a)	(*M*-1)-1/2
M-1	Step 3(a)	(*M*-2)
	Step 3(b)	*M*-2
. . .	. . .	. . .
1	Step 3(a)	0.5
1	Step 3(a)	0
	Step 3(b)	0

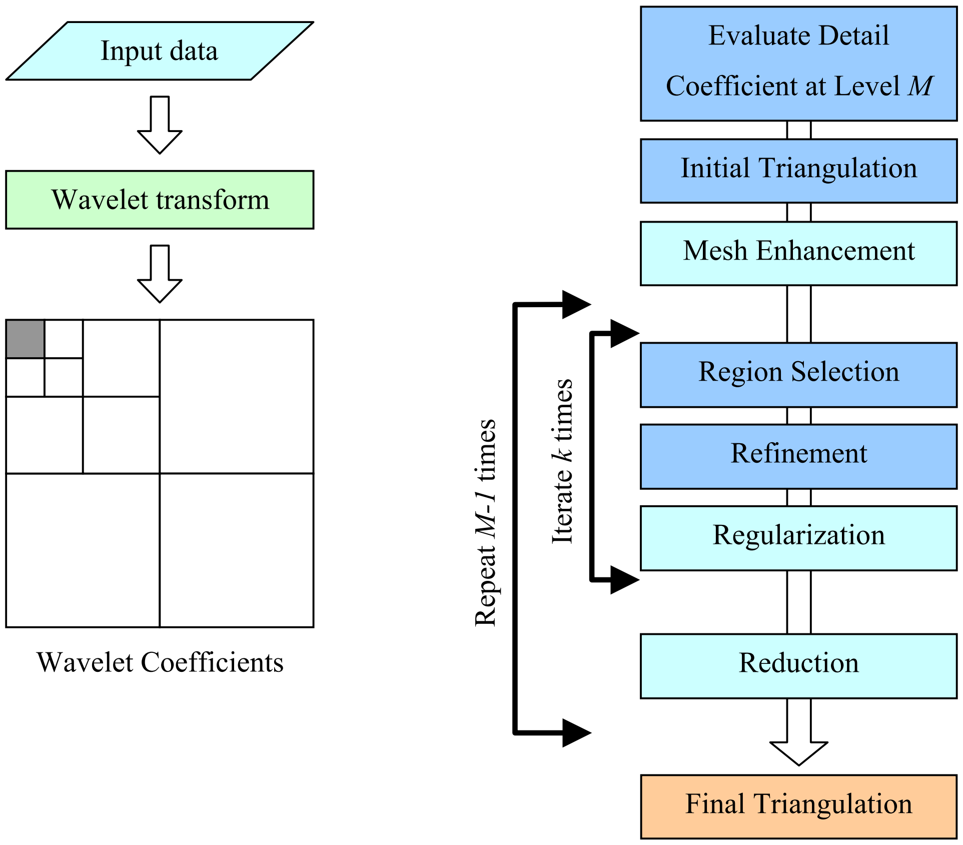


Figure 4.20 Flow diagram of the algorithm.

Table 4.3 Summary of the parameters used in the approximation algorithm.

Parameter		
ς	Approximation error for a triangle	$E(T_i) \le \varsigma$
δ_d (4.12)	Wavelet detail energy	$e_{m,i,j} = \left\|d^1_{m,i,j}\right\|^2 + \left\|d^2_{m,i,j}\right\|^2 + \left\|d^3_{m,i,j}\right\|^2 > \delta_d$
δ_p (4.16)	Pairing criterion	$\frac{1}{\\|\mathbf{g}_1\\|\\|\mathbf{c}\\|}\left\|\mathbf{g}_1 \bullet \mathbf{c}\right\| > \delta_p$ and $\frac{1}{\\|\mathbf{g}_2\\|\\|\mathbf{c}\\|}\left\|\mathbf{g}_2 \bullet \mathbf{c}\right\| > \delta_p$, $0.5 < \delta_p < 1$
δ_v (4.21)	Vertex movement threshold	$\left\|v_i - v_k\right\| < \delta_v, \quad k = a \ or \ b$
α (4.22)	Mesh regularity factor	$\xi(T_i, T_j) = \alpha\Re_A + (1-\alpha)\Re_E$ $\Re_A = \frac{\min \measuredangle(t_i, t_j)}{\min \measuredangle(T_i, T_j)}, \quad \Re_E = \frac{E(T_i) + E(T_j) + 1}{E(t_i) + E(t_j) + 1}$
δ_λ (4.32)	Vertex removal criterion	$\lambda_i < \delta_\lambda \ for \, i = 1, 2$
δ_c (4.34)	Edge collapsing criterion	$\left\|\mathbf{e}_{21} \bullet \mathbf{e}_{22}\right\| > \delta_c$

Chapter 5

Results and Performance Analysis

In the preceding chapters, I have presented methods for an initial triangulation and for iterative mesh refinement. The new algorithm utilizes wavelet detail coefficients in determining proper templates, selecting candidate regions for refinement, and refining the candidate regions. This chapter is devoted to analyzing both the efficiency of the new algorithm and the quality of its results. The chapter will start with theoretical analysis of the time complexity and memory usage. The analysis shows that the time complexity is linear with respect to the number of faces. Then, some experiments are performed to demonstrate the algorithm's capabilities of multiresolution mesh generation, level-of-detail (LOD) control, and selective triangulation. The actual running time is measured in terms of the number of faces at the final approximations.

5.1 Time Complexity and Memory Usage

One of our primary design goals is to produce an efficient surface approximation algorithm using triangular meshes. The algorithm that has been discussed in previous chapters is quite fast, but it requires a substantial amount of memory because the algorithm uses a linked list data structure for fast reference to neighborhood triangles. In this section, the time complexity and memory usage will be examined. The proposed algorithm consists of two phases: initial triangulation and iterative refinement. In section 3.5.1, the cost for the wavelet transform was examined to find that it could be achieved in less than 0.5 second with common grid sizes with the latest DSP processors. Furthermore, the initial triangular mesh is constructed using simple search for appropriate templates. Therefore, the execution time for the initial triangulation will be ignored in following analysis.

Since our time complexity analysis will be represented in terms of mesh size that varies as the resolution level decreases, the analysis focuses on a particular resolution level. For simplicity of the representation, let $n = |\Gamma_m|$ be the number of triangles in the triangulation Γ_m, and $r = |\mathbb{V}_m|$ the number of vertices in Γ_m. In addition, define a *vertex degree h* as a number of edges that connected to a particular vertex. In this analysis, a constant h is used and $h \ll n$ is assumed.

5.1.1 Time Complexity

Asymptotic time complexity is very important to evaluate an algorithm [Cormen et al. 90]. In this section, the worst case of the time complexity is examined to provide an upper bound of the time cost. The worst case at a given resolution corresponds to the situation that all detail energy values are above a threshold such that the algorithm needs to subdivide all triangles at this resolution level.

The algorithm begins with region selection, which is performed in $O(n)$ or $O(r)$ time. It depends on which neighborhood the algorithm is using. A face-based neighborhood requires $O(n)$ time. Then, the shape-preserving split operation requires $3n$ comparisons to determine valid pairs, that is, $O(3n)$ complexity. Assuming that all triangles are in valid pairs, the split operation determines split positions on common edges in $O(n)$. The split operation yields n new triangles and $\frac{n}{2}$ new vertices. As a result, the regularization stage involves $2n$ triangles and evaluates the energy function defined in section 4.5.1 in $O(6n)$ time complexity. Finally, mesh reduction requires calculation of an autocorrelation matrix $\mathbf{R}(v)$ with complexity $O(r+\frac{n}{2})$, and re-triangulation of the remaining holes takes $O([r+\frac{n}{2}]h\log h)$ time. When a vertex of degree h is removed, the remaining hole is a polygon that has h

sides and can be triangulated in time $O(h\log h)$ [Preparata and Shamos 85]. Furthermore, using Euler's formula,

$$r - e + n = 2\,, \tag{5.1}$$

where e represents the number of edges, and using the property that each vertex has degree ≥ 3, we can conclude that the number of vertices $r + \frac{n}{2}$ satisfies

$$r + \tfrac{n}{2} \leq 2n - 4 + \tfrac{n}{2} < 3n\,, \tag{5.2}$$

when $n \gg 8$ [Preparata and Shamos 85].

Overall, the time complexity for the mesh reduction can be bounded by $O(3n \cdot h\log h)$. Table 5.1 summarizes the costs for these individual operations. The total time complexity is

$$O(n) + O(3n) + O(n) + O(6n) + O(3n) + O(3n \cdot h\log h) \;=\; O(14n) + O(3n \cdot h\log h)\,. \tag{5.3}$$

The use of vertex-based neighborhood for region selection is also bounded by $O(14n)$ + $O(3n \cdot h\log h)$ since it is assumed that $r < n$ for most practical cases. At the beginning of this section, we assumed the use of constant vertex degree $h \ll n$, and our experiments show that it is rare for h to exceed 20.

As a consequence, the approximation algorithm has time complexity $O(14n)$ + $O(3n \cdot h\log h)$ for each resolution level. In order to calculate overall time complexity, we assume that a dataset is decomposed up to level *M*, and that a single iterations is used for refinement steps. Since an initial triangulation represents approximation at level $M-1$, there will be $(M-1)$ resolution levels. Consequently, the overall time complexity for region selection operation is expressed by a summation of time cost at each resolution level;

$$O(n) + O(2n) + \cdots O(2^{M-m-1}n) + \cdots + O(2^{M-2}n) = \sum_{m=1}^{4} O(2^{M-m-1}n)\,, \tag{5.4}$$

where *m* represents the resolution level. Similar analysis is applied for the rest of operations in Table 5.1 and yields $\sum\limits_{m=1}^{M-1} O(2^{M-m-1}14n)$ + $\sum\limits_{m=1}^{M-1} O(2^{M-m-1}3n \cdot h\log h)$ for the overall time complexity of the proposed algorithm (Table 5.2). Accordingly, for a given value of *M*, the time complexity of the algorithm is $O(n)$, and a typical value of *M* is 5 or 6 in practical situation.

Table 5.1 Time complexity of the surface approximation at each resolution level.

Operations	Time Complexity
Region selection	$O(n)$ or $O(r)$
Pairing	$O(3n)$
Split	$O(n)$
Regularization	$O(6n)$
Reduction	$O(3n)+O(3n\cdot h\log h)$
Total	$O(14n)+O(3n\cdot h\log h)$

Table 5.2 Overall time complexity of the surface approximation.

Operations	Time Complexity
Region selection	$\sum_{m=1}^{M-1} O(2^{M-m-1}n)$
Pairing	$\sum_{m=1}^{M-1} O(2^{M-m-1}3n)$
Split	$\sum_{m=1}^{M-1} O(2^{M-m-1}n)$
Regularization	$\sum_{m=1}^{M-1} O(2^{M-m-1}6n)$
Reduction	$\sum_{m=1}^{M-1} O(2^{M-m-1}3n)+\sum_{m=1}^{M-1} O(2^{M-m-1}3n\cdot h\log h)$
Total	$\sum_{m=1}^{M-1} O(2^{M-m-1}14n)+\sum_{m=1}^{M-1} O(2^{M-m-1}3n\cdot h\log h)$

This analysis is based on the worst case that assumes that all triangles are involved in refinement process. There are three possibilities in which the algorithm runs at the worst case of time complexity. First of all, large variations in height values may result in this time complexity for all resolution levels, but it is not common in practical situations. Second is for which most detail coefficients have significant energy, so that the entire parametric domain is selected for the refinement process. However, at lower resolution levels where this is most likely to occur, the number of triangles is relatively small compared to the higher resolution levels. Finally, when the threshold δ_d for detail energy is extremely low, it may cause the worst case by selecting all triangles as a candidate region for all resolution levels. Accordingly, except for those extreme cases, it is expected that the actual running time is far less than this upper bound.

5.1.2 Memory Usage

This section considers memory usage of the proposed algorithm as another measure of efficiency. Due to the evolutionary development of VLSI technology and falling prices of computer memory, memory usage becomes less of a concern nowadays. However, in some situations such as embedded system design, memory consumption may still be a major concern.

The basic data structure, which the algorithm has utilized, is the "half edge" data structure that is described in Appendix I. In the half edge data structure, each face is described by a *doubly connected cyclic linked list* of its half edges [Berg et al. 97]. For each half edge, pointers to its incident *face* and *edge* are maintained in order to follow the mesh efficiently along adjacent faces, which is useful when the algorithm accesses adjacent faces. In addition, each mesh element (i.e., vertices, edge, and faces) is stored in a doubly linked list structure. Table 5.3 summarizes the memory breakdown of the half edge data structure. The table includes memory requirement for the use of input data with size *N*×*N*. Recall that $n = |\Gamma_m|$ and $r = |\mathbb{V}_m|$, and let the number of edges in Γ_m be e. The total memory usage is approximated by the use of $e \approx n$ and $r \approx n$ since both e and r are less than n in practical situations.

Table 5.3 Memory breakdown of data structure for surface approximation

	Variables	Size	Data Type	Memory Use (bytes)
Vertices	Position	$3r$	FLOAT	$12r$
	Selection	r	BYTE	r
	Link to HalfEdge	r	WORDS	$4r$
	Linked List	$2r$	WORDS	$8r$
HalfEdge	Link to Vertex	$3n$	WORDS	$12n$
	Link to Edge	$3n$	WORDS	$12n$
	Link to Loop	$3n$	WORDS	$12n$
	Linked List	$6n$	WORDS	$24n$
Edge	Link to HalfEdges	$2e$	WORDS	$8e$
	Length	e	FLOAT	$4e$
	Linked List	$2e$	WORDS	$8e$
Loop	Link to HalfEdge	n	WORDS	$4n$
	Link to Face	n	WORDS	$4n$
Face	Normal	$3n$	FLOAT	$12n$
	Selection	n	BYTE	n
	Link to Loop	n	WORDS	$4n$
	Linked List	$2n$	WORDS	$8n$
Wavelet coefficient		$N \times N$	FLOAT	$4N^2$
Input data		$N \times N$	FLOAT	$4N^2$
Total				$25r+93n+20e+8N^2$ $\approx 183n+8N^2$

Each vertex stores its position, a flag for region selection, and link to a half edge. Each half edge contains links to a vertex, an edge, and a loop. A loop just acts as a bridge between a face and a cyclic linked list of half edges so that each half edge is connected with a face through the link to a loop (see Appendix I). Each edge stores its length as well as links to two half edges by which connect the adjacent faces. Each face has its surface normal, a flag for region selection, and a link to a loop. As mentioned earlier, each element is stored in doubly linked list data structures, and consumes 2 WORDS of memory for next and previous links for an individual element.

5.2 Mesh Size and Regularity

5.2.1 Sample Datasets and Error Metrics

To demonstrate the performance of the proposed surface approximation algorithm, we consider the six different height fields shown in Figure 5.1. The datasets `Ball` and `Box` are synthetic, with sizes 256 × 256. These datasets represent smooth surfaces with sharp discontinuities. The range of values for these synthetic datasets is from 0 to 96 units for `Ball`, 0 to 250 for `Box`. The next two datasets are terrain data of the big island of Hawaii and of Crater Lake, Oregon, downloaded from the USGS FTP site [USGS DEM]. The elevation information of the terrains is stored in digital elevation model (DEM) format. The elevation of the dataset `Hawaii` ranges from 0 (sea level) to 4,245 meters (13,796 feet). The terrain contains smooth variations of elevation along the ridges and has sharp variations along the coastline. This terrain dataset was clipped and sub-sampled to have the size of 256 × 256. The dataset `CraterLake` also has a size of 256 × 256 and ranges from 1,730 meters (5,675 feet) to 2,477 meters (8,126 feet). The `CraterLake` set has many shape variations in a mountainous region and in the flat area of the lake.

We have also used range images acquired from laser range finders. The `Board` dataset is obtained using a sheet-of-light method, and is used to detect wane and holes on rough lumber [Lee et al. 00a, Lee et al. 00b]. Since the rough boards are unplaned, the surface of the boards is relatively rough and bark debris still remain in the wane area. Height values for this dataset range from 0 to 45 in steps of 1/16 inch. Another range image `Perc15` was downloaded from the University of South Florida [USF DB], and was converted to have 39 cm for farthest point (the background wall) from the viewer and 276.8 cm for nearest point (the foremost floor point). The dataset `Perc15` is of size 512 × 512, and was obtained by a Perceptron sensor that computes the phase shift between an outgoing laser beam and its returned signal [Perceptron 93, Dorum et al. 94]. This image was taken in the CESAR lab at Oak Ridge National Laboratory, Tennessee. The dataset contains a considerable amount of noise and also has peak noise around surface discontinuities. We have reduced the noise by applying a 5 × 5 median filter.

For all approximated surfaces displayed in this chapter, we used a Lambertian reflection model with a cyan and magenta color map, in which each face is flat-shaded. Lower values are rendered in cyan, and higher values are colored in magenta. The interior of each face is filled with colors obtained by bilinear interpolation of the vertex colors. While smooth shading such as Gouraud and Phong is more common in practice, the use of flat shading makes the structure of the approximated surfaces more apparent.

The quality of approximation is measured with the L_∞ norm that is defined by

$$\left\| f - f_{\hat{\Gamma}_m} \right\|_\infty = \max_{(x,y)\in\Omega} \left| d(f,\, f_{\hat{\Gamma}_m}) \right|, \tag{5.5}$$

where $d(f,\, f_{\hat{\Gamma}_m})$ represents the normal distance measures between the input dataset and the approximation at resolution level *m* at a point (x, y) in the parametric domain $\Omega \subset \mathbb{R}^2$. The use of L_∞ norm does not allow even one point to deviate from an error bound so that it is overly sensitive to any noise that might be present in acquiring height fields. Throughout the experiments in this chapter, we use the L_∞ norm.

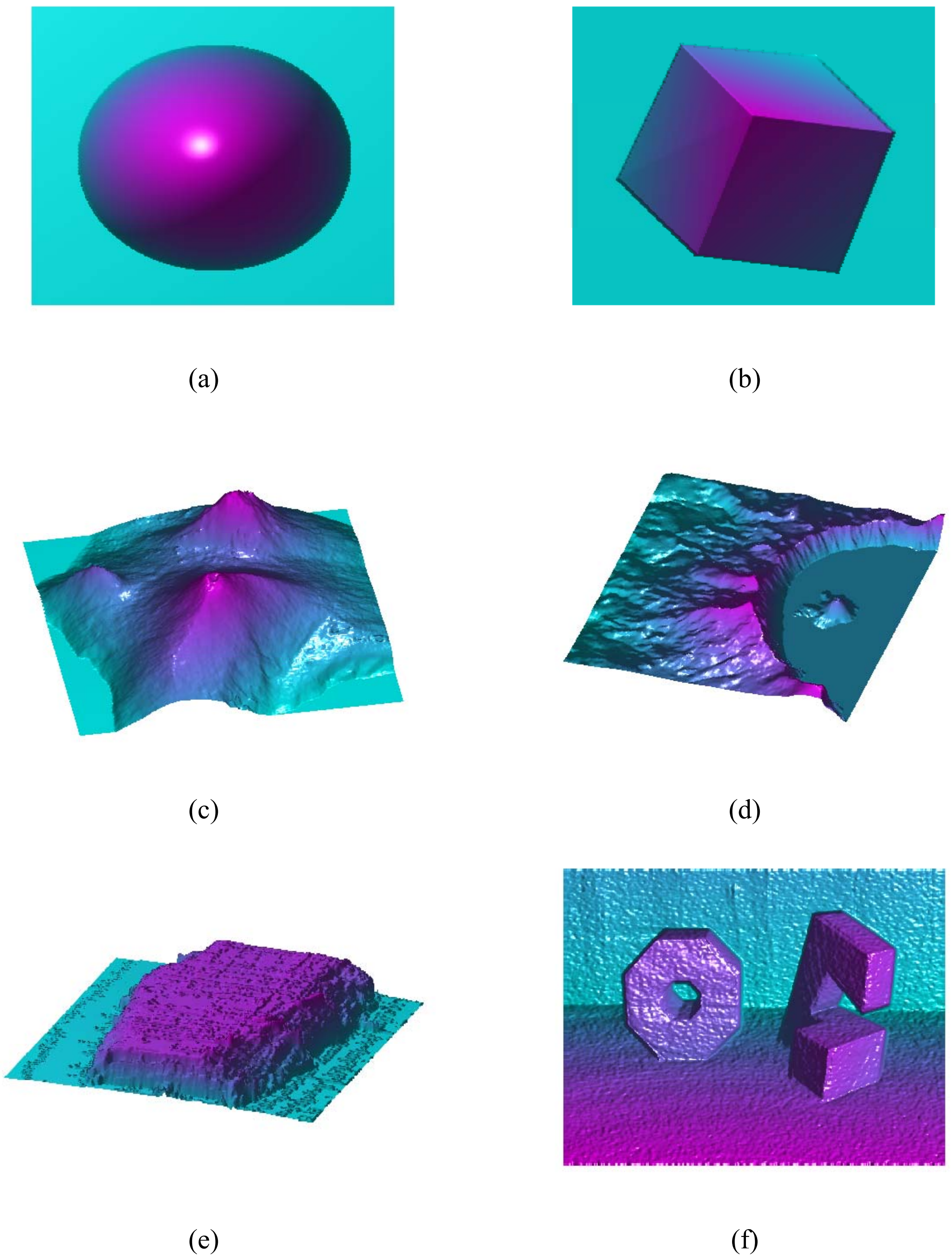


Figure 5.1 Sample datasets. (a) `Ball`. (b) `Box`. (c) `Hawaii`. (d) `CraterLake`. (e) `Board`. (f) `Perc15`.

5.2.2 Split and Removals

The mesh size that the proposed algorithm generates is largely dependent on three criteria: detail energy e_m (section 4.3.2), mesh regularity ξ (4.5.1), and, eigenvalues of the autocorrelation matrix $\mathbf{R}(v)$ (section 4.5.2), each of which is related to region selection, edge swap operations, and vertex removal, respectively. This section will explore the effects on the generated mesh size according to those criteria. Although the number of iterations for the refinement step (section 4.2.2) may vary the mesh size, the explicit experiments on the effect will not be included here. Throughout the experiments in this chapter, the sample datasets are decomposed up to level $M = 5$ for the grid sizes 256×256, and up to level $M = 6$ for grid sizes 512×512. The datasets are decomposed with the spline wavelet for the most experiments. In addition, face-based neighborhoods are used for region selection.

The number of splits and removals are compared according to the various settings of the thresholds δ_d for wavelet detail energy and vertex removal criterion δ_λ, as described in sections 4.3.2 and 4.5.2, respectively. Tables 5.4 and 5.5 summarize the results for the dataset `Hawaii`. Since the dataset is decomposed up to level $M = 5$ and 2 iterations are used for the refinement step, there are four intermediate resolution levels and the corresponding triangulations are $\tilde{\Gamma}_{3.5}$, $\tilde{\Gamma}_{2.5}$, $\tilde{\Gamma}_{1.5}$, and $\tilde{\Gamma}_{0.5}$ as shown in section 4.6. The proposed initial triangulation method produces a triangulation $\tilde{\Gamma}_4$ and the mesh enhancement procedure removes 10 vertices along with edge swap operations resulting an enhanced triangulation $\hat{\Gamma}_4$. Then, two successive executions of the refinement step produce triangulations $\tilde{\Gamma}_{3.5}$ and $\tilde{\Gamma}_3$. The later is enhanced to $\hat{\Gamma}_3$.

Table 5.4 shows the effect on the split and vertex removal operations according to the various thresholds δ_d of wavelet detail energy. The threshold δ_d varies from 15 to 1 for the dataset `Hawaii`, and other parameters are fixed. As expected, the number of splits increases for each resolution level as the wavelet detail energy δ_d decreases, and so does the number of removals. According to the region selection scheme of the proposed algorithm, the lower threshold δ_d selects the more triangles as active triangles, which causes an increase in the number of splits. The split operation may produce redundant triangles that can be removed by the vertex removal and edge collapse operations (sections 4.5.2 and 4.5.3). The more split the algorithm conducts, the more redundant it can generate. Accordingly, the number of removals increases as δ_d decreases. Also, at a fixed δ_d, the number of splits increases first and then decreases later as the resolution moves from coarse to fine (that is, from 4 to 0). The decrease in the number of splits at finer resolution levels is because the wavelet detail energy at finer levels typically low unless there are sudden changes of values in datasets.

Since the split operation introduces a vertex to a triangulation, the total number of splits is equivalent to the number of vertex insertions to a given triangulation. Therefore, for $\delta_d = 1$, the split operations insert 1,176 vertices in total to the initial triangulation $\tilde{\Gamma}_4$, and the removal operations remove 149 vertices in total. Consequently, the net vertex insertion becomes 1,027 (= 1,176 - 149). The net vertex insertion also increases as δ_d decreases, by which the complexity of the mesh increases and it is expected to obtain more accurate approximations.

For the next experiment that is summarized in Table 5.5, we have applied several different thresholds for the vertex removal criterion δ_λ to see its effect on the number of splits and removals. Other parameters are fixed, and in particular the threshold for the wavelet detail energy is set to $\delta_d = 1$. As expected, the number of removals decreases for each resolution level when the criterion δ_λ decreases. This experiment shows that the number of splits also depends on the criterion δ_λ. The reason for this is that the vertex removals may create large triangles that do not satisfy the error criterion, and the next refinement step splits those triangles again. As a result, it is not a good strategy to use large δ_λ, since a large δ_λ may cause more splits in the next refinement process, resulting in more processing time.

Table 5.4 Number of edge splits and vertex removals for dataset `Hawaii` as a function of δ_d. The designation N/A indicates that no operations are performed in this step.

($\varsigma = 2$, $\delta_p = 0.9$, $\delta_v = 2$, $\delta_\lambda = 0.01$, $\alpha = 0.6$, and $\delta_c = 0.98$)

(a) Number of edge splits

Resolution Levels	δ_d				Γ
	15	10	5	1	
4	N/A	N/A	N/A	N/A	$\tilde{\Gamma}_4$
3.5	168	168	168	168	$\tilde{\Gamma}_{3.5}$
3	198	222	222	222	$\tilde{\Gamma}_3$
2.5	180	226	229	229	$\tilde{\Gamma}_{2.5}$
2	92	146	196	196	$\tilde{\Gamma}_2$
1.5	59	90	129	129	$\tilde{\Gamma}_{1.5}$
1	26	37	80	82	$\tilde{\Gamma}_1$
0.5	23	40	83	85	$\tilde{\Gamma}_{0.5}$
0	20	27	61	65	$\tilde{\Gamma}_0$
Total	766	956	1168	1176	

(b) Number of vertex removals

Resolution Levels	δ_d				Γ
	15	10	5	1	
4	10	10	10	10	$\hat{\Gamma}_4$
3.5	N/A	N/A	N/A	N/A	
3	31	31	31	31	$\hat{\Gamma}_3$
2.5	N/A	N/A	N/A	N/A	
2	28	42	44	44	$\hat{\Gamma}_2$
1.5	N/A	N/A	N/A	N/A	
1	23	31	48	48	$\hat{\Gamma}_1$
0.5	N/A	N/A	N/A	N/A	
0	8	10	17	16	$\hat{\Gamma}_0$
Total	100	124	150	149	

Table 5.5 Number of edge splits and vertex removals for dataset `Hawaii` as a function of δ_λ.

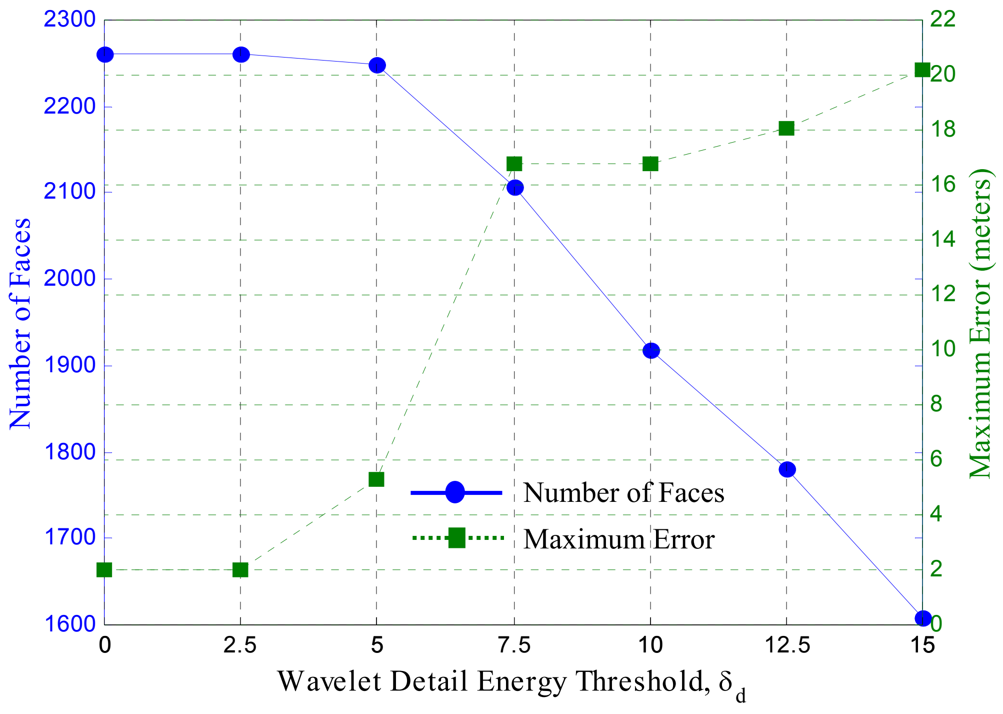
($\varsigma = 2$, $\delta_d = 1$, $\delta_p = 0.9$, $\delta_v = 2$, $\alpha = 0.6$, and $\delta_c = 0.98$)

(a) Number of edge splits

Resolution Levels	δ_λ				Γ
	.05	.01	.005	.001	
4	N/A	N/A	N/A	N/A	$\tilde{\Gamma}_4$
3.5	168	168	166	166	$\tilde{\Gamma}_{3.5}$
3	222	222	222	222	$\tilde{\Gamma}_3$
2.5	273	229	227	226	$\tilde{\Gamma}_{2.5}$
2	211	196	206	202	$\tilde{\Gamma}_2$
1.5	184	129	125	124	$\tilde{\Gamma}_{1.5}$
1	101	82	83	85	$\tilde{\Gamma}_1$
0.5	148	85	75	75	$\tilde{\Gamma}_{0.5}$
0	145	65	56	56	$\tilde{\Gamma}_0$
Total	1452	1176	1160	1156	

(b) Number of vertex removals

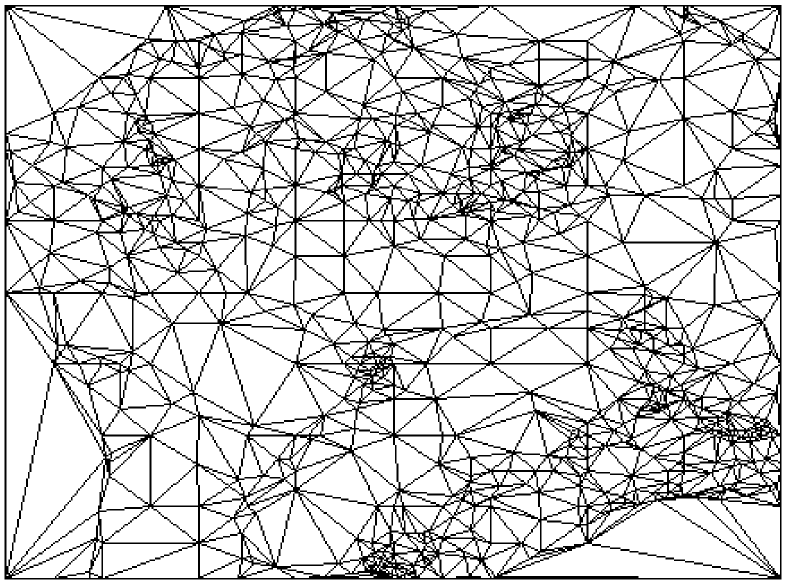
Resolution Levels	δ_λ				Γ
	.05	.01	.005	.001	
4	10	10	9	9	$\hat{\Gamma}_4$
3.5	N/A	N/A	N/A	N/A	
3	79	31	16	5	$\hat{\Gamma}_3$
2.5	N/A	N/A	N/A	N/A	
2	156	44	32	18	$\hat{\Gamma}_2$
1.5	N/A	N/A	N/A	N/A	
1	145	48	44	32	$\hat{\Gamma}_1$
0.5	N/A	N/A	N/A	N/A	
0	23	16	12	11	$\hat{\Gamma}_0$
Total	413	149	113	75	

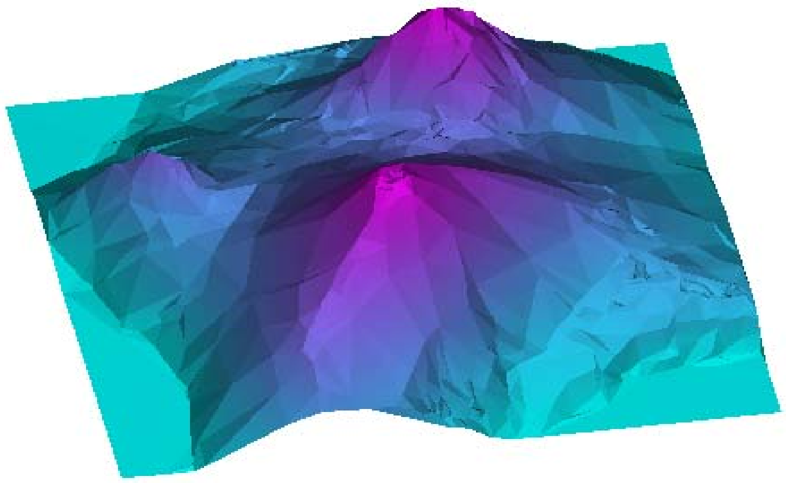


$(\varsigma = 2,\ \delta_p = 0.9,\ \delta_v = 2,\ \delta_\lambda = 0.01,\ \alpha = 0.6,\ \text{and}\ \delta_c = 0.98)$

Figure 5.2 Number of faces and maximum error for the dataset `Hawaii` according to the threshold for wavelet detail energy, δ_d.

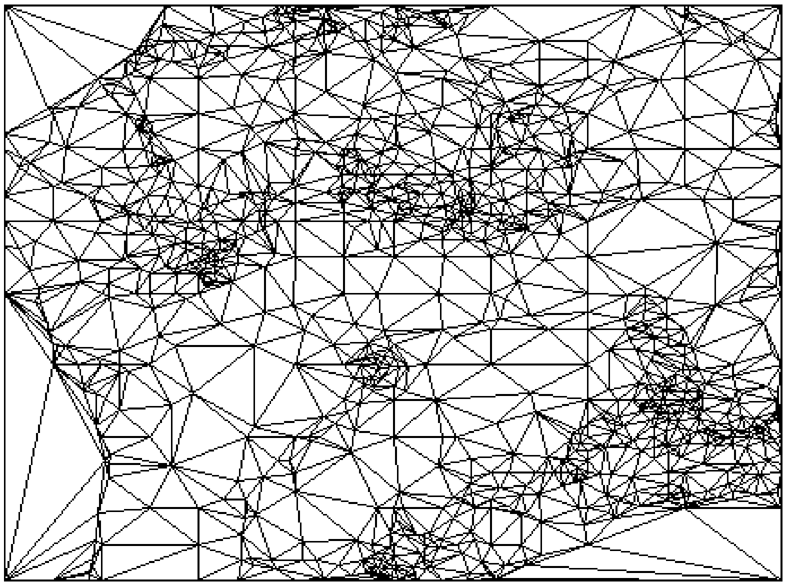
Figure 5.2 shows the resulting number of faces and maximum error including the results in Table 5.4. The numbers of faces are 2,261, 2,249, 1,918, and 1,609 for the threshold δ_d = 0, 5, 10, and 15, respectively. The corresponding maximum errors are 1.999, 5.281, 16.757, and 20.188. The maximum error increases and the number of faces decreases as the threshold δ_d increases. Consequently, to obtain an approximation that satisfies the error criterion $\varsigma = 2$ meters, we have to use at most $\delta_d = 2.5$. Figure 5.3 shows the resulting triangulations and their rendered outputs for the thresholds δ_d = 2.5, and δ_d = 15. The triangulation in Figure 5.3 (b) is denser and the approximation shows many more details compared to the results in (a). The effects on δ_d will be discussed again in later section.

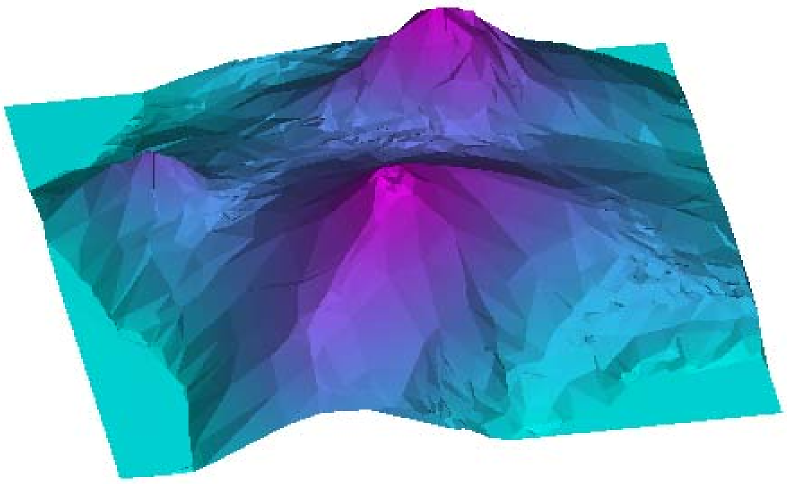




(a)

Figure 5.3 Triangulations and rendered outputs for the dataset `Hawaii` according to different choices of δ_d. (a) Results for $\delta_d = 15$. $\left|\hat{\Gamma}_0\right| = 1609$ and $E_{max} = 20.188$.





(b)

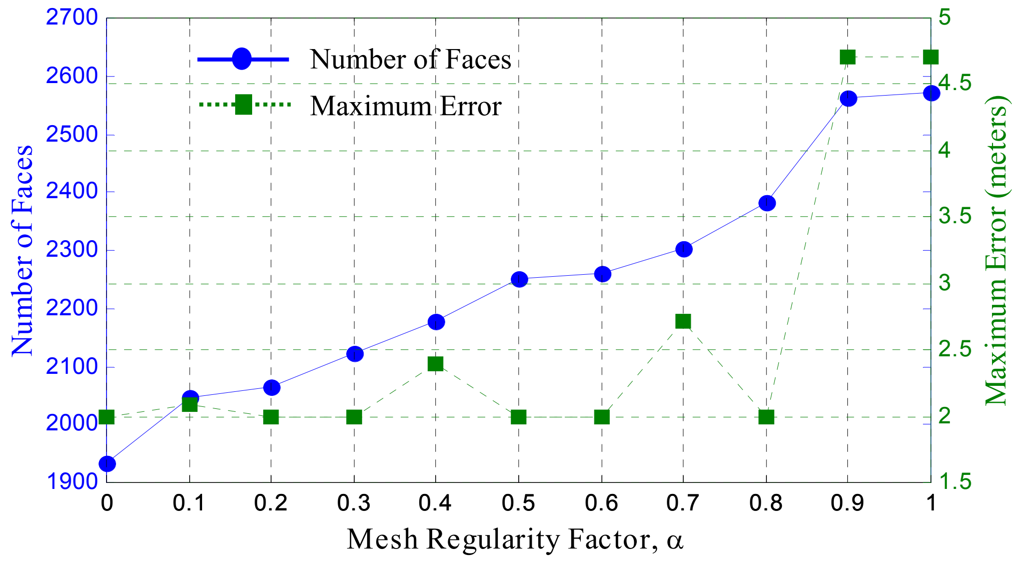
Figure 5.3 (continued) (b) Result for $\delta_d = 2.5$. $\left|\hat{\Gamma}_0\right| = 2261$ and $E_{max} = 1.999$.

5.2.3 Mesh Regularity Factor

In the proposed algorithm, the mesh regularity factor α mentioned in section 4.5.1 controls the regularity of triangular meshes. Setting $\alpha = 1$ is equivalent to Delaunay triangulation, while $\alpha = 0$ produces pure data-dependent triangulation. Typically, Delaunay triangulation produces more triangles than data-dependent triangulations for the same error criterion. Thus, for a given error criterion, pursuing the high regularity of meshes could generate many more triangles and require significant increment of processing time. In contrast, pure data-dependent triangulations produce fewer triangles so that we can reduce the processing time. However, pure data-dependent triangulation often creates very narrow triangles, and these slivers could subjectively degrade the quality of rendered objects. Our primary goal is to achieve triangulations such that they represent given height fields with an appropriate balance of triangle count and low error and as small number of slivers.

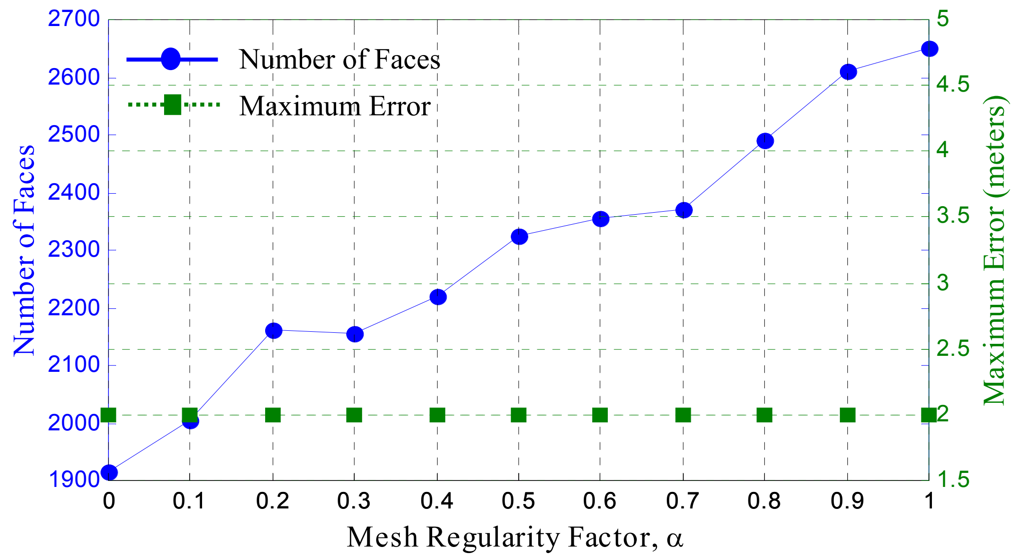
In this experiment, we used the same settings for other parameters as in the previous cases, along with $\delta_d = 1$ and $\delta_\lambda = 0.01$. As the mesh regularity factor α increases from 0 to 1, the number of triangles increases monotonically (Figure 5.4), where 2 iterations are used for refinement step. The triangulation for each value of α satisfies the error criterion $\varsigma = 2$ meters in roughly half of the cases. For the cases that do not satisfy the error criterion, we can easily achieve the desired accuracy by changing the number of iterations to 3 (Figure 5.5). Note that the numbers of faces are slightly higher for the case of 3 iterations than for 2 iterations. The triangulations and the rendered outputs are depicted in Figure 5.6.

The triangulation and approximation results are displayed in Figure 5.6. The results are created using two different α (α = 0.2 and α = 0.8) and 2 iterations for refinement. Both approximations satisfies the error criterion. The triangulation shown in Figure 5.6(a) contains many slivers, which is pretty common in data-dependent triangulations. In contrast, the triangles in triangulation shown in Figure 5.6(b) are well shaped and well distributed. Aa a compromise, most of the later experiments use α = 0.6.



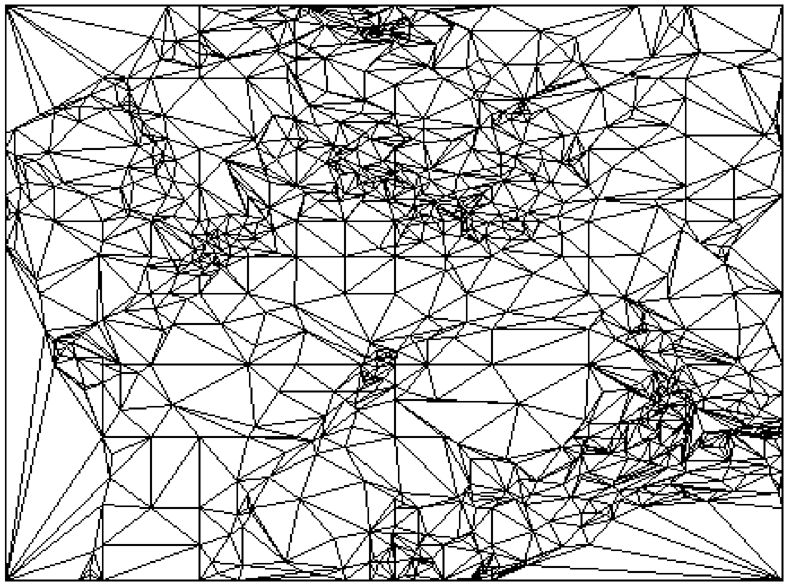
($\varsigma = 2$, $\delta_d = 0.01$, $\delta_p = 0.9$, $\delta_v = 2$, $\delta_\lambda = 0.01$, and $\delta_c = 0.98$)

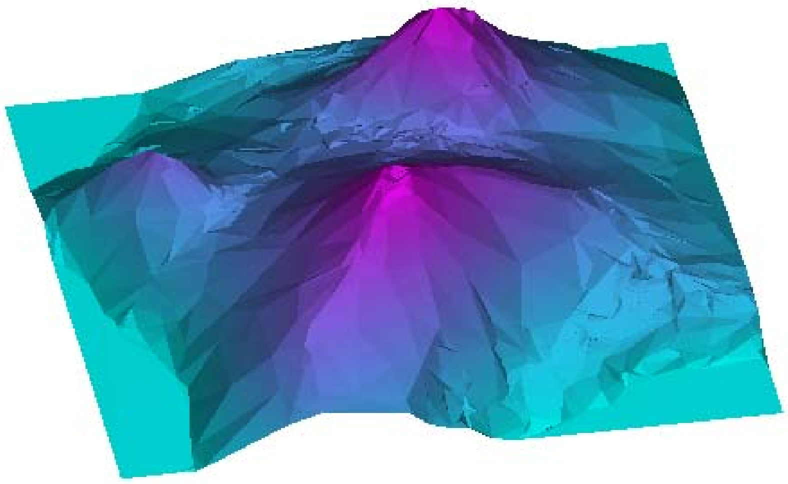
Figure 5.4 Number of faces and maximum error for dataset `Hawaii` according to mesh regularity factor α. 2 iterations are used for the refinement step. In some cases, the error criterion is not satisfied.



($\varsigma = 2$, $\delta_d = 0.01$, $\delta_p = 0.9$, $\delta_v = 2$, $\delta_\lambda = 0.01$, and $\delta_c = 0.98$)

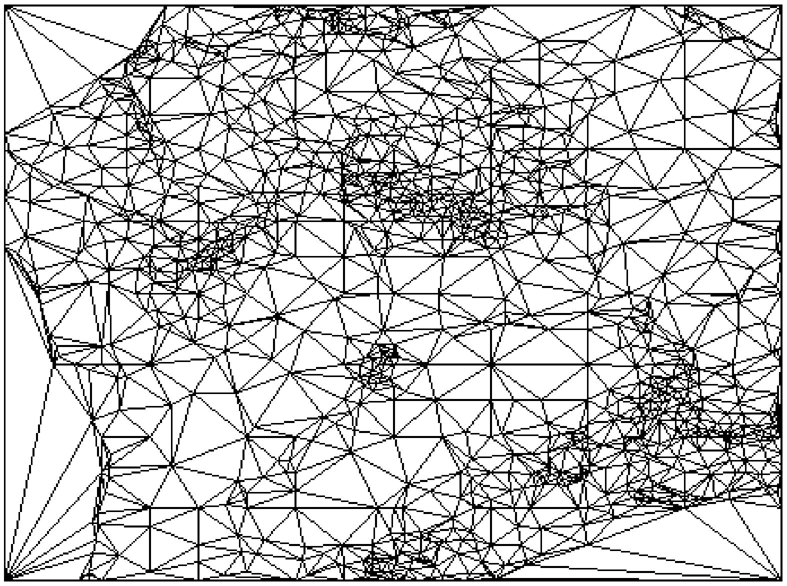
Figure 5.5 Number of faces and maximum error for dataset `Hawaii` according to mesh regularity factor α. 3 iterations are used for the refinement step.

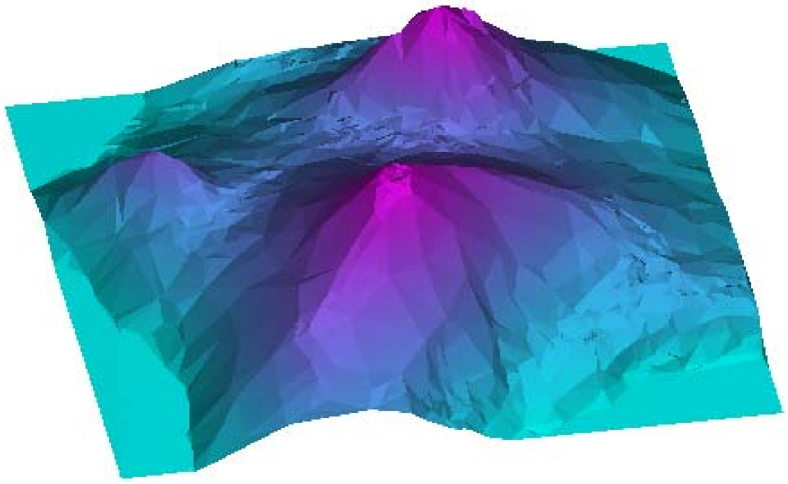




(a)

Figure 5.6 Triangulations and rendered outputs for Hawaii according to different α. Two iterations for refinement step are used for this example. (a) Results for $\alpha = 0.2$. $\left|\hat{\Gamma}_0\right| = 2065$ and $E_{max} = 1.999$.





(b)

Figure 5.6 (continued) (b) Results for $\alpha = 0.8$. $\left|\hat{\Gamma}_0\right| = 2383$ and $E_{max} = 1.999$.

5.3 Multiresolution Mesh Generation.

This section explores the ability of the proposed algorithm to produce multiresolution approximations of input datasets. By its nature, the wavelet transform generates multiresolution information for a signal being analyzed. The proposed surface approximation method utilizes this multiresolution information to construct triangular meshes. More specifically, an initial triangulation, which represents the coarsest approximation of input height fields, becomes finer and finer as the resolution increases. The new approach to the edge split operation directly estimates split points by using the detail information at each resolution level.

For the next experiment, we used 2 iterations of refinement for the synthetic datasets and 3 iterations for terrain datasets. Error criterion ς = 1 unit and ς = 2 meters are used for the synthetic and terrain datasets, respectively. Since the datasets are decomposed up to $M = 5$ and we disregard intermediate levels, there are five resolution levels and the level 0 is the finest level. In order to represent approximation at each resolution level, a triangulation that is output of mesh enhancement step (section 4.5) is used. Table 5.6 summarizes the results, and Figure 5.7 and Figure 5.8 show the results graphically. The number of triangles increases and the maximum error decreases, as the resolution increases.

The dataset `Ball` represents smooth surfaces, except at the occluding boundary of the ball. It therefore has low magnitude of detail coefficients except on the boundary of the ball, so that a lot of activity occurs in that region throughout the progress of the refinement and the mesh enhancement. Similarly, the dataset `Box` represents a simple shape with flat faces on each side of the box object. Because of the flat surfaces, the number of triangles does not increase as much as for the dataset `Ball`. The discontinuities along the edges of the box cause high approximation errors at early stages of the triangulation, and even at $m = 1$ the approximation error has a high value of 21.49 (Figure 5.8). Similarly, the dataset `CraterLake` produces high error values at coarse approximations because of its discontinuities between the peaks and the lake in the dataset. In processing the dataset `Hawaii`, the maximum error is higher at the resolution level $m = 3$, compared to the previous resolution level. This may happen when the mesh regularization step tries to avoid slivers.

Table 5.6 Number of faces and maximum error for 4 datasets at each resolution level.

	Ball		Box		Hawaii		CraterLake	
m	$\lvert\hat{\Gamma}_m\rvert$	$E_{\max}$	$\lvert\hat{\Gamma}_m\rvert$	$E_{\max}$	$\lvert\hat{\Gamma}_m\rvert$	$E_{\max}$	$\lvert\hat{\Gamma}_m\rvert$	$E_{\max}$
4	156	9.31	164	152.61	292	23.82	266	156.56
3	498	9.04	254	150.00	1409	32.84	2122	18.17
2	700	7.72	310	81.74	2021	16.76	3428	7.00
1	766	1.32	322	21.49	2173	16.76	3689	3.20
0	782	0.98	360	1.00	2239	2.00	3758	2.00

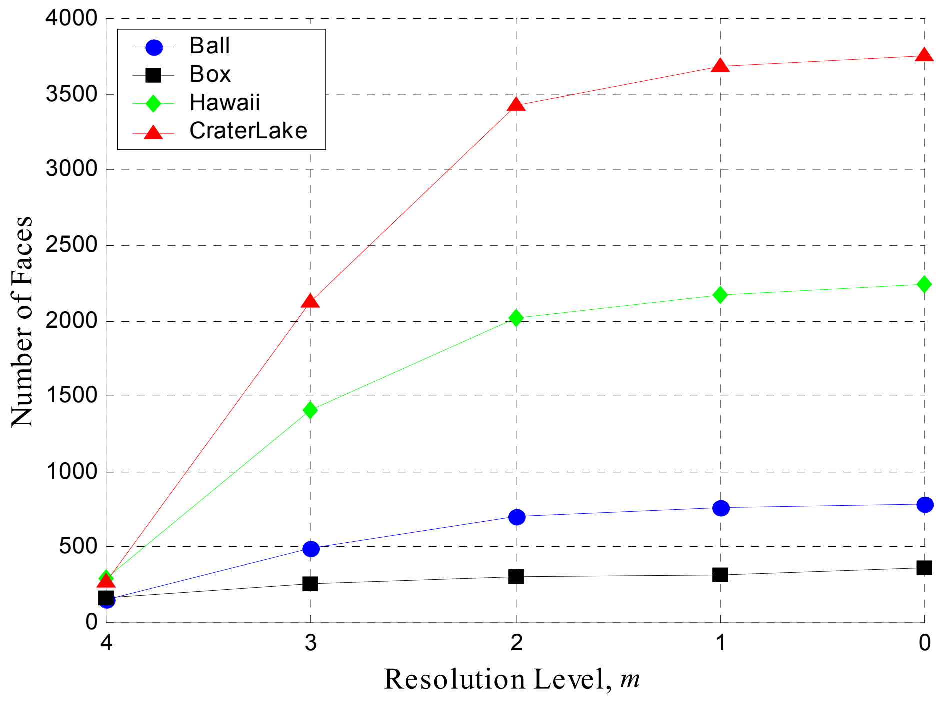


Figure 5.7 Number of faces at each resolution level. The number of triangles increases as the resolution increases.

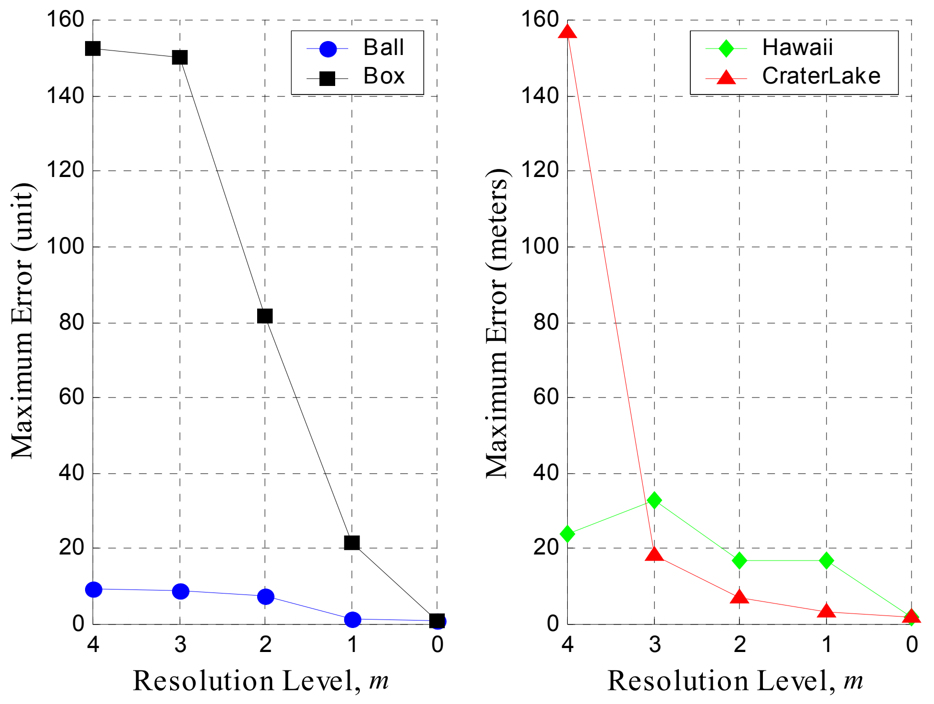
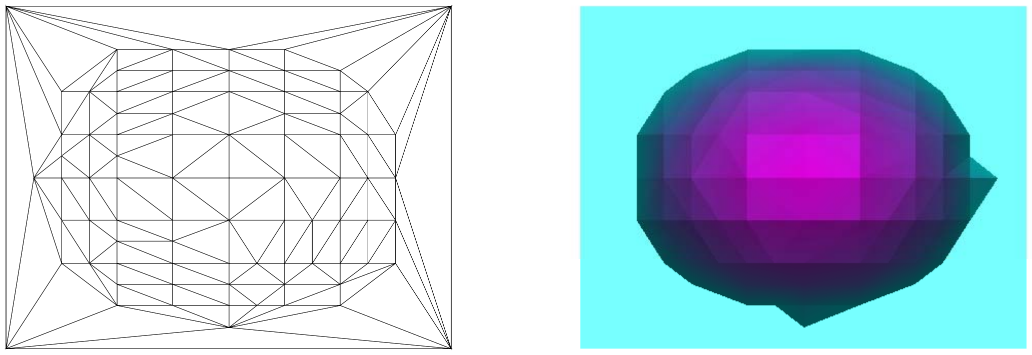
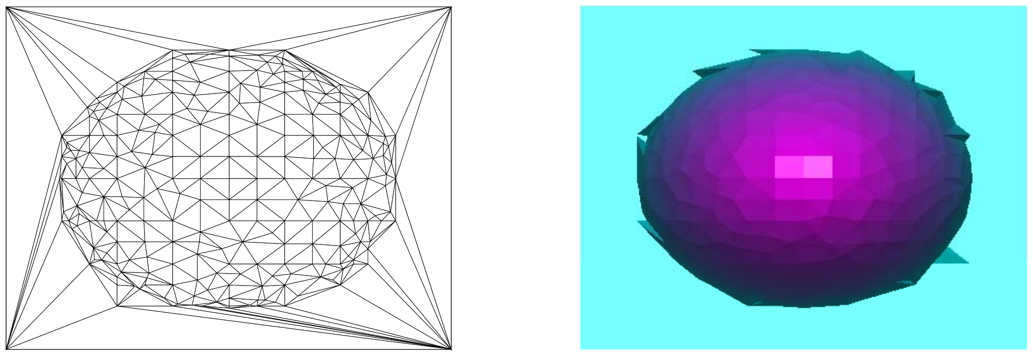


Figure 5.8 Maximum errors for multiresolution approximation for 4 datasets. In most cases, the errors decrease as the resolution increases.

Figure 9.9 shows the triangulations and the rendered outputs of the dataset `Ball` for each resolution level. At the coarsest level $m = 4$, the triangulation $\hat{\Gamma}_4$, which is obtained after the mesh enhancement of initial triangulation, represents overall shape of the dataset but the rendered output $f_{\hat{\Gamma}_4}$ is quite rough especially around the boundary of the ball object. Note that the edges of each triangle try to line up with the level contour of the object, making circular loops. As the refinement succeeds, new triangles are added to generate finer approximations. At $m = 3$, the triangles inside of the ball are fairly evenly distributed and well shaped. However, the boundary of the ball is covered with tiny triangles, and the approximation $f_{\hat{\Gamma}_3}$ is still rough around the boundary. The center of the ball object has not been changed much in triangulation at $m = 2$, $\hat{\Gamma}_2$, where the wavelet detail coefficients have little energy, but considerable activities have occurred near the boundary of the object. The approximation $f_{\hat{\Gamma}_2}$ is quite good except for a few triangles that cause high approximation error. At m = 1, the maximum approximation error is $E_{\max} = 1.32$. The triangulation at this level is fairly well distributed, with good symmetry at the interior of the object, and with tiny and well-shaped triangles around the boundary. The approximation $f_{\hat{\Gamma}_1}$ could therefore be used as a useful output. However, the error criterion is not met at this resolution level, and the triangulation needs to be refined once again. With only 16 more triangles, the goal is achieved at m = 0.

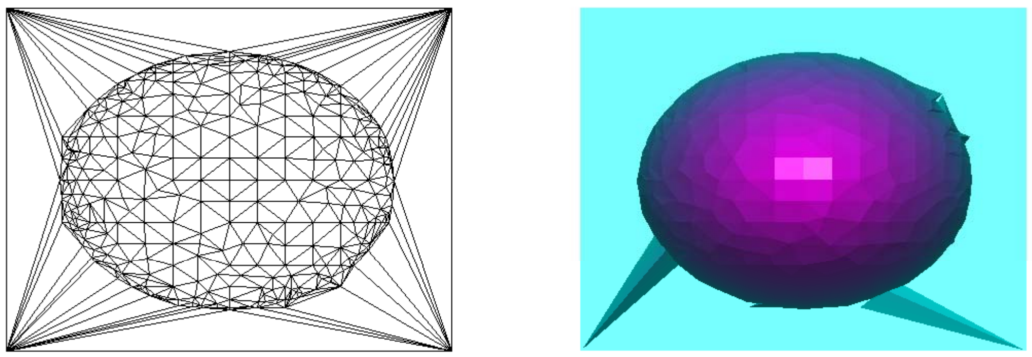


(a) $m = 4$

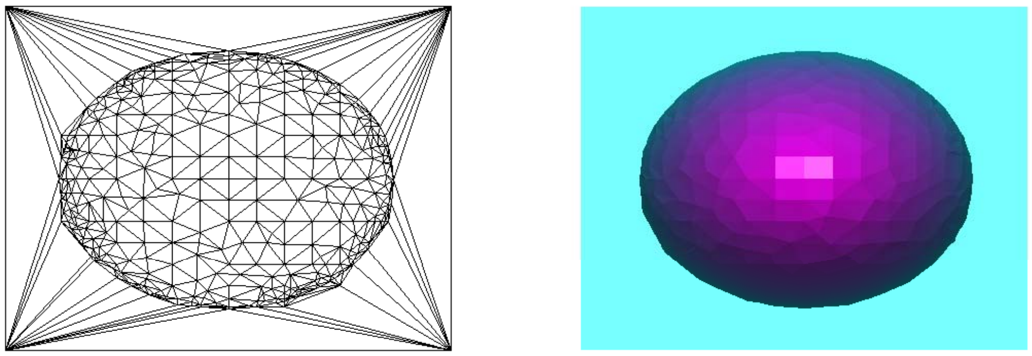


(b) $m = 3$

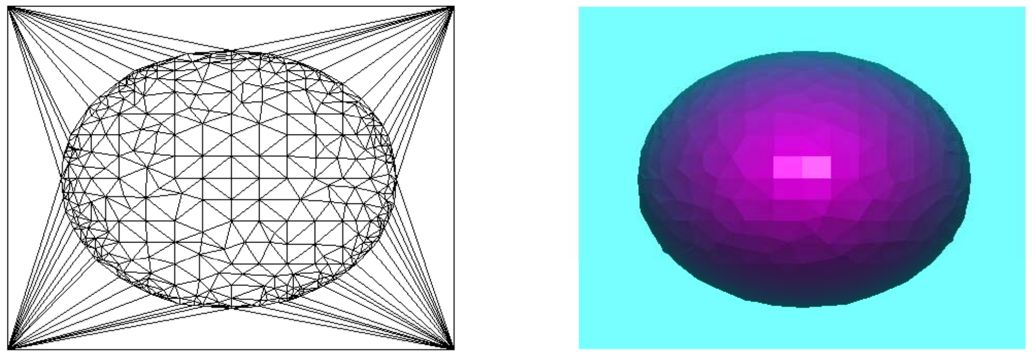
Figure 5.9 Multiresolution approximation of dataset `Ball`. The parameters are set as follows: $\varsigma = 1$, $\delta_d = 0.1$, $\delta_p = 0.9$, $\delta_v = 2$, $\alpha = 0.8$, $\delta_\lambda = 0.001$, and $\delta_c = 0.98$.



(c) $m = 2$



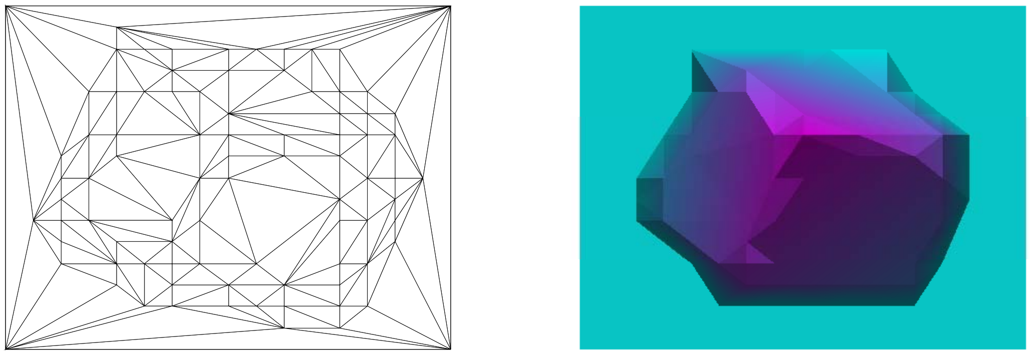
(d) $m = 1$



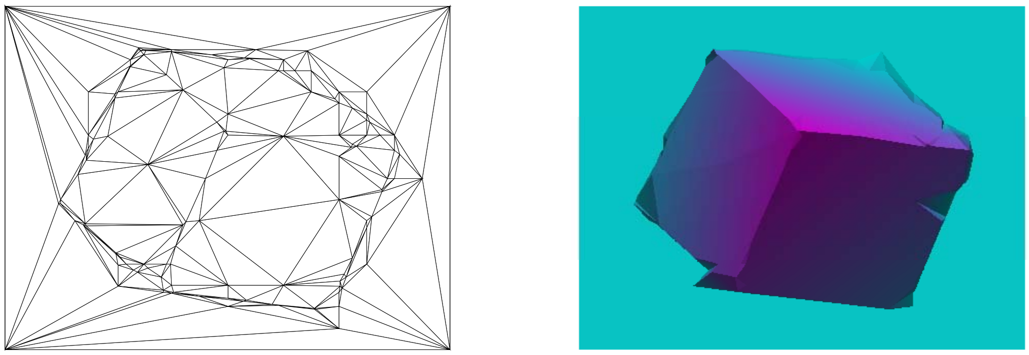
(e) $m = 0$

Figure 9.9 (continued). The approximation at the finest level is quite good, with a maximum error of 1 unit.

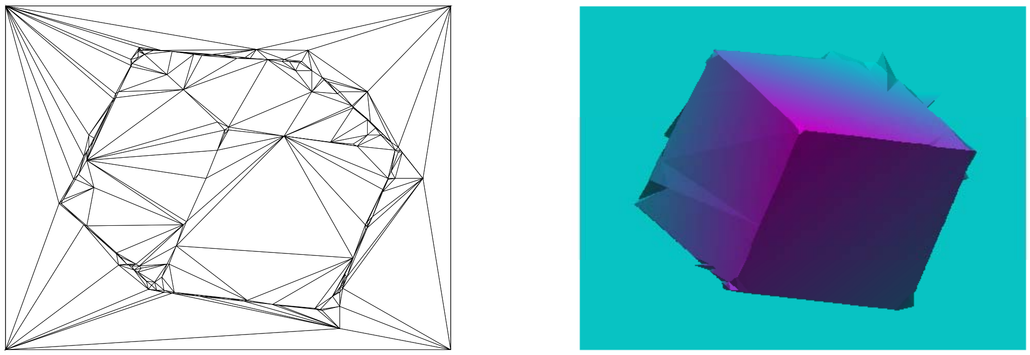
The multiresolution approximation of the dataset `Box` is presented in Figure 9.10. The data values range from 0 to 250 units. The coarsest approximation $f_{\hat{\Gamma}_4}$ is quite rough, but still reasonably represents the overall shape of the dataset, which can be recognizable as a box. Its maximum error at this level is $E_{\max} = 152.61$. Those high errors are common at the early stages of refinement when the datasets contain substantial discontinuities. At $m = 3$, the visual appearance of the approximation $f_{\hat{\Gamma}_3}$ is much improved and almost all of important features of a box are well represented with the triangulation $\hat{\Gamma}_3$. However, the maximum error is still high, but only few triangles in the vicinity of the discontinuities have that high error. At this level of resolution, the triangles are slim around discontinuities and become larger on each face of the box object. Also, the triangle edges are placed along with the edges of the box object, and the vertices are located around corners of the box. However, the triangulation $\hat{\Gamma}_3$ is not an optimal solution in locating triangle edges and vertices, because the proposed algorithm does not search for optimal locations for them. Instead, the algorithm modifies the initial triangulation by the iterative operations of edge split and vertex removal. As a result, at the next level $m = 2$, the appearance of the triangulation $\hat{\Gamma}_2$ is quite different from the previous one, having vertices closer to the critical points (corner points of the box object) in the dataset and placing the triangle edges closer to the edges of the box object. At $m = 1$, the triangles are slimmer around the discontinuities and the box edges are well represented with the triangle edges. The approximation $f_{\hat{\Gamma}_1}$ is very good except at the leftmost corner and the right bottom of the box. However, the approximation error $E_{\max} = 21.49$ is still considerable. At the highest resolution level, the approximation finally meets the error criterion, and the triangulation has 360 faces.



(a) $m = 4$

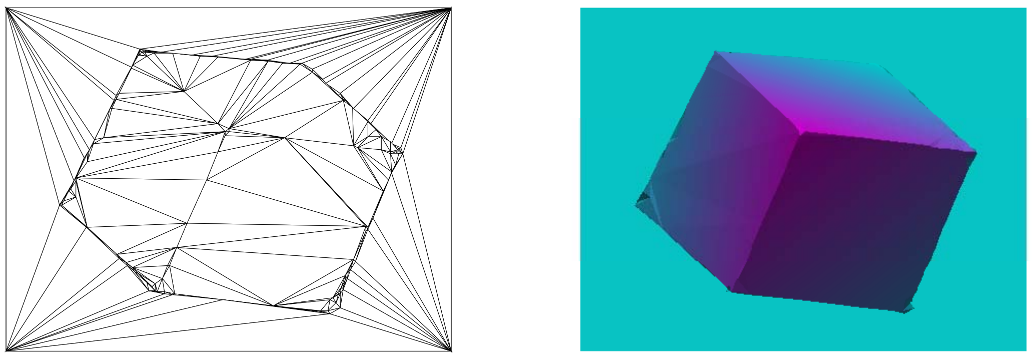


(b) $m = 3$

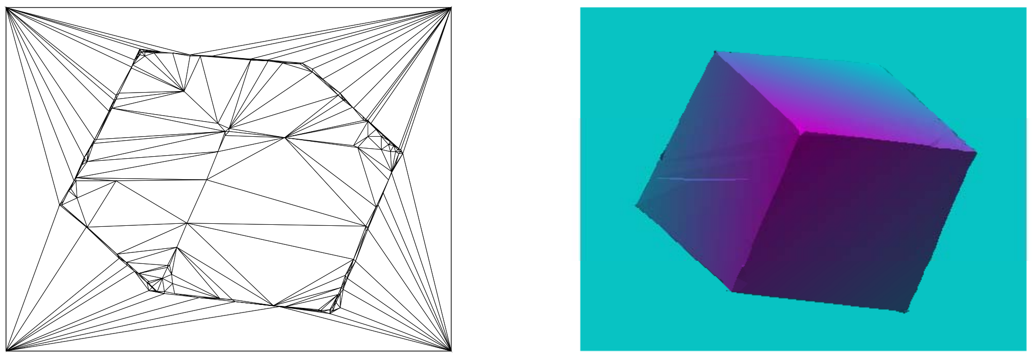


(c) $m = 2$

Figure 5.10 Multiresolution approximation of dataset `Box`. The parameters are set as follows: $\varsigma = 1$, $\delta_d = 0.1$, $\delta_p = 0.9$, $\delta_v = 2$, $\alpha = 0.6$, $\delta_\lambda = 0.01$, and $\delta_c = 0.98$.



(d) $m = 1$

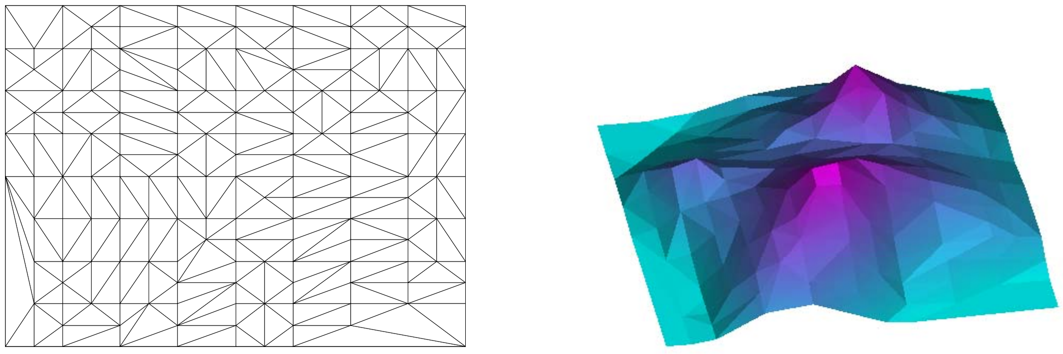


(e) $m = 0$

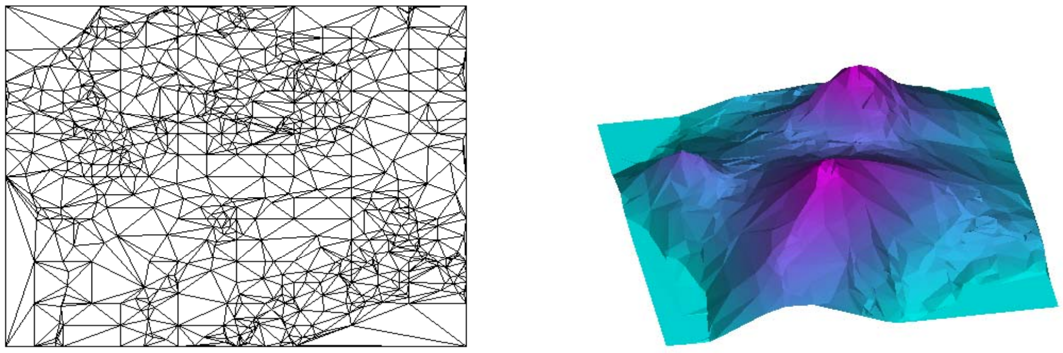
Figure 9.10 (continued). As the resolution level increases, the triangles get slimmer around the discontinuities and represent the box edges. The approximation $f_{\hat{\Gamma}_1}$ is very good except at the leftmost corner and the right bottom of the box. The approximation error at this level is still high ($E_{\max} = 21.49$) because of discontinuities in the dataset. At the highest resolution level, the approximation finally meets the error criterion, and the triangulation has 360 faces.

Figure 9.11 shows the result of the multiresolution mesh generation of the dataset `Hawaii`. At the coarsest triangulation $\hat{\Gamma}_4$, it is hard to see the overall shape of input dataset. Only the large triangles at the left and right bottom corners are noticeable. The approximation $f_{\hat{\Gamma}_4}$ does not show the crater on top of the mountain Mauna Loa located at the middle of the dataset. At the resolution level $m = 3$, the triangulation around the peaks becomes dense while leaving large triangles in the sea and on the sides of each mountain. In the approximation $f_{\hat{\Gamma}_3}$, the crater on top of Mauna Loa starts to appear. As the resolution level decreases and accordingly as the resolution increases, the triangulation becomes finer around peaks and along the coastline, but some of triangles remain unchanged in flat areas such as the sides of the mountains and the sea. At $m = 2$, much details are present in the approximation $f_{\hat{\Gamma}_2}$, and now the crater is clearly noticeable. To complete the approximation, more details are added to $f_{\hat{\Gamma}_2}$ around peaks and along the coastline, generating finer approximations $f_{\hat{\Gamma}_1}$ and $f_{\hat{\Gamma}_0}$.

The dataset `CraterLake` represents mountainous terrain in which many sharp peaks are present. Therefore, the wavelet detail energy even at high-resolution levels is considerably high almost everywhere except the lake region. This causes a sharp increment of the number of triangles when refining the triangulation $\hat{\Gamma}_4$ to produce $\hat{\Gamma}_3$. In addition, the discontinuities between the peaks and the lake cause high approximation error at the coarsest level. At $m = 3$, the number of triangles $\left|\hat{\Gamma}_3\right|$ is 2,122, almost ten times the previous case (Table 5.6). Even though it is in an early stage of the refinement, the approximation captures almost all the features in the mountainous area, including all ridges and peaks. The island in the lake is also fairly well represented. The approximation error at this level is only 18.17 meters, which is relatively low compared to the previous error of 152.17 meters. Denser triangulation is achieved at $m = 2$, and its approximation $f_{\hat{\Gamma}_2}$ is much more improved. One noticeable change can be found in the island, in which the crater has appeared on the top of the island that was not apparent in previous level. At levels $m = 1$ and 0, a slight increment of the number of triangles adds even more details, decreasing the approximation error to 3.2 and 2 meters.

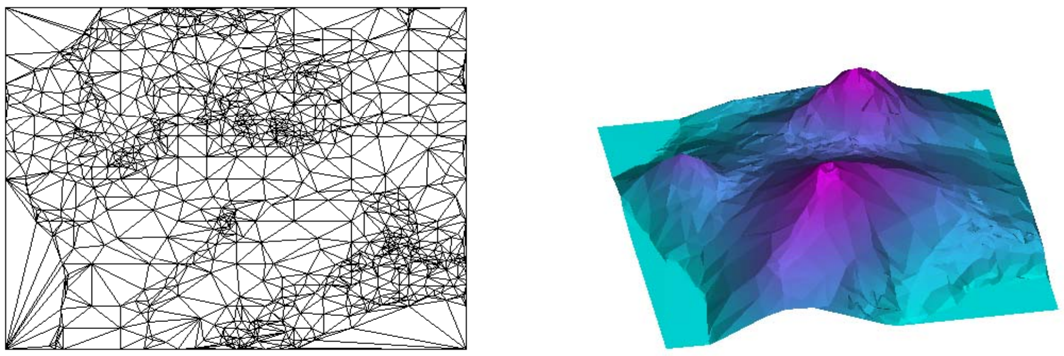


(a) $m = 4$

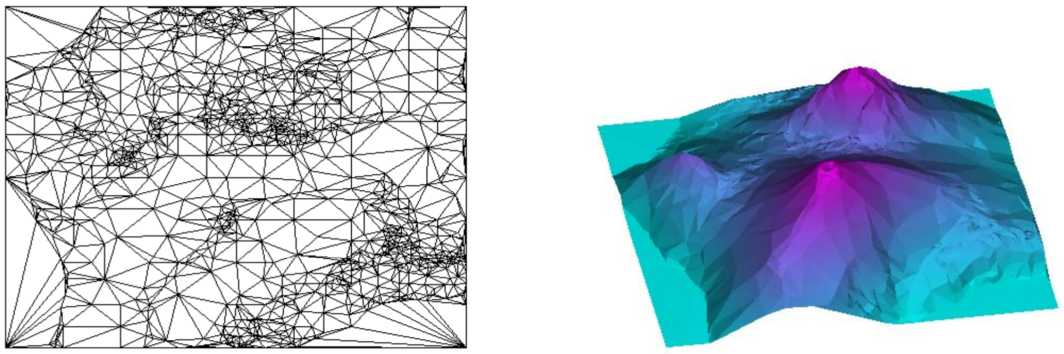


(b) $m = 3$

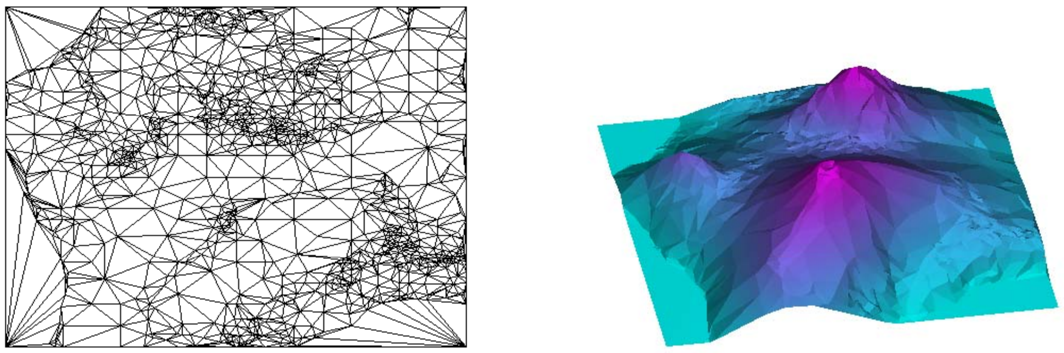
Figure 5.11 Multiresolution approximation of dataset `Hawaii`. The parameters are set as follows: $\varsigma = 2$, $\delta_d = 0.1$, $\delta_p = 0.9$, $\delta_v = 2$, $\alpha = 0.6$, $\delta_\lambda = 0.01$, and $\delta_c = 0.98$. At the coarsest level, it is hard to see the overall shape of input dataset. However, as the resolution increases, the triangulation around the peaks becomes dense while leaving large triangles in the sea and on the sides of each mountain.



(c) $m = 2$

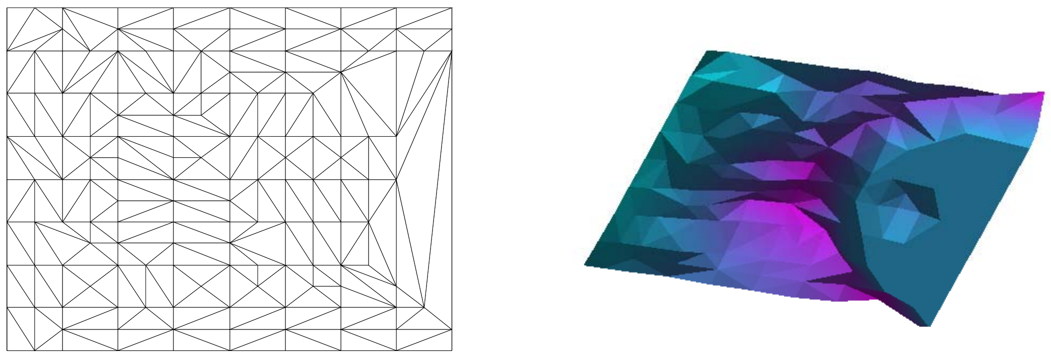


(d) $m = 1$

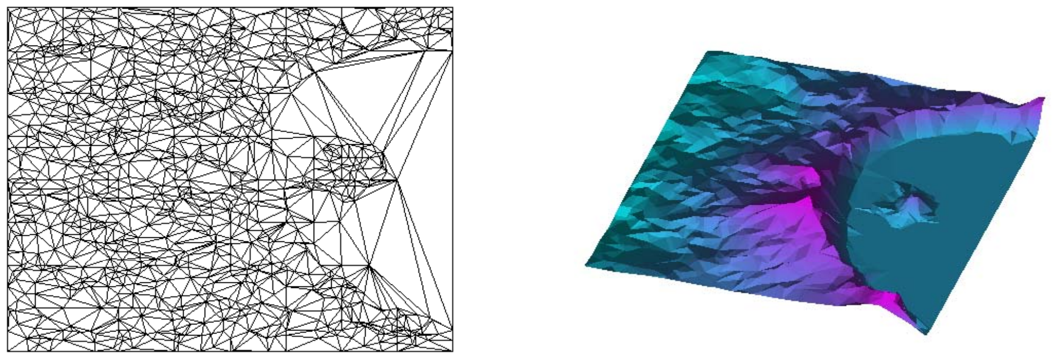


(e) $m = 0$

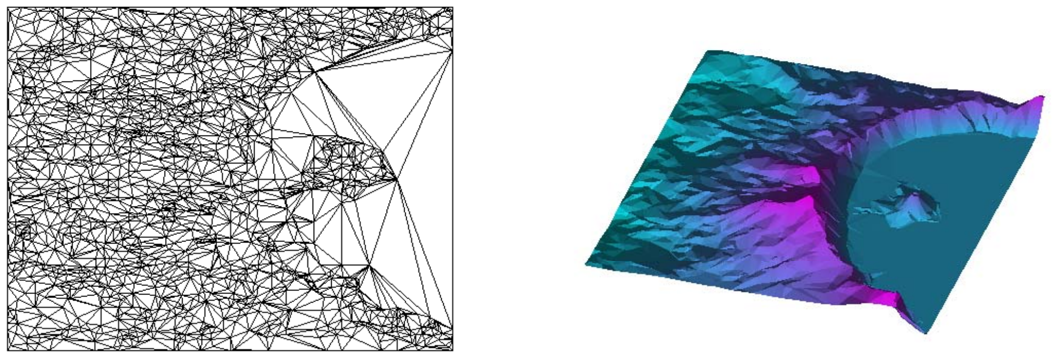
Figure 9.11 (continued). Much details are present in the approximation $f_{\hat{\Gamma}_2}$, and more details are added to it around peaks and along the coastline, generating finer approximations $f_{\hat{\Gamma}_1}$ and $f_{\hat{\Gamma}_0}$.



(a) $m = 4$

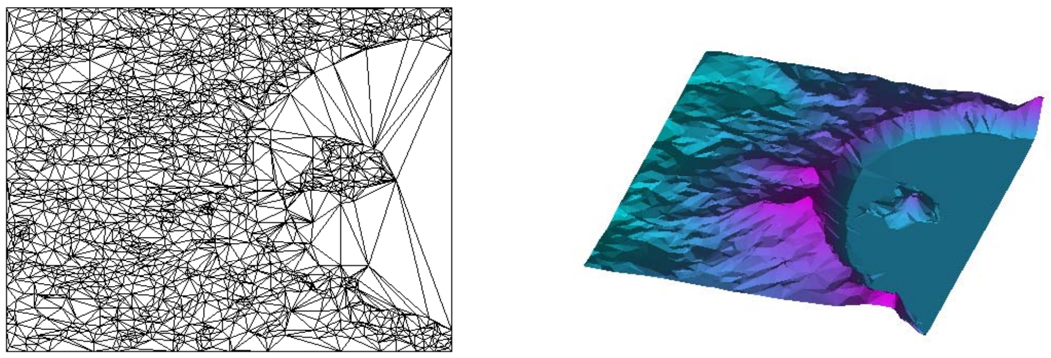


(b) $m = 3$

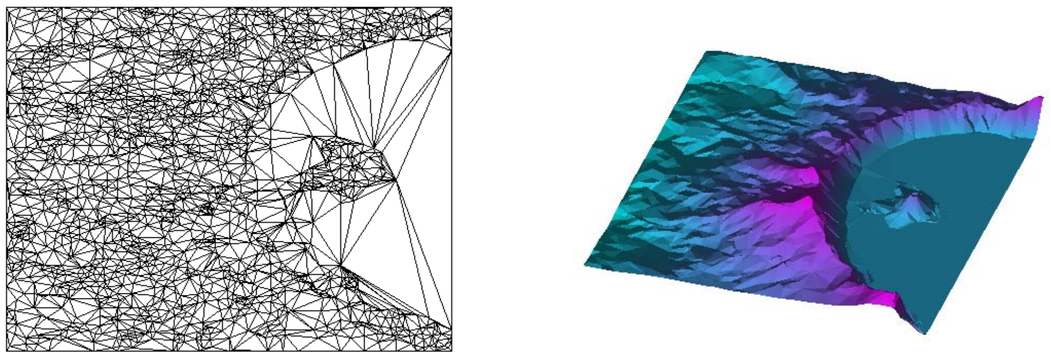


(c) $m = 2$

Figure 5.12 Multiresolution approximation of dataset `CraterLake`. The parameters are set as follows: $\varsigma = 2$, $\delta_d = 0.1$, $\delta_p = 0.9$, $\delta_v = 2$, $\alpha = 0.6$, $\delta_\lambda = 0.01$, and $\delta_c = 0.98$.



(d) $m = 1$



(e) $m = 0$

Figure 9.12 (continued). The dataset contains many sharp peaks, so that the wavelet detail energy even at high-resolution levels is considerably high almost everywhere except the lake region. This causes a sharp increment of the number of triangles when refining the triangulation $\hat{\Gamma}_4$ to produce $\hat{\Gamma}_3$. Denser triangulations are achieved and the approximation error decreases as the resolution level increases. At the finest level, $\left|\hat{\Gamma}_0\right| = 3,758$ and $E_{\max} = 2$ meters.

5.4 Level of Detail Control

The proposed surface approximation algorithm is fairly suitable for level of detail (LOD) control. This section will describe and demonstrate the ability of the proposed algorithm to do this. The LOD control in approximation models is an important feature required in computer graphics applications. With this capability, the graphics system will draw a portion of interest with full resolution, while at the same time leaving the regions outside of the interest area in low resolution. One reason for LOD control is to reduce the rendering time in graphics systems, and another is to direct computational resources toward particular objects. LOD control with the use of wavelet transform is straightforward because the manipulation of wavelet coefficients directly affects the resolution of reconstructed signals. Likewise, the proposed algorithm, which utilizes the wavelet detail coefficients for refinement process, provides for LOD control by thresholding the wavelet detail energy.

As an experiment for the LOD control of the proposed algorithm, we have used the `CraterLake` dataset with 3 iterations for refinement and $\varsigma = 2$ meters as the error criterion. In the proposed refinement method, the wavelet detail energy determines whether each triangle needs to be subdivided in order to obtain a finer approximation at the next resolution level. Therefore, the threshold δ_d for the wavelet detail energy is an important factor to determine the quality of the final approximation in both accuracy and appearance. By changing δ_d from 0.1 to 10, the size of the final triangulation is gradually reduced and the maximum error increases as shown in Table 5.7 and Figure 5.13. At δ_d = 0.1, the number of triangles is 3,758 and the error criterion $\varsigma = 2$ is satisfied. The number of vertices used in this triangulation is 1,916, which is only 2.92 % of the original data points in a dataset of size 256 × 256. Thus, in this example, the use of a small fraction of data can approximate the input dataset within 2 meters deviation. As described before, since the height value in the dataset `CraterLake` ranges from 1,730 to 2,477 meters, a 2-meter maximum error is reasonably accurate for many applications in computer graphics and computer visions.

Table 5.7 Numerical results for LOD control according to wavelet detail energy.

δ_d	0.1	0.5	1	2	3	4	5	6	7	8	9	10
$\lvert\hat{\Gamma}_0\rvert$	3758	3689	3526	2872	2254	1811	1582	1393	1280	1180	1063	1065
$\lvert\hat{\mathbb{V}}_0\rvert$	1916	1889	1800	1470	1160	941	828	735	678	631	569	570
%	2.92	2.88	2.75	2.25	1.77	1.44	1.26	1.12	1.03	0.96	0.87	0.87
$E_{\max}$	2	3.4	7.0	11.4	13.9	16.9	19.8	28.0	28.6	29.5	30.5	30.5

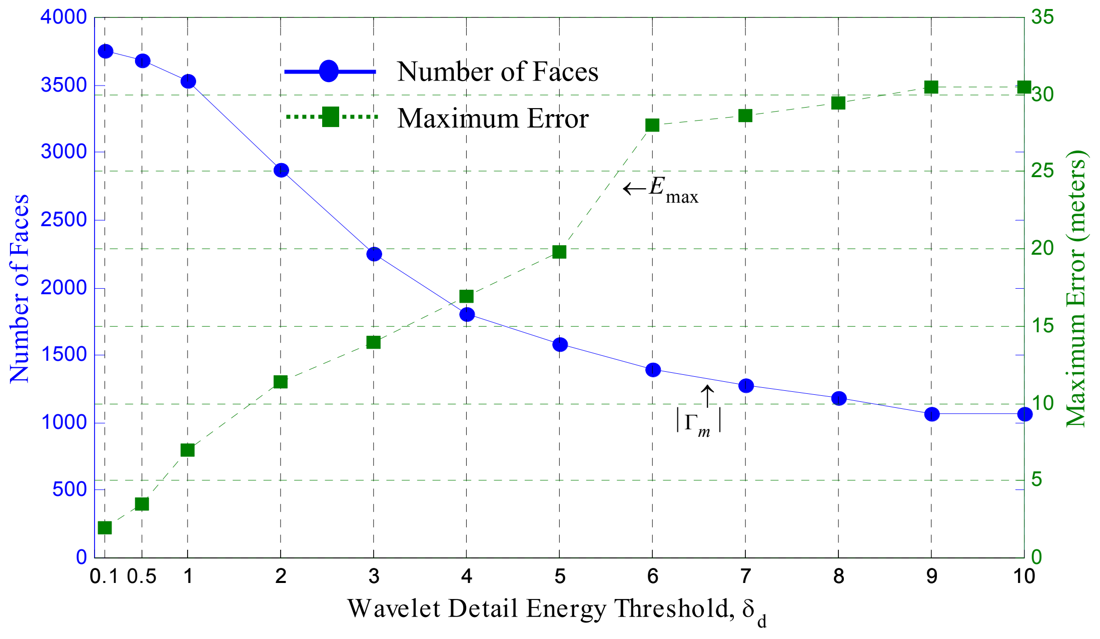
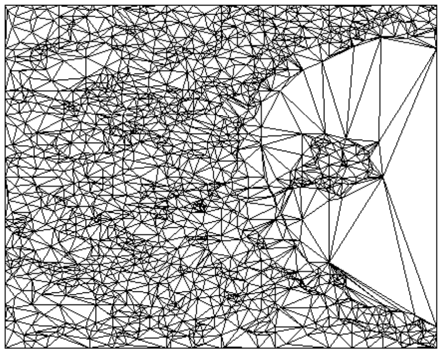
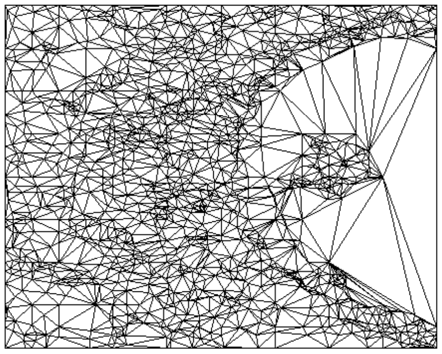


Figure 5.13 Number of faces and maximum error with respect to different δ_d. The proposed approximation method has a capability of LOD control by using different values for wavelet detail energy threshold δ_d.

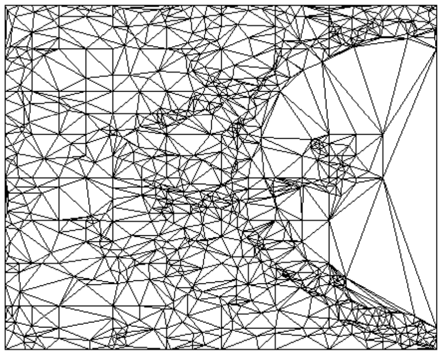
Six different triangulations and their approximations are shown in Figure 5.14 and 5.15, respectively. A triangulation of 3,526 faces is produced with δ_d = 1 and its maximum approximation error is 7.0 meter. The number of vertices used in the triangulation is 1,800, which is 2.75 % of the input dataset of size 256 × 256. The rendered output (Figure 5.15(a)) still keeps the overall shape of the input dataset (compare to the original data shown in Figure 5.1(d)), and almost all peaks of the mountainous area remain largely unchanged. Triangle density is fairly high in almost everywhere except the lake. At $\delta_d = 2$, we can obtain a fairly good approximation with the use of only 2.25 % of data points, even though the maximum error is 11.4 meter and some of finer details have been removed. Dense triangles still remain around the peaks and make a good approximation for the ridges of mountains. However, large triangles are created to cover the plain area of the input dataset. As δ_d increases, the triangulations become coarser and coarser, in which case larger triangles are created for almost all areas. At $\delta_d = 6$, most fine details are disappeared, but dense triangles still remain in the peaks along the lake. The use of higher values of δ_d produces even coarser triangulations in which the shape of the island in the lake is noticeably degraded.



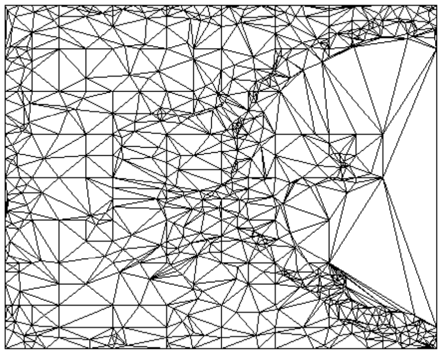
(a) $\delta_d = 1$



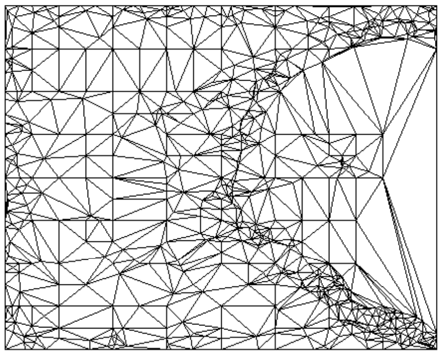
(b) $\delta_d = 2$



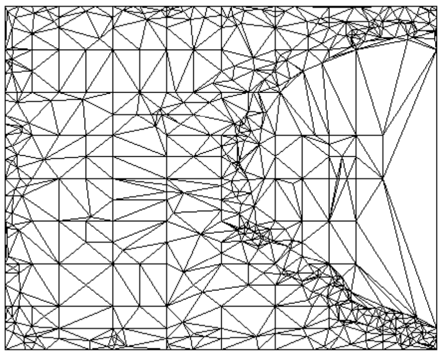
(c) $\delta_d = 4$



(d) $\delta_d = 6$

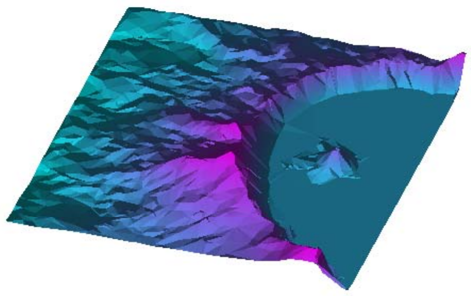
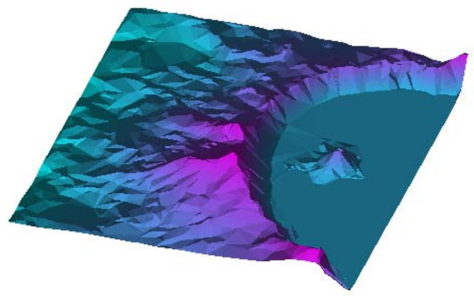


(e) $\delta_d = 8$

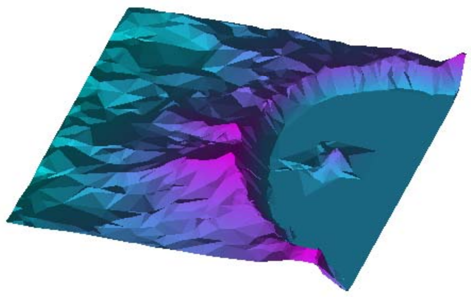
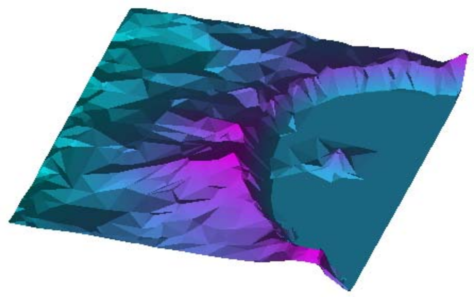


(f) $\delta_d = 10$

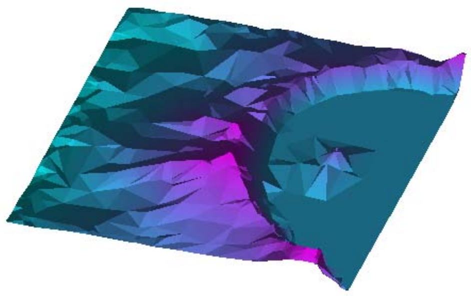
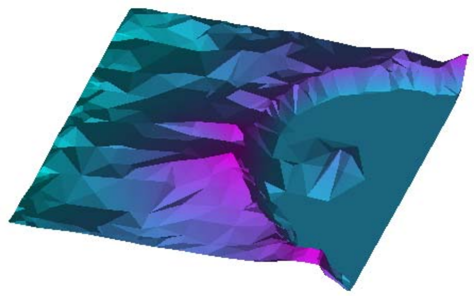
Figure 5.14 Triangulations reflecting LOD control with different δ_d. At low values of the threshold δ_d, dense triangulations are created while sparse triangulations are obtained with high δ_d.

(a) $\delta_d = 1$ (b) $\delta_d = 2$

(c) $\delta_d = 4$ (d) $\delta_d = 6$

(e) $\delta_d = 8$ (f) $\delta_d = 10$

Figure 5.15 Approximated surfaces with different LODs. Simply by changing the threshold δ_d, we can control the level of detail in the resulting approximations.

5.5 Performance on Noisy Data

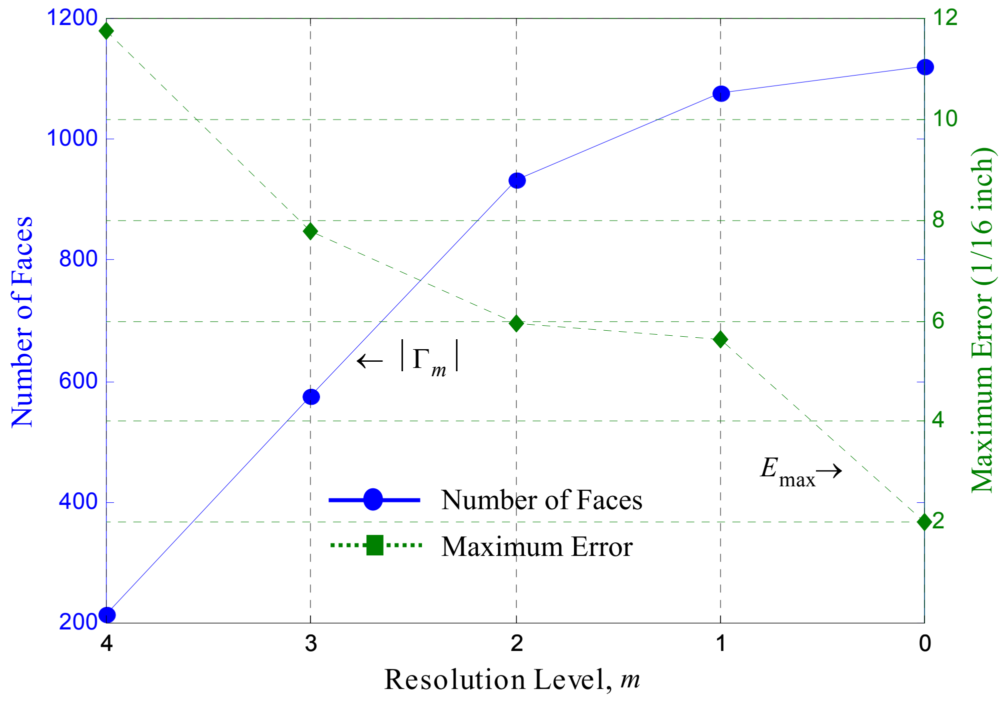
No explicit mechanism exists in the proposed approximation method to handle noisy datasets. Any data point in a dataset is considered as a possible vertex in triangulation. The best strategy is to reduce the noise by applying an appropriate filter on the noisy data before decomposing the dataset. For example, we applied a 5×5 median filter to remove peak noise in the dataset `Perc15` in Figure 5.1(f). Another approach for the noise reduction is to utilize the de-noising capability of wavelet transforms. The typical wavelet de-noising procedure thresholds the wavelet detail coefficients, and then reconstructs the data based on the original approximation coefficients and the modified detail coefficients. In this section, we are going to approximate the datasets `Board` and `Perc15`. Both datasets contain a considerable amount of noise and the resulting triangulations reflect the noise by increasing the number of triangles.

The numerical results of the approximations are summarized in Table 5.8. For the dataset `Board`, the approximation error ς = 1/8 inch is allowed for each triangle. The number of triangles gradually increases and the maximum error decreases as the level of resolution decreases (Figure 5.16). At the final approximation, only 0.874 % of data points are used to construct the approximation with 1/8-inch accuracy.

Even better efficiency is achieved in the dataset `Perc15`, because only 0.317 % of data points are used to construct an approximation of 2 cm accuracy. The approximation error at early stages is pretty high due to the discontinuities. Figure 5.17 shows the behavior of the multiresolution mesh generation graphically.

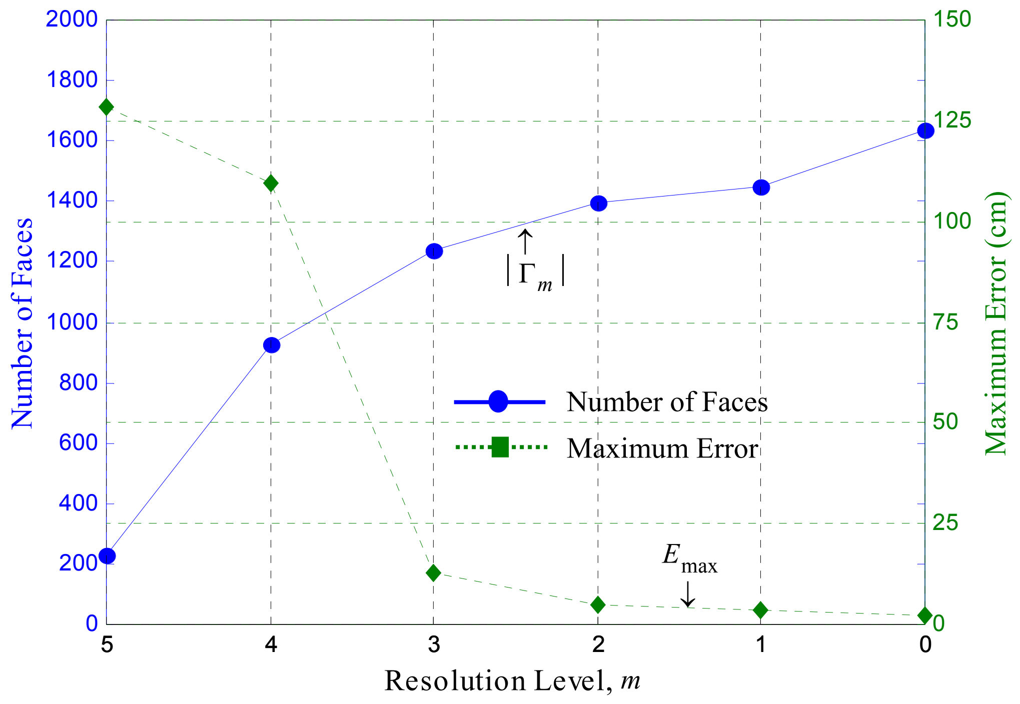
Table 5.8 Number of faces and maximum error at each resolution level for noisy datasets.

M	`Board`				`Perc15`			
	$\lvert\hat{\Gamma}_m\rvert$	$\lvert\hat{\mathbb{V}}_m\rvert$	%	$E_{\max}$	$\lvert\hat{\Gamma}_m\rvert$	$\lvert\hat{\mathbb{V}}_m\rvert$	%	$E_{\max}$
5	-	-	-	-	231	129	0.049	128.51
4	214	120	0.183	11.74	927	477	0.182	109.67
3	574	229	0.349	7.79	1239	631	0.241	12.90
2	934	480	0.732	5.93	1397	712	0.272	4.69
1	1078	551	0.841	5.64	1451	738	0.282	3.65
0	1122	573	0.874	2.0	1637	832	0.317	2.0



$(\delta_d = 0.05,\ \delta_p = 0.9,\ \delta_v = 2,\ \alpha = 0.6,\ \delta_\lambda = 0.01,\text{ and } \delta_c = 0.96)$

Figure 5.16 Number of faces and maximum error for the dataset `Board` ($\varsigma = \frac{1}{8}$ inch).

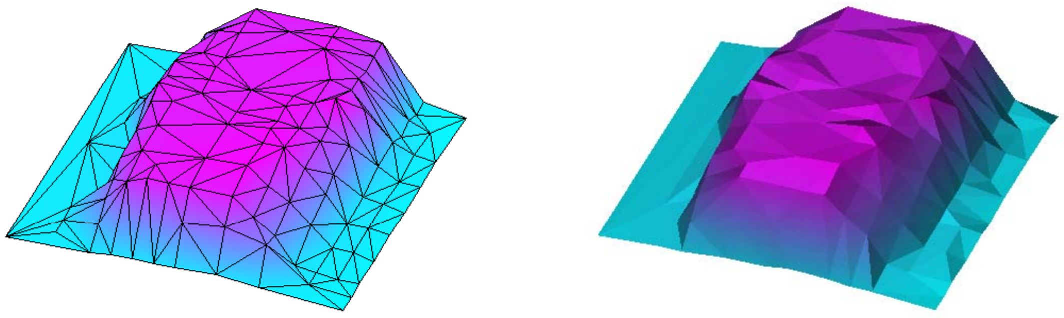


($\delta_d = 0.05$, $\delta_p = 0.9$, $\delta_v = 5$, $\alpha = 0.6$, $\delta_\lambda = 0.01$, and $\delta_c = 0.96$)

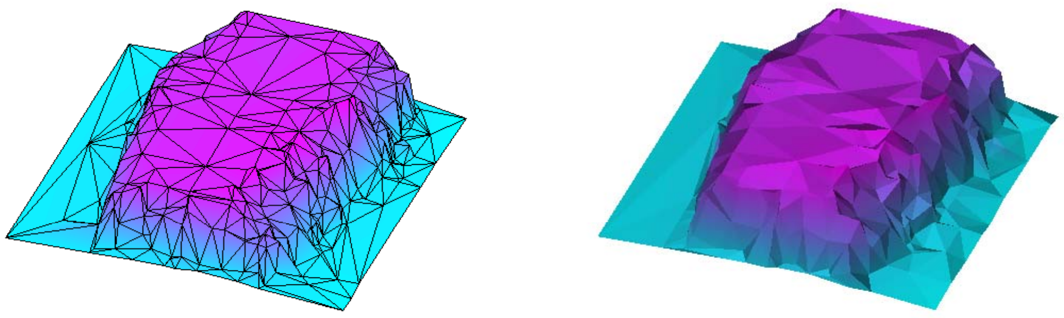
Figure 5.17 Number of faces and maximum error for the dataset `Perc15` ($\varsigma = 2$ cm).

Figure 5.18 shows the results of multiresolution approximation for the dataset `Board`. For each resolution level, the wire frame models (left) and the rendered outputs (right) are displayed. As in previous renderings, the Lambertian reflection model is used. At the coarsest approximation (m = 4), large triangles are created on the top of the board data and in the background. Note that the triangle edges represent the board edges well. This portion of a board is called wane, which is caused by remaining bark or a beveled end. Therefore, it is possible to use the proposed initial triangulation method for detecting wane area of rough lumber. As described in section 3.6, the initial triangulation can be achieved very fast making the method feasible for real-time application. In order to locate the wane boundary, the surface characteristics are analyzed based on the surface normal of each triangle, and the triangle edges that encounter high variation of surface normals can be considered as a rough estimate of wane of rough lumber.

At $m = 3$, the triangulation on top of the board does not change much while a lot more activities take places in wane area. Even though there is noise on top of the board, the approximation shows very reliable result. As the resolution level decreases, the same phenomenon occurs, leaving unchanged on top of the board and producing dense triangulation in the wane area. At $m = 1$ and $m = 0$, the results look almost the same since the difference in the number of triangles is only 44.

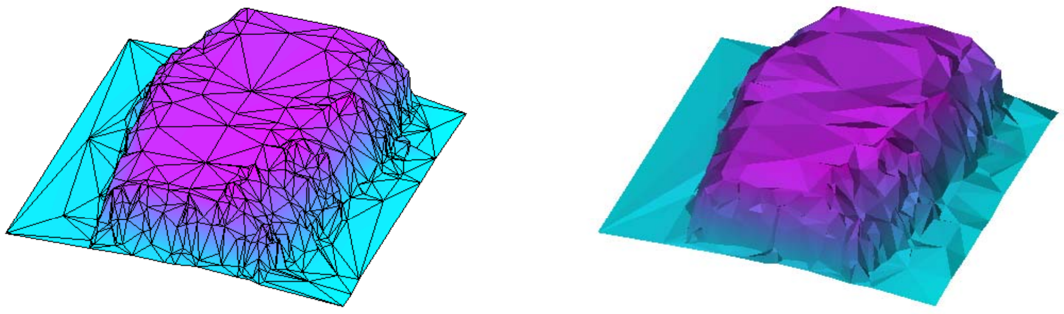


(a) $m = 4$

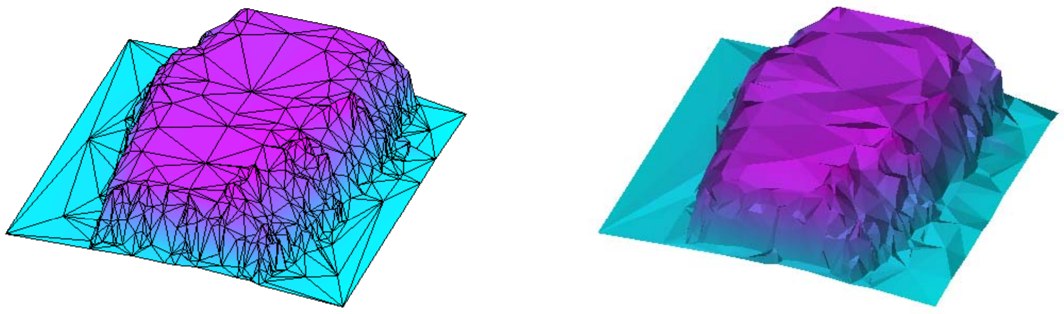


(b) $m = 3$

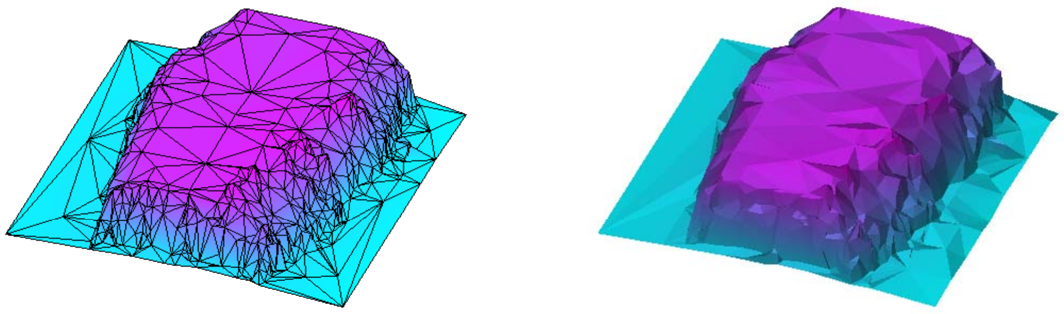
Figure 5.18 Multiresolution approximation of noisy dataset `Board`. The parameters are set as follows: $\varsigma = \frac{1}{8}$, $\delta_d = 0.05$, $\delta_p = 0.9$, $\delta_v = 2$, $\alpha = 0.6$, $\delta_\lambda = 0.01$, and $\delta_c = 0.96$. For each resolution level, the wire frame models (left) and the rendered outputs (right) are displayed.



(c) $m = 2$



(d) $m = 1$



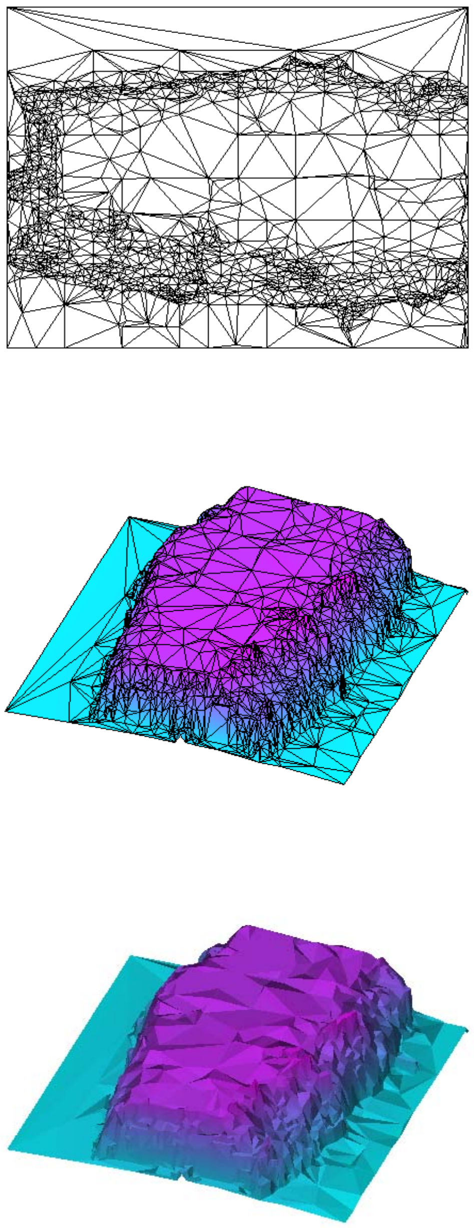
(e) $m = 0$

Figure 5.18 (continued). Triangulations on top of the board do not change much while a lot more activities take places in wane area. Even though there is noise on top of the board, the approximation shows very reliable result. The proposed initial triangulation method can be used as a wane detection method for rough lumber.

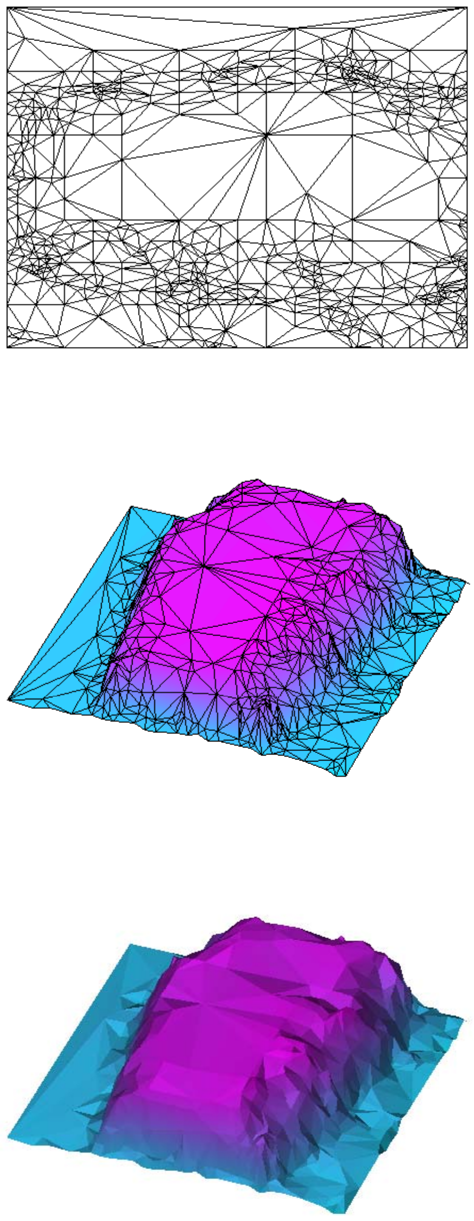
In order to explore the effect of de-noising, we recreated the surface approximations with ς = 1/16 inch while keeping remaining all other parameters the same. The triangulation and its approximation are shown in Figure 5.19(a). Much denser triangulation is obtained compared to the case when ς = 1/8 inch is used, and smaller triangles are produced even on top of the board. $\left|\hat{\mathbb{V}}_0\right|$ = 1,529 (2.33 % of input) vertices are used and $\left|\hat{\Gamma}_0\right|$ = 3,015 triangles are created for this approximation. However, in this triangulation, most of triangles are redundant and unnecessary for our goal that locates wane boundary.

On the other hand, we have applied the de-noising technique for the noisy input data, and instead of using the original data, the reconstructed data is used for the height value. For de-noising purposes, there are various threshold selection rules [Donoho and Johnstone 94, Johnstone and Silverman 97] and thresholding methods (hard or soft) [Donoho 95], but the discussion on this subject is beyond our scope. We have selected the threshold values by using Stein's unbiased risk estimate method [Stein 81] and used soft-thresholding for detail coefficients at each decomposition level.

The resulting triangulation is shown in Figure 5.19(b). Even though an error criterion of ς = 1/16 inch is used, the rendered outputs are very similar to the ones in Figure 5.18(e), and the surface characteristics are well recovered with fewer triangles than ones in (a). With the de-noised data, $\left|\hat{\mathbb{V}}_0\right|$ = 638 (0.97 % of input) vertices are used, which is less than half of that for the previous approximation, and $\left|\hat{\Gamma}_0\right|$ = 1,202 triangles are created. As shown in Figure 5.18(e), large triangles are created on top of the board while tiny triangles represent the wane area. While detail features are recovered at the rendered output in Figure 5.19(a), a much smoother approximation is obtained in (b). Since we have already decomposed the input data, a little effort can reconstruct the de-noised signals. First, apply thresholding the detail coefficients and then perform the inverse wavelet transform to reconstruct signals. Therefore, the de-noising step can be done very fast quickly.



(a)



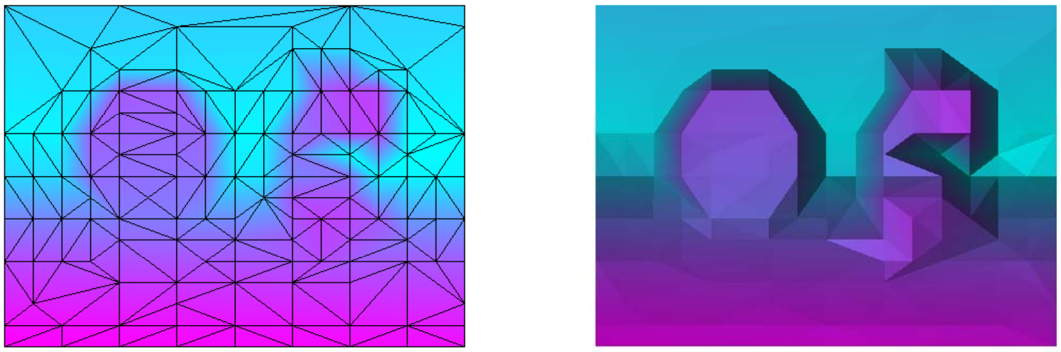
(b)

Figure 5.19 Illustration of the effect of de-noising. (a) Original dataset, without noise suppression. For the 1/16 inch error criterion, $\left|\hat{\Gamma}_0\right|$ = 3,015 and $\left|\hat{\mathbb{V}}_0\right|$ = 1,529. (b) De-noised data. For the same error criterion, $\left|\hat{\Gamma}_0\right|$ = 1,202 and $\left|\hat{\mathbb{V}}_0\right|$ = 638.

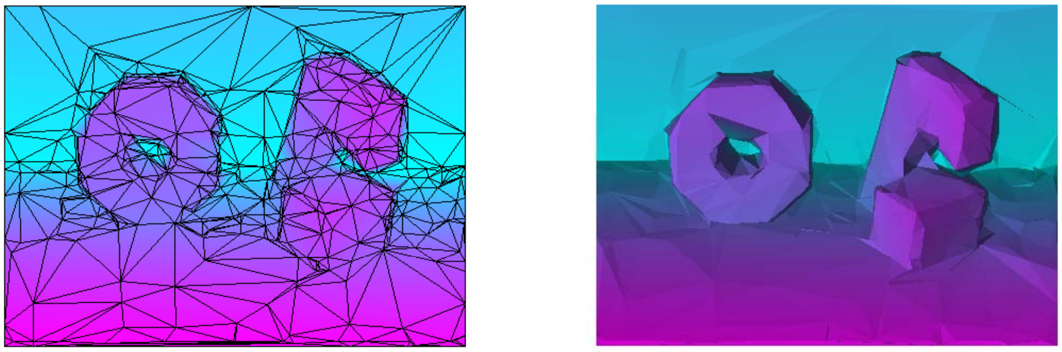
Figure 5.20 shows another example of noisy data. There exist two polyhedral objects in the scene: `object1` on the left side and `object2` on the right side. The actual objects have flat faces. However, because of noise in the measurement process, the object faces do not appear to be flat, and the corner points of the objects are corrupted with noise. Also, the background wall and floor are not smooth. The noise values range up to 3 cm. For this experiment, the input data is decomposed up to level 6 so that the initial triangulation procedure evaluates 8 × 8 coefficients for horizontal, vertical, and diagonal coefficients. A 2 cm error criterion is used, and $\delta_p = 0.9$, $\delta_v = 5$, $\alpha = 0.6$, $\delta_\lambda = 0.01$, $\delta_c = 0.96$ and $\delta_d = 0.05$ are used for other parameters.

The coarsest approximation at m = 5 displays the rough shape of the two objects. The outline of `object1` is represented with an octagon that closely follows the true outline of the object, but the hole in the middle of the object has not been recovered. In contrast, it is difficult to recognize `object2` at this level. At the next resolution level, the triangulation has improved a lot so that the hole in `object1` has appeared and the overall shape of the `object2` is quite recognizable. Tiny triangles have begun to develop around the discontinuities, and triangle vertices are placed in the vicinity of the critical points (corner points). The maximum error at this level is still high; $E_{\max}$ = 109.67 cm. At m = 3, the approximation is much improved, and the error has drastically reduced to 12.9 cm. Other noticeable improvements at this level are that the triangle edges are reasonably aligned with the object edges almost everywhere, that large triangles have replaced small ones for each face of the objects, and that tiny and narrow triangles are generated to approximate the discontinuities. The hole in `object1` has been much refined. As the resolution level decreases, vertices in the triangulations are placed at almost all corner points and some of unnecessary vertices are removed. Since the vertex locations are corrupted by noise, it is not easy for the algorithm to place triangle vertices at the right places. In addition, as the vertex removal and edge collapse operations do not remove all unnecessary triangles, there may still exist redundant triangles.

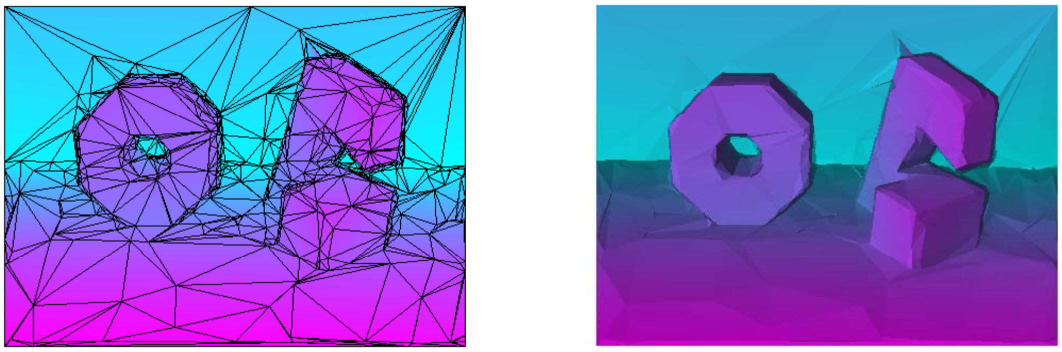
Meanwhile, the background wall and floor are covered with rather large triangles from the early stage of the approximation procedure. However, we can see the repeated operations of removals and splits on the floor as the approximation progresses. Again, the noise causes repeated operations. The next section will show experimental results for selective triangulation.



(a) $m = 5$

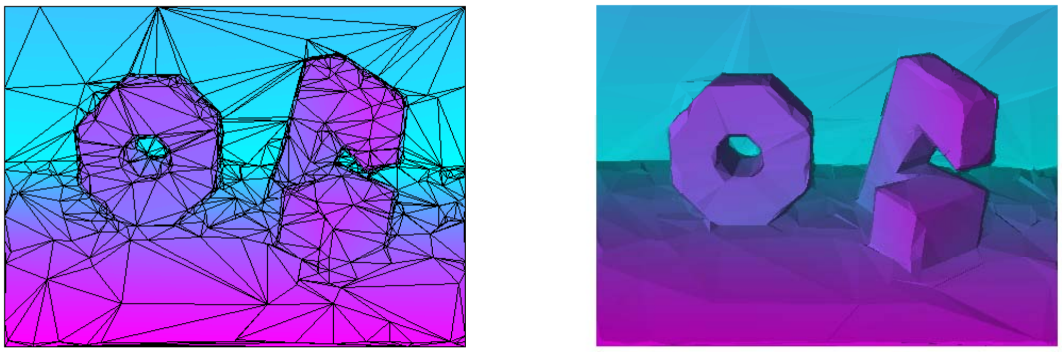


(b) $m = 4$

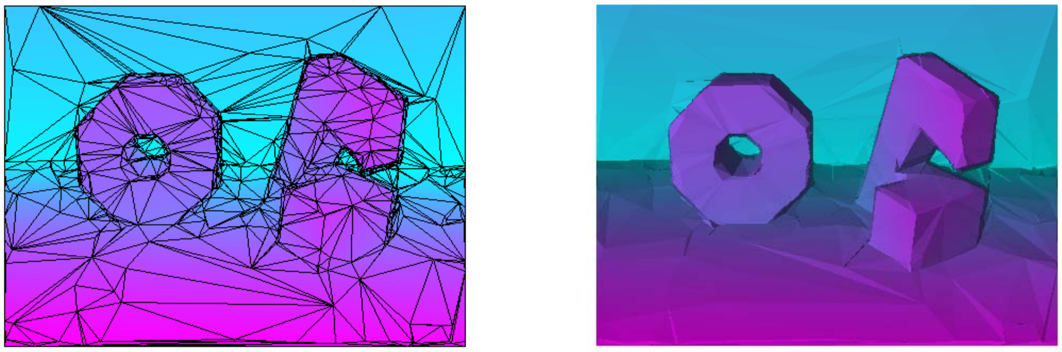


(c) $m = 3$

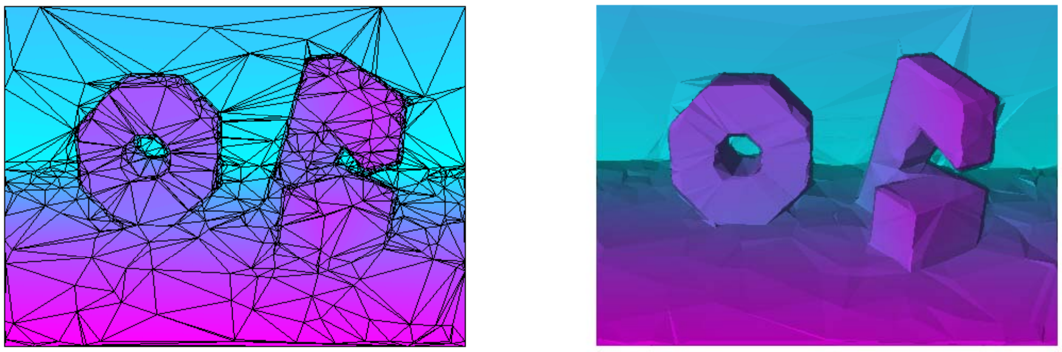
Figure 5.20 Multiresolution approximation of noisy dataset `Perc15`.



(d) $m = 2$



(e) $m = 1$



(f) $m = 0$

Figure 5.20 (continued).

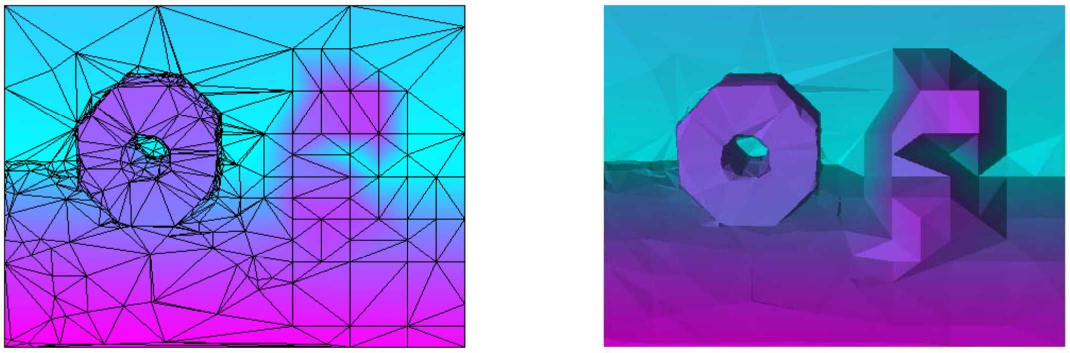
5.6 Selective Triangulation

One of the benefits of using the wavelet transform for surface approximation from height fields is easy control of level-of-detail (LOD). In section 5.4, we controlled the LOD by altering thresholds for wavelet detail energy. However, the LOD control by a threshold provides a global effect on the approximated results. In order to obtain a local LOD control, a so-called *wavelet space filter* can be applied [Gross et al. 96]. The wavelet space filter was designed to weight wavelet coefficients that correspond to the region of interest in the original dataset. A similar approach can be used for our approximation method. Instead of designing an elaborate wavelet space filter, however, we will use a *mask* that suppresses all wavelet coefficients corresponding to a region of interest. In order to apply the mask, a priori knowledge of object locations is required.

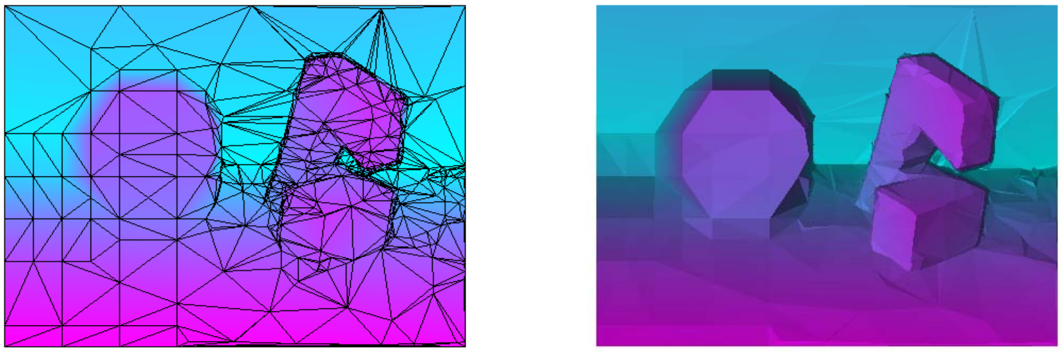
The most common approach for selective triangulation in computer graphics applications is to use a view dependent refinement, in which the entire scene of datasets is represented with different LODs depending on viewpoint and other parameters such as lighting [Xia and Varshney 96, Hoppe 97, Hoppe 98]. Although numerous researchers have worked on selective triangulation for terrain datasets and other graphical objects, there has been no such an effort to tessellate range images with different LODs. The selective triangulation of range datasets has several benefits in computer vision applications. For example, autonomous navigation systems that utilize 3D information may need to handle urgent matters such as the closest object with higher priority, and database query engines that search 3D shapes may want to continue matching only with similar objects at the coarsest level of resolution.

In order to fulfill those requirements, the proposed triangulation method can be used as an efficient selective triangulation scheme with an appropriate mask automatically constructed based on depth or shape information at the coarsest level. Based on this, a mask is constructed so that wavelet coefficients corresponding to the scene outside of a predefined depth are suppressed. For the database querying, shape estimation is performed at the coarsest resolution level and then a mask is constructed so that the unrelated region is suppressed.

For experimental purposes, we used the dataset `Perc15` in which two separate objects exist, as described in the previous section. First, a manually generated mask is applied on the right side of the dataset to suppress the wavelet coefficients corresponding to the `object2`. The approximation results are displayed in Figure 5.21(a). The `object1` is fully reconstructed at resolution level $m = 0$ while the `object2` remains in its primitive stage. The left side of the triangulation in Figure 5.21(a) is only slightly different from the one in Figure 5.20(f). This shows that the proposed triangulation method is strongly dependent on neighboring triangles. For the second experiment, the mask is applied on the left side of the input data, and all wavelet coefficients corresponding to the region are suppressed. As expected, the resulting approximation in Figure 5.21(b) shows a fully reconstructed `object2` on the right side and an undeveloped `object1` on the left side. Similar to the previous result, a slightly different triangulation is obtained for the `object2`. A small portion of `object1`, the rightmost side of the object, has been involved in the refinement process since it is identified as a *dependent region* that was defined in section 4.3.1.



(a)



(b)

Figure 5.21 Examples of selective triangulation. (a) Wavelet detail coefficients corresponding to `object2` are suppressed so that no refinement takes place in that region. (b) The region containing `object1` is suppressed.

5.7 Experimental Running Time

This section will show the actual running time of the proposed approximation algorithm. The time complexity analysis in section 5.1.1 has shown that the proposed algorithm runs in $O(14n) + O(3n \cdot h \log h)$ time cost at each resolution level, and that the overall time complexity is $\sum_{m=1}^{M-1} O(2^{M-m-1} 14n) + \sum_{m=1}^{M-1} O(2^{M-m-1} 3n \cdot h \log h)$, where n and M represent the number of triangles at an initial triangulation and the wavelet decomposition level, respectively. The analysis assumed the worst case in which each refinement process increases the number of triangles by a factor of 2. Since a constant *h* is assumed, the algorithm has linear time complexity for a given value of *M*.

The algorithm has been implemented using C++ and executed on a system with a 1 GHz Intel Pentium III processor and 256 MB of main memory. Although the implementation has not been optimized for speed, the actual running time is quite satisfactory for many applications. Figure 5.22 shows the running time information. The elapsed times are measured on the dataset `CraterLake` with 18 different sets of parameters. Each set of parameters produces different approximations with different numbers of triangles. The measurement includes times for only region selection, refinement, and mesh enhancement, and the input and output times are excluded. The algorithm takes less than 1 second to produce about 2,000 triangles (roughly corresponds to 1,000 vertices) and requires about 2.5 seconds for approximations with 4,000 triangles (roughly corresponds to 2,000 vertices). In this curve, 51 seconds elapse to create an approximation of 17,269 triangles. Our experiments on other datasets obtained similar curves for running time.

The curve for running time appears to be nonlinear, particularly beyond 8000 triangles. The main reason for this is the use of un-optimized code that inefficiently searches for neighborhood triangles and that re-triangulates remaining holes with $O(h^3)$ algorithm after vertex removals.

Using the same dataset, Garland and Heckbert have achieved 5 seconds running time for Delaunay greedy insertion of 5,000 data points [Garland and Heckbert 95] while our un-optimized code requires about 14 seconds for the creation of an approximation having the same number of vertices. Although direct comparison of the two algorithms is not possible, their approximation error measured in a root mean square (RMS) error of about 3 meters after 5,000 point insertions, but our approximation error (measured in terms of maximum normal distance) is less than 1 meter in an approximation having 4,132 vertices.

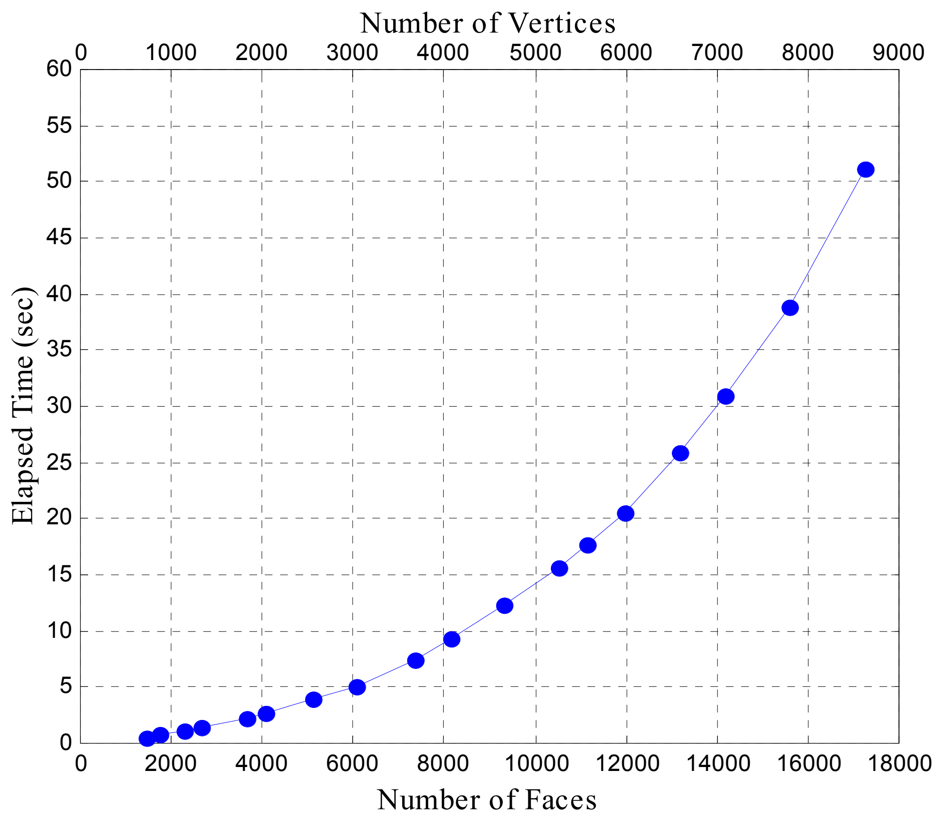


Figure 5.22 Running time information for surface approximation. The number of faces refers to the number of triangles at the finest resolution level for a given parameter set. On top of the graph, the number of vertices is shown for each approximation. The algorithm is executed with 18 different sets of parameters on the same dataset `CraterLake` so that the final approximations have different numbers of triangles. Only refinement and enhancement times are measured; input and output times are excluded.

Chapter 6

Conclusion and Future Direction

6.1 Summary and Conclusion

In this dissertation, a new fast algorithm for approximation of a dense height field using multiresolution analysis with wavelets has been developed. The main goal of the approximation scheme is to achieve a fast and reasonable approximation of a 3D surface with a small number of triangles.

The algorithm extracts 3D shape information by analyzing wavelet detail coefficients at different levels of resolution, and then constructs triangular meshes by using this shape information. A refinement approach is used for the triangulation. Accordingly, the proposed approximation method starts with an initial triangulation created by using a set of predefined templates, and then refines the initial triangulation by subdividing triangles corresponding to high wavelet detail energy. The templates are designed to reflect the underlying 3D shape of the input data, and are used for fast triangular tessellation of the 2D parametric domain yielding initial triangulations. The proposed refinement method again utilizes the shape information in order to perform shape-preserving splits for triangular meshes. While the refinement method increases mesh complexity by adding triangles into initial triangulation, the mesh enhancement procedure tries to reduce mesh sizes based on the analysis of geometrical shape of meshes. In addition, the refinement stage tends to produce pure data-dependent triangulation, but the mesh enhancement step regularizes the mesh shapes. The proposed approximation method iteratively applies the refinement and mesh enhancement as the wavelet decomposition level decreases. Instead of searching for optimal triangulation over a given local area, the iterative approach is more efficient in terms of execution time.

This dissertation has presented an analysis of time complexity and memory usage for the proposed approximation algorithm. The algorithm has been tested on height fields formed by several acquisition devices. The experimental results have shown that the approximation method is feasible for fast construction of triangular meshes so that the approximated surfaces represent the original datasets with desired accuracy. Also, it has been shown that the algorithm is useful for level-of-detail control. Currently the algorithm runs using un-optimized code.

The proposed algorithm produces piecewise planar surfaces from dense height fields. Therefore, it can be used in graphical model generation for applications in computer graphics, computer vision, and reverse engineering. Because the algorithm can produce approximations at different levels of resolution, it is useful for level-of-detail control, fast coarse-to-fine 3D object recognition, and analysis of surface characteristics at different resolution levels.

6.2 Future Directions

There are three natural extensions of this work. The first extension is to make the algorithm handle pure three-dimensional (3D) datasets in which data points on a surface of a 3D object are represented with 3D coordinates, not necessarily tied to a planar sampling grid. The proposed approximation method deals with only height fields that are represented in $2\frac{1}{2}$ dimensions. It is not designed to approximate pure 3D datasets. Therefore, in order to approximate 3D objects, several approximations can be constructed from several range images scanned at different viewpoints, and then be seamlessly sewed together [Turk and Levoy 94, Hausler and Karbacher 97, Pulli 99]. However, the seamless sewing method is not appropriate for objects with arbitrary topology. For this case, new wavelet analysis techniques need to be developed in order to decompose surface data on 3D objects. The three-dimensional wavelet decomposition [Mallat 98], which is a direct extension of 2D wavelets, could be a solution, but it would be more appropriate to design a special wavelet scheme in order to decompose data points on surfaces. For this purpose, it could be adequate to use second-generation wavelets with a lifting scheme that allows constructing a broader range of bases than the first generation wavelets. Details on the second-generation wavelets and their theory can be found in [Schroeder and Sweldens 95, Daubechies and Sweldens 96, Sweldens and Schröder 96, Sweldens 98].

As the second extension, the proposed approximation algorithm can be implemented with hardware. The algorithm largely consists of two parts: initial triangulation and mesh refinement. The initial triangulation is constructed from three sub-procedures including wavelet decomposition of height dataset, shape analysis based on detail coefficients, and triangular mesh construction by selecting appropriate templates. Templates are sets of pre-defined triangular meshes and easily stored with small amount of memory. Therefore, hardware implementation for the three sub-procedures is straightforward and can be done with FPGAs (field-programmable gate arrays) programmed to have a set of linear filters and a look-up table for templates. However, hardware implementation for the mesh refinement is rather challenging and may need high-performance DSP (digital signal processor) chips.

The final extension concerns the transmission of massive 3D model data through communication networks. Interest in this subject has been growing as the network infrastructure has developed and the demand for multimedia has increased. Transmission of complex 3D data to a remote terminal, especially linked with wireless networks, is slow. Thus, it is important to reduce the amount of data being transmitted. An attractive way for displaying a complex object over a network is to begin with a low-resolution version that can be quickly transmitted and rendered, and then be progressively improved by transmitting detail information. For the proposed algorithm, encoded wavelet coefficients can be transmitted fast over networks, and then the remote terminal can reconstruct height field data and perform approximation. Another way to achieve fast transmission is to send first the initial triangulation and then other information related to split and mesh enhancement.

Therefore, another future extension for this work would be to develop a hierarchical data structure to efficiently store the information related to split and mesh enhancement.

Bibliography

[Abbott and Ahuja 90] A. L. Abbott and N. Ahuja, "Active surface reconstruction by integrating focus, vergence, stereo, and camera calibration", *Proceedings: Third International Conference on Computer Vision*, December 1990, pp. 489-492.

[Aho et al. 83] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley Publishing Co., 1983.

[Ahuja and Abbott 93] N. Ahuja and A. L. Abbott, "Active Stereo: Integrating disparity, vergence, focus, aperture, and calibration for surface estimation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 15, no. 10, October 1993, pp. 1007-1029.

[Aksoy and Haralick 01] S. Aksoy, R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval", *Pattern Recognition Letters*, vol.22, no.5, May 2001, pp. 563-582.

[Algorri and Schmitt 96] M.-E. Algorri and F. Schmitt, "Mesh simplification", *Computer Graphics Forum*, 15(3), September 1996, pp. C77-C86.

[Arneodo et al. 94] A. Arneodo, E. Bacry and J.-F. Muzy, "Solving the inverse fractal problem from wavelet analysis", *Europhysics Letters*, vol. 25, no. 7, March 1994, pp. 478-484.

[Bajaj and Schikore 96] C. L. Bajaj and D. R. Schikore. "Error-bounded reduction of triangle meshes with multivariate data", *Proceedings of Visual Data Exploration and Analysis III*, *SPIE,* vol. 2656, 1996, pp. 34-45.

[Berg et al. 97] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[Berkmann and Caelli 94] J. Berkmann and T. Caelli, "Computation of surface geometry and segmentation using covariance techniques", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16 Issue 11 , November 1994 pp. 1114 –1116.

[Berman et al. 94] D. Berman, J. Bartell and D. Salesin, "Multiresolution painting and compositing", *Computer Graphics Annual Conference Series*, July 1994, pp. 85-90.

[Bern et al. 92] M. Bern, D. Dobkin and D. Eppstein, "Triangulating polygons without large angles", *Proceedings of the 8th annual ACM Symposium on Computational Geometry*, 1992, pp. 222-231.

[Bertram et al. 01] M. Bertram, M. A. Duchaineau, B. Hamann, and K. I. Joy, "Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization", *Proceedings of IEEE Visualization 2000*, 2000, pp. 389 -396, pp. 579.

[Besl and Jain 86] P. J. Besl and R. C. Jain, "Invariant surface characteristics for 3D object recognition in range images", *Computer Vision, Graphics, and Image Pro*cessing, vol. 33, 1986, pp. 33-80.

[Beylkin et al. 91] G. Beylkin, R. Coifman and V. Rokhlin, "Fast wavelet transforms and numerical algorithms", *Communications on Pure and Applied Mathematics*, vol. 44, 1991, pp. 141-183.

[Bolle and Vemuri 91] R. M. Bolle and B. C. Vemuri, "On three-dimensional surface reconstruction methods", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no 1, January, 1991, pp. 1-13.

[Bonneau 98] G. Bonneau, "Multiresolution analysis on irregular surface meshes", *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 4, October-December 1998, pp. 365-378.

[Bonneau and Gerussi 98] G. Bonneau and A. Gerussi, "Level of detail visualization of scalar data sets on irregular surface meshes", *Proceedings of IEEE Visualization '98*, 1998, pp. 73-77.

[Bonneau et al. 96] G. Bonneau, S. Hahmann, and G. M. Nielson. "BLaC-Wavelets: A multiresolution analysis with non-nested spaces" *Proceedings of IEEE Visualization '96*, October 1996, pp. 43-48.

[Burt and Adelson 83] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code", *IEEE Transactions on Communications*, vol. 31, no. 4, April 1983, pp. 532-540.

[Canny 86] J. Canny, "A computational approach to edge detection", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. PAMI-8, no. 6, November 1986, pp. 679-698.

[Carmo 76] M. P. do Carmo, *Differential geometry of curves and surfaces*, Prentice-Hall, 1976.

[Chen and Medioni 92] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images", *International Journal of Image and Vision Computing*, vol. 10, no. 3, April 1992, pp. 145-155.

[Chen and Schmitt 94] X. Chen and F. Schmitt, "Surface modeling of range data by constrained triangulation", *Computer Aided Design,* 26(8), 1994, pp. 632-645.

[Chen and Stockman 96] J. L. Chen and G. Stockman, "Determining pose of 3D objects with curved surfaces", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, 1996, pp. 57-62.

[Chetverikov and Khenokh 99] D. Chetverikov and Y. Khenokh, "Matching for shape defect detection", *Lecture Notes in Computer Science*, vol. 1689, Berlin, Springer, 1999, pp. 367-374.

[Choo et al. 99] K. Choo, I. D. Yun and S. U. Lee, "Edge-based approach to mesh simplification", *Proceedings: Second International Conference on 3-D Digital Imaging and Modeling*, 1999, pp. 368 -377

[Chouraqui and Elber 96] P. Chouraqui and G. Elber, "Physically based adaptive triangulation of freeform surfaces", *International Proceedings on Computer Graphics*, 1996, pp. 144-153.

[Chui 92] C. Chui, *An Introduction to Wavelets*, Academic Press, 1992.

[Ciampalini et al. 97] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno, "Multiresolution Decimation based on Global Error", *The Visual Computer,* Springer International*,* 13(5), 1997, pp.228-246.

[Cohen et al. 92] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets", *Communication on Pure and Applied Mathematics*, vol. 45, 1992, pp. 485-560.

[Cohen et al. 96] J. D. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, "Simplification Envelopes", *Proceedings of SIGGRAPH '96*, 1996, pp. 119-128.

[Cormen et al. 90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[Curless and M. Levoy 96] B. Curless and M. Levoy, "A volumetric method for building complex models from range images", *Proceedings of SIGGRAPH '96*, 1996, pp.301-312.

[Daubechies 88] I. Daubechies, "Orthonormal bases of compactly supported wavelets", *Communication on Pure and Applied Mathematics*, vol. 41, November 1988, pp. 909-996.

[Daubechies 92] I. Daubechies, *Ten lectures on wavelets*, SIAM, 1992.

[Daubechies and Sweldens 96] I. Daubechies and W. Sweldens. "Factoring wavelet transforms into lifting steps", Technical report, Bell Laboratories, Lucent Technologies, 1996.

[De Floriani et al. 84] L. De Floriani, B. Falcidieno, G. Nagy and C. Pienovi, "A hierarchical structure for surface approximation", *Computers and Graphics*, vol. 8, no. 2, 1984, pp. 183-193.

[De Floriani et al. 85] L. De Floriani B. Falcidieno and C. Pienovi, "Delaunay-based representation of surfaces defined over arbitrarily shaped domains", *Computer Vision, Graphics, and Image Processing*, vol. 32, 1985, pp. 127-140.

[DeRose et al. 94] T. D. DeRose, M. Lounsbery, and J. Warren, "Multiresolution analysis for surfaces or arbitrary topological type", SIGGRAPH '94 Course Notes, 1994.

[DeVore et al. 92] R. DeVore, B. Jawerth and B. Lucier, "Image compression through wavelet transform coding", *IEEE Transactions on Information Theory*, vol. 38, no. 2, March 1992, pp. 719-746.

[Donoho 95] D. L. Donoho, "Denoising by soft-thresholding", *IEEE Transactions on Information Theory*, vol. 41, 1995, pp. 613-627.

[Donoho and Johnstone 94] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, 1994, pp. 425-455.

[Dorum et al. 94] O. H. Dorum, A. Hoover and J. P. Jones, "Calibration and control for range imaging in mobile robot navigation", *Proceedings of Vision Interface '94*, Banff, Alberta, Canada, May 1994.

[Dreger et al. 98] A. Dreger, M. H. Bross, and J. Schlegel, "Multiresolution triangular B-spline surface", *International Proceedings on Computer Graphics*, 1998, pp. 166-177.

[Dyn et al. 90] N. Dyn, D. Levin and S. Rippa, “Data dependent triangulations for piecewise linear interpolation”, *IMA Journal of Numerical Analysis*, vol. 10, no. 1, January 1990, pp. 137-154.

[Eck et al. 95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, and M. Lounsbery, W. Stuetzle, “Multiresolution analysis of arbitrary meshes”, *Proceedings of SIGGRAPH '95*, Los Angeles, California, August 1995, pp. 173-182.

[Edelsbrunner 87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, 1987.

[Faugeras 93] Faugeras, O., *Three-Dimensional computer Vision, a geometric viewpoint*, The MIT Press, Cambridge, Massachussets, 1993.

[Faugeras et al. 83] O. D. Faugeras, M. Hebert, and E. Pauchon. “Segmentation of range data into planar and quadratic patches”, *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, June 1983, pp. 8-13.

[Finkelstein and Salesin 94] A. Finkelstein and D. Salesin, “Multiresolution curves”, *Computer Graphics Annual Conference Series*, July 1994, pp. 261-268.

[Friedrich et al. 98] A. Friedrich, K. Polthier, and M. Schmies, “Interpolation of triangle hierarchies”, *Proceedings of IEEE Visualization '98*, 1998, pp. 391-396, pp. 554.

[Garcia 95] M. A. Garcia, “Fast approximation of range images by triangular meshes generated through adaptive randomized sampling”, *Proceedings of IEEE International Conference on Robotics and Automation*, vol.2, 1995, pp. 2043-2048.

[Garcia et al. 97] M. A. Garcia, A. D. Sappa and L. Basanez, “Efficient approximation of range images through data-dependent adaptive triangulations”, *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 628 -633.

[Garland and Heckbert 95] M. Garland, P. Heckbert, “Fast Polygonal Approximation of Terrains and Height Fields”, Carnegie Mellon University, Technical report. Rep. CMU-CS-95-181.

[Garland and Heckbert 97] M. Garland and P. S. Heckbert, “Surface simplification using quadric error metrics”, *Proceedings of SIGGRAPH '97*, Los Angeles, California, August 1997, pp. 209-215.

[Garland and Heckbert 98] M. Garland and P. S. Heckbert, “Simplifying surfaces with color and texture using quadric error metrics”, *Proceedings of IEEE Visualization '98*. pp. 263-269, 542.

[Gieng et al. 97] T. S. Gieng, B. Hamann, K. I. Joy, G. L. Schussman and I. J. Trotts, “Smooth hierarchical surface triangulations”, *Proceedings of IEEE Visualization '97*, 1997, pp. 379 -386.

[Gortler and Cohen 95] S. J. Gortler and M. F. Cohen, “Hierarchical and variational geometric modeling with wavelets”, *Symposium on Interactive 3D Graphics*, 1995, pp. 35-42.

[Gray 98] A. Gray, *Modern Differential Geometry of Curves and Surfaces with Mathematica*, Boca Raton, CRC Press, 1998.

[Gross and Yellen 99] J. Gross and J. Yellen, *Graph Theory and Its Applications*, CRC Press, 1999.

[Gross et al. 96] M. H. Gross, O. G. Staadt, and R. Gatti, “Efficient triangular surface approximations using wavelets and quadtree data structures”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 2, June 1996, pp. 130-143.

[Grünbaum 67] B. Grünbaum, *Convex Polytopes*, Wiley-Interscience, New York, 1967.

[Guibas and Stolfi 85] L. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams”, *ACM Transactions on Graphics*, vol. 4 no. 2, 1985, pp.75-123.

[Guo 95] B. Guo, “A multiscale model for structure-based volume rendering”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, No. 4, December 1995, pp. 291-301.

[Guskov et al 99] I. Guskov, W. Sweldens, and P. Schroder, “Multiresolution signal processing for meshes”, *Proceedings of SIGGRAPH '99*, 1993, pp.35-44.

[Hamann 94] B. Hamann, “A data reduction scheme for triangulated surfaces”, *Computer Aided Geometric Design*, vol. 11, 1994, pp. 197-214.

[Häusler and Karbacher 97] G. Häusler and S. Karbacher, “Reconstruction of smoothed polyhedral surfaces from multiple range images”, *3D Image Analysis and Synthesis ’97*, In B. Girod, H. Niemann, and H.-P. Seidel (Eds.), Infix Verlag, Sankt Augustin, 1997, pp. 191-198.

[Heckbert and Garland 94] P. S. Heckbert and M. Garland, “Multiresolution modeling for fast rendering”, *Proceedings of Graphics Interface ’94*, May 1994, pp.43-50.

[Heckbert and Garland 97] P. S. Heckbert and M. Garland, “Survey of polygonal surface simplification algorithms” In “Multiresolution Surface Modeling”, *SIGGRAPH '97 Course Note*, 1997.

[Hongwei and Zhongkang 96] C. Hongwei and S. Zhongkang, “Application of wavelet packets theory in maneuver target tracking”, *Proceedings of the IEEE National Aerospace and Electronics Conference,* vol. 1, 1996, pp. 157 -162.

[Hoppe 96] H. Hoppe, “Progressive meshes”, *Proceedings of SIGGRAPH '96,* 1996, pp. 99-108.

[Hoppe 97] H. Hoppe**, “**View-dependent refinement of progressive meshes”, *Proceedings of SIGGRAPH '97, 1997*, pp. 189-198.

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

[Hoppe et al. 92] H. Hoppe, T. DeRose, and T. Duchamp, “Surface reconstruction from unorganized points”, *Proceedings of SIGGRAPH '92*, Chicago, July 1992, pp. 71-78.

[Hoppe et al. 93] H. Hoppe, T. DeRose, and T. Duchamp, “Mesh optimization”, *Proceedings of SIGGRAPH '93*, Anaheim, California, August 1993, pp. 19-26.

[Hu and Stockman 89] G. Hu and G. Stockman, “3-D surface solution using structured light and constraint propagation”, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 11, no. 4, April 1989, pp. 390-402.

[Hughes et al. 96] M. Hughes, A. Lastra, and E. Saxe, “Simplification of global-illumination meshes”, *Proceedings of Eurographics '96, Computer Graphics Foru,.*1996, pp. 339-345.

[Huttenlocher and Rucklidge 93] D. P. Huttenlocher and W. J. Rucklidge, “A multi-resolution technique for comparing images using the Hausdorff distance”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1993, pp. 705-706.

[Ji et al. 98] Q. Ji, M. S. Costa, R. M. Haralick and L. G. Shapiro, “An integrated technique for pose estimation from different geometric features”, *Proceedings of Vision Interface '98*, Vancouver, June 1998, pp. 77-84.

[Johnson and Hebert 98] A. E. Johnson and M. Hebert, “Efficient multiple model recognition in cluttered 3-D scenes”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 671-677.

[Johnstone and Silverman 97] I. M. Johnstone and B. W. Silverman, “Wavelet threshold estimators for data with correlated noise”, *Journal of the Royal Statistical Society*, Series B 59, 1997, pp. 319-351.

[Journet and Bazin 00] B. Journet and G. Bazin, “A low-cost laser range finder based on an FMCW-like method”, *IEEE Transactions on Instrumentation and Measurement*, vol. 49 no. 4, August 2000, pp. 840 -843.

[Kahn et al. 96] R. Kahn, M. Swain, P. Prokopowicz, and R. Firby, “Gesture recognition using the perseus architecture”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 734-741.

[Klein et al. 96] R. Klein, G. Liebich, and W. Straber, “Mesh reduction with error control”, *Proceedings of IEEE Visualization '96*, October 1996, pp. 311-318.

[Kobbelt et al. 98] L. Kobbelt, S. Campagna, and H. P. Seidel, “A general framework for mesh decimation”, *Proceedings on Graphics Interface ’98*, June 1998, pp. 43-50.

[Lee et al 00a] S. M. Lee, A. L. Abbott, and D. L. Schmoldt, “Using an embedded-processor camera for surface scanning of unplaned hardwood lumber”, *Review of Progress in Quantitative Nondestructive Evaluation*, In D. O. Thompson and D. E. Chimenti (Eds.) New York: Plenum Press, 2000, pp. 845-852.

[Lee et al. 00b] S. M. Lee, A. L. Abbott, and D. L. Schmoldt, “Wane detection on rough lumber using surface approximation”, *Proceedings of the 4th International Conference on Image Processing and Scanning of Wood*, In D. E. Kline and A. L. Abbott (Eds.), USDA Forest Service, Southern Research Station, Asheville NC, 2000, pp. 115-126.

[Lewis and Knowles 92] A. S. Lewis and G. Knowles, “Image compression using the 2-D wavelet transform”, *IEEE Transactions on Image Processing*, vol. 1, no. 2, April 1992, pp. 244-250.

[Liang and Todhunter 90] P. Liang and J. S. Todhunter, “Representation and recognition of surface shapes in range images”, *Computer Vision, Graphics and Image Processing*, vol. 52, no. 10, 1990, pp. 78-109.

[Linnainmaa et al. 88] S. Linnainmaa, D. Harwood and L. Davis, “Pose determination of a three-dimensional object using triangle pairs”, *IEEE Transactions on Pattern Analysis and machine Intelligence*, vol. 10, no. 5, September 1988, pp. 634-647.

[Liu et al. 94] Z. Liu, S. J. Gortler and M. F. Cohen, “Hierarchical space-time control”, *Computer Graphics Annual Conference Series*, July 1994, pp. 22-35.

[Lounsbery et al. 97] M. Lounsbery, T.D. DeRose, and J. Warren, “Multiresolution analysis for surfaces of arbitrary topological type”, *ACM Transactions on Graphics*, vol. 16, no. 1, January 1997, pp. 34–73.

[Madhuram et al. 97] C. R. Madhuram, S. S. Kim, R. Guha, G. Schiavone, and R. Mohapatra, “Multiresolution representation of non-uniformly sampled terrain databases using wavelets”, *Conference on Signals, Systems and Computers*, 1997, pp. 988-992.

[Maeda et al. 97] M. Maeda, K. Kumamaru, H. Zha, K. Inoue and S. Sawai, “3-D surface recovery from range images by using multiresolution wavelet transform”, *Proceedings of IEEE International Conference on Computational Cybernetics and Simulation*, vol. 4, 1997, pp. 3654-3659.

[Mallat 89] S. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation”, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 11, no. 7, July 1989, pp. 674-693.

[Mallat 96] S. Mallat, “Wavelets in Computer Vision”, *Proceedings* of the IEEE, vol. 84, no. 4, April 1996, pp. 604-614.

[Mallat 98] S. Mallat, *Wavelet Tour of Signal Processing*. Academic Press, 1998.

[Mallat and Hwang 92] S. Mallat and W. L. Hwang, “Singularity detection and processing with wavelets”, *IEEE Transactions on Information Theory*, vol. 38, no. 2, March 1992, pp. 617-643.

[Mallat and Zhong92] S. Mallat and S. Zhong, “Characterization of signals from multiscale edges”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14 no. 7, July 1992, pp. 710 –732.

[Miller and Goldman 95] J. R. Miller and R.N. Goldman, “Geometric algorithms for detecting and calculating all conic sections in the intersection of two natural quadratic surfaces”, *Graphical Models and Image Processing*, vol. 57, no. 1, 1995, pp. 55-66.

[Morris and Kanade 00] D. D. Morris and T. Kanade, “Image-consistent surface triangulation”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, pp. 332 –338.

[Nielson and Hamann 91] G. M. Nielson and B. Hamann, “The asymptotic decider: removing the ambiguity in marching cubes”, *Proceedings of IEEE Visualization '91*, 1991, pp. 83-91.

[Nielson et al. 97] 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 IEEE Visualization '97*, 1997, pp. 143-149, pp. 536.

[O’Neill 66] B. O’Neill, *Elementary Differential Geometry*, Academic Press, 1966.

[Olson 97] C. Olson, “Mobile robot self-localization by iconic matching of range maps”, *Proceedings of International Conference on Advanced Robotics*, 1997, pp. 447-452.

[Pajarola 98] R. Pajarola, “Large scale terrain visualization using the restricted quadtree triangulation”, *Proceedings of IEEE Visualization '98*, 1998, pp. 19 -26, 515.

[Park et al. 99] I. K. Park, I. D. Yun, and S. U. Lee, “Constructing NURBS surface model from scattered and unorganized range data”, *Proceedings of Second International Conference on 3-D Digital Imaging and Modeling*, 1999, pp. 312-320.

[Paster and Rodriguez 99] L. Paster and A. Rodriguez, “Surface approximation of 3D objects from irregularly sampled clouds of 3-D points using spherical wavelets”, *Proceedings of International Conference on Image Analysis and Processing*, 1999, pp. 70-75.

[Paulik and Wang 98] M. J. Paulik and Y. D. Wang, “Three-dimensional object recognition using vector wavelets”, *Proceedings of IEEE International Conference on Image Processing*, 1998, pp. 586 -590.

[Pentland 92] A. P. Pentland, “Fast solutions to physical equilibrium and interpolation problems”, *Visual Computer*, vol. 8, no. 5, June 1992, pp. 303-314.

[Perceptron 93] Perceptron Inc. *LASAR Harware Manual*, 1993.

[Pottier et al. 99] E. Pottier, D. L. Schuler, J. S. Lee and T. L. Ainsworth, “Estimation of the terrain surface azimuthal/range slopes using polarimetric decomposition of POLSAR data”, *Proceedings of International Geoscience and Remote Sensing Symposium*, vol.4, 1999, pp. 2212 -2214.

[Prenter 75] P. M. Prenter, *Splines and Variational Methods*, John Wiley & Sons, New York, 1975.

[Preparata and Shamos 85] F. P. Preparata and M. I. Shamos, *Computational Geometry, An Introduction*, Springer-Verlag, New York, 1985.

[Pulli 99] K. Pulli, "Multiview registration for large data sets", *Proceedings of Second International Conference on 3D Digital Imaging and Modeling*, Ottawa, Canada, October 1999, pp. 160-168.

[Pulli et al. 98] K. Pulli, H. Abi-Rached, T. Duchamp, L. G. Shapiro and W. Stuetzle, "Acquisition and visualization of colored 3D objects", *Proceedings of International Conference on Pattern Recognition*, vol. 1, 1998, pp. 11 -15.

[Puzicha et al. 97] J. Puzicha, T. Hofmann and J. M. Buhmann, "Non-parametric similarity measures for unsupervised texture segmentation and image retrieval", *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, San Juan, 1997, pp. 267-272.

[Ramamootrhi and Arvo 99] R. Ramamootrhi and J. Arvo, "Creating generative models from range images", *Proceedings of SIGGRAPH '99*, Los Angeles, California, August 1999, pp. 195-204.

[Rioul and Vetterli 91] O. Rioul and M. Vetterli, "Wavelets and signal processing", *IEEE Signal Processing Magazine*, vol. 8, no. 4, October 1991, pp. 14-38.

[Ronfard and Rossignac 96] R. Ronfard and J. Rossignac, "Full-range approximation of triangulated polyhedra", *Computer Graphics Forum*, vol. 15 no. 3, September 1996, pp. C67-C76, pp. C462.

[Rossignac and Borrel 93] J. Rossignac and P. Borrel, "Multiresolution 3D approximations for rendering complex scenes", *Modeling in Computer Graphics: Methods and Applications*, 1993, pp. 455-465.

[Rubner and Tomasi 98] Y. Rubner and C. Tomasi, "Texture metrics", *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 1998, pp. 4601-4607.

[Rucklidge 96] W.J. Rucklidge, "Efficient visual recognition using the Hausdorff distance", Number 1173 Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1996.

[Sacchi et al. 99] R. Sacchi, J. F. Poliakoff; P. D. Thomas and K.-H. Hafele, “Curvature estimation for segmentation of triangulated surfaces”, *Proceedings of Second International Conference on 3-D Digital Imaging and Modeling*, 1999, pp. 536 -543.

[Scarlatos and Pavlidis 92] L. Scarlatos and T. Pavlidis, “Hierarchical triangulation using cartographic coherence”, *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 2, March 1992, pp. 147-161.

[Schroeder 96] P. Schroeder, “Wavelets in computer graphics”, *Proceedings* of the IEEE, vol. 84, no. 4, April 1996, pp. 615-625.

[Schroeder and Sweldens 95] P. Schroeder and W. Sweldens, “Spherical wavelets: efficiently representing functions on the sphere”, *Proceedings of SIGGRAPH '95*, 1995, pp. 161-172.

[Schroeder et al. 92] W. J. Schroeder, J. A. Zarge, W. E. Lorensen, “Decimation of triangle meshes”, *Proceedings of SIGGRAPH '92*, Chicago, July 1992, pp. 65-70.

[Seibold and Joy 99] W. Seibold and K. I. Joy, “Near-optimal adaptive polygonization”, *Proceedings of Computer Graphics International '99*, 1999, pp. 206-212.

[Seibold and Wyill 98] W. Seibold and G. Wyill, “Towards an understanding of surfaces through polygonization”, *Proceedings of Computer Graphics International '98*, 1998, pp. 416-425.

[Shapiro 93] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, *IEEE Transactions on Signal Processing*, vol. 41, no. 12, December 1993, pp. 3445-3462.

[Silva et al. 95] C. T. Silva, j. S. B. Mitchell and A. E. Kaufman, “Automatic generation of triangular irregular networks using greedy cuts”, *Proceedings of IEEE Visualization '95*, 1995, pp. 201-208, pp. 453.

[Sim and Dudek 99] R. Sim and G. Dudek, “Learning and evaluating visual features for pose estimation”, *International Conference on Computer Vision '99*, 1999, pp. 1217-1222.

[Soucy and Laurendeau 96] M. Soucy and D. Laurendeau, “Multiresolution surface modeling based on hierarchical triangulation*”, Computer Vision and Image Understanding*, vol. 63, no. 1, January 1996, pp. 1-14.

[Staadt et al. 97] O. G. Staadt, M. H. Gross and R. Weber, “Multiresolution compression and reconstruction”, *Proceedings of IEEE Visualization '97*, 1997, pp. 337-346.

[Starck et al. 98] Starck, J.-L., F. Murtagh and A. Bijaoul, *Image Processing and Data Analysis. The Multiscale Approach*. Cambridge University Press, 1998.

[Stark and Bowyer 96] Stark, L. and K. Bowyer, *Generic Object Recognition Using Form and Function*, World Scientific, Singapore, 1996.

[Stein 81] C. Stein, “Estimation of the mean of a multivariate normal distribution”, *The Annals of Statistics*, vol. 9, 1981, pp. 1135-1151.

[Stollnitz et al. 95a] E. J. Stollnitz, T.D. DeRose, and D.H. Salesin, “Wavelets for computer graphics: A Primer, Part 1”, *IEEE Computer Graphics and Applications*, vol. 15, no.3, 1995, pp. 76-84.

[Stollnitz et al. 95b] E. J. Stollnitz, T.D. DeRose, and D.H. Salesin, “Wavelets for computer graphics: A Primer, Part 2”, *IEEE Computer Graphics and Applications*, vol. 15, no.4, 1995, pp. 75-85.

[Stollnitz et al. 96] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, *Wavelets for Computer Graphics: Theory and Applications*, Morgan Kaufmann, San Francisco, 1996.

[Sweldens 98] W. Sweldens. “The lifting scheme: A construction of second generation wavelets.” *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, March 1998, pp. 511-546.

[Sweldens and Schröder 96] W. Sweldens and P. Schröder. “Building your own wavelets at home.” In “Wavelets in Computer Graphics”, SIGGRAPH '96 Course Notes, 1996.

[Szeliski 90] R. Szeliski, “Fast surface interpolation using hierarchical basis functions”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, June 1990, pp. 513 -528.

[Takahashi et al. 97] S. Takahashi, Y. Shinagawa, and T. Kunii, “Curve and surface design using multiresolution constraints”, *Proceedings of Computer Graphics International '97*, 1997, pp. 121-130, 249.

[Thomas and Finney 95] G. B. Thomas and R. Finney, *Calculus and Analytic Geometry***,** Addison Wesley Longman, Inc. 1995.

[Thorpe 79] J. A. Thorpe, *Elementary Topics in Differential Geometry*, New York, Springer-Verlag, 1979.

[TI 00] Texas Instruments. *TMS320C6000 CPU and Instruction Set Reference Guide*, TI Inc., 2000.

[Tieng and Boles 97] Q. M. Tieng and W. W. Boles, “Recognition of 2D object contours using the wavelet transform zero-crossing representation”**,** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 8, August 1997, pp. 910-916.

[Turk and Levoy 94] G. Turk and M. Levoy, “Zippered polygon meshes form range images”, *Proceedings of SIGGRAPH '94*, 1994, pp. 311-318.

[Unser et al. 93] M. Unser, A. Aldroubi, and M. Eden, “A family of polynomial spline wavelet transforms”, *Signal Processing*, vol. 30, 1993, pp. 141-162.

[USF DB] http://marathon.csee.usf.edu/range/DataBase.html.

[USGS DEM] ftp://edcftp.cr.usgs.gov/pub/.

[Valette et al. 99] S. Valette, Y. Kim, H. Jung, I. Magnin, and R. Prost, “A multiresolution wavelet scheme for irregularly subdivided 3D triangular mesh”, *Precedings of International Conference on Image Processing*, vol. 1, 1999, pp. 171-174.

[Vetterli and Kovacevic 95] Vetterli, M. and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, New Jersey, 1995.

[Williams 83] L. Williams, “Pyramidal parametrics”, *Proceedings of SIGGRAPH '83*, vol. 17, no. 3, July 1983, pp. 1-11.

[Xia and Varshney 96] J. Xia and A. Varshney, “Dynamic view-dependent simplification for polygonal models”, *Proceedings of IEEE Visualization '96*, 1996, pp. 327-334.

[Yaou and Chang 94] M. Yaou and W. Chang, “Fast surface interpolation using multiresolution wavelet transform”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 7, July 1994, pp. 673-688.

[Yemez and Schmitt 99] Y. Yemez and F. Schmitt, “Progressive multilevel meshes from octree particles”, *Proceedings of Second International Conference on 3-D Digital Imaging and Modeling*, 1999, pp. 290-299.

[Yu and Ra 99] H. J. Yu and J. B. Ra, “Fast triangular mesh approximation of surface data using wavelet coefficients”, *The Visual Computer*, vol. 15, Springer-Verlag, 1999, pp. 9-20.

Appendix I.

Half-Edge Data Structure[†]

A common way to represent a polygon mesh is a shared list of vertices and a list of faces storing pointers for its vertices. Although this representation is both convenient and efficient for many purposes, it is ineffective in some applications that require a frequent search for neighborhood information. As an example, our surface approximation uses edge split, vertex removal, edge swap, and edge collapse operations in which require adjacency relationships between components of the mesh, such as the faces and the vertices. The edge split operation requires a search for which faces are adjacent to a particular face, while the vertex removal operation needs to know which faces share a particular vertex of interest. Also, the edge swap operation searches faces that border a particular edge to be swapped, and the edge collapse operation needs to know which vertices are connected to a particular edge to be collapsed. While it is possible to implement these operations on the simple mesh representation mentioned above, they would most likely be costly; the operations will require a search through the entire list of faces or vertices, or possibly even both.

To order to obtain a fast access to neighborhood information for a given mesh component, more sophisticated boundary representations have been developed which explicitly model the vertices, edges, and faces of the mesh with additional adjacency information stored inside. One of the most common of these types of representations is the half-edge data structure

[†] A full implementation of the half-edge data structure can be found at Harvard Graphics Archive Mesh Library (http://www.cs.deas.harvard.edu/~xgu/mesh). This appendix is based on a tutorial written by Max McGuire at http://www.flipcode.com.

where two half-edges are stored to represent an edge. As the name implies, a half-edge is a half (one side) of an edge. Thus, two half-edges make up an edge and they are cross-referred through the edge they are representing. The diagram in Figure A.1 shows a half-edge representation of a triangle. A triangle is represented with three vertices (gray dots) and edges (orange bars). The arrows in the diagram represent pointers. The half-edges that border a triangle form a circular linked list around its perimeter and are drawn in blue bars. This list can either be oriented clockwise or counter-clockwise around the triangle just as long as the same convention is used throughout. Each of the half-edges in the loop stores a pointer to the face it borders (not shown in the diagram), the vertex at its end point and a pointer to an edge that it makes up. Having pointers to two half-edges, each edge acts as a bridge that links two opposite half-edges, by which two faces are connected each other. Each face contains only a point to a half-edge, and each vertex stores location as well as a pointer to one of half-edges emanating from the vertex.

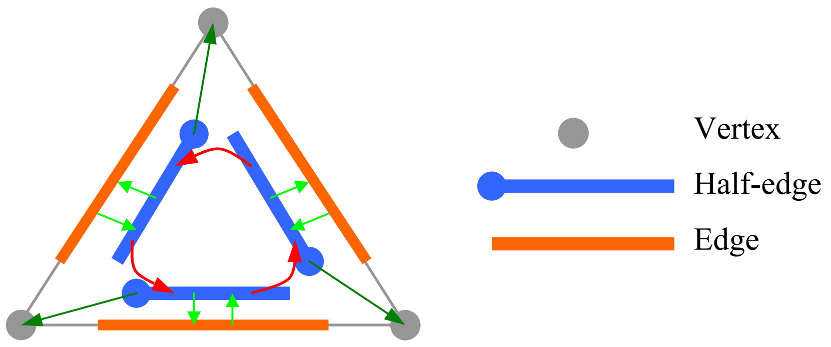


Figure A.1 A half-edge representation of a triangle.

Figure A.2 shows a half-edge representation of a small portion of triangular mesh. Here, the pointers to vertices in `HalfEdge` class are omitted. At any given vertex, there will be more than one half-edge that starts from it, and these half-edges are easily reached via pointers to edges and next half-edges.

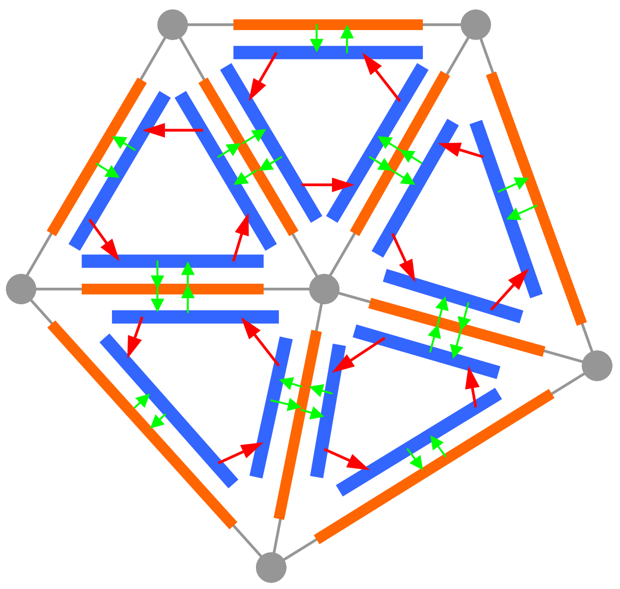


Figure A.2 A half-edge representation of a triangular mesh.

Thus, the C++ implementation of the half-edge data structure looks like this:

```
class Vertex {

private:
   float       x, y, z;
   HalfEdge    *he;

};
```

```
class HalfEdge {

private:
   Face      *face;
   Edge      *edge;
   Vertex    *vert;

   HalfEdge *next;
};
```

```
class Edge {

private:
   HalfEdge *he1;
   HalfEdge *he2;
};
```

```
class Face{

private:
   HalfEdge *he;
};
```

Vita

Sang-Mook Lee was raised in Seoul, Korea, and graduated from KangSeo High School in 1986. He earned a B.S. degree and a M.S. degree in Electronic Engineering from Inha University, Inchon, Korea in 1990 and 1992, respectively. From March 1992 to July 1997, he was with Agency for Defense Development (ADD), TaeJeon, Korea, where he was involved in research and development of Fire Control System. He has then pursued a Ph.D. degree in Electrical and Computer Engineering at Virginia Polytechnic Institute and State University from the fall semester of 1997. His research interests are in Computer Vision, Image Processing, Digital Signal Processing, and Polygonal Approximation of Range Images.