AN ADAPTIVE METHOD
FOR TERRAIN SURFACE APPROXIMATION
BASED ON TRIANGULAR MESHES

By

Hélio Pedrini

An Abstract of a Thesis Submitted to the Graduate Faculty of Rensselaer Polytechnic Institute in Partial Fulfillment of the Requirements for the Degree of DOCTOR OF PHILOSOPHY

Major Subject: Computer and Systems Engineering

The original of the complete thesis is on file in the Rensselaer Polytechnic Institute Library

Examiner Committee:

William Randolph Franklin, Thesis Adviser
Badrinath Roysam, Member
Michael Wozny, Member
Thomas Yu, Member

Rensselaer Polytechnic Institute
Troy, New York
May 2000
© Copyright 2000

by

Hélio Pedrini

All Rights Reserved
CONTENTS

LIST OF TABLES vi
LIST OF FIGURES vii
ACKNOWLEDGMENTS xi
ABSTRACT xii

1 INTRODUCTION 1
  1.1 Motivation .................................................. 1
  1.2 Research Goals ............................................ 4
  1.3 Thesis Organization ......................................... 6

2 BACKGROUND AND RELATED WORK 8
  2.1 Digital Terrain Models ..................................... 8
    2.1.1 Tessellation Models ................................. 10
      2.1.1.1 Regular Tessellations .......................... 10
      2.1.1.2 Nested Tessellations ........................... 13
      2.1.1.3 Irregular Tessellations ......................... 17
    2.1.2 Contour Line Models ................................ 18
    2.1.3 Mathematical Models ................................. 18
  2.2 Surface Representation ................................... 20
    2.2.1 Subdivision Surfaces ................................ 21
    2.2.2 Multiresolution Models .............................. 22
      2.2.2.1 Image Pyramids ................................ 23
      2.2.2.2 Volume Pyramids ................................ 24
      2.2.2.3 Wavelets ........................................ 25
      2.2.2.4 Hierarchical Subdivision Models ............... 25
  2.2.3 Polygonal Surface Simplification ..................... 30
    2.2.3.1 Refinement Methods .............................. 31
    2.2.3.2 Decimation Methods .............................. 33

3 ADAPTIVE TERRAIN SURFACE APPROXIMATION 39
  3.1 C₀ Continuous Triangular Patches ....................... 42
    3.1.1 Data Structures .................................... 52
    3.1.2 Mesh Compression .................................... 54
      3.1.2.1 Geometry Compression .......................... 55
      3.1.2.2 Topology Compression ........................... 55
    3.1.3 Geometric Primitives ................................. 58
  3.2 C¹ Continuous Triangular Patches ....................... 61
3.2.1 Barycentric Coordinates ............................................. 61
3.2.2 Bernstein Polynomials .............................................. 62
3.2.3 Triangular Bézier Patches .......................................... 63
3.2.4 Continuity Conditions ............................................. 63
3.2.5 Derivative Estimation ............................................... 66
3.2.6 Quadratic Interpolant ............................................... 67
3.2.7 Cubic Interpolant ................................................... 71

4 ACCURACY ASSESSMENT ................................................. 74
  4.1 USGS Digital Elevation Models .................................... 74
  4.2 Accuracy of USGS DEMs ............................................ 76
  4.3 Accuracy Measures .................................................. 78
    4.3.1 Quantitative Measures ......................................... 78
      4.3.1.1 Root Mean Square Error (RMSE) ......................... 78
      4.3.1.2 Surface Curvature ......................................... 79
      4.3.1.3 Spatial Autocorrelation .................................. 81
      4.3.1.4 Texture Descriptors ...................................... 82
      4.3.1.5 Volume ........................................................ 84
      4.3.1.6 Slope and Aspect ........................................... 84
      4.3.1.7 Drainage Networks ......................................... 86
      4.3.1.8 Visibility .................................................... 89
      4.3.1.9 Hausdorff Distance ........................................ 90
    4.3.2 Qualitative Measures ......................................... 92
      4.3.2.1 Elevation Map ............................................... 92
      4.3.2.2 Contour Map ............................................... 92
      4.3.2.3 Shaded Relief Map ........................................ 93
      4.3.2.4 Slope and Aspect Map ...................................... 97

5 RESULTS ........................................................................ 99
  5.1 Terrain Data Sets ..................................................... 99
  5.2 Accuracy .............................................................. 102
  5.3 Computational Time .................................................. 125
  5.4 Memory Requirements ............................................... 125
  5.5 Implementation Issues .............................................. 126

6 CONCLUSIONS ............................................................. 129
  6.1 Summary and Contributions ........................................ 129
  6.2 Future Directions ..................................................... 130

REFERENCES .................................................................. 132

A TRIANGULATION ALGORITHMS ..................................... 158
  A.1 Delaunay Triangulation ............................................. 158
    A.1.1 Constrained Delaunay Triangulation ....................... 161
A.1.2 Conforming Delaunay Triangulation . . . . . . . . . . . . . . . . . . . . . 163
A.2 Delaunay Triangulation Algorithms . . . . . . . . . . . . . . . . . . . . . 164
   A.2.1 Divide-and-Conquer Algorithms . . . . . . . . . . . . . . . . . . . . . 164
   A.2.2 Growth Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
   A.2.3 Incremental Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . 168
   A.2.4 Sweepline Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
A.3 Data-Dependent Triangulation . . . . . . . . . . . . . . . . . . . . . . . . 173
A.4 Optimization Criteria for Generating Meshes . . . . . . . . . . . . . . . 174

B DATA STRUCTURES FOR TINs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
   B.1 Point-Based Data Structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
   B.2 Edge-Based Data Structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178
   B.3 Triangle-Based Data Structures . . . . . . . . . . . . . . . . . . . . . . . . . . 180
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Surface types defined by the signs of the mean and Gaussian curvatures.</td>
<td>80</td>
</tr>
<tr>
<td>5.1</td>
<td>Summary of results for sample of 24 random USGS DEMs.</td>
<td>103</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of results for Lake Champlain.</td>
<td>104</td>
</tr>
<tr>
<td>5.3</td>
<td>Summary of results for Crater Lake.</td>
<td>105</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary of results for the Martian terrain.</td>
<td>105</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison between maximum vertical error and new vertex selection criterion.</td>
<td>106</td>
</tr>
<tr>
<td>5.6</td>
<td>Statistics of the differences in visibility indices.</td>
<td>116</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary of pairwise comparison of visibility indices.</td>
<td>117</td>
</tr>
<tr>
<td>5.8</td>
<td>Mesh quality measures for sample of 24 random USGS DEMs.</td>
<td>118</td>
</tr>
<tr>
<td>5.9</td>
<td>Viewshed determination at five locations of the Lake Champlain DEM.</td>
<td>123</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 Overall schematic view of the method. ........................................... 4
1.2 Thesis organization ................................................................. 7

2.1 A classification of digital terrain models. ........................................ 10
2.2 Regular tessellations ............................................................... 11
2.3 The USGS Glens Falls West DEM ................................................ 12
2.4 (a) A region $R$; (b) binary array; (c) block decomposition of the region;
    (d) quadtree representation of the blocks. .................................. 14
2.5 (a) The Morton code addressing scheme for labeling pixels; (b) the
    Morton code addresses for the blocks of Figure 2.4(c). ............... 15
2.6 A triangulated irregular network (TIN). ...................................... 17
2.7 Contour line model ................................................................. 19
2.8 Neighborhood of a given vertex (black dot). (a) manifold; (b) mani-
    fold with boundary; (c) non-manifold vertex neighborhood; (d) non-
    manifold edge neighborhood. ................................................. 20
2.9 Loop subdivision scheme. (a) initial mesh; (b) new mesh created by
    the splitting step (new vertices are shown as black dots); (c) mesh
    created by the averaging step. .................................................. 22
2.10 Pyramid representation .......................................................... 24
2.11 Discontinuities along an edge in a quadtree-based surface model. ... 26
2.12 A quaternary triangulation ...................................................... 26
2.13 A ternary triangulation ........................................................... 27
2.14 Subdivision strategies for preserving cartographic coherence. ........ 28
2.15 A hierarchical Delaunay triangulation ........................................ 28
2.16 (a) A Delaunay sequence and (b) its corresponding Delaunay pyramid. 29
2.17 (a) $3 \times 3$ filter; (b) elevation interpolation .................................. 32
2.18 Vertex decimation ................................................................. 34
2.19 Edge decimation ................................................................. 34
2.20 Triangle decimation .............................................................. 34

3.1 Example of a TIN representing terrain surfaces. ........................... 40
3.2 (a) Invalid triangulation; (b) a surface discontinuity exists if $z$-
    coordinate of the point $C$ is not on line $BD$. .......................... 41
3.3 The Douglas-Peucker algorithm for line simplification ..................... 43
3.4 Redundant triangulation generated by incremental triangulation algo-
    rithm, Figure adapted from Garland and Heckbert (1995). ............. 44
3.5 Alternate refinement and decimation steps .................................. 45
3.6 Maximum vertical error .......................................................... 46
3.7 Criterion for vertex removal .................................................... 47
3.8 An invalid retriangulation ........................................ 48
3.9 Approximation of an object (using 100 vertices) constructed by (a) our combined refinement/decimation method and (b) conventional incremental insertion algorithm ........................................ 49
3.10 Pseudocode for refinement step .................................. 49
3.11 Pseudocode for decimation step .................................. 50
3.12 (a) Breakline represented by segment AC is not a Delaunay edge because the circumcircle of triangle ABC contains point D; (b) new point F is added on the constrained edge to satisfy the Delaunay criterion 50
3.13 Topographic features in triangular meshes ................ 52
3.14 Local adjust of resolution at an arbitrary location .......... 52
3.15 Quad-edge structure representing three edges incident on one vertex .................................. 53
3.16 A triangle strip .................................................. 57
3.17 Area of a triangle .................................................. 59
3.18 The orientation test ................................................ 59
3.19 The InCircle test .................................................. 60
3.20 Incircle of a triangle .............................................. 60
3.21 Barycentric coordinates .......................................... 62
3.22 (a) Quadratic and (b) cubic triangular Bernstein-Bézier patches and their corresponding Bernstein polynomials .................................. 64
3.23 Two adjacent triangular patches .................................. 65
3.24 Continuity conditions for triangular Bézier patches .......... 66
3.25 Powell-Sabin subdivision ........................................... 68
3.26 Clough-Tocher subdivision ........................................ 71
4.1 Surface curvature .................................................. 79
4.2 Calculation of entropy in a 3×3 window ......................... 83
4.3 Drainage network. (a) eight nearest neighbors; (b) DEM; (c) flow direction; (d) flow path .................................. 86
4.4 Drainage networks derived from the Crater Lake DEM ........ 88
4.5 (a) Single flow direction; (b) multiple flow direction .......... 88
4.6 Visibility in a terrain profile ....................................... 90
4.7 Elevation map for Crater Lake ..................................... 93
4.8 Contour map for Crater Lake ....................................... 94
4.9 (a) Geometry of reflection; (b) spherical coordinate system .... 95
4.10 Shaded relief map for Lake Champlain ......................... 96
4.11 Slope map for Lake Champlain .................................... 97
4.12 Aspect map for Lake Champlain ................................... 98
5.1 Sample of 24 random USGS DEMs ............................... 100
5.2 The USGS Lake Champlain West DEM .......................... 101
5.3 The USGS Crater Lake West DEM ................................ 101
5.4 A Martian terrain of 964×960 sample points .................. 102
5.5 Approximations of Crater Lake DEM constructed by using (a) our combined refinement/decimation method (230 triangles) and (b) incremental insertion algorithm (263 triangles) ............................................ 107
5.6 Approximations of the Lake Champlain DEM ............................................. 108
5.7 Approximations of the Crater Lake DEM ................................................... 109
5.8 Approximations of the Martian terrain ....................................................... 110
5.9 Approximation error for three terrain data sets ......................................... 111
5.10 (a) Three-dimensional plot of the elevation error for Lake Champlain (using 5% of the input points); (b) error profile at 26th row; (c) error profile at 150th column ........................................... 112
5.11 (a) 1-D cross section of original and approximate surfaces of the Lake Champlain at 26th row; (b) 1-D cross section of original and approximate surfaces at 150th column ........................................... 113
5.12 Distribution of the differences between two sets of visibility indices .......... 114
5.13 Visibility indices for Lake Champlain DEM. (a) original; (b) approximation using 16% of the original points .......................................................... 115
5.14 Search area bounded by a circle ............................................................... 116
5.15 Approximation error for Lake Champlain DEM using each interpolation method .......................................................... 119
5.16 Differences in elevation for Lake Champlain DEM (using 1000 points). (a) linear; (b) quadratic; (c) cubic .................................................. 120
5.17 Differences in elevation for Crater Lake DEM (using 1000 points). (a) linear; (b) quadratic; (c) cubic .................................................. 121
5.18 Approximations of the Crater Lake DEM (using 1000 points). (a) linear; (b) quadratic; (c) cubic .................................................. 122
5.19 Viewsheds computed at point $x = 412$ and $y = 268$ for Lake Champlain DEM. (a) original; (b) linear; (c) quadratic; (d) cubic. White pixels represent the areas that are visible from the viewpoint ............... 124
5.20 Running time ............................................................................................. 126
5.21 Original Lake Champlain DEM ................................................................. 127
5.22 Crater Lake DEM (10,000 points) .............................................................. 128
5.23 Martian Terrain (100,000 points) ............................................................... 128

A.1 The circle property. (a) Triangle $T_1$ is not a Delaunay triangle since a data point $P$ lies inside its circumcircle; (b) triangle $T_2$ is a Delaunay triangle .......................................................... 159
A.2 The max-min angle property. The diagonal $e_A$ in (a) is replaced with diagonal $e_B$, as in (b), in order to increase the minimum of the six internal angles ........................................................................ 159
A.3 The Voronoi diagram of a data point set (dashed lines) and the corresponding Delaunay triangulation (solid lines) .............................................................. 161
A.4 (a) A set of constraint segments; (b) the corresponding visibility graph; (c) a constrained triangulation .......................................................... 162
| A.5 | An illustration of a triangle in a constrained Delaunay triangulation that satisfies the constrained version of the circle property. | 162 |
| A.6 | A conforming Delaunay triangulation. | 163 |
| A.7 | Illustration of the upper and lower common tangents of two convex hulls. | 165 |
| A.8 | Merge step of two triangulations in the divide-and-conquer algorithm. The dashed edges have to be removed. | 166 |
| A.9 | Selection of a Delaunay neighbor. | 167 |
| A.10 | Radial sweep algorithm. (a) Radiating lines from the central point; (b) the radiating lines are connected; (c) the convex hull is covered by triangles; (d) final result of the triangulation. | 169 |
| A.11 | Illustration of the steps of Watson's algorithm. | 170 |
| A.12 | Insertion of a point $P$ into the triangulation. Dashed lines indicate edges that need to be checked by the algorithm. | 171 |
| A.13 | (a) Original Voronoi diagram; (b) transformed Voronoi diagram. | 172 |
| B.1 | Point-based data structure. | 178 |
| B.2 | The dual-edge data structure. | 178 |
| B.3 | (a) A triangulation and (b) its corresponding dual-edge data structure. | 179 |
| B.4 | The winged-edge data structure. | 179 |
| B.5 | The quad-edge data structure. | 180 |
| B.6 | (a) A subdivision and (b) its corresponding quad-edge data structure. | 181 |
| B.7 | (a) A subdivision and (b) its corresponding triangle-based data structure. | 181 |
| B.8 | Triangle-based data structure. | 182 |
ACKNOWLEDGEMENTS

I would like to thank my thesis adviser, Dr. William Randolph Franklin, for his constant encouragement, support, and valuable guidance through this work. I am also very grateful to Dr. Badrinath Roysam, Dr. Michael Wozny, and Dr. Thomas Yu for providing helpful suggestions and comments. My special thanks to Dr. George Nagy for sharing his knowledge and time.

Appreciation is also extended to my friends Alessandro, Aluíso, Manoel, Marcelo, and all members of Rensselaer DocLab, who made this journey more enjoyable.

I would like to dedicate this thesis to my parents for their support, endurance, and inspiration throughout the years. I am also deeply grateful to all the members of my family for their encouragement and support.

Partial support of this work was provided by a grant from Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, Brazil.
ABSTRACT

The development of methods for storing, manipulating, and rendering large volumes of data efficiently is a crucial task in terrain modeling. As an alternative representation to regular grid digital elevation models (DEM), we present an improved method for adaptively approximating terrain surfaces based on triangulated irregular networks (TIN). Alternate refinement and decimation steps are applied to the triangular model, incrementally determining a better distribution of the data points, while a specified error tolerance is preserved. This technique provides an effective compromise between fidelity and time requirements, producing high quality approximations with great flexibility.

The method uses a Delaunay triangulation to maintain the topology of the data points, whose vertices lie at a subset of the input data. A new local error metric is used to select points from the original terrain data set, based on the maximum vertical error weighted by the standard deviation calculated in a neighborhood of the candidate point. Conversely, a measure of angle between surface normals is used to determine whether a vertex should be removed from the triangulation.

Automatic procedures for construction of smooth terrain surfaces defined as a network of curved triangular patches are also investigated. The method has been implemented and tested on a number of real terrain data sets.
CHAPTER 1
INTRODUCTION

1.1 Motivation

The substantial development of computer-based technology during the last decades has contributed to the expansion of numerous scientific applications. The increasing power of computer systems, associated with efficient geometric and modeling algorithms, are now capable of producing models with high degree of detail. On the other hand, the availability and complexity of these models are growing rapidly due to the improved capabilities for collecting and distributing data, and due to the need for higher accuracy.

Examples of applications involving large volumes of data and complex domain knowledge include remote sensing, satellite imaging, planetary exploration, computer vision, computer-aided design, and medical image analysis. In digital terrain modeling, our target application, the growing availability of terrain data at higher and higher resolutions, and the integration of spatial data from multiple sources, such as maps, aerial photographs, remotely sensed data, and video images of maps create a need for more efficient storage strategies.

Careful design of models and algorithms is necessary in order to make tasks more computationally tractable. Maintaining redundant information or using trivial analysis procedures may demand computational requirements far from of current or anticipated hardware capabilities. One of the most critical research problems encountered in the analysis and visualization of large volumes of data is the development of methods for storing, manipulating, and rendering massive data sets efficiently. Unless data reduction or compression methods are used, extremely large data sets cannot be analyzed or visualized in real time.
A common method for approximating topographic surfaces uses a regular grid digital elevation model (DEM), in which a set of sampled points representing measures of altitude or elevation are stored at regular intervals. Different interpretations can be considered to calculate grid points from sampled elevation values. If the distance between elevation values is coincident with the grid size, then no interpolation is necessary. Otherwise, an interpolation or convolution method can be used to resample the grid. Bilinear interpolation, for instance, calculates a new value for a grid point from the four elevation points surrounding it, whereas a cubic convolution uses sixteen neighbor points and a nonlinear weight for the distance.

Alternatively, triangulated irregular networks (TINs) represent the terrain surface as a mesh of adjacent triangles, whose vertices are the elevation points. The points need not lie in any particular pattern and the density may vary over space. There are many advantages associated with TINs. First, terrain data are commonly not regularly distributed in space, therefore, the structure of the mesh can be adjusted to reflect the density of the data. Consequently, cells become larger where data are sparse, and smaller where data are dense. Second, terrain features can be incorporated into the model. For instance, vertices in a TIN can describe nodal terrain features such as peaks, pits or passes, while edges can represent linear terrain features such as break, ridge or channel lines. Third, TINs can be organized into a hierarchical model so that they can represent a terrain in various levels of detail. Finally, triangles are simple geometric objects which can be easily manipulated and rendered.

Besides their application in terrain modeling, polygonal surfaces are frequently used to represent three-dimensional data sets in several other scientific areas, due mainly to their simplicity and flexibility. They are supported by the vast majority of modeling and rendering packages, such as OpenGL [226], VRML [16], VTK [281], and Data Explorer [167], and polygonal surface data are widely available. Hard-
ware support for polygon rendering is also becoming more popular. In the last few years, several polygonal surface simplification algorithms have been proposed in the literature (see Section 2.2.3 for more details) to generate a surface containing fewer polygons. This is important for processing, visualizing, or transmitting larger surface data sets than the available capabilities of software, computers, and networks permit.

On the basis of the considerations mentioned above, we construct a terrain model based on irregular triangular meshes which preserves topographic features and provides an approximation of the terrain surface with accuracy within a specified error tolerance. A hybrid technique using repeated refinement and coarsening\(^1\) passes is used to adaptively approximate the terrain surface, incrementally determining a better distribution of the data points in the mesh. Even though not asymptotically optimal or monotonically convergent, it produces approximations that are, experimentally, significantly better than those generated by straightforward incremental triangulation algorithms. The method is also suitable to support multiresolution surface representation, that is, the terrain can be approximated at different levels of accuracy and detail. This is particularly relevant to reduce space requirements and the complexity of manipulating large amounts of data.

Although this piecewise linear approximation is simple in concept and generates compact, accurate surfaces, the generalization to a piecewise smooth representation is a natural and, in many cases, a necessary extension. Certain regions of interest may consist of smoothly curved areas that meet along sharp curves. Modeling such regions as piecewise linear surfaces ($C^0$-continuity surfaces) requires a large number of triangles, whereas a curved surface can provide a more accurate and compact model of the true surface. Smooth surfaces can also produce superior results for rendering purposes, reducing certain perceptual problems such as the appearance of Mach Bands along element boundaries. In order to investigate a possible improvement

\(^1\)The term decimation is also used in the literature.
over linear interpolation, we describe a method for constructing smooth surfaces on the irregular triangular mesh. The approximating surface is represented by a $C^1$ piecewise triangular Bézier surface. High-order elements are generally more expensive to evaluate, therefore, they should be used only where the error is high and the extra effort is justified. Figure 1.1 shows a schematic representation of our method.

![Diagram](https://example.com/diagram.png)

**Figure 1.1:** Overall schematic view of the method.

### 1.2 Research Goals

The main purpose of our research work is to produce an approximate terrain representation using a smaller subset of points selected from the original elevation
data, while preserving the most relevant topographic features.

Although our primary focus is on generating high-quality approximations, it is also desirable to create compact models in order to reduce data necessary for storage and transmission requirements, and improve rendering performance.

The main criteria that have influenced the design of our approximation method include:

- **adaptability**: the algorithm should be easily controlled to meet a specified level of accuracy. In addition, important topographic information contained in the original data should be retained by the generated surfaces;

- **compactivity**: the method should significantly reduce the storage requirements with no or little loss in accuracy;

- **accuracy**: the terrain model should maintain a high degree of fidelity to the original data. The accuracy of the model should be estimated on the basis of quantitative and qualitative measures;

- **versatility**: the approach should be flexible enough to accommodate a number of applications without requiring significant changes;

- **efficiency**: performance of the algorithm (time efficiency) associated with compactivity of the model (storage efficiency) should be compatible with the application requirements;

- **simplicity**: the method should require simple data structures and a small number of rules;

- **locality**: the region over which a point influences the shape of the surface should be small;
• **continuity:** the terrain representation should produce continuous and visually pleasing surfaces. The locations where possibly no derivative exists (e.g. breaklines) should be either explicitly given or extracted from the original data.

The relative importance of each of these factors is a function of the overall requirements of a given application. Some of these criteria can be quantitatively measured, such as speed and space efficiency for a particular model. However, it is very difficult to provide quantitative measures for abstract factors of versatility or simplicity.

The general methodology of our approach includes the formal definition of terrain models, the development of algorithms for rapidly and accurately approximate topographic surfaces from large sets of data points, the formulation of quantitative and qualitative measures to evaluate the fidelity of our approximations, and the application of the method to a number of real terrain data sets.

### 1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 provides an overview of the main methods for modeling digital terrain data. The relevant characteristics, advantages, and disadvantages of each representation are described.

Chapter 3 presents our adaptive method for terrain surface approximation based on triangular meshes. Initially, the construction of planar TIN models is considered, then the method is generalized to curved surface approximations. A number of techniques for mesh compression are also described.

Chapter 4 analyzes several quantitative and qualitative criteria for evaluating the accuracy of our models. Examples of some measures include surface curvature, root mean square error, slope, visibility information, and drainage networks.

In Chapter 5, the proposed method is applied to several real terrain data sets. Implementation issues and experimental results are presented and discussed.

Chapter 6 concludes with some final remarks and directions for future research.
The main contributions of this work are also summarized.

Finally, two appendices are included to review technical considerations mentioned only briefly in the chapters.

Figure 1.2 illustrates the main topics focused on this thesis and a diagram of its framework.

Figure 1.2: Thesis organization.
CHAPTER 2
BACKGROUND AND RELATED WORK

2.1 Digital Terrain Models

The objective of terrain modeling is to provide efficient, flexible, and powerful tools to represent and manipulate topographic surfaces. The history of terrain modeling predates the development of geographic information systems\(^1\) (GIS) and developed from efforts of researchers working in disciplines such as cartography, photogrammetry, surveying, geology, geography, hydrology, and civil engineering. The heterogeneous and independent nature of these application fields resulted in the development of a large number of data models.

A data model can be defined as an abstraction of real world objects, which incorporates only the levels of detail necessary to meet the purposes of a given application. This abstraction of reality should be preferably independent of implementation conventions or computing environment restrictions. A data structure, on the other hand, is a representation of a data model in computer form.

Spatial data generally refers to any data distributed in multidimensional space. More specifically, geographic data are spatial data which are associated with a location relative to the earth’s surface. Geographic data have traditionally been represented by means of maps. The basic units used for representing the spatial location of geographic data are points, lines, polygons, and volumes.

The modeling of geographic data is generally very complex due to a number of factors. For instance, natural phenomena tend to occur in irregular, complex patterns. Spatial entities tend to be inexact and context dependent. In addition, the relationships among spatial entities are typically very numerous.

\(^1\)Geographic information systems originated in the 1960s as the result of refinements in cartographic techniques, advances in computer systems, and quantitative development in spatial analysis.
The design of general-purpose data models is extremely difficult, particularly when dealing with complex phenomena. The more perfectly a model represents reality, the more robust and flexible the model will be in the application. However, the more precisely the model fits a single application, the more efficient it will tend to be in terms of storage requirements and ease of use.

It is common to distinguish two major categories of spatial data models available in GIS. The first model, known as vector model, allows the representation of geographic space in a more intuitive and reminiscent abstraction of the traditional analog map. This model represents the spatial locations of items explicitly. The second model, known as raster model, divides space as a series of units, each of which represents a limited, but defined, amount of the earth’s surface. Although absolute location is not explicitly part of the raster data model, it is implied by the relative locations of the space units. Raster data from different sources may use different pixel sizes, orientations, positions, and projections. The traditional advantages and disadvantages between vector and raster models have been extensively documented in the literature [75, 244, 298]. Many authors [143, 221, 243] have also suggested that a GIS should be able to accept, store, retrieve, manipulate, and display both vector and raster models (known as hybrid models), as well as convert from one model to the other.

Despite the increased capacity of digital storage systems, the availability of terrain data at higher and higher resolutions requires a need for more efficient storage strategies. Moreover, the integration of data from multiple sources demands the development of more flexible spatial data models designed specifically to store terrain data.

The term digital terrain model (DTM) is largely attributed to Miller and LaFlamme [216], who describe its use for a variety of terrain analysis problems applied to civil engineering. DTM has been the generic term to refer to digital
representations of topographic surfaces. Burrough [31] argues that the term digital elevation model (DEM) is more suitable for models which contain only elevation data, since terrain often implies other types of attributes of a landscape.

Several different strategies for representing terrain data are available in the literature. The most relevant digital terrain models are described in the following sections. The objective is to provide a insight into how spatial data handling techniques may be combined in a more systematic and efficient manner for practical applications.

A classification of these models is introduced in Figure 2.1.

Figure 2.1: A classification of digital terrain models.

2.1.1 Tessellation Models

2.1.1.1 Regular Tessellations

A simple model for representing terrains is to use regular cells to describe a finite set of sampled points in the surface domain. The only three possible regular tessellations (mosaic or polygonal mesh) which can form the basis of planar decompositions are rectangular, triangular, or hexagonal [4]. These tessellations are shown in Figure 2.2.

The rectangular (or square) tessellations have historically been the most widely
used since it is natural to store the data in a two-dimensional array of grid cells. A characteristic of interest in regular hexagonal tessellations is that all neighboring cells are equidistant, which makes this model advantageous for radial search and retrieval functions. The critical difference among rectangular, triangular, and hexagonal tessellations in the plane is that only the rectangular grid cells can be recursively subdivided into smaller cells of the same shape and orientation as the original cells. Triangles can be subdivided into other triangles, but the triangles do not all have the same orientation. Hexagons cannot be subdivided into other smaller hexagons. Ahuja [4] describes these basic geometric differences in detail.

**Regular Grid Digital Elevation Model:** The most common form of DTM is probably the regular grid digital elevation model (DEM), in which a set of sampled points representing measures of altitude or elevation are stored at regular intervals. These elevation values can be stored as elements in a two-dimensional array, such that the regular grid spacing of points allows the search for a point to be implied directly from its coordinates. Since the coordinates of the origin and the grid spacing are known, the $x$ and $y$ coordinates of each point need not be stored in the data structure.

Digital elevation data have been produced by many government agencies, such as the United States Geological Survey (USGS) or the Australian Surveying and Land Information Group (AUSLIG). The USGS, for instance, provides public access to digital elevation data for the entire United States. These data have been predominantly produced by photogrammetric and cartographic processes. Five different digital ele-
vation products are currently distributed by the USGS in the standard DEM format. Although all are similar in the manner the data are structured, each product varies in sampling interval, geographic reference system, areas of coverage, and accuracy. For instance, 1:250,000-scale topographic maps [315] are the primary source of 1-degree DEM (3- by 3-arc-second\(^2\) data spacing). Larger scale maps, such as 1:100,000 and 1:24,000, are used to generate higher accuracy DEM.

Figure 2.3 shows the 1-degree USGS Glens Falls West DEM.

![Figure 2.3: The USGS Glens Falls West DEM.](image)

The DEM is easy to manipulate, process, and display for a variety of applications. While digital elevation models have traditionally been developed to store only one attribute for each grid cell, different types of attributes can be handled by creating data layers or overlays, each of which represents one different thematic content (e.g., soil, hydrology, topography, land use). However, the analysis of relationships among the themes is limited, since each grid cell location must be addressed individually.

\(^2\)arc second: 1/3600th of a degree (1 second) of latitude or longitude.
A disadvantage of the DEM is its inherent spatial invariability, since the structure is not adaptive to the irregularity of the terrain. This may produce a large amount of data redundancy, especially where the topographic information is minimal. Compression methods can provide significant enhancements in storage requirements and, therefore, they have been frequently integrated into spatial data models. The basic principle of data compression methods is to determine the minimal data required to retain the necessary information. This is achieved by taking advantage of the redundancy that is inherent in image data. Data compression methods are generally classified into two groups. Lossless compression methods preserve the information in such a way that the original data can be recovered exactly from the compressed data. On the other hand, lossy compression methods allow a certain loss of information, typically providing high compression ratios.

Several techniques have been developed for data compression, such as Huffman encoding [164], run-length coding [154], Lempel-Ziv-Welch (LZW) coding [327, 332], arithmetic coding [131], predictive techniques [262], vector quantization [133], fractal compression [14, 106], transform-based methods (cosine, Fourier, Karhunen-Loève, Hadamard, Walsh, wavelets). There is an increasing effort to establish standards in data compression, such as JPEG [173] and JBIG [172].

2.1.1.2 Nested Tessellations

a) Quadtrees: There are several advantages of using a regular, recursive tessellation of the plane as a spatial data model. In particular, the quadtree data model, first proposed by Klinger [179], has been the focus of much research over the past years [209, 270, 271, 272]. Although quadtrees are commonly used to represent the decomposition of a region, quadtree variants for point [103, 232] and line [274, 290] data have also been proposed for use in cartography.

The region quadtree representation is a hierarchical structure based on the regular, recursive decomposition of a square region (grid) into quadrants and subquadrants,
until homogeneous blocks (possibly pixels) are obtained. To illustrate this concept, consider the region shown in Figure 2.4(a), which is represented by a $2^3 \times 2^3$ binary array in Figure 2.4(b). The blocks resulting from the decomposition of the region quadtree are shown in Figure 2.4(c), and the corresponding quadtree in Figure 2.4(d). The root node corresponds to the entire region, the four children of the root node correspond to the quadrants, and the terminal nodes correspond to those blocks of the array for which no further subdivision is necessary. The quadrants are labeled in order NW, NE, SW, and SE.

Figure 2.4: (a) A region $R$; (b) binary array; (c) block decomposition of the region; (d) quadtree representation of the blocks.

Several data structures for quadtrees have been proposed. One class of structures explicitly represents the relationships among all internal (non-leaf) and terminal (leaf) nodes of the quadtree by using pointers [165, 180, 263]. In this structure, each non-leaf node has five pointers associated with it, that is, four to its children and one to its parent. Although the resulting quadtree provides a fast access from one place
in the tree to another by following a chain of pointers, it requires a large amount of storage. The linear quadtree, proposed by Gargantini [124], is a version that requires no pointers, where only the leaf nodes form a linear list which implicitly encodes the original data. Many variations on this model exist, for example associating a color or an address value with each node [2, 182, 208, 270]. The addressing schemes commonly used are variations on one suggested by Morton [223], known as Morton addressing. Morton addresses are created by interleaving the bits of the binary representation of the pixel coordinates. For instance, Figure 2.5(a) shows a $2^3 \times 2^3$ array with each pixel labeled by a 3-bit binary representation. Each pixel's address is formed by bit interleaving such that the $y$ bit precedes the $x$ bit at each position. The NW quadrant is encoded with 0, the NW with 1, the SW with 2, and the SE with 3. Each leaf node is then encoded with digits 0, 1, 2, and 3 in base 4, where each successive digit represents the quadrant from which it originates. The Figure 2.5(b) shows the Morton code addresses for the blocks of Figure 2.4(c).

![Figure 2.5](image)

Figure 2.5: (a) The Morton code addressing scheme for labeling pixels; (b) the Morton code addresses for the blocks of Figure 2.4(c).

When the pixel's addresses are sorted in increasing order, the result is equivalent to a depth-first traversal. Given the address of any pixel and a list of leaf nodes
ordered by their addresses, finding the leaf containing that pixel reduces to searching a sorted list. Since files may contain a large number of leaf nodes, a B-tree structure, for instance, may be used to store large sorted lists and provide efficient insertion, deletion, and search algorithms.

Quadtrees have numerous advantages for GIS applications because they can operate at multiple levels of resolution, allow a good compactness of storage, and are specially good for overlay operations. Examples of GIS applications based on quadtrees can be found in [273, 285].

b) K-d Trees: The multidimensional binary search tree or k-d tree, proposed by Bentley [17], offers an efficient data structure in storage requirements. In two dimensions, it divides the area into two parts instead of four at each point, resulting a tree of degree 2. The first point becomes the root of the tree, each point falling into the left half-plane is inserted into the left subtree, and each point falling into the right half-plane is inserted into the right subtree. For points that divide regions horizontally, the points in the upper half-plane are inserted into the left subtree, whereas points in the lower half-plane are inserted into the right subtree. Therefore, the direction of this subdivision is rotated among the coordinates for successive levels of the tree. Assuming that the root is at level 0, nodes at even levels in the tree divide the set of points into left and right half-regions, and nodes at odd levels divide a region into upper and lower half-regions.

In general, k-d trees are superior to point quadtrees. However, since point quadtrees are inherently parallel data structures [270], comparison operations can be performed in parallel using such representations, whereas this cannot be done for the k-d tree.
2.1.1.3 Irregular Tessellations

The most important advantage of an irregular tessellation is that the structure of the mesh can be adjusted to reflect the density of the data. This offers a variable resolution model in the sense that the size and density of its elements may vary over space. As a result, cells become larger where data are sparse, and smaller where data are dense. Since the structure of topographic surfaces is commonly not regular, irregular tessellations offer a suitable model for digital terrains. On the other hand, the topology of the tessellation needs to be maintained.

**Triangulated Irregular Network:** The irregular tessellation most frequently used as a spatial model is probably the *triangulated irregular network* (TIN). It consists of a number of irregularly spaced points connected by a set of edges to form a planar triangulation. Each vertex of the triangulation has an elevation value. If a point does not lie on a vertex, and edge or in the triangle, its elevation may be obtained by linear interpolation. Therefore, a TIN is a piecewise linear model that can be seen as a connected set of contiguous nonoverlapping triangles. Figure 2.6 illustrates a TIN and its equivalent representation in three-dimensional space.

![Figure 2.6: A triangulated irregular network (TIN).](image)

The TIN model for representing terrain data has been used since the 1970s [114, 128, 240, 242]. The first implementation of a TIN in GIS was probably developed by
Franklin [114]. A newer version of that program, Franklin [115], can process large terrain data sets.

A common problem with irregular triangular networks is that several different triangulations can be generated from the same set of points. In consequence, many triangulation algorithms have been used for surface modeling. Most TIN methods are based on procedures for selecting a set of points to best approximate the terrain surface, according to a number of established criteria. More detailed information on methods and data structures used to construct triangular networks is described in Appendices A and B.

2.1.2 Contour Line Models

Contours are a set of imaginary lines which connect points of equal elevation. These lines are known as contour lines or isolines. The distance between lines, known as contour interval, is based on the difference in elevation. Closely spaced contour lines represent steep slopes, whereas widely spaced contour lines represent slow changes in elevation.

Contours are usually stored as an ordered sequence of points along each contour line. Figure 2.7 shows the contour line model generated for the digital elevation model presented in Figure 2.3.

2.1.3 Mathematical Models

Topographic surfaces can also be represented by numerous analytical [113] and numerical methods. Mathematical models for surface representation are commonly based on interpolation and approximation techniques. Interpolation techniques aim to represent the surface passing exactly through all elevation values (or control points), whereas the surface described by approximation techniques passes as close as possible to all elevation values, introducing some errors into the model. Since topographic surfaces are commonly complex, exact interpolation is not practicable for digital
terrain modeling.

Mathematical models can be characterized as *global* or *local*. Global methods use a single function to represent the surfaces through the data elevations. These methods are usually inappropriate for terrain modeling since they tend to generate oscillations between data points. In addition, although global methods can be computationally efficient in terms of storage requirements, they require highly complex functions to represent surface of even moderate complexity. On the other hand, local methods divide the model into local domains and define separate functions for each of these *surface patches*. Each set of local functions can be relatively simple and still result a satisfactory representation.

There are several mathematical approaches that can be applied to terrain modeling, including the use of splines, double Fourier series, polynomials, fractals, and wavelets.
2.2 Surface Representation

In order to characterize the domain of our data sets, some useful concepts from topology are described. A manifold is a connected region where each point on the surface has an infinitesimal neighborhood topologically equivalent to an open disk\(^3\) (in two dimensions) or an open ball\(^4\) (in three dimensions). A manifold with boundary is a region where the neighborhood of all points is topologically equivalent to either a disk or a half-disk (in two-dimensions).

Examples of manifold and non-manifold geometry are shown in Figure 2.8.

![Figure 2.8: Neighborhood of a given vertex (black dot). (a) manifold; (b) manifold with boundary; (c) non-manifold vertex neighborhood; (d) non-manifold edge neighborhood.]

A manifold is orientable if it has two distinguishable sides, that is, if there is a clear distinction between the inside and the outside of the surface. The torus and the sphere are examples of orientable manifolds. The Möbius strip and the Klein bottle are examples of non-orientable manifolds.

A polygonal surface is a piecewise-linear surface defined by a set of polygons, typically a set of triangles. Polygonal surfaces are probably the most widely used representation in practical applications, since they are flexible and supported by the majority of modeling and rendering packages. Hardware support for polygon rendering is also becoming more available.

---

\(^3\)An open disk is the portion of a plane which is enclosed within some circle, but not including the circle.

\(^4\)An open ball is the portion of three-dimensional space which is enclosed within some sphere, but not including the sphere.
Piecewise-polynomial models are a common alternative to polygonal models. By using higher-order polynomials, smooth surfaces can be approximated more accurately than with planar polygons. A model composed of piecewise-polynomial elements discretizes the model into a fixed set of patches. For certain applications, however, a fixed set of surface elements may not be appropriate. More adaptive methods have been developed, such as subdivision surfaces [37, 82, 201], hierarchical splines [109, 110], and models composed of triangular Bézier patches [97, 98].

2.2.1 Subdivision Surfaces

In 1974, Chaikin [40] introduced a method for generating a smooth curve from a control polygon by recursively cutting off the corners of the polygon. This is perhaps the first method of constructing smooth curves of arbitrary topological type.

Catmull and Clark [37] and Doo and Sabin [82] generalized the idea to surfaces. In such schemes, a subdivision surface is defined by repeatedly refining an initial control mesh $M^0$ to produce a sequence of meshes $M^1, M^2, ...$ that converge to a limit surface.

By using systematic subdivision rules, it is possible to show that the limit surface presents tangent plane continuity. The Catmull-Clark scheme generalizes bicubic uniform B-spline surfaces, and the Doo and Sabin [82] scheme generalizes biquadratic uniform B-spline surfaces.

The Loop scheme [201] is probably the simplest subdivision method for triangular meshes. Each edge of the mesh is split into two, and new vertices are reconnected to form four new triangles. Vertices are rearranged by using an averaging step. Figure 2.9 shows this subdivision procedure.

This scheme is based on the three-directional quartic box spline, which produces $C^2$-continuous surfaces in the regular meshes.

The Butterfly scheme, proposed by Dyn et al. [86], is a subdivision scheme that recursively divides each triangular face of the control polygon into four triangular
faces interpolating the old control points. Like other interpolating schemes, the subdivision step retains the existing vertices and splits each edge segment at its midpoint. The butterfly scheme is proven to achieve tangent plane continuity when applied to regular meshes. A modification of the Butterfly scheme was proposed by Zorin [333], which guarantees that the scheme produces $C^1$-continuous surfaces for arbitrary meshes.

### 2.2.2 Multiresolution Models

Polygonal surface models are typically composed of a fixed set of vertices and faces, therefore, providing a single fixed resolution representation of an object. However, this single resolution may not be suitable for many applications, particularly when they require complex, highly detailed models to support variable levels of realism. In computer graphics, for instance, flight simulators typically require compact storage, realistic images, and real-time display rates. For such applications, rendering algorithms are often either slow or coarse due to an inappropriate level of detail at which the objects of interest in the scene are displayed. It is desirable that the rendering system can allow dynamic changes to the geometry of the objects with no or little loss in image quality, resulting in higher display rates. In geographic
applications, terrain data are becoming available at finer resolutions, while terrain models have remained static and limited for new application demands. Alternative representations of geological structures and local characteristics of terrain at multiple levels of detail are desirable.

Multiresolution modeling provides an abstraction for representing, manipulating, and visualizing large volumes of spatial data at multiple levels of detail and accuracy. For a given application, a coarse representation can then be used to describe less relevant areas, while high resolution can be focused on specific parts of interest. In order to be effective, a multiresolution model should allow different levels of resolution by means of a compact and flexible representation associated with efficient algorithms for manipulating it. Although multiresolution modeling is not a recent concept, there are few existing automatic tools for supporting variable resolution.

The following sections present the most relevant approaches for multiresolution modeling proposed in the literature.

2.2.2.1 Image Pyramids

*Image pyramids* provide a natural way of constructing multiresolution models for applications in image processing and analysis [260] and texture mapping [151]. An image pyramid stores successively reduced representations of the information in the input image. For instance, each level might represent a version of the original image with dimensions reduced by half from level to level. Figure 2.10 shows an image and its reduced versions.

An alternative representation, known as Laplacian pyramid [32], stores a base image and a sequence of successively larger images representing the differences between two images at successive levels. Such representation provides a compact and appropriate model for a variety of tasks such as multiresolution image editing [239, 334].
2.2.2.2 Volume Pyramids

Octree representation [168, 214, 331] is a geometric modeling technique used in computer vision, robotics, and computer graphics to represent arbitrary 3D objects.

The octree is based on the successive subdivision of an object array into octants. Each node is assigned a label, full if the array is completely enclosed by the 3D object, empty if the array contains no part of the object, and partial if the array partly intersects the object. A node with label full or empty has no children. A node with label partial is subdivided into octants, suboctants, and so on, until cubes (possibly single voxels) consisting of full or empty nodes are obtained. This subdivision process is represented by a tree of degree 8 in which the root node represents the entire object and the leaf nodes correspond to those cubes of the array for which no further subdivision is necessary.

Octrees provide a suitable representation for a number of applications, for instance, hidden surface removal can be easily achieved by traversing the octree in a fixed order specified by the viewpoint. Similar to 2D image pyramids, volume pyramids are appropriate for multiresolution volume modeling, however, they are bulky as a surface representation [152].
2.2.2.3 Wavelets

Wavelet methods provide a mathematical framework for the decomposition of a surface into a base mesh, followed by a sequence of detail coefficients that measure the error between successive finer approximations. Wavelet techniques have been used for producing multiresolution representation in several applications, such as computer graphics [301], radiosity [132], volume rendering [137], and signal decomposition [203].

A method for generating a wavelet decomposition of surfaces with subdivision connectivity is proposed by Lounsbery et al. [202]. However, since a large number of triangles may be used to preserve the subdivision connectivity, their method may produce approximations far from optimal. Wavelet decompositions are also generally unable to resolve breaklines on the surface unless they fall along edges in the base mesh [160].

Eck et al. [88] propose a method for constructing wavelet representations of arbitrary manifold surfaces. The surface must initially be remeshed in order to preserve subdivision connectivity and the topology of the model must remain fixed at all levels of detail. Their method is also unable to properly preserve sharp corners and other discontinuities on the surface.

Gross et al. [136, 138] describe a method for constructing surface approximations by using a wavelet decomposition of the input data samples. Wavelet space filters are introduced in order to control the quality of the surface approximations in local regions. A quadtree-based representation is used to adaptively refine the mesh.

2.2.2.4 Hierarchical Subdivision Models

Hierarchical subdivision models provide an appropriate structure for representing topographic surfaces at different levels of resolution. These models are based on a recursive subdivision of the surface into nested polygons, typically rectangles or triangles. Quadtrees and k-d trees, mentioned in Section 2.1.1.2, are examples of rectangular subdivisions.
Chen and Tobler [43] evaluate different techniques for approximating a surface defined by a quadtree in terms of accuracy, computational speed, and storage costs. The basic problem with quadtree-based models is the occurrence of surface discontinuities along some edges of the subdivision (Figure 2.11). Von Herzen and Barr [156] propose a method for adaptive triangulation of parametric surfaces using a variant of quadtrees, called *restricted quadtree*, which avoids discontinuities.

![Figure 2.11: Discontinuities along an edge in a quadtree-based surface model.](image)

Hierarchical triangulation models are defined by the recursive subdivision of an initial triangle into a set of nested subtriangles with vertices at data points. A *quaternary triangulation model* is proposed by Gómez and Guzmán [130], in which each triangle is recursively subdivided into four subtriangles until a maximum error tolerance is achieved. A significant point near the midpoint of each edge is chosen, and the triangle is split into four nearly congruent triangles, as shown in Figure 2.12. However, since the new vertices are not constrained to lie on the edges, the surface may have discontinuities.

![Figure 2.12: A quaternary triangulation.](image)

A *ternary triangulation model* is described by De Floriani et al. [61], in which each triangle is split into three subtriangles by connecting the three vertices of the
triangle to an internal data point (Figure 2.13). This internal point is selected as
the point that maximizes the approximation error inside the triangle. However, this
approach tends to produce long, thin triangles, which may cause numerical errors
and also produce display artifacts. Ponce and Faugeras [247] suggest an adjustment
step to avoid these problems.

![Figure 2.13: A ternary triangulation.](image)

Schmitt and Gholizadeh [279] propose a method for an adaptive polyhedral ap-
proximation of surfaces whose points are located on a regular grid. The triangular
faces of an initial coarse polyhedral surface are progressively split in order to ob-
tain a finer approximation, which satisfies a quality criterion. A triangle may be
divided into two, three, or four subtriangles by splitting one, two, or three edges of
the triangle.

Scarlatos and Pavlidis [276, 277] present an algorithm that produces a hierar-
chical triangulation based on terrain features. Their method starts with a coarse
triangulation and recursively splits triangles until a given error tolerance is achieved
for each level of resolution. The algorithm adds edges that approximate critical lines
on the surface, in order to preserve the coherence of topographic features such as
valleys and ridges. Figure 2.14 illustrates the five ways that a triangle may be split.

De Floriani and Puppo [69, 70] propose a hierarchical Delaunay triangulation
in order to reduce the occurrence of elongated triangles. The method consists of
refining a triangulation by inserting one vertex at a time until a specified resolution
is satisfied. The refinement process inserts points at which the maximum errors occur
along the edges, and then inserts additional points in the interior of the triangle until

27
the required error tolerance is met. The interior of the triangle is retriangulated using Delaunay triangulation. This refinement process is illustrated in Figure 2.15.

Although this hierarchical approach maintains a local Delaunay triangulation inside each triangle, the global subdivision usually does not represent a Delaunay triangulation.

A problem in using Delaunay triangulations to create a hierarchical structure is that the insertion of a new point may cause a modification which affects several areas of the tessellation. An alternative method for creating a hierarchy of Delaunay triangulations is proposed by De Floriani [60, 68], called Delaunay pyramid. A sequence
of increasingly refined Delaunay triangulations is created, where every triangulation is obtained by iteratively inserting the data which has the maximum approximation error, until the current triangulation satisfies the required accuracy. A set of pointers is used to describe the conceptual links between any pair of consecutive levels in the pyramid. An example of a Delaunay sequence and its corresponding Delaunay pyramid is shown in Figure 2.16. Unlike the hierarchical Delaunay triangulation, the Delaunay pyramid guarantees the equiangularity property to be globally satisfied. However, the Delaunay pyramid does not allow easy local refinements in areas of interest.

Figure 2.16: (a) A Delaunay sequence and (b) its corresponding Delaunay pyramid.

De Berg and Dobrindt [58, 59] present a hierarchical representation for a sequence of Delaunay triangulations, whose structure is based on a scheme proposed
by Kirkpatrick [176] to support point location. A Delaunay triangulation is initially constructed using the entire dataset. From this high resolution triangulation, a number of points are iteratively removed to create successively coarser levels of approximation. An algorithm that extracts a representation at variable resolution is described. However, the resulting surface is not guaranteed to meet the desired accuracy requirements across the domain.

Voigtmann et al. [318, 319] present a hierarchical structure called constrained Delaunay pyramid, which reconstructs a given topographic surface at a variety of predefined resolutions.

Cignoni et al. [51] propose a data structure, called hypertriangulation, for representing all layers of the Delaunay pyramid while avoiding duplication of triangles that belong to more than one layer. The adjacency links between a given triangle and all its adjacent triangles in the different layers containing it are maintained. Such data structure is based on embedding triangles forming the pyramid in three-dimensional space, and each triangle is characterized by its creation and elimination time. The model allows representations to be retrieved at variable degrees of resolution.

2.2.3 Polygonal Surface Simplification

The purpose of polygonal surface simplification algorithms is to reduce the data required to represent complex meshes, improving storage and transmission capacities, and rendering performance. Surface simplification has received increasing attention in recent years. Several approaches have been proposed, a survey of the relevant work in this field is given by Heckbert and Garland [153].

Most surface simplification algorithms can be categorized as refinement and decimation methods. Refinement methods start with a minimal initial approximation of the surface and add new points to the triangulation until the model satisfies a specified approximation criterion. Decimation methods start with a triangulation containing the entire set of data points and iteratively simplify it, until the desired
approximation criterion is achieved.

2.2.3.1 Refinement Methods

Refinement methods start with an initial approximation of the surface and repeatedly add new points to the triangulation until an approximation criterion is reached.

An intuitive approach is to identify the most significant terrain characteristics and construct a triangulation based on these features. Topographic features typically include peaks, pits, passes, valleys and ridges. For instance, a peak is a point of relative maximum, while a point lies on a ridge if its neighborhood can be subdivided by a line passing through it. In this approach, the chosen points become the vertex set and the chosen lines become edges in a constrained triangulation.

The most commonly used technique for extracting topographic features from DEMs uses linear or nonlinear filters to examine elevation patterns in a \( n \times n \) neighborhood of points. Johnston and Rosenfeld [171], and Peucker and Douglas [241] describe methods for detecting topographic features of terrains based on such filters.

Christensen [48] presents a technique for creating a triangle mesh that fits a terrain surface represented by digitized contour lines. The triangulation uses a medial axis transformation, which reduces the structural shape of a region to a set of lines that are equidistant to the boundary. Triangles are formed by connecting vertices located on the axis and contour lines.

Fowler and Little [112] suggest a two-step method for constructing TINs. The first step uses a \( 2 \times 2 \) filter to identify topographic features, similarly suggested by Peucker and Douglas. An initial TIN is constructed using these points. In the second step, additional points are progressively added to the TIN until the model satisfies a specified error tolerance. Their algorithm uses a Delaunay triangulation in order to have the advantage that only a local area must be retriangulated when a new point is inserted.
Chen and Guevara [42] describe a method for selecting *very important points* (VIP) from DEMs, which evaluates the importance of a given point by calculating its contribution to the terrain surface. The importance of a point is based on the difference between its elevation and an estimated elevation from its eight neighbors. All points in the DEM are ordered in terms of importance. Point selection is based on either a predetermined significance level or a specified number of points.

Although this technique is simple and fast, it has several problems due to the local consideration of the importance measure. First, the peak of a small sharp hill, for instance, can be considered more significant than the peak of a larger one, but having smooth slope. Second, many unnecessary points can be chosen in places where topographic features such as ridges and valleys follow a straight line. Finally, the VIP procedure is very sensitive to noise and high frequency.

The VIP procedure is illustrated in Figure 2.17. In a $3 \times 3$ filter (Figure 2.17(a)), a given point $P$ has eight neighbors, forming four diametrically opposite pairs of neighbors. For each pair, the VIP procedure calculates the perpendicular distance $d$ from the point $P$ to the line connecting two neighbors, for instance, $P_1$ and $P_8$ (Figure 2.17(b)). The average of these four perpendicular distances is then used as the importance measure of the point $P$.

![Figure 2.17: (a) $3 \times 3$ filter; (b) elevation interpolation.](image)

Polis and McKeown [245, 246] describe an algorithm that iteratively computes the absolute error at each point in the DEM. Points of maximal absolute error are selected, and these are added to the triangulation, one at a time, since a proximity
criterion is satisfied. Points that are too close to a point already in the current triangulation are not inserted. The algorithm allows user to define a number of operational criteria, such as median error, maximal error, or number of points used to construct the TIN.

A technique called *adaptive triangular mesh filtering* is proposed by Heller [155]. The algorithm constructs an initial triangulation using a set of boundary points (points lying on perimeter) and significant extremes. The priority of the remaining points is calculated as the vertical distance to the current triangular mesh weighted by the inverse of a local tolerance. A heap is used to store the priority of each point. In each step, the point with highest priority is inserted into the triangulation. The insertion of a point requires a local triangulation, which consists of swapping edges of affected triangles to maintain the Delaunay criterion, and readjusting the priorities of all affected points. Significant extremes are determined by checking if each point is local minimum or maximum within a height tolerance.

### 2.2.3.2 Decimation Methods

Decimation methods, in contrast to refinement methods, start with a triangulation containing the entire set of data points and iteratively simplify it, until a specified approximation criterion is achieved. Decimation methods are frequently designed to preserve the original topology of the mesh.

Most decimation algorithms can be classified into three categories, according to the geometric entities (vertices, edges, or faces) used for removal (see Figures 2.18 - 2.20).

1. **vertex decimation**: delete a vertex; retriangulate its neighborhood;
2. **edge decimation**: delete one edge and two triangles; merge two vertices;
3. **triangle decimation**: delete one triangle and three edges; merge three vertices; retriangulate their boundary;
Figure 2.18: Vertex decimation.

Figure 2.19: Edge decimation.

Figure 2.20: Triangle decimation.

The fundamental goal of a decimation algorithm is to reduce the total number of triangles in a mesh, while preserving as accurately as possible important features.

Lee [188] proposes a drop heuristic method for simplifying terrains. The algorithm uses a vertex decimation approach, removing a vertex in each pass. An initial triangulation is created by adding an edge that connects two diagonal points in each 2×2 neighborhood. The error, measured as differences in elevation between surfaces, is computed for each remaining point in the triangulation, and the point with the lowest error is deleted. The process continues until a predefined tolerance level, or a specified number of points is achieved.
The drop heuristic method is computationally expensive since each vertex is visited on every step, and a dense initial triangulation must be created. A comparison of his algorithm to three other methods is evaluated by Lee [190].

Scarlatos and Pavlidis [278] present a method that uses curvature\textsuperscript{5} equalization for improving a triangulation. First, the size of the triangles with highest curvature is reduced by moving each vertex of the triangle. Unnecessary pairs of adjacent triangles are then removed by collapsing their common edge into a single point. In most cases, the method reduced the number of triangles, however, it also increased the maximum error when explicit error tests were not used.

Schröder and Roßbach [280] present a decimation method in which the importance of a vertex is evaluated according to a measure of the roughness of the terrain at that vertex. A vertex is removed from the mesh if it does not affect the overall representation significantly. The area around the removed point is retriangulated.

A general vertex decimation algorithm is presented by Schroeder et al. [283]. The algorithm performs multiple passes over an existing triangulation, removing vertices until a specified error is achieved. The error at a vertex is calculated as the distance from the point to the average plane of the surrounding vertices. Since errors are measured with respect to the previous approximations, not relative to the input points, errors can accumulate. Schroeder [282] improves the calculation of error measures in later versions of the algorithm.

Turk [313] presents a method for simplifying polygonal surfaces by triangulating a new set of vertices which replaces the original one. An iterative point repulsion procedure is used to distribute the new set of points over the surface, concentrating more points in regions of higher curvature. The original points are then removed one by one, resulting a triangulation that preserves the topology of the original surface.

\textsuperscript{5}For more information on surface curvature, see Section 4.3.1.2 on page 79.

35
range views is presented by Soucy and Laurendeau [296]. A sequential optimization process is used to iteratively remove the vertex minimizing the retriangulation error. A constrained Delaunay triangulation is used to retriangulate the surface and a version of Lawson's local optimization procedure [184] is proposed to improve the triangulation. To compute the retriangulation error at each iteration, a vertex is temporarily removed, the area affected by its removal is retriangulated, and then the minimum distance between the removed vertex and the local retriangulation is computed. The vertex minimizing the error is effectively removed and the retriangulation error of all neighbors of the removed vertex is recomputed. An extension of this method is described by Soucy and Laurendeau [297].

Garland and Heckbert [126] describe an algorithm that uses iterative contractions of vertex pairs to simplify polygonal models. A quadratic error metric is used to characterize the error at each vertex.

An algorithm for reducing triangular meshes based on the Hausdorff distance\(^6\) between two surfaces is proposed by Klein et al. [177]. The simplification algorithm starts with a coarse triangulation and successively removes vertices and retriangulates the resulting holes until a predefined threshold between the original and the simplified meshes is achieved. A modification of this algorithm to allow a multiresolution representation is described in [178]. Methods for extracting triangulations at uniform and variable levels of detail are described. Ciampalini et al. [50] present a vertex-based decimation algorithm that also uses global error measures.

Hoppe et al. [162] present a method for 3D surface simplification based on mesh optimization. Coarser triangulations are obtained from an initial fine triangulation by minimizing an energy function. This function consists of three terms: a distance energy that measures the closeness of a fit, a representation energy that penalizes meshes with a large number of points, and a spring energy that penalizes long edges.

\(^6\)Hausdorff distance is described in more details in Section 4.3.1.9 on page 90.
in the triangulation. The optimization process uses heuristics and random selection to either collapse, split, or swap an edge. Nonlinear optimization techniques are used to find the vertex positions that minimize the global error for a given topology. Topological changes are kept if they reduce the global error, otherwise they are discarded. A simplified version of this method is presented by Hoppe [160], which uses a sequence of edge collapses to construct **progressive meshes**. An approach that attempts to extract variable levels of detail from progressive meshes through more sophisticated data structures is proposed by Xia and Varshney [330].

Hamann [144] presents a triangle-based decimation technique, where a **weight** is assigned to each triangle and is used as criterion to select triangles to be removed. The weight of a triangle is determined by considering the principal curvatures estimates at its vertices and a measure of equiangularity. Long, thin triangles and low curvature triangles are removed first. The region affected by the removal of a triangle is retriangulated, and weights are computed for all new triangles introduced. An extension of this method is presented in [127], which is used to generate a sequence of triangular meshes. Other criteria are also proposed to assign a weight to each triangle.

Kalvin and Taylor [174] present an algorithm for simplifying polyhedral meshes within a prespecified error tolerance based on a bounded approximation criterion. The algorithm initially partitions the surface into patches, called **surfaces**. Each patch is formed by merging nearly coplanar faces. Additional tests prevent degenerate surfaces from being created. The algorithm also simplifies the surface perimeters using a split-and-merge technique. Finally, the patches are retriangulated by decomposing each surface into star polygons and then triangulating each star polygon. Their method uses similar ideas to those proposed by Hinker and Hansen [157], in which polygons are grouped into nearly coplanar sets and then retriangulated into fewer simple polygons than originally required.
An edge-based decimation algorithm for simplifying surfaces is described by Guéziec and Hummel [140]. To simplify the surface, edges are collapsed into a vertex, respecting certain geometric and topological criteria. The simplification process minimizes a measure of error volume using linear programming. An edge collapse is considered valid if it satisfies the following conditions: the topology of the surface is preserved; the normals of the modified faces change little; the new triangles are well shaped; and the error associated with the new vertex is within a threshold. A related algorithm is presented by Guéziec in [139].

Lindstrom and Turk [196, 197] propose a simplification method that repeatedly selects the edge with a minimum cost, collapses this edge, and then recomputes the cost of the edges affected by this edge collapse. The new vertex position from an edge collapse is chosen such that certain geometric properties (e.g. area and volume) are minimized. Using a different approach, Lindstrom et al. [195] present a method based on a regular grid representation to approximate terrains at different levels of detail. The algorithm reduces the number of rendered polygons and allows for smooth changes in resolution across areas of the surface, achieving high frame rates.

Cohen et al. [54] propose the idea of simplification envelopes for generating a hierarchy of level-of-detail approximations for a given polygonal model. Their approach guarantees that all points of the approximate model are within a specified tolerance from the original model. Successive modifications are applied to the original model such that the resulting surface lies between two envelope surfaces.
CHAPTER 3

ADAPTIVE TERRAIN SURFACE APPROXIMATION

The surface representation of a terrain can be generally viewed as a $2\frac{1}{2}$-dimensional modeling problem, where a bivariate function $z = f(x, y)$ expresses the elevation $z$ of the surface at a point $(x, y)$ of the Euclidean plane. Therefore, any line parallel to $z$ axis penetrates the surface at most once. Although this representation covers most natural terrains, it excludes some features such as caves and faults.

As mentioned in Chapter 2, two models are commonly used to represent terrain surfaces, regular grid digital elevation models (DEMs) and triangulated irregular networks (TINs). A DEM consists of a finite set of elevation values stored at regular intervals. Alternatively, a TIN model approximates a topographic surface by a set of nonoverlapping contiguous triangular faces generated from a finite set of sampled data points.

The advantages and disadvantages between DEMs and TINs have been extensively discussed in the literature [31, 206, 242]. Although a definite supremacy between both these models is difficult to establish due mainly to the wide variety of problems in terrain analysis and the computational resources available, TINs have become increasingly used because of their efficiency in storing terrain data.

The main advantages of using TINs are:

- the structure of the triangulation can be adjusted to reflect the high irregularity of terrain data. Consequently, cells become larger where data are sparse, and smaller where data are dense;

- the triangulation can incorporate terrain features. For instance, vertices in a TIN can describe nodal terrain features such as peaks, pits or passes, while edges can represent linear terrain features such as break, ridge or channel lines;
• TINs can be organized into a hierarchical model so that they can represent a terrain in various levels of detail;

• triangles are simple geometric objects which can be easily manipulated and rendered.

The purpose of constructing a TIN from dense terrain data is to extract a set of irregularly spaced elevation points which are as few as possible, while maintaining as much information as possible about topographic structures.

An example of a TIN representing terrain surfaces is illustrated in Figure 3.1. The terrain model must provide a continuous surface and define spatial relationships of the surface.

![Figure 3.1: Example of a TIN representing terrain surfaces.](image)

In order to prevent discontinuities between triangles in the surface, the model must contain a collection of triangles provided that an edge shares a maximum of two adjacent triangles. Figure 3.2 shows an example of a triangulation which does not satisfy such condition.

The triangulation of a set of data points in the plane can be defined in terms of a planar graph in which pairs of vertices are connected by edges intersected only at their endpoints, forming triangular faces. The topology of the triangulation can be generally chosen either using only the $xy$ projections of the data points or using
the elevations of the points as well. These approaches are described in details in Appendix A.

The most common triangulation method that uses only the $xy$ projections is the Delaunay triangulation. The Delaunay triangulation has the property that the circumcircle of any triangle in the triangulation contains no other data points in its interior, known as circle property. The Delaunay triangulation generates the triangulation that maximizes the minimum angle of all triangles. This property is known as max-min angle property. The equivalence between the circle property and the max-min angle property was shown by Lawson [184] and Sibson [291]. In a Delaunay triangulation, most of its triangles are nearly equiangular, which helps to minimize the occurrence of thin and long triangles since they can lead to undesirable behavior, affecting numerical stability and producing visual artifacts. Another interesting property is that Delaunay triangles define nearest natural neighbors in the sense that the data points at the vertices are closer to their circumcenter than is any other data point. These circumcenters are the positions of vertices in the geometrically dual Voronoi diagram, also known as Dirichlet, Thiessen or Wigner-Seitz tessellation.

Due to these properties of the Delaunay triangulation, associated with the fact that many efficient algorithms for its construction exist, our method adopts the
Delaunay triangulation to represent terrain surfaces.

The following sections describe an adaptive method for representing terrain surfaces based on triangular meshes. Initially, a piecewise linear interpolation is constructed by creating triangular elements at the data points. A scheme for constructing $C^1$ continuous piecewise surface patches over the data triangulation is then presented and applied to several real terrain data sets.

### 3.1 $C^0$ Continuous Triangular Patches

The problem of representing terrain surfaces with a piecewise linear interpolation has been proposed by several authors (for instance, see Floriani et al. [62], and Lee and Schachter [186]). A basic algorithm starts with an initial triangulation that covers the boundary of the domain, and iteratively adds points from the data set until a specified error tolerance is achieved.

Although many programs for extracting a TIN from a set of elevations exist, most methods have been demonstrated only on small terrain data sets. Most methods also produce poor approximations in regions of discontinuity or poor vertex selection in the presence of noise. This occurs because there is no obvious strategy for determining the optimal vertex locations in advance, and vertices inserted (or deleted) early in the refinement (or decimation) process may become later unnecessary by better vertices. In fact, Agarwal and Suri [3] prove that the problem of approximating surfaces while minimizing the number of vertices for a given accuracy is $NP$-hard. Practical solutions found in the literature are often based on heuristics that attempt to produce an approximate model by either iteratively adding new vertices to a coarse triangulation or iteratively removing points from an initial triangulation built over the entire data set.

The selection of points to build a TIN can be thought of as two-dimensional version of the line simplification problem, whose purpose is to find the minimum number of points that represent a line within an error tolerance. The algorithm
by Douglas and Peucker [83], probably one of the best known line simplification algorithms\(^1\), initially starts by connecting the two endpoints of the original line. The distances from each point to this line are then calculated. If all these distances are within some prespecified tolerance, the process stops. Otherwise, the point that is farthest from the approximation is selected, and the line is split into two segments. This process is then repeated on the two new lines. This algorithm is illustrated in Figure 3.3.

\[
\begin{array}{cc}
\text{(a)} & \text{(b)} \\
\text{(c)} & \text{(d)}
\end{array}
\]

Figure 3.3: The Douglas-Peucker algorithm for line simplification.

Since the use of additional vertices can result in a nonsmooth surface approximation or in an inadequate data distribution, an approach that might be useful is to alternate refinement and decimation steps, inserting several vertices with an incremental triangulation algorithm, and then removing a few vertices that appear the least important.

An example caused by the short-sightedness of most incremental triangulation algorithms is given by Garland and Heckbert [125], where the object to be approximated contains a step discontinuity or cliff (Figure 3.4). The left half of the grid has constant height 0 and the right half has constant height 1. From a \(100 \times 100\) grid, 99 vertices were selected to achieve zero error. Since only 8 vertices suffice, the

\(^1\)Similar algorithms were independently developed by Duda and Hart [84], and Ramer [253].
triangulation algorithm generated several unnecessary vertices.

Figure 3.4: Redundant triangulation generated by incremental triangulation algorithm. Figure adapted from Garland and Heckbert (1995).

We propose a new method for adaptively approximating terrain surfaces through repeated refinement and decimation passes [236], providing higher quality triangulations with great flexibility. Initially, a minimal approximation consisting of two triangles is constructed. This mesh is then incrementally refined until either a specified error tolerance is achieved or a given number of points is reached. Once the desired goal has been satisfied, the approximation is simplified by eliminating a small number of points based on a vertex removal criterion. Finally, the approximation is again refined to the given error tolerance and partially resimplified. This alternate refinement and decimation process is repeated until no further improvement in the accuracy of the approximation can be achieved. The repeated application of these steps can be viewed through the diagram shown in Figure 3.5.

The method, analogous in some aspects to other approximation techniques such as stepwise linear regression techniques, incrementally improves the triangulation by determining a better distribution of the data points, while maintaining a specified error tolerance. Although the method is not necessarily optimal or monotonically convergent, it produces approximations that are significantly better than those generated by straightforward incremental triangulation algorithms.
As described above, the first step is to generate a coarse piecewise linear approximation of the surface according to a predefined error tolerance. This initial triangulation is then refined by iteratively adding new points, updating it after each point is inserted. The Delaunay triangulation is used to incrementally construct the mesh.

The vertex selection criterion is crucial during the triangulation process since it determines the degree of fidelity between the original data and the approximation. The magnitude of the error can be estimated by using the $L_n$ function norm, defined as

$$L_n(\phi) = \|\phi\|_n = \sqrt[n]{\int_\Phi |\phi(x, y)|^n \, dx \, dy}$$

where $\phi$ is a function defined over domain $\Phi$.

When used to characterize the error of an approximation, $\phi(x, y)$ represents the difference between the original and approximate surface. The norm can be computed over a limited region to estimate local error or over the entire domain to measure global error. Global error measures usually produce better approximations, however, the resulting algorithms are significantly slower than those using local metrics. The most commonly used norms are the $L_1$, $L_2$, and $L_\infty$ norms. The $L_1$ norm of the approximation error, $\epsilon$, corresponds to the volume between the surfaces. The $L_2$ norm provides a measure of the average root mean square (RMS) error between the original and
the approximation. Another common measure is based on the maximum difference between the actual elevation data and the surface approximation (Figure 3.6), known as $L_\infty$ norm.

![Figure 3.6: Maximum vertical error.](image)

A variation to the conventional maximum vertical error measure is proposed as the vertex selection criterion, which is based on the maximum vertical error weighted by the standard deviation calculated in a neighborhood of the candidate point, given by

$$C = \frac{h(p) - z(p)}{\sigma(p)}$$

where $h(p)$ is the height value of point $p$, $z(p)$ is the height value of the interpolated surface at point $p$, and $\sigma(p)$ is the standard deviation calculated in a $3\times3$ neighborhood of the candidate point $p$.

The idea is to associate greater importance to the points in regions where the local variability of the data is high, allowing the surface to conform to the local trends in the data. Our experiments have demonstrated that this vertex selection criterion is slightly superior to the vertical error measure.

A triangulation $T_k$ satisfies a given error tolerance $\epsilon$ if its maximum deviation over all points is less than or equal to $\epsilon$. A priority queue stores the sequence of vertices used to refine the triangulation, ordered by increasing approximation error. For each refinement step, only those vertices affected by the insertion process need
to have their approximation error recalculated.

Since this strategy for determining the vertex locations is based on a local heuristic, suboptimal insertions can eventually be performed by the refinement process. The approach proposed to deal with this problem is to identify and remove those points that may have become unnecessary by later insertions.

A decimation algorithm is then defined to produce a sequence of triangulations through a set of vertex removal operations. At each iteration, the vertex with the smallest error is removed and the area affected by its removal is retriangulated. This process is repeated until a specified error tolerance is achieved.

The criterion for removing a vertex \( v \) is computed by averaging the surface normals \( n_i \) of the triangles surrounding \( v \) weighted with their areas \( A_i \) (see Figure 3.7) and taking the maximum angle, \( \alpha_{\text{max}} \), between the averaged normal, \( n_{\text{av}} \) and the surrounding triangles, that is

\[
\alpha_{\text{max}} = \max \left( \arccos \frac{n_{\text{av}} \cdot n_i}{|n_{\text{av}}| \cdot |n_i|} \right)
\]

where \( n_{\text{av}} = \frac{\sum n_i A_i}{\sum A_i} \) and \( 0 \leq n_i \leq \pi \).

![Figure 3.7: Criterion for vertex removal.](image)

This criterion describes a measure of roughness, indicating a local variability of the data. Each vertex having a value less than the specified maximum angle will be removed during the decimation process. The area around the removed point is retriangulated, which requires careful consideration, in particular when such area
is not a convex polygon. The edges that form the triangulation of the polygon surrounding the vertex $v$ must be checked to determine if they do not intersect one another. Figure 3.8, for instance, illustrates an invalid retriangulation.

![Figure 3.8: An invalid retriangulation.](image)

Similarly to the refinement algorithm, for each vertex affected by the local retriangulation, the approximation error is recalculated and stored in the priority queue. The algorithm stops when the smallest retriangulation error of a vertex becomes larger than error tolerance $\varepsilon$.

An example of triangulation obtained by our method is given in Figure 3.9(a), while the same object is approximated by a conventional incremental insertion algorithm, shown in Figure 3.9(b). It can be noticed that our combined refinement/decimation method improves the result of the triangulation by adding and/or removing vertices based on the criteria described above.

The object shown above was defined as the function

$$F(x, y) = \begin{cases} 
0.0 & \text{if } \sqrt{(x - 0.5)^2 + (y - 0.5)^2} \geq 0.3 \\
1.0 & \text{if } \sqrt{(x - 0.5)^2 + (y - 0.5)^2} < 0.3 
\end{cases}$$

Pseudocodes for the refinement and decimation steps are presented in Figures 3.10 and 3.11, respectively.

Another important aspect of our method is that a priori information about topographic characteristics of the terrain can be incorporated into the triangulation.
Figure 3.9: Approximation of an object (using 100 vertices) constructed by (a) our combined refinement/decimation method and (b) conventional incremental insertion algorithm.

```plaintext
// refinement process
construct initial triangulation using domain boundary;
if ∃ constrained vertices and edges
    include them in the triangulation;
compute approximation error for each vertex;
while (maximum error > error tolerance ε) {
    find vertex v with maximum error;
    insert v;
    update triangulation;
    recomputes error for vertices affected by the local update;
}
```

Figure 3.10: Pseudocode for refinement step.

This information can describe nodal features (such as peaks, pits, or passes) and linear features (such as ridges, rivers, roads, channels, or cliffs). These nodal and linear features are inserted in the triangulation as constrained vertices and edges, respectively, in such a way that subsequent operations will preserve them.

Heller [155] describes an algorithm where linear features (breaklines) are inserted in an existing triangulation. The reorganization of the mesh to adjust such breaklines seems to produce regions of long and thin triangles.

In our approach, the constraints are included as the first elements of the triangulation, then reducing the number of edge swaps necessary to update the mesh each time a new breakline needs to be inserted. Breaklines are positioned along the edges.
// decimation process
while (maximum error < error tolerance ε) {
    find vertex v with minimum error;
    find vertices \( w_i = w_1, ..., w_k \) adjacent to v;
    if v satisfies decimation criteria {
        remove v;
        delete all triangles connected to v;
        retriangulate polygon defined by vertices \( w_1, ..., w_k \);
        recompute error for each \( w_i \);
        modify list of vertices;
    }
}

Figure 3.11: Pseudocode for decimation step.

of triangles, which is another surface behavior that cannot be easily handled by grid-based methods. Additionally, new points can be added on the constrained edges to guarantee that the final triangulation satisfies the Delaunay criterion. Figure 3.12 shows an example of this procedure.

![Figures 3.12](image)

Figure 3.12: (a) Breakline represented by segment \( AC \) is not a Delaunay edge because the circumcircle of triangle \( ABC \) contains point \( D \); (b) new point \( F \) is added on the constrained edge to satisfy the Delaunay criterion.

Although numerous methods for extracting 3D features from range images have been developed in recent years, more research is still needed to automatically segment significant structural properties of the original image. Many systems are based on restrictive assumptions about the structure of the sensed image data. The lack of explicit use of knowledge of the application usually limits the surface reconstruction
process to very specific domain [235, 237]. For instance, more precise features might
be directly measured from aerial photographs acquired from lower platform (that is,
having higher resolution than the basic elevation data) in order to make better use
of the original image data and provide a more accurate surface description for the
feature extraction step.

In computer vision field, a great amount of effort has been expended on automati-
cally detecting characteristics of 2D and 3D objects. Many types of features can
be used for object recognition [25]. Global features usually represent characteristics
of regions such as area, perimeter, Fourier descriptors, and moments. Local features
represent a distinguishable small area in an image, usually based on significant inten-
sity changes or surface discontinuities. For instance, reflectance, shadow boundaries,
corners, and edges play an important role in object recognition. Relational features
are based on the relative positions of entities (such as contours, regions, or local
features), which are useful to form composite features.

One common method for detecting surface curvature is to use differential ge-
ometry² concepts to identify surface shape of objects. Basic surface types can be
determined by analyzing the local behavior of the Gaussian and mean curvature.
These two surface curvature functions have many interesting properties, an impor-
tant aspect is that they are invariant to translations, rotations, and changes in pa-
rameterization. Since the computation of surface derivatives is extremely sensitive
to noise or other small fluctuations, image smoothing is usually required to reduce
the effects of noise. However, such smoothing can also suppress relevant information
or cause edge displacement. The use of a binomial mask [24, 25], designed in a way
that the detected discontinuity edges are not crossed, is suggested here.

Triangular models also provide support for the extraction of topographic fea-
tures. This can be achieved by analyzing the structure of the triangles in the mesh.

²See Section 4.3.1.2 on page 79 for more details.
Figure 3.13 illustrates some of the most common topographic features, which are classified according to the angle between the normals of adjacent triangles.

![Topographic features in triangular meshes.](image)

Our method also offers the possibility of locally changing the level of resolution in different regions of the surface, therefore allowing a coarser approximation of less relevant areas and a finer approximation within specific regions of interest. This dynamic adjust of resolution produces a *zoom* effect in the terrain model. An example of this effect applied to a particular portion of the Lake Champlain DEM is demonstrated in Figure 3.14.

![Local adjust of resolution at an arbitrary location.](image)

### 3.1.1 Data Structures

Our method is based on the quad-edge data structure described by Guibas and Stolfi [142]. The quad-edge has the ability to represent simultaneously a general subdivision of a two-dimensional manifold and its dual, such as a Delaunay triangulation
and its corresponding Voronoi diagram. Another advantage is that the quad-edge structure requires only two topological operators to build and modify arbitrary diagrams.

Each quad-edge structure represents an undirected edge in a subdivision by four directed edges. Each directed edge has two pointers, a next pointer to the next counterclockwise edge around its origin, and a data pointer to geometric and other nontopological information, such as the coordinates of its origin. In Figure 3.15, three edges incident on the same vertex are represented using the quad-edge data structure [198]. The vertex corresponds to the inner cycle of pointers, whereas the remaining three cycles correspond to the three faces meeting at the vertex. An example of a triangulation and its corresponding representation with the quad-edge data structure is given in Appendix B on page 181.

![Quad-edge structure](image)

Figure 3.15: Quad-edge structure representing three edges incident on one vertex.

A classical problem in computational geometry is to determine which region of a subdivision contains a given point. Several theoretically optimal algorithms have been proposed in the literature, examples of efficient algorithms are given by Kirkpatrick [176], Sarnak and Tarjan [275], Clarkson and Shor [52], Asano et al. [8], and Seidel [284]. However, many of these methods do not necessarily produce good practical performance due to the preprocessing time and additional storage requirements.
A simple method for locating a point in a triangulation is the \textit{jump-and-walk} strategy [134]. The idea is to start from an arbitrary place in the triangulation and then move in the direction of the query point until the containing triangle is reached, taking expected time $O(n^{1/2})$ for each point, or $O(n^{3/2})$ in total. In many practical applications, successive points tend to be close to each other, therefore, if the point location procedure uses the edge returned in the previous call as its starting point, each insertion may take approximately constant time on the average [142]. Due to its simplicity and efficiency, we use the jump-and-walk strategy in our method for locating points in the Delaunay triangulation.

\subsection{Mesh Compression}

The triangular surface models produced by our algorithm are stored using a representation in uncompressed format, which describes the list of vertices in a sequence such that each vertex can be uniquely identified by an integer reference, and the list of triangles containing three vertex references for each triangle. The conversion from this representation to other 3D format models such as STL [1], VRML [16], and DXF [10] is straightforward. In STL, for instance, each triangular facet is described by its unit normal and three points representing the vertices of the triangle.

Because of the increasing use of TINs as terrain models, the study of techniques for compressing triangular meshes has received great attention in the last few years. The use of compact structures can significantly reduce storage and transmission time requirements.

We now summarize the most relevant methods for compressing triangular meshes, since they can be directly applied to reduce the size of our terrain models. Such methods usually address two different tasks, the compression of numerical information associated with each vertex (such as position, elevation, texture, normal vectors) and the compression of information describing the connectivity between the surface components. These approaches are here organized into two categories, geometry
compression and topology compression, respectively.

3.1.2.1 Geometry Compression

Geometric data associated with each vertex are usually reduced using lossy methods based on quantization, and lossless methods based on entropy encoding such as Huffman or arithmetic coding.

The concept of geometry compression was introduced by Deering [72]. In his lossy compression method, vertex positions, normals, and colors are quantized to less than 16 bits. The quantized vertex positions are encoded by using a delta compression followed by a modified Huffman encoding. This technique can compress geometric data with a factor of 6 to 10 times smaller than uncompressed geometry, without significant loss in image quality.

Taubin and Rossignac [304] describe a method where vertex positions are quantized within a desired level of accuracy, and a vertex spanning tree is used to predict the position of each vertex from its ancestors in the tree. An entropy encoding technique is used to compress the coordinate corrections between the predicted and the actual data. Their method uses approximately 12 bits per vertex, that is, an average of 4 bits per coordinate to represent complex models, which is significantly more compact than 32-bit single precision IEEE floating-point numbers or 64-bit double precision numbers.

A comparison of several lossless and lossy compression methods for elevation data is presented by Franklin [116]. An efficient lossy compression method is applied to a number of DEM files in Franklin and Said [118]. Visibility indices are calculated and compared to estimate the performance of their compression method.

3.1.2.2 Topology Compression

A simple way to represent connectivity is to use a triangle-vertex incidence table, which associates each triangle with its three bounding vertices. Since the number of
triangles is approximately twice the number of vertices, the use of efficient techniques for compressing the triangle-vertex incidence table becomes an important issue.

A representation storing each triangle as a list of 12-bit integer coordinates for each one of its three vertices would require 108 bits per triangle. Since the location of a vertex is repeated six times on average, it becomes expensive to store multiple representations of each vertex. An alternative is to store a table containing the vertex data in a sequence and a table containing three vertex references for each triangle. A vertex reference uniquely identifies the position of a vertex in the vertex data table. Since we need at most $\lceil \log_2 n \rceil$ bits per vertex reference in a triangulation with $n$ vertices, this scheme requires a connectivity cost of $3 \log_2 n$ bits per triangle. Bar-Yehuda and Gotsman show that a buffer size of $12.72 \sqrt{n}$ suffices to render any triangular mesh with $n$ vertices, such that each vertex is transferred only once. Rossignac [265] estimates that this improvement leads to a connectivity cost of $1.25 \log_2 n + 9.75$ bits per triangle.

A technique supported by OpenGL [226] and other graphic libraries decomposes a triangular mesh into triangle strips. A triangle strip is a path of triangles where their vertices are specified in a zig-zag order (see Figure 3.16). However, the straightforward use of triangle strips does not result in high compression rates. Each vertex is encoded twice on average, and it is also difficult to obtain long strips from a generic mesh [92]. Long strips are desirable since the first two bits are the overhead for each strip.

Deering [72] proposes the use of a buffer of vertices to avoid that a vertex is encoded more than once. Following this idea, Evans et al. [92] discuss the impact of buffer sizes on triangle strip performance, and Chow [47] proposes heuristics to improve the decomposition of triangular meshes into triangle strips. Rossignac [265] suggests modifications to the idea of using a buffer of 16 positions proposed by Deering, estimating a connectivity cost of $3.75 + 0.062 \log_2 n$ bits per triangle, when
a vertex is used twice on average.

![Figure 3.16: A triangle strip.](image)

Progressive meshes, developed by Hoppe [160, 161], provide a technique for transferring a mesh progressively, starting from a coarse approximation and then iteratively inserting a sequence of new vertices. A new vertex is created by expanding a vertex into an edge, which is the inverse of the edge collapse operation used in many mesh simplification techniques. Each vertex is transferred only once and 5 bits are used to identify two vertices among those adjacent, giving a total connectivity cost of approximately \( \lceil \log_2 n \rceil + 5 \) bits.

Li and Kuo [194] combine progressive transmission of connectivity refinements with progressive transmission of vertex data. A vertex decimation method produces a sequence of operations that reduces the level of detail of the model. The inverse connectivity refinement operations are encoded by identifying the base triangle where the vertex must be inserted and by labeling the surrounding edges to indicate which of these must be flipped to restore the correct incidence. This approach leads to an average connectivity cost of \( 0.5 \log_2 n + 5 \) bits per triangle.

The connectivity information for a simple mesh can be described as a triangulated planar graph. Turan [312] shows that labeled planar graphs can be encoded using slightly less than \( 12n \) bits. Keeler and Westbrook [175] improve on Turan’s results, encoding planar graphs with \( 4.6n \) bits or, equivalently, 2.3 bits per triangle. A triangle-spanning tree is built to encode the connectivity.
Methods based on vertex permutation are proposed by Snoeyink and Kreveld [293, 294], and Denny and Sohler [76]. Snoeyink and Kreveld present a method that constructs a Delaunay triangulation in linear time from an ordered set of vertices. As a consequence, only the vertex coordinates need to be stored to compress the Delaunay triangulation. However, since the connectivity information is omitted, the decompression algorithm requires the use of an incremental Delaunay triangulation to reconstruct the mesh. Denny and Sohler show that there are less than \(2^{8.2n + O(\log n)}\) valid triangulations of a planar set of \(n\) points and, for sufficiently large \(n\), each triangulation can be associated with a different permutation of these points.

A compression method based on traversing a triangulation in a *shelling order* (that is, radially around an initial triangle), is proposed by De Floriani et al. [66]. It encodes each vertex once, and produces less than 2.25 bits of connectivity information for each triangle.

An efficient method for compressing connectivity of 3D triangular meshes is presented by Rossignac [265]. This scheme, called *Edgebreaker*, produces results between 1.3 and 2 bits per triangle in simply connected manifold triangular meshes. It also supports meshes with holes and handles by using additional storage.

### 3.1.3 Geometric Primitives

Some useful geometric primitives used in the implementation of our Delaunay triangulation method are:

- area of a triangle: the magnitude of the cross product of two vectors \(\mathbf{r}\) and \(\mathbf{s}\) corresponds to the area of the parallelogram determined by these vectors. Since any triangle can be viewed as half of a parallelogram (see Figure 3.17), then \(\frac{1}{2} |\mathbf{r} \times \mathbf{s}|\) is the area of a triangle \((P_1, P_2, P_3)\), which can be described as
the determinant

\[ \text{Area}(P_1, P_2, P_3) = \frac{1}{2} |\mathbf{r} \times \mathbf{s}| = \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \]  

(3.1)

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{triangle.png}
\caption{Area of a triangle.}
\end{figure}

- orientation test: This test determines whether a point lies to the left of a directed line defined by two other points, (see Figure 3.18), which can be implemented as

\[ \text{Left}(P_1, P_2, P_3) = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} > 0 \]  

(3.2)

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{orientation.png}
\caption{The orientation test.}
\end{figure}

Note the connection to Equation 3.2, that is, \( P_3 \) is to the left of \( (P_1, P_2) \) if and only if the area of the counterclockwise triangle is positive.
• InCircle test: this test determines whether a point $P_4$ lies inside an oriented circle formed by $P_1P_2P_3$ (see Figure 3.19), given by

$$InCircle(P_1, P_2, P_3, P_4) = \begin{vmatrix} x_1 & y_1 & x_1^2 + y_1^2 & 1 \\ x_2 & y_2 & x_2^2 + y_2^2 & 1 \\ x_3 & y_3 & x_3^2 + y_3^2 & 1 \\ x_4 & y_4 & x_4^2 + y_4^2 & 1 \end{vmatrix} > 0 \quad (3.3)$$

![Figure 3.19: The InCircle test.](image)

• Incenter: the incenter of a triangle is the intersection point of the angle bisectors (see Figure 3.20).

![Figure 3.20: Incenter of a triangle.](image)

The incenter is defined as

$$Incenter(P_1, P_2, P_3) = \frac{|P_2 - P_3|P_1 + |P_1 - P_3|P_2 + |P_1 - P_2|P_3}{\text{Perimeter}(P_1, P_2, P_3)} \quad (3.4)$$
where

\[ \text{Perimeter}(P_1, P_2, P_3) = |P_1 - P_2| + |P_2 - P_3| + |P_1 - P_3| \quad (3.5) \]

- estimated elevation: in order to estimate the height of a point \( P(x, y, z) \) that lies in the interior of a triangle \( (P_1, P_2, P_3) \), we can use the equation of the triangular facet, that is,

\[ z = Ax + By + C \]

where \( A, B, \) and \( C \) are the coefficients that define the normal to the triangular facet.

### 3.2 \( C^1 \) Continuous Triangular Patches

This section describes two methods for generating smooth surfaces over a triangular mesh using piecewise polynomial patches. The first interpolation scheme uses quadratic polynomials and the second one uses cubic polynomials to construct \( C^1 \) surfaces. Both methods divide each triangle into subtriangles and fits an approximating function over each subtriangle. Certain continuity conditions must be satisfied at every boundary between two patches in order for the entire surface to be smooth. Since the methods require only elevation and derivative values at the vertices of the triangulation, such approaches are relatively simple to implement.

Before describing the interpolation methods, some preliminary concepts are introduced.

#### 3.2.1 Barycentric Coordinates

The use of barycentric coordinates is a natural way of representing triangular patches, since this guarantees a symmetric influence of all three triangle corners. Let \( T \) be a planar triangle defined by the vertices \( V_1, V_2, V_3 \). Any point \( V \) in \( T \) can be
expressed in terms of the barycentric coordinates \((r, s, t)\) defined by

\[
V = rV_1 + sV_2 + tV_3
\]

where \(r + s + t = 1\) and \(0 \leq r, s, t \leq 1\).

If \(V\) is joined to \(V_1, V_2, V_3\), then \(T\) is partitioned into three subtriangles (see Figure 3.21). The areas of these triangles are proportional to the barycentric coordinates \(r, s, t\) of \(V\), that is

\[
\begin{align*}
    r &= \frac{\text{Area}(V, V_2, V_3)}{\text{Area}(V_1, V_2, V_3)} \\
    s &= \frac{\text{Area}(V, V_1, V_3)}{\text{Area}(V_1, V_2, V_3)} \\
    t &= \frac{\text{Area}(V, V_1, V_2)}{\text{Area}(V_1, V_2, V_3)}
\end{align*}
\]

![Figure 3.21: Barycentric coordinates.](image)

An important property of barycentric coordinates is their invariance under affine transformation (translation, rotation, scaling, or shear).

### 3.2.2 Bernstein Polynomials

Bernstein polynomials of degree \(n\) over a triangle \(T\) can be defined in terms of barycentric coordinates \((r, s, t)\) expressed as

\[
B_{i,j,k}^n(r, s, t) = \frac{n!}{i! j! k!} r^i s^j t^k
\]
which form a basis for all bivariate polynomials of degree \( n \).

### 3.2.3 Triangular Bézier Patches

The parametric equation for a single triangular Bernstein–Bézier patch is

\[
p(r, s, t) = \sum_{i+j+k=n \atop i, j, k \geq 0} b_{i,j,k} B^n_{i,j,k}(r, s, t)
\]  

(3.6)

where the coefficients \( b_{i,j,k} \) are called the Bézier control points of \( p(r, s, t) \).

These points form a regular triangular array of \( (n + 1)(n + 2)/2 \) points in the triangle \( T \).

Taking \( n = 2 \), Equation 3.6 gives

\[
p(r, s, t) = r^2 b_{2,0,0} + s^2 b_{0,2,0} + t^2 b_{0,0,2} + 2rs b_{1,1,0} + 2rt b_{1,0,1} + 2st b_{0,1,1}
\]

Similarly, taking \( n = 3 \)

\[
p(r, s, t) = r^3 b_{3,0,0} + s^3 b_{0,3,0} + t^3 b_{0,0,3} + 3r^2 s b_{2,1,0} + 3r^2 t b_{2,0,1} + 3s^2 t b_{1,2,0} + 3s^2 t b_{0,2,1} + 3t^2 s b_{1,0,2} + 3t^2 b_{0,1,2} + 6rst b_{1,1,1}
\]

Figure 3.22 shows examples of two triangular patches and their corresponding Bernstein polynomials.

### 3.2.4 Continuity Conditions

Farin [98, 99] provides a comprehensive description of the conditions for derivative continuity on the common boundary between two adjacent triangular patches. Here we discuss only the cases relevant to achieve \( C^1 \) continuity. Figure 3.23 illustrates two polynomials \( p \) and \( q \) on the adjacent triangles \( T_1 \) and \( T_2 \).
Figure 3.22: (a) Quadratic and (b) cubic triangular Bernstein-Bézier patches and their corresponding Bernstein polynomials.

Expressing the polynomials \( p \) and \( q \) in the form of Equation 3.6, we obtain

\[
p(r_p, s_p, t_p) = \sum_{i+j+k=n} b_{i,j,k} \frac{n!}{i! j! k!} r_p^i s_p^j t_p^k
\]

\[
q(r_q, s_q, t_q) = \sum_{i+j+k=n} c_{i,j,k} \frac{n!}{i! j! k!} r_q^i s_q^j t_q^k
\]

Along the common edge

\[
t_p = 0 \quad t_q = 0
\]

\[
r_p = r_q \quad s_p = s_q
\]
Taking these conditions and writing the polynomials along the common edge

\[
\sum_{i+j=n} b_{i,j,0} \frac{n!}{i! j!} r_p^i s_p^j = \sum_{i+j=n} c_{i,j,0} \frac{n!}{i! j!} r_q^i s_q^j = 0
\]

therefore,

\[
b_{i,j,0} = c_{i,j,0} \quad \forall i + j = n
\]

This ensures that the two triangles have \(C^0\) continuity along their common edge. To ensure \(C^1\) continuity, the first derivatives of \(p\) and \(q\) must join continuously across the shared edge [6]. Let \(\partial p\) denote the differentiation in the direction of a vector in the plane (e.g. an edge of the triangle). Thus

\[
\partial p = \partial \sum_{i+j+k=n} b_{i,j,k} \frac{n!}{i! j! k!} r_p^i s_p^j t_p^k = \\
= \sum_{i+j+k=n} \frac{n!}{i! j! k!} b_{i,j,k} (i \partial r_p r_p^{i-1} s_p^j t_p^k + r_p^i j \partial s_p s_p^{j-1} t_p^k + r_p^i s_p^j k \partial t_p t_p^{k-1})
\]
Reorganizing the expression in Bernstein-Bézier form

\[
\partial \mathbf{p} = \sum_{i+j+k=n-1} \overline{b}_{i,j,k} \frac{(n-1)!}{i! j! k!} \partial_{p}^{i} \partial_{s}^{j} \partial_{t}^{k}
\]

where

\[
\overline{b}_{i,j,k} = n(b_{i+1,j,k} \partial r_{p} + b_{i,j+1,k} \partial s_{p} + b_{i,j,k+1} \partial t_{p})
\]

The differentiation with respect to \( q \) is defined similarly. Examining the coefficients, we obtain the condition

\[
c_{i,j,1} = r_{p} b_{i,j,1} + s_{p} b_{i+1,j,0} + t_{p} b_{i,j+1,0} \quad \forall \ i + j = n - 1
\]

Therefore, the two patches are \( C^1 \) continuous if all pairs of shaded subtriangles shown in Figure 3.24 are coplanar.

![Figure 3.24: Continuity conditions for triangular Bézier patches.](image)

3.2.5 Derivative Estimation

Our interpolation methods compute the coefficients of the polynomial for each triangle based only on the elevation values and the estimated values of the first partial derivatives (tangent vectors) at the three vertices of the triangle.

Several methods for estimating derivatives have been proposed in the literature [5,
181, 184, 227, 255, 299]. In our algorithms, the derivative at a vertex is computed as
the weighted average of tangents of the triangles adjacent to the vertex.

Given a data point \((x_i, y_i)\), the \(n\) triangles surrounding this vertex are found. Then, the cross product of any pair of sides from each triangle is computed. This three-dimensional vector is perpendicular to the triangle, and its length is twice the area of the triangle (see page 58 for more details). For example, the cross product vector \((x_p, y_p, z_p)\) for a triangle defined by the points \(P_1(x_1, y_1, z_1)\), \(P_2(x_2, y_2, z_2)\), and \(P_3(x_3, y_3, z_3)\) is given by

\[
\begin{align*}
    x_p &= (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \\
    y_p &= (z_2 - z_1)(x_3 - x_1) - (z_3 - z_1)(x_2 - x_1) \\
    z_p &= (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)
\end{align*}
\]

Since the triangle vertices are ordered in counterclockwise direction in our implementa-
tion, values \(z_p\) are always positive. This agrees with the fact that these values
represent a height component.

To estimate the derivative at a data point, the cross products are accumulated
for all triangles involving that point, and the resulting normals at the vertex are
averaged. The estimated gradient plane can be thought of as a spatial average of
slopes and sizes of the surrounding triangles [324].

### 3.2.6 Quadratic Interpolant

This section describes one approach to constructing \(C^1\) piecewise quadratic sur-
faces over the triangulation associated with the terrain data points. The method
is based on a scheme originally described by Powell and Sabin [248] and further
developed by Cendes and Wong [39].

The scheme subdivides each triangular element into six subtriangles to generate
smooth surfaces, as shown in Figure 3.25. The points \(P_2\), \(P_4\), and \(P_6\) are the vertices
of the triangle. The point \( P_0 \) is the inscribed center of the triangle and the points \( Q_0, Q_1, \) and \( Q_2 \) are the inscribed centers for the adjacent triangles. The six subtriangles are formed by connecting the point \( P_0 \) with the triangle vertices and with the three points \( Q_i \). Boundary triangles are subdivided into six by adding one point at an arbitrary location along each triangle boundary. In our algorithm, the midpoint of the edge on the boundary of the mesh is used.

![Figure 3.25: Powell-Sabin subdivision.](image)

The interpolation requires only function and derivatives values at each data vertex. The quadratic surface is calculated for each subtriangle by evaluating the six control points and the barycentric coordinates relative to the subtriangle. Then, for the subtriangle \( P_0 P_3 P_4 \), the \( C^1 \) quadratic surface can be represented as

\[
p(r, s, t) = r^2 \mathbf{b}_0 + s^2 \mathbf{b}_3 + t^2 \mathbf{b}_4 + 2rs \mathbf{b}_{15} + 2rt \mathbf{b}_{16} + 2st \mathbf{b}_9
\]

where \((r, s, t)\) are the barycentric coordinates of a point \((x, y)\) in the subtriangle and the six coefficients denote the data values at the control points.
As described in Section 3.2.4, \( C^1 \) continuity can be guaranteed between two adjacent triangle patches by requiring that each pair of adjacent subtriangles along the common edge be coplanar.

In order to satisfy the coplanarity conditions in the Bézier triangle scheme, we need to specify data values and derivatives at the 19 control points. The vertex values at \( P_2, P_4, \) and \( P_6 \) are already specified. The derivative values at the vertices, denoted as \( z_i^0 \) and \( z_i^1, i = 2, 4, 6, \) are estimated by using the technique described in Section 3.2.5.

Each of the six subtriangles \( i \) in Figure 3.25 is conveniently identified by the indices of the control points given by \( 0, 12 + i, 13 + i \mod 6, i, 6 + i, \) and \( 1 + i \mod 6, \) where \( 1 \leq i \leq 6. \)

The 19 Bézier control points \( b_i, 0 \leq i \leq 18, \) are defined as follows

\[
\begin{align*}
b_0 &= r \ b_{14} + s \ b_{16} + t \ b_{18} \\
b_1 &= \beta b_{12} + (1 - \beta) b_7 \\
b_2 &= z_2 \\
b_3 &= \gamma b_8 + (1 - \gamma) b_9 \\
b_4 &= z_4 \\
b_5 &= \alpha b_{10} + (1 - \alpha) b_{11} \\
b_6 &= z_6
\end{align*}
\]
\[ b_7 = b_2 + c_3 z_1^x + d_3 z_1^y \]
\[ b_8 = b_2 + c_4 z_1^x + d_4 z_1^y \]
\[ b_9 = b_4 + c_5 z_2^x + d_5 z_2^y \]
\[ b_{10} = b_4 + c_6 z_2^x + d_6 z_2^y \]
\[ b_{11} = b_6 + c_7 z_3^x + d_7 z_3^y \]
\[ b_{12} = b_6 + c_8 z_3^x + d_8 z_3^y \]
\[ b_{13} = \beta b_{18} + (1 - \beta) b_{14} \]
\[ b_{14} = b_2 + c_0 z_1^x + d_0 z_1^y \]
\[ b_{15} = \gamma b_{14} + (1 - \gamma) b_{16} \]
\[ b_{16} = b_4 + c_1 z_2^x + d_1 z_2^y \]
\[ b_{17} = \alpha b_{16} + (1 - \alpha) b_{18} \]
\[ b_{18} = b_6 + c_2 z_3^x + d_2 z_3^y \]

where the coefficients \( c_i, 0 \leq i \leq 8 \), are defined as

\[
\begin{align*}
  c_0 &= \frac{1}{2}(x_0 - x_2) \\
  c_2 &= \frac{1}{2}(x_0 - x_6) \\
  c_4 &= \frac{1-\gamma}{2}(x_4 - x_2) \\
  c_6 &= \frac{1-\alpha}{2}(x_6 - x_4) \\
  c_8 &= \frac{1-\beta}{2}(x_2 - x_6) \\
  c_1 &= \frac{1}{2}(x_0 - x_4) \\
  c_3 &= \frac{\beta}{2}(x_6 - x_2) \\
  c_5 &= \frac{\gamma}{2}(x_2 - x_4) \\
  c_7 &= \frac{\alpha}{2}(x_4 - x_6)
\end{align*}
\]

The coefficients \( d_i, 0 \leq i \leq 8 \), are defined similarly by substituting \( y_j \) for \( x_j \) in
the above equations. Finally, $\alpha$, $\beta$, and $\gamma$ represent the distance ratios defined as

$$
\alpha = \frac{P_6P_5}{P_5P_4} \quad \beta = \frac{P_2P_1}{P_2P_6} \quad \gamma = \frac{P_4P_3}{P_4P_2}
$$

where $P_iP_j$ is the distance between nodes $i$ and $j$.

### 3.2.7 Cubic Interpolant

The Clough-Tocher [53] interpolation scheme, originally developed as a technique in finite element analysis, was used to produce a piecewise cubic polynomial surface over the triangulation. For its construction, each triangle is subdivided at the centroid into three subtriangles, and a cubic Bernstein-Bézier polynomial is defined over each subtriangle. Figure 3.26 illustrates the Clough-Tocher interpolation scheme.

![Figure 3.26: Clough-Tocher subdivision.](image)

In a manner similar to the quadratic interpolation, data and derivative values at the vertices of the triangle are required to ensure $C^1$ continuity. Our interpolation method is based on the work developed by Quak and Schumaker [251]. Their paper provides a construction such that derivative continuity is achieved on each shared
triangle edge.

Initially, the derivatives at each data vertex are computed by estimating the cross product of the tangents for each adjacent triangle. The resulting derivatives at the vertex are averaged. Then, three cubic triangular Bézier patches are constructed over each subtriangle. A cubic Bézier patch is defined by 10 control points as shown in Figure 3.22(b) on page 64. The 10 control points of each subtriangle provide the degrees of freedom required to ensure continuity across the element boundaries.

The 19 Bernstein-Bézier coefficients are given by the following equations

\[
\begin{align*}
    b_1 & = z_1 \\
    b_2 & = z_2 \\
    b_3 & = z_3 \\
    b_4 & = ((x_2 - x_1)z_1^x + (y_2 - y_1)z_1^y)/3 + z_1 \\
    b_5 & = ((x_4 - x_1)z_1^x + (y_4 - y_1)z_1^y)/3 + z_1 \\
    b_6 & = ((x_3 - x_1)z_1^x + (y_3 - y_1)z_1^y)/3 + z_1 \\
    b_7 & = ((x_3 - x_2)z_1^x + (y_3 - y_2)z_1^y)/3 + z_2 \\
    b_8 & = ((x_4 - x_3)z_2^x + (y_4 - y_3)z_2^y)/3 + z_2 \\
    b_9 & = ((x_1 - x_2)z_2^x + (y_1 - y_2)z_2^y)/3 + z_2 \\
    b_{10} & = ((x_1 - x_3)z_3^x + (y_1 - y_3)z_3^y)/3 + z_3 \\
    b_{11} & = ((x_4 - x_3)z_3^x + (y_4 - y_3)z_3^y)/3 + z_3 \\
    b_{12} & = ((x_2 - x_3)z_3^x + (y_2 - y_3)z_3^y)/3 + z_3 \\
    b_{13} & = (b_5 + b_8 + (\theta_1 - 1)b_1 + (2 - 3\theta_1)b_4 + (3\theta_1 - 1)b_9 - \theta_1 b_2)/2 \\
    b_{14} & = (b_8 + b_{11} + (\theta_2 - 1)b_2 + (2 - 3\theta_2)b_7 + (3\theta_2 - 1)b_{12} - \theta_2 b_3)/2 \\
    b_{15} & = (b_{11} + b_5 + (\theta_3 - 1)b_3 + (2 - 3\theta_3)b_{10} + (3\theta_3 - 1)b_6 - \theta_3 b_1)/2 \\
    b_{16} & = (b_{15} + b_5 + b_{13})/3
\end{align*}
\]
\[ b_{17} = \frac{(b_{13} + b_8 + b_{14})}{3} \]
\[ b_{18} = \frac{(b_{14} + b_{11} + b_{15})}{3} \]
\[ b_{19} = \frac{(b_{18} + b_{16} + b_{17})}{3} \]

where
\[
\begin{align*}
\theta_1 &= \frac{(x_4 - x_1)(x_2 - x_1) + (y_4 - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
\theta_2 &= \frac{(x_4 - x_2)(x_3 - x_2) + (y_4 - y_2)(y_3 - y_2)}{(x_3 - x_2)^2 + (y_3 - y_2)^2} \\
\theta_3 &= \frac{(x_4 - x_3)(x_1 - x_3) + (y_4 - y_3)(y_1 - y_3)}{(x_1 - x_3)^2 + (y_1 - y_3)^2}
\end{align*}
\]

The Bernstein-Bézier representation with respect to the subtriangle \( P_0P_1P_{18} \) is
\[
p(r, s, t) = r^3 b_{18} + s^3 b_0 + t^3 b_1 + 3r^2s b_{15} + 3r^2t b_6 + 3rs^2 b_4 + 3s^2t b_3 + 3rt^2 b_7 + 3st^2 b_8 + 6rst b_{12}
\]

where \((r, s, t)\) are the barycentric coordinates of a point \((x, y)\) relative to the subtriangle. The \( C^1 \) cubic surface for subtriangles \( P_1P_2P_{18} \) and \( P_0P_2P_{18} \) are computed analogously.
CHAPTER 4
ACCURACY ASSESSMENT

The accuracy with which a model is assumed to represent terrain surface is an important issue. In order to assess the fidelity of a model, it is important to recognize the type and implications of errors that are inherent in the use of geographic information data. Several factors can contribute to the occurrence of errors. A major source of errors is due to the process used to acquire the real surface. This occurs during the preparation of input data, where errors are caused by human limitations, instrumental imperfections, and environmental effects. A second important source of error is the data processing and transformation. When data are transformed through a set of operations, new errors or uncertainty may be introduced into the derivative products.

This chapter initially discusses the type and accuracy of the data used in the generation of our models. Several accuracy measurements are then described and evaluated in detail.

4.1 USGS Digital Elevation Models

The United States Geological Survey (USGS) provides public access to a number of digital cartographic data. Digital elevation models (DEMs) correspond to one of the categories of data for which certain standards have been established. The USGS DEMs are currently available in three classes [314]:

- **large scale**: these data include the 7.5- and 15-minute DEMs. The 7.5-minute units correspond to the USGS 1:24,000 and 1:25,000 scale topographic quadrangle map series for all of the United States and its territories. Each 7.5-minute DEM is based on 30- by 30-meter data spacing in the Universal Transverse Mer-
cator (UTM) projection coordinate system. The 15-minute DEM data consist of a regular array of elevations referenced horizontally to the geographic (latitude/longitude) coordinate system of the North American 1927 Datum (NAD 27) or the North American 1983 Datum (NAD 83). The spacing between elevations is 2 arc seconds of latitude by 3 arc seconds of longitude. The 7.5 minute DEM data are currently derived from digital line graph\(^1\) (DLG) data by incorporating selected elements from hypsography (contours, spot elevations) and hydrography (shorelines, lakes, drainage). The data for both 7.5- and 15-minute DEMs are ordered from south to north in profiles that are ordered from west to east. Elevation units are in meters or feet relative to National Geodetic Vertical Datum of 1929 (NGVD 29).

- **intermediate scale:** these data correspond to the 30-minute DEMs. The spacing between elevations is 2- by 2-arc seconds referenced horizontally to the geographic coordinate system of the World Geodetic System 1972 (WGS 72). The 30-minute DEM is distributed as four 15-minute DEM units, corresponding to one half of a 1:100,000 scale map. The USGS uses two processes to collect the digital elevation data, either by converting 1:24,000-scale and 1:100,000-scale hypsography DLG data or by resampling scanned raster files of USGS 1:24,000-scale or 1:100,000-scale map series. The elevation data are ordered from south to north in profiles that are ordered from west to east. Elevation units are in meters or feet relative to NGVD 29.

- **small scale:** these data correspond to the 1-degree DEMs. The spacing between elevations is 3- by 3- arc seconds referenced horizontally to the geographic system of NAD 27 or NAD 83. The majority of the 1-degree DEMs are produced by the Defense Mapping Agency (DMA) from photographic and

\(^1\)Digital line graph (DLG) is a digital representation of topographic and planimetric map features such as transportation, hydrography, and boundaries.
cartographic sources. Elevation data from photographic sources are collected by using manual and automated correlation techniques. Elevation data from cartographic sources are normally collected from 1:250,000-scale map series. Elevations are in meters relative to NGVD 29.

4.2 Accuracy of USGS DEMs

Many existing digital elevation data are derived from topographic maps, consequently, their accuracy depends on the quality of the original source of the data.

The accuracy of most USGS DEMs is computed by comparison of interpolated elevations in the DEM with corresponding true elevations that include points on contour lines, field control, or spot elevations [314]. However, since these known elevations may also contain errors, it is apparent that DEMs in general have inherent inaccuracies in their constituent data, requiring that users "become aware of the nature and the types of errors" [35].

According to Caruso [36], a factor influencing the accuracy of a DEM is its horizontal and vertical dimension. **Horizontal accuracy** of DEM data is dependent upon the units of resolution of the elevation array. Most terrain features, within a standard DEM, are generalized by being reduced to grid nodes regularly spaced in the horizontal plane, reducing the ability to recover feature positions less than the grid spacing and resulting in a filtering or smoothing of the surface during gridding. On the other hand, **vertical accuracy** of the DEM data is dependent upon the horizontal grid spacing, quality of the source data, collection and processing procedures, and digitizing systems.

DEM errors are classified into three categories [36, 308]:

- **Blunders**: errors typically of major proportions, exceeding the maximum error permitted for each DEM level and, therefore, easily identifiable and removed during interactive editing.
**systematic**: errors that are introduced by data collection procedures or systems, typically predictable but not easily correctable. These error artifacts include, for instance, vertical elevation shifts, fictitious ridges, misinterpretation of terrain surface due to trees, buildings or shadows.

**random**: these errors result from unknown or accidental causes.

The USGS provides most of its elevation models with root mean square error (RMSE), globally indicating accuracy between the interpolated elevations in the DEM and corresponding true elevations from the published maps. A collection of test points is used as representative of the terrain.

The quality of the USGS DEM data is classified into three levels:

- **Level 1**: A vertical RMSE of 7 meters is the desired accuracy standard. An RMSE of 15 meters is the maximum permitted. All 2-arc second DEMs (intermediate scale) derived for 7.5-minute DEMs are categorized as level 1. The majority of the 7.5-minute DEMs (large scale) are categorized as level 1.

- **Level 2**: An RMSE of one-half contour interval is the maximum permitted. DEM data derived from hypsographic and hydrographic information are categorized as level 2. All 2-arc-second DEMs (intermediate scale) from contours are level 2.

- **Level 3**: An RMSE of one-third of the contour interval is the maximum permitted. DEMs derived from DLG data by incorporating selected elements from both hypsography and hydrography. All 1-degree DEMs (small scale) are categorized as level 3.

Additional information on DEM specifications can be found in the USGS National Mapping Program’s Standards for Digital Elevation Models [316].
4.3 Accuracy Measures

This section describes several accuracy measures which may be useful in assessing
the degree of fidelity between the original surface and the approximate surface at a
level of resolution. Although quantitative measures can provide important informa-
tion regarding the accuracy of the surface, it is often desirable to use qualitative
analysis in order to help detect errors which may not be easily identified by quanti-
tative measures.

4.3.1 Quantitative Measures

Quantitative measures are generally obtained by means of statistical analysis of
spatial data, allowing the identification and description of relationships present in
the data. Geostatistics is sometimes referred to as a set of techniques for the spatial
analysis of data. Typical measurements are based on mean, standard deviation,
correlation, and regression.

4.3.1.1 Root Mean Square Error (RMSE)

A common measure for describing accuracy between two data sets is the root
mean square error (RMSE):

\[
RMSE = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (z_{ij} - h_{ij})^2}
\]

where \( z_{ij} \) and \( h_{ij} \) are two corresponding elevation values, \( n \) and \( m \) are the grid
dimensions.

The larger the RMSE, the greater the difference between the two sets of data val-
ues. However, since the RMSE provides a single global measure of deviation, spatial
variation in error over the DEM surface is not revealed by such measure. Wood [329]
suggests a dimensionless accuracy ratio in order to reduce effects of relative relief
from measurement deviation, where the RMSE is divided by the standard deviation
of elevation values.

4.3.1.2 Surface Curvature

In differential geometry [79], the estimation of surface curvature is normally based on the curves that lie in the surface. These curves are determined by the intersection between the surface and planes perpendicular to the tangent plane at a point on the surface (Figure 4.1). The curvatures of these resulting curves are called normal curvatures at the point. The minimum and maximum normal curvatures, \( k_1 \) and \( k_2 \), are called the principal curvatures. The Gaussian curvature is the product of the principal curvatures

\[
K = k_1 k_2
\]

and the mean curvature is the average of the principal curvatures

\[
H = \frac{k_1 + k_2}{2}
\]

![Figure 4.1: Surface curvature.](image)

Based on the combinations of signs for \( K \) and \( H \), the local shape of the surface at each point can be classified as eight types according to Table 4.1.

The concept of curvature is significant in digital image processing. For instance, edges [149] and corners [261, 264] are regions of high surface curvature. Surface
<table>
<thead>
<tr>
<th></th>
<th>$K &lt; 0$</th>
<th>$K = 0$</th>
<th>$K &gt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H &lt; 0$</td>
<td>peak</td>
<td>ridge</td>
<td>saddle ridge</td>
</tr>
<tr>
<td>$H = 0$</td>
<td>-</td>
<td>flat</td>
<td>minimal</td>
</tr>
<tr>
<td>$H &gt; 0$</td>
<td>pit</td>
<td>valley</td>
<td>saddle valley</td>
</tr>
</tbody>
</table>

Table 4.1: Surface types defined by the signs of the mean and Gaussian curvatures.

curvature is also related to extraction of topographic structures such as peaks, pits, ridges, and valleys [148, 171, 310] in digital images.

A local quadratic trend surface to fit the surface of the terrain over a $3 \times 3$ subgrid is proposed by Evans [93]

$$z = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2$$ (4.1)

The coefficients are estimated using the method of least squares, which is greatly simplified for the $3 \times 3$ neighborhood. The expressions for the six coefficients are given as functions of the neighborhood cells as follows

$$a_{00} = \frac{2(zi_{i,j} + zi,j-1 + zi,j+1 + zi+1,j) - (zi_{i-1,j-1} + zi_{i-1,j+1} + zi_{i+1,j-1} + zi_{i+1,j+1}) + 5zi_{i,j}}{9}$$

$$a_{10} = \frac{(zi_{i-1,j} + zi_{i,j+1} + zi_{i+1,j+1}) - (zi_{i-1,j-1} + zi_{i,j-1} + zi_{i+1,j-1})}{6\Delta}$$

$$a_{01} = \frac{(zi_{i-1,j-1} + zi_{i,j} + zi_{i-1,j+1}) - (zi_{i+1,j-1} + zi_{i,j} + zi_{i+1,j+1})}{6\Delta}$$

$$a_{11} = \frac{(zi_{i-1,j+1} + zi_{i,j+1}) - (zi_{i-1,j-1} + zi_{i,j+1})}{4\Delta^2}$$

$$a_{20} = \frac{(zi_{i-1,j-1} + zi_{i,j-1} + zi_{i,j+1} + zi_{i+1,j-1} + zi_{i+1,j-1} + zi_{i+1,j+1})}{6\Delta^2} - \frac{(zi_{i-1,j} + zi_{i,j} + zi_{i+1,j})}{3\Delta^2}$$

$$a_{02} = \frac{(zi_{i-1,j-1} + zi_{i,j-1} + zi_{i,j+1} + zi_{i+1,j-1} + zi_{i+1,j} + zi_{i+1,j+1})}{6\Delta^2} - \frac{(zi_{i,j-1} + zi_{i,j} + zi_{i,j+1})}{3\Delta^2}$$

where $\Delta$ is the grid spacing.

A simple expression for estimating the curvature of the surface is the Laplacian [234]:

$$\eta = \frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}$$

80
Taking Equation 4.1, the second derivatives are
\[
\frac{\partial^2 z}{\partial x^2} = 2a_{20} \quad \frac{\partial^2 z}{\partial y^2} = 2a_{02}
\]
therefore,
\[
\eta = 2(a_{20} + a_{02})
\]

A measure of surface smoothness is proposed by Briggs [28]:
\[
C = \sum_{i=1}^{m} \sum_{j=1}^{n} (C_{ij})^2
\]
where \(C_{ij}\) is the curvature at the point \((x_i, y_j)\). The curvature \(C_{ij}\) is a function of the point \((x_i, y_j)\) and some neighboring point values.

4.3.1.3 Spatial Autocorrelation

Spatial autocorrelation is a measure of the degree of spatial association between data values. In the case of DEMs, each cell has an attribute which is a measure of the elevation at its location. A DEM is considered highly spatially autocorrelated if its adjacent cells have similar elevation measures.

An autocorrelation function can be described as
\[
A = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} c_{ij}
\]
where \(w_{ij}\) is a measure of spatial proximity and \(c_{ij}\) is a measure of attribute similarity.

Spatial autocorrelation is commonly measured using Geary’s ratio and Moran’s
index. Geary’s ratio is defined as

\[ G = \frac{(n - 1)}{2} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (z_i - z_j)^2}{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \sum_{i=1}^{n} (z_i - \bar{z})^2} \]

where \( z_i \) and \( z_j \) are elevations of cell \( i \) and cell \( j \), and \( \bar{z} \) is the average elevation of all cells. The spatial weight \( w_{ij} \) is usually assigned 1 to the cells in the horizontal and vertical directions, and \( \sqrt{2} \) to the diagonal neighboring cells. The computation of Geary’s ratio results in a value within the range of 0 and 2. A value of 1 indicates no spatial autocorrelation (random distribution), 0 indicates a strong positive spatial autocorrelation (clustering of values), and 2 indicates a strong negative spatial autocorrelation (neighboring data points have dissimilar values).

On the other hand, Moran’s index is defined as follows

\[ I = n \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (z_i - \bar{z})(z_j - \bar{z})}{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \sum_{i=1}^{n} (z_i - \bar{z})^2} \]

Moran’s index is similar to the conventional correlation coefficients, where a value of 1 indicates positive autocorrelation, 0 indicates no spatial autocorrelation, and -1 negative autocorrelation.

### 4.3.1.4 Texture Descriptors

Textural characteristics of geophysical images can be important to distinguish surface types based on roughness. One of the simplest measures of texture is the local variance

\[ \text{Variance} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (z_{ij} - \bar{z})^2 \]
When the variance increases, the degree of roughness also increases.

Another measure of texture that can be used to characterize surfaces is the local entropy, defined as

\[
\text{Entropy} = - \sum_{i=1}^{k} (p_i \log p_i)
\]

where \( p_i \) is the proportion of the cells in the neighborhood with value \( i \), and \( k \) is the number of pixels in the neighborhood. Figure 4.2 shows the calculation of entropy for two 3×3 neighborhoods. As the entropy increases, the roughness index also increases.

\[
\begin{array}{|c|c|c|}
\hline
i & p_i & p_i \log p_i \\
\hline
1 & 6/9 & -0.1174 \\
2 & 3/9 & -0.1590 \\
\hline
\end{array}
\]

Entropy = 0.2764

\[
\begin{array}{|c|c|c|}
\hline
i & p_i & p_i \log p_i \\
\hline
1 & 3/9 & -0.1590 \\
2 & 2/9 & -0.1452 \\
3 & 2/9 & -0.1452 \\
4 & 1/9 & -0.1060 \\
6 & 1/9 & -0.1060 \\
\hline
\end{array}
\]

Entropy = 0.6614

Figure 4.2: Calculation of entropy in a 3×3 window.

Cooccurrence matrices [150] describe the spatial arrangement of local features within an image. Although being primarily developed as texture descriptors based on gray-scale images, cooccurrence matrices can also be applied to DEMs.

Given an image quantized into \( k \) classes of cell attributes, the cooccurrence matrix \( C \) consists of \( k \times k \) cells, where \( C(x, y) \) is an estimate of the probability that the attribute pair \((x, y)\) will be found at a pair of image pixels separated by a displacement \( \Delta = (\Delta x, \Delta y) \). Such displacement has traditionally been defined along the horizontal and vertical directions. Highly positively autocorrelated surfaces will have most of elements concentrated along the main diagonal of the matrix, since it represents the probability that two adjacent pixels belong to the same class.
4.3.1.5 Volume

Given a terrain surface defined by a set of triangular facets, the volume under the surface can be determined by summing the volumes of each triangular prism. Assuming that the base of the prism lies on the plane \( z = 0 \), the volume of a prism is

\[
Volume = \left| \frac{(z_1 + z_2 + z_3) A_\Delta}{3} \right|
\]

where \( z_1, z_2, \) and \( z_3 \) are the elevations of each point of the triangle, and \( A_\Delta \) is the projected area of the triangle onto the plane \( z = 0 \).

4.3.1.6 Slope and Aspect

Slope and aspect provide important information for a variety of environmental applications, such as choice of planting sites [288], erosion and depositing modeling [259], accessibility and harvesting methods [288], solar energy research [259], and avalanche prevention.

Slope is generally described as a measure of change in surface value over distance, expressed in degrees or as a percentage. Mathematically, slope is referred to as the first derivative of the surface. Aspect is defined as the compass direction in which the surface is facing [266], usually measured in degrees from North in a clockwise direction. Most of the methods for calculating slope and aspect from DEMs are based on neighboring elevation values in a \( 3 \times 3 \) window that is moved over the grid.

The slope is usually defined as

\[
Slope = \arctan \sqrt{\left( \frac{\partial z}{\partial x} \right)^2 + \left( \frac{\partial z}{\partial y} \right)^2}
\]

(4.2)

whereas aspect is defined as

\[
Aspect = \arctan \left( \frac{\partial z/\partial y}{\partial z/\partial x} \right)
\]

(4.3)
The simplest way to estimate the partial derivatives employs finite difference methods using either four or eight of neighboring cell values within a window centered on the cell in question. A second-order finite difference method is proposed by Fleming and Hoffer [107]

\[
\begin{align*}
\frac{\partial z}{\partial x} &= \frac{z_{i+1,j} - z_{i-1,j}}{2\Delta x} \\
\frac{\partial z}{\partial y} &= \frac{z_{i,j+1} - z_{i,j-1}}{2\Delta y}
\end{align*}
\]

where \( \Delta x \) and \( \Delta y \) are the spacing between points in the horizontal and vertical direction, respectively.

A third-order finite difference method was proposed by Sharpnack and Akin [288], where the slope is defined as

\[
\begin{align*}
\frac{\partial z}{\partial x} &= \frac{(z_{i+1,j+1} + z_{i+1,j} + z_{i+1,j-1}) - (z_{i-1,j+1} + z_{i-1,j} + z_{i-1,j-1})}{6\Delta x} \\
\frac{\partial z}{\partial y} &= \frac{(z_{i+1,j+1} + z_{i,j+1} + z_{i-1,j+1}) - (z_{i+1,j-1} + z_{i,j-1} + z_{i-1,j-1})}{6\Delta y}
\end{align*}
\]

Horn [163] presented a modified version of Sharpnack and Akin’s method using unequal weights for the closer elevation values

\[
\begin{align*}
\frac{\partial z}{\partial x} &= \frac{(z_{i+1,j+1} + 2z_{i+1,j} + z_{i+1,j-1}) - (z_{i-1,j+1} + 2z_{i-1,j} + z_{i-1,j-1})}{8\Delta x} \\
\frac{\partial z}{\partial y} &= \frac{(z_{i+1,j+1} + 2z_{i,j+1} + z_{i-1,j+1}) - (z_{i+1,j-1} + 2z_{i,j-1} + z_{i-1,j-1})}{8\Delta y}
\end{align*}
\]

(4.4)

Skidmore [292] and Hodgson [159] compare several methods, including the above mentioned, suggesting that the one proposed by Horn results in good accuracy without requiring extensive CPU time.
4.3.1.7 Drainage Networks

The drainage network for a section of terrain is a map of its main rivers and streams. One of the first methods for determining drainage networks was proposed by Peucker and Douglas [241]. Their algorithm flags the highest elevation cell in a 2×2 window, which is moved over the DEM. The unflagged cells remaining after this process form the drainage network. The maps derived by this approach, however, usually generates many interrupted channel segments and isolated points.

A set of thinning and joining procedures is proposed by Band [13], where channels derived using the previous algorithm determine connected networks and watershed boundaries. O’Callaghan and Mark [228], and Jenson and Domingue [170] propose a method based on quantifying the drainage accumulation at each cell of the DEM. Water on a grid cell is allowed to flow to one of its eight nearest neighbors, as shown in Figure 4.3(a). Initially, a drainage direction matrix is computed in order to store the flow direction for each cell. The slope relative to the central cell and each adjacent cell within a 3×3 window is calculated (elevation values for diagonal neighbors are divided by √2), and a direction code indicating the steepest descent is assigned to the central cell.

![Figure 4.3: Drainage network. (a) eight nearest neighbors; (b) DEM; (c) flow direction; (d) flow path.](image)

If cells have no neighbors at a lower elevation, known as *pits*, then no downslope flow path can be assigned to a neighbor cell. Pits are often due to errors during DEM generation and truncation of interpolated values on output [94, 207, 228].
True pits or closed depressions are rare in natural Earth topography, restricted to a few special geomorphic environments, such as deserts, glaciated terrain, and karst topography. Several authors propose methods for eliminating pits, most of them are based on local filtering or smoothing operation. O'Callaghan and Mark [228], and Jenson and Domingue [170] propose methods based on filling operation. In flat areas, where neighboring cells have the same elevation, the flow direction can be calculated using a combination of neighborhood techniques and iterative region growing procedures [170, 210].

A drainage accumulation matrix is used to store the total accumulated drainage at each point (Figure 4.3(d)). In order to model different drainage related applications, a weight matrix can be defined. Examples of different weight matrices include [207]:
(a) if each element of the weight matrix is set to one, then each cell of the drainage accumulation matrix will represent the number of cells that flow into it; (b) if each element of the weight matrix is set to the cell area, then the resulting values will give the total contributing drainage area; (c) if only one cell of the weight matrix is set to one (for instance, the cell having the steepest descent), with others set to zero, then each cell of the drainage accumulation matrix with a value equal to one will represent the path from that cell to its outlet. Once the drainage accumulation is calculated, a specified threshold is applied to produce a channel network. The choice of the threshold value affects the drainage density. The smaller the threshold values, the more detailed channel network will be generated. An example of (synthetic) drainage networks derived from the Crater Lake DEM is shown in Figure 4.4.

Most of the methods mentioned above assign flow from each cell to only one of its neighbors (Figure 4.5(a)). A simple strategy is to choose the neighbor located in the direction of steepest descent within a 3×3 window. Multiple flow direction methods have also been suggested as an attempt to solve certain limitations arising from the discretization of flow into only one of eight possible directions, separated by
45°. Quinn et al. [252] and Freeman [119] propose a representation where flow from a pixel is distributed among all of its lower elevation neighbors, proportionally to their slope (Figure 4.5(b)). Although multiple direction methods require additional costs in terms of computation, they can produce superior results for terrains where flow is typically divergent. A method that models flow path routing in two dimensions is described by Costa-Cabral and Burges [56], where the dispersal area of each pixel is calculated by constructing a set of flow tubes.

![Figure 4.4: Drainage networks derived from the Crater Lake DEM.](image)

![Figure 4.5: (a) Single flow direction; (b) multiple flow direction.](image)
4.3.1.8 Visibility

The determination of visibility information on digital terrain models has several practical applications, such as navigation [215, 306], siting of television, radio, and cellular telephone transmitters and receivers [189], siting of observation towers for monitoring forest fires [189, 311], scenic landscape assessment [105], shading applications [33, 211, 300], and military surveillance [254]. Visibility analysis uses elevation data to determine the regions that are visible (viewshed) from a particular location in the terrain.

A number of methods for computing terrain visibility have been proposed in the literature. The choice of the terrain representation in general affects the visibility computation. The algorithms usually operate on grid-based DEMs [191, 217, 295, 321] or TINs [65, 71]. Computing visibility information from terrain data is usually very time-consuming, particularly when the size and density of data increase. Due to the improved capabilities for collecting and distributing data and due to the need for higher accuracy, careful design of models and algorithms is necessary in order to make tasks more computationally tractable.

A survey describing useful applications, algorithms for computing visibility, and a list of references is given by Nagy [225]. De Floriani et al. [63, 64] discuss the use of visibility algorithms applied to gray-scale images and topographic surfaces.

Most algorithms calculate visibility comparing elevation values between the path connecting the observer and the target point. If this path, known as line of sight, is blocked by any intermediate point, then the target is not visible from the viewpoint (Figure 4.6).

The concept of visibility index is used as a measure for the area of visible regions on a terrain, indicating the number of data points that are visible from each point of the terrain [117, 317].

Some other issues of visibility on terrains include the impact of DEM error on
visibility calculation [101, 104, 233], networks of interconnected sites [67, 71], moving points of view [19], optimal site location [49, 55, 63, 287], and parallel computation [77, 217, 250, 305, 306].

4.3.1.9 Hausdorff Distance

The Hausdorff distance, named after Felix Hausdorff (1868-1942), is a measure defined between two point sets at fixed positions. Given two point sets $A$ and $B$, the Hausdorff distance is defined as

$$H(A, B) = \max[h(A, B), h(B, A)]$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

and $\|a - b\|$ refers to some measure of distance between two points $a$ and $b$, for instance, the $L_2$ (or Euclidean) norm.

The function $h(A, B)$ is called the directed Hausdorff distance [166, 267] from $A$ and $B$. The intuitive idea is that each point of $A$ must be within distance $h(A, B)$ of some point $B$, and there is some point of $A$ that is exactly distance $h(A, B)$ from the nearest point of $B$.

The Hausdorff distance can be used as an appropriate measure of the approximation error between the original and the simplified meshes. Given a point $p$ and a
surface $S$, the distance $d(p, S)$ is defined as

$$d(p, S) = \min_{q \in S} d_E(p, q)$$

where $d_E$ is the Euclidean distance between two points in $\mathbb{R}^3$. The directed Hausdorff distance between two surfaces $S_1$ and $S_2$ is then defined as

$$h(S_1, S_2) = \max_{p \in S_1} d(p, S_2)$$

Since there exist surfaces such that $h(S_1, S_2) \neq h(S_2, S_1)$, the Hausdorff distance can be obtained by taking the maximum of $h(S_1, S_2)$ and $h(S_2, S_1)$, that is, $H(S_1, S_2) = \max(h(S_1, S_2), h(S_2, S_1))$. Therefore, if the Hausdorff distance between the original surface $S_1$ and a simplified mesh $S_2$ is less than a predefined error tolerance $\epsilon$, then

$$\forall p \in S_1, \exists q \in S_2 \mid d(p, q) < \epsilon$$

and

$$\forall q \in S_2, \exists p \in S_1 \mid d(p, q) < \epsilon$$

A practical approach to evaluating the difference between the two surfaces is to adopt a surface sampling, where the surface of the first mesh is sampled, and the distance to the second mesh is computed for each face. A possible idea for the sampling step is to generate a number of random points in the interior of each face, with the number of samples proportional to the facet area. This approach can also be used to iteratively evaluate the approximation error between two meshes for each refinement or simplification step, where only a small (local) region of the triangulation changes in each step.
4.3.2 Qualitative Measures

Visualization techniques traditionally provide an effective way of understanding complex spatial relationships. Maps, photographs, histograms, 3-D perspective diagrams, variograms are some example of pictorial representations that provide visual appreciation of spatial data.

This section examines how some visual representations can be used as suitable tools for analyzing elevation data accuracy.

4.3.2.1 Elevation Map

This representation associates elevation classes with uniform colors or patterns. The purpose is to give the appearance of thick colored contours filling the surface, giving a general impression of topography. In cartography, these maps, called choropleth maps, are commonly used to specify spatial divisions of a surface according to a particular variable, such as population density, use of land, or type of forest cover.

However, although some differences can be seen between the surfaces, it is difficult to make a valued judgment about the data quality of each.

Figure 4.7 shows an elevation map for the Crater Lake DEM.

4.3.2.2 Contour Map

A contour map is probably the most common way of representing topography. In such representation, contour lines are constructed by connecting points on a terrain surface that have the same elevation. Contour lines can be visualized as the intersection of the terrain surface with a sequence of equally spaced, horizontal planes that pass through this surface. The vertical distance separating these planes is known as contour interval.

Figure 4.8 shows a contour map for Crater Lake DEM.
4.3.2.3 Shaded Relief Map

Hill-shading techniques provide a way of understanding topography by means of shaded relief maps. Such maps are commonly used to visually enhance terrain features by simulating the appearance of the effects of sunlight across a terrain surface.

A variety of hill-shading algorithms is available [163, 238, 322], most of them use a local neighborhood to indicate variations in elevation. A convenient representation for the apparent brightness of a surface element is the reflectance map [163], which gives the distribution of radiance as a function of the orientation of the surface element. A surface that has equal radiance when viewed from different angles is known as Lambertian surface. In this special case where the surface reflects all incident light and appears equally bright from all directions, the radiance is proportional to the cosine of the incident angle.

Figure 4.9 illustrates a surface element $S$ at $(x, y, z)$ with surface normal $\mathbf{n}$ illu-
minated by an infinitely distant light source from direction \( \mathbf{s} \) and observed from the 
\( z \)-axis. The orientation of the surface element \( S \) can be described in terms of its slope 
\( p \) in the \( x \) (west-to-east) direction and its slope \( q \) in the \( y \) (south-to-north) direction. 
The slopes \( p = \partial z / \partial x \) and \( q = \partial z / \partial y \), which can be calculated by Equation 4.4 on 
page 85, are the components of the gradient vector \( (p, q) \).

The surface normal is a vector perpendicular to the local tangent plane. Since the 
vectors \( (1, 0, p)^T \) and \( (1, 0, q)^T \) are tangent to the surface, their cross-product defines 
the surface normal

\[
\mathbf{n} = (1, 0, p)^T \times (0, 1, q)^T = (-p, -q, 1)^T
\]

A unit surface normal \( \hat{\mathbf{n}} \) can be obtained by dividing the vector \( \mathbf{n} \) by its magni-
tude, that is
\[
\hat{n} = \frac{(-p, -q, 1)^T}{\sqrt{1 + p^2 + q^2}}
\]

In the spherical coordinate system shown in Figure 4.9(b), the azimuth angle $\phi$ is measured from the $x$-axis in the plane perpendicular to the $z$-axis, and the zenith angle $\theta$ is measured from the $z$-axis. The direction of the light source can also be written as a function of the variables $p$ and $q$. In a similar manner to the surface normal, the vector $(-p_s, -q_s, 1)^T$ denotes the direction of the light source.

A surface element will be maximally illuminated when the surface normal points at the light source. The azimuth and zenith angle of the light source are then

\[
\sin \phi_s = \frac{-q_s}{\sqrt{p_s^2 + q_s^2}} \quad \text{and} \quad \cos \phi_s = \frac{-p_s}{\sqrt{p_s^2 + q_s^2}}
\]

while

\[
\sin \theta_s = \frac{\sqrt{p_s^2 + q_s^2}}{\sqrt{1 + p_s^2 + q_s^2}} \quad \text{and} \quad \cos \theta_s = \frac{1}{\sqrt{1 + p_s^2 + q_s^2}}
\]

Therefore,

\[
p_s = -\cos \phi_s \tan \theta_s \quad \text{and} \quad q_s = -\sin \phi_s \tan \theta_s
\]
As mentioned above, the reflectance map \( R(p,q) \) for a Lambertian surface is proportional to the cosine of the incident angle, can be derived using a normalized dot product of the unit vector normal \((-p, -q, 1)\) and the unit illumination vector \((-p_s, -q_s, 1)\), given by

\[
R(p,q) = \frac{(-p, -q, 1)}{\sqrt{1 + p^2 + q^2}} \cdot \frac{(-p_s, -q_s, 1)}{\sqrt{1 + p_s^2 + q_s^2}} = \frac{1 + pp_s + qq_s}{\sqrt{1 + p^2 + q^2} \sqrt{1 + p_s^2 + q_s^2}}
\]

In cartography, the light source is typically assumed to be in northwest position \( (\phi_s = 135^\circ) \), with a zenith \( \theta_s \) of \( 45^\circ \). In this particular case, the reflectance map becomes

\[
R(p,q) = \frac{1 + (p - q)/\sqrt{2}}{\sqrt{2} \sqrt{1 + p^2 + q^2}}
\]

Figure 4.10 on page 96 shows a shaded image of the Lake Champlain calculated by using Equation 4.5.

Figure 4.10: Shaded relief map for Lake Champlain.
4.3.2.4 Slope and Aspect Map

Slope and aspect are two important measures for characterizing topographic surfaces. Among other operations, their values can be used to indicate magnitude of local variation in elevation and also as a diagnostic tool for assessing the impact of errors when operations are applied to elevation data.

Slope and aspect can be calculated using Equations 4.2 and 4.3, respectively, on page 84. Examples of slope and aspect maps for Lake Champlain are illustrated in Figures 4.11 and 4.12, respectively.

Figure 4.11: Slope map for Lake Champlain.
Figure 4.12: Aspect map for Lake Champlain.
CHAPTER 5

RESULTS

5.1 Terrain Data Sets

We have used several digital elevation models for our tests. The data sets represent different terrain types including flat regions, hills, plateaus, and mountains. Most of these data sets are publicly available from the U.S. Geological Survey, which are originally encoded in a regular grid. Elevations are represented by integer values.

In particular, a sample of 24 random USGS DEMs [315] formed by the first available DEMs starting with each letter of the alphabet (except X and Z) was used. The data files, which correspond to the east half of the map, are Aberdeen, Baker, Caliente, Dalhart, Eagle Pass, Fairmont, Gadsden, Hailey, Idaho Falls, Jackson, Kalispell, La Crosse, Macon, Nashville, O’Neill, Paducah, Quebec, Racine, Sacramento, Tallahassee, Ukiah, Valdosta, Waco, and Yakima. These DEMs, shown in Figure 5.1, are derived from 1:250,000-scale maps, providing coverage in 1- by 1-degree blocks. The samples consist of 1,201×1,201 points and the spacing of the elevations along each profile is 3 arc-seconds\(^1\) (90 meters).

Three other terrain data sets were used to test our algorithms. The USGS Lake Champlain West DEM, shown in Figure 5.2, contains elevations with range between 15m and 1591m, standard deviation of 247.8m, and 3- by 3-arc-second data spacing. The Crater Lake West DEM, shown in Figure 5.3, contains elevations with range between 1533m and 2478m, standard deviation of 162.6m, and 30- by 30-meter data spacing. A 964×960 Martian terrain was also used to evaluate our method. The elevations range between 1983m and 3500m, standard deviation of 218.3, resolution

\(^1\)arc second: 1/3600th of a degree (1 second) of latitude or longitude.
Figure 5.1: Sample of 24 random USGS DEMs.
Figure 5.2: The USGS Lake Champlain West DEM.

Figure 5.3: The USGS Crater Lake West DEM.
of 0.93 Km per point. The data set, shown in Figure 5.4, is part of the Planetary Data System CD-ROM volume MG-2007, *Mars Mosaicked Digital Image Model (MDIM) and Digital Terrain Model (DTM), version 2.0*. This CD-ROM is available from the National Space Science Data Center.

### 5.2 Accuracy

Some results for the sample of 24 random DEMs are summarized in Table 5.1. The number of triangles, number of points, and RMS error for each terrain are indicated. Although less than 16% of the original points are used to approximate all terrain data sets, the models still provide an accurate representation of the terrain surface, that is, approximations with RMS error smaller than 1 meter.

Note that several terrain data are approximated by a reduced number of points (less than 2% of the original data points). This illustrates the adaptive characteristic of the triangular meshes, whose structure reflects the irregular density of the data.
Large, nearly planar regions are therefore approximated by large triangles, whereas regions of high curvature are approximated by small triangles.

<table>
<thead>
<tr>
<th>DEM File</th>
<th>Elevation Values (meters)</th>
<th>% of Points</th>
<th>Number of Triangles</th>
<th>RMS Error (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aberdeen E</td>
<td>379</td>
<td>683</td>
<td>420.54</td>
<td>36.48</td>
</tr>
<tr>
<td>Baker E</td>
<td>546</td>
<td>2,521</td>
<td>1,260.90</td>
<td>376.94</td>
</tr>
<tr>
<td>Caliente E</td>
<td>678</td>
<td>2,865</td>
<td>1,540.20</td>
<td>334.95</td>
</tr>
<tr>
<td>Dalhart E</td>
<td>1,054</td>
<td>1,524</td>
<td>1,246.10</td>
<td>86.54</td>
</tr>
<tr>
<td>Eagle Pass E</td>
<td>155</td>
<td>720</td>
<td>287.98</td>
<td>87.97</td>
</tr>
<tr>
<td>Fairmont E</td>
<td>286</td>
<td>472</td>
<td>370.87</td>
<td>34.06</td>
</tr>
<tr>
<td>Gadsden E</td>
<td>118</td>
<td>549</td>
<td>257.63</td>
<td>73.74</td>
</tr>
<tr>
<td>Hailey E</td>
<td>954</td>
<td>3,600</td>
<td>1,974.10</td>
<td>516.28</td>
</tr>
<tr>
<td>Idaho Falls E</td>
<td>1,310</td>
<td>3,231</td>
<td>1,500.10</td>
<td>144.88</td>
</tr>
<tr>
<td>Jackson E</td>
<td>13</td>
<td>158</td>
<td>64.77</td>
<td>30.73</td>
</tr>
<tr>
<td>Kalispell E</td>
<td>849</td>
<td>2,926</td>
<td>1,360.00</td>
<td>342.79</td>
</tr>
<tr>
<td>La Crosse E</td>
<td>191</td>
<td>442</td>
<td>306.22</td>
<td>48.89</td>
</tr>
<tr>
<td>Macon E</td>
<td>24</td>
<td>162</td>
<td>78.59</td>
<td>22.18</td>
</tr>
<tr>
<td>Nashville E</td>
<td>116</td>
<td>398</td>
<td>199.04</td>
<td>37.13</td>
</tr>
<tr>
<td>O'Neil E</td>
<td>369</td>
<td>720</td>
<td>565.73</td>
<td>74.08</td>
</tr>
<tr>
<td>Paducah E</td>
<td>90</td>
<td>305</td>
<td>134.03</td>
<td>26.56</td>
</tr>
<tr>
<td>Quebec E</td>
<td>0</td>
<td>914</td>
<td>350.43</td>
<td>140.25</td>
</tr>
<tr>
<td>Racine E</td>
<td>176</td>
<td>289</td>
<td>185.02</td>
<td>17.00</td>
</tr>
<tr>
<td>Sacramento E</td>
<td>40</td>
<td>3,104</td>
<td>1,102.10</td>
<td>695.34</td>
</tr>
<tr>
<td>Tallahassee E</td>
<td>0</td>
<td>110</td>
<td>36.51</td>
<td>26.02</td>
</tr>
<tr>
<td>Ukiah E</td>
<td>9</td>
<td>2,244</td>
<td>526.28</td>
<td>514.34</td>
</tr>
<tr>
<td>Valdosta E</td>
<td>3</td>
<td>76</td>
<td>36.44</td>
<td>9.24</td>
</tr>
<tr>
<td>Waco E</td>
<td>66</td>
<td>213</td>
<td>132.45</td>
<td>25.25</td>
</tr>
<tr>
<td>Yakima E</td>
<td>157</td>
<td>1,888</td>
<td>662.02</td>
<td>304.62</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of results for sample of 24 random USGS DEMs.
Tables 5.2 - 5.4 present the percentage of selected points, number of triangles, approximation error, and running time for fifteen different levels of resolution generated from Lake Champlain, Crater Lake, and Martian terrain data sets, respectively. The maximum error and the root mean square (RMS) error are calculated for each level. The running times are given in CPU seconds measured on an SGI O2 workstation (IRIX 6.5, R5000 with a 200MHz MIPS processor and 64 Mbytes of main memory).

<table>
<thead>
<tr>
<th>% of Points</th>
<th>Number of Triangles</th>
<th>Maximum Error (meters)</th>
<th>RMS Error (meters)</th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>28,554</td>
<td>23.00</td>
<td>6.53</td>
<td>35.25</td>
</tr>
<tr>
<td>2.0</td>
<td>57,228</td>
<td>12.97</td>
<td>3.60</td>
<td>42.31</td>
</tr>
<tr>
<td>3.0</td>
<td>85,926</td>
<td>9.03</td>
<td>2.55</td>
<td>48.62</td>
</tr>
<tr>
<td>4.0</td>
<td>114,679</td>
<td>7.00</td>
<td>1.99</td>
<td>54.79</td>
</tr>
<tr>
<td>5.0</td>
<td>143,435</td>
<td>5.66</td>
<td>1.64</td>
<td>60.79</td>
</tr>
<tr>
<td>6.0</td>
<td>172,179</td>
<td>4.75</td>
<td>1.41</td>
<td>66.58</td>
</tr>
<tr>
<td>7.0</td>
<td>200,927</td>
<td>5.28</td>
<td>1.19</td>
<td>72.38</td>
</tr>
<tr>
<td>8.0</td>
<td>229,686</td>
<td>5.34</td>
<td>1.06</td>
<td>78.34</td>
</tr>
<tr>
<td>9.0</td>
<td>258,451</td>
<td>3.14</td>
<td>0.93</td>
<td>83.74</td>
</tr>
<tr>
<td>10.0</td>
<td>287,225</td>
<td>4.00</td>
<td>0.86</td>
<td>89.36</td>
</tr>
<tr>
<td>11.0</td>
<td>316,014</td>
<td>5.00</td>
<td>0.76</td>
<td>95.43</td>
</tr>
<tr>
<td>12.0</td>
<td>344,805</td>
<td>2.33</td>
<td>0.70</td>
<td>101.01</td>
</tr>
<tr>
<td>13.0</td>
<td>373,601</td>
<td>2.00</td>
<td>0.65</td>
<td>107.18</td>
</tr>
<tr>
<td>14.0</td>
<td>402,394</td>
<td>2.00</td>
<td>0.61</td>
<td>112.32</td>
</tr>
<tr>
<td>15.0</td>
<td>431,180</td>
<td>1.80</td>
<td>0.55</td>
<td>118.26</td>
</tr>
</tbody>
</table>

Table 5.2: Summary of results for Lake Champlain.
<table>
<thead>
<tr>
<th>% of Points</th>
<th>Number of Triangles</th>
<th>Maximum Error (meters)</th>
<th>RMS Error (meters)</th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>3,011</td>
<td>24.25</td>
<td>6.76</td>
<td>3.07</td>
</tr>
<tr>
<td>2.0</td>
<td>6,053</td>
<td>23.44</td>
<td>3.99</td>
<td>3.70</td>
</tr>
<tr>
<td>3.0</td>
<td>9,110</td>
<td>10.15</td>
<td>2.93</td>
<td>4.30</td>
</tr>
<tr>
<td>4.0</td>
<td>12,171</td>
<td>7.92</td>
<td>2.33</td>
<td>4.85</td>
</tr>
<tr>
<td>5.0</td>
<td>15,231</td>
<td>6.45</td>
<td>1.91</td>
<td>5.43</td>
</tr>
<tr>
<td>6.0</td>
<td>18,290</td>
<td>6.71</td>
<td>1.66</td>
<td>5.93</td>
</tr>
<tr>
<td>7.0</td>
<td>21,355</td>
<td>4.66</td>
<td>1.41</td>
<td>6.51</td>
</tr>
<tr>
<td>8.0</td>
<td>24,426</td>
<td>4.00</td>
<td>1.26</td>
<td>7.07</td>
</tr>
<tr>
<td>9.0</td>
<td>27,495</td>
<td>3.66</td>
<td>1.15</td>
<td>7.59</td>
</tr>
<tr>
<td>10.0</td>
<td>30,609</td>
<td>4.45</td>
<td>1.04</td>
<td>8.14</td>
</tr>
<tr>
<td>11.0</td>
<td>33,639</td>
<td>3.00</td>
<td>0.95</td>
<td>8.66</td>
</tr>
<tr>
<td>12.0</td>
<td>36,708</td>
<td>4.00</td>
<td>0.89</td>
<td>9.23</td>
</tr>
<tr>
<td>13.0</td>
<td>39,780</td>
<td>2.50</td>
<td>0.82</td>
<td>9.72</td>
</tr>
<tr>
<td>14.0</td>
<td>42,855</td>
<td>2.40</td>
<td>0.77</td>
<td>10.27</td>
</tr>
<tr>
<td>15.0</td>
<td>45,927</td>
<td>2.20</td>
<td>0.71</td>
<td>10.80</td>
</tr>
</tbody>
</table>

Table 5.3: Summary of results for Crater Lake.

<table>
<thead>
<tr>
<th>% of Points</th>
<th>Number of Triangles</th>
<th>Maximum Error (meters)</th>
<th>RMS Error (meters)</th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>17,654</td>
<td>23.46</td>
<td>7.32</td>
<td>24.67</td>
</tr>
<tr>
<td>2.0</td>
<td>35,348</td>
<td>14.42</td>
<td>4.21</td>
<td>29.51</td>
</tr>
<tr>
<td>3.0</td>
<td>53,064</td>
<td>9.21</td>
<td>3.13</td>
<td>34.07</td>
</tr>
<tr>
<td>4.0</td>
<td>70,801</td>
<td>7.74</td>
<td>2.38</td>
<td>38.45</td>
</tr>
<tr>
<td>5.0</td>
<td>44,413</td>
<td>6.17</td>
<td>1.92</td>
<td>42.75</td>
</tr>
<tr>
<td>6.0</td>
<td>106,266</td>
<td>5.24</td>
<td>1.65</td>
<td>46.32</td>
</tr>
<tr>
<td>7.0</td>
<td>123,999</td>
<td>5.78</td>
<td>1.36</td>
<td>50.76</td>
</tr>
<tr>
<td>8.0</td>
<td>141,740</td>
<td>4.21</td>
<td>1.19</td>
<td>54.91</td>
</tr>
<tr>
<td>9.0</td>
<td>159,475</td>
<td>3.50</td>
<td>1.08</td>
<td>58.34</td>
</tr>
<tr>
<td>10.0</td>
<td>177,205</td>
<td>4.78</td>
<td>0.97</td>
<td>62.26</td>
</tr>
<tr>
<td>11.0</td>
<td>194,950</td>
<td>5.20</td>
<td>0.91</td>
<td>66.92</td>
</tr>
<tr>
<td>12.0</td>
<td>212,677</td>
<td>3.27</td>
<td>0.79</td>
<td>70.65</td>
</tr>
<tr>
<td>13.0</td>
<td>230,425</td>
<td>2.64</td>
<td>0.73</td>
<td>75.13</td>
</tr>
<tr>
<td>14.0</td>
<td>248,182</td>
<td>2.43</td>
<td>0.66</td>
<td>78.83</td>
</tr>
<tr>
<td>15.0</td>
<td>265,901</td>
<td>2.17</td>
<td>0.59</td>
<td>82.87</td>
</tr>
</tbody>
</table>

Table 5.4: Summary of results for the Martian terrain.

105
Table 5.5 shows a comparison between the RMS error using the conventional maximum vertical distance and the new vertex selection criterion for Crater Lake.

<table>
<thead>
<tr>
<th>% of Points</th>
<th>RMS Error (meters)</th>
<th>Max. Vertical Distance</th>
<th>New Selection Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>7.05</td>
<td>6.76</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>4.17</td>
<td>3.99</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>3.06</td>
<td>2.93</td>
<td></td>
</tr>
<tr>
<td>4.0</td>
<td>2.44</td>
<td>2.33</td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>2.00</td>
<td>1.91</td>
<td></td>
</tr>
<tr>
<td>6.0</td>
<td>1.74</td>
<td>1.66</td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>1.48</td>
<td>1.41</td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>1.32</td>
<td>1.26</td>
<td></td>
</tr>
<tr>
<td>9.0</td>
<td>1.20</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td>1.09</td>
<td>1.04</td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>1.00</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>12.0</td>
<td>0.93</td>
<td>0.89</td>
<td></td>
</tr>
<tr>
<td>13.0</td>
<td>0.86</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>14.0</td>
<td>0.81</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td>15.0</td>
<td>0.75</td>
<td>0.71</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Comparison between maximum vertical error and new vertex selection criterion.

Although the differences may look not significant when we compare the approximation errors, the new vertex selection criterion generates triangulations with much less points. Figures 5.5(a)-(b) show approximations obtained by applying our method and a conventional incremental algorithm, respectively, to the Crater Lake DEM. The triangulation produced by our method has 126 vertices and 230 triangles, whereas the other triangulation contains 140 vertices and 263 triangles. The corresponding root mean square (RMS) error for both approximations is 29m.
Figure 5.5: Approximations of Crater Lake DEM constructed by using (a) our combined refinement/decimation method (230 triangles) and (b) incremental insertion algorithm (263 triangles).

Figures 5.6 - 5.8 show successive approximations for the digital elevation model of Lake Champlain, Crater Lake, and Martian terrain, respectively, using 1.0%, 5.0%, and 10.0% of the original points. Although the meshes shown in Figures 5.6(c) and 5.7(c) have only 1% of the original number of points, the models still provide an adequate representation.
Figure 5.6: Approximations of the Lake Champlain DEM.
Figure 5.7: Approximations of the Crater Lake DEM.

109
Figure 5.8: Approximations of the Martian terrain.
Figure 5.9 shows the RMS error for Lake Champlain, Crater Lake, and the Martian terrain. Even though the point selection criterion does not guarantee that the triangular surface at a given step is more accurate than the one calculated at the previous step, however, experiments have shown that the behavior of the approximation errors are decreasing, although not monotonically.

![Figure 5.9: Approximation error for three terrain data sets.](image)

The vertical error of the approximate surface of the Lake Champlain using 5% of the input points is shown in Figure 5.10(a), which is quite flat. Figure 5.11 shows the error profile in horizontal (parallel to the x-axis) and vertical direction (parallel to the y-axis). 1-D cross-sections of elevations in both directions are given in Figures 5.10(b) and 5.10(c), where the solid and dotted lines represent the original and the approximate elevation values, respectively. The vertex with maximum vertical error is used as the intersection point of the two cross-sections. Note that there is no significant deviation between the original and the approximate surfaces.
Figure 5.10: (a) Three-dimensional plot of the elevation error for Lake Champlain (using 5% of the input points); (b) error profile at 26th row; (c) error profile at 150th column.
Figure 5.11: (a) 1-D cross section of original and approximate surfaces of the Lake Champlain at 26th row; (b) 1-D cross section of original and approximate surfaces at 150th column.
Although the accuracy of the terrain models is usually evaluated by computing the differences in elevation between the original and the approximate surfaces, other criteria can be used. For instance, terrain characteristics such as visibility, drainage networks, and slope can also be calculated and compared.

The visibility index for a sample of terrain data was calculated using a program (experimental implementation) of Franklin [117]. The algorithm estimates the number of targets visible from each point of the terrain. For each viewpoint, it is assumed an observer whose height is 10 meters above the surface. The visibility index is measured within a range of 100 points from observer to target.

Figure 5.13 on page 115 shows the visibility index for each point of the original and approximate Lake Champlain DEM (using 16% of the original points). Brighter points represent higher visibility indices.

The degree of agreement between the visibility indices for both data sets can be evaluated by measuring the point-by-point differences. A histogram showing the distribution of these point-by-point differences is given in Figure 5.12.

![Figure 5.12: Distribution of the differences between two sets of visibility indices.](image)

114
Figure 5.13: Visibility indices for Lake Champlain DEM. (a) original; (b) approximation using 16% of the original points.
Consider now each visibility index within the interval between 0 (indicating that no target is visible from the viewpoint) and 100 (indicating that all possible targets are visible from the viewpoint). Some basic statistics of the visibility index for Lake Champlain are given in Table 5.6.

<table>
<thead>
<tr>
<th>Visibility Index - Lake Champlain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Std. Deviation</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>25-percentile</td>
</tr>
<tr>
<td>75-percentile</td>
</tr>
</tbody>
</table>

Table 5.6: Statistics of the differences in visibility indices.

These results suggest a high degree of agreement between the visibility indices calculated from the original and approximate terrain model. Since an important application of visibility information is the identification of candidate point sites, we selected a set of points of highest visibility from each data set, and compared them. The visibility indices of the points in the original an approximate Lake Champlain DEM were ranked in descending order, and the first 500 points were selected from each data set. The relation of correspondence between these two set of points was evaluated based on a distance function within a local area. This area is bounded by a circle of radius $R$ centered at a point $P$, as shown in Figure 5.14.

![Figure 5.14: Search area bounded by a circle.](image)

Table 5.7 shows the results obtained by comparing each point $P_i$ in a set against each point $Q_j$ in the other set, using three different radius values. From these two
samples of 500 points, 78.6% of the points coincide exactly in their coordinates (represented as $P_i = Q_j$), and approximately 20% have their location less than two pixels apart (represented as $P_i - Q_j \leq R$). Therefore, only a few points differ from each other (represented as $P_i \neq Q_j$).

<table>
<thead>
<tr>
<th>Radius</th>
<th>Pairwise Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 1$</td>
<td>393 92 15</td>
</tr>
<tr>
<td>$R = 2$</td>
<td>393 99 8</td>
</tr>
<tr>
<td>$R = 3$</td>
<td>393 103 4</td>
</tr>
</tbody>
</table>

Table 5.7: Summary of pairwise comparison of visibility indices.

The quality of the triangular meshes was also estimated by three different measures:

- valence: corresponds to the number of edges sharing a single vertex. A triangular mesh that has 6 edges at a vertex is considered optimal [34].

- sliveriness: poor shape elements can affect efficiency of numerical integration and produce visual artifacts. The sliveriness of a triangle was defined as

$$Sliveriness = \frac{Perimeter^2}{Area}$$

An equilateral triangle has a sliveriness ratio equals $36/\sqrt{3} \approx 20.78$. The average sliveriness for the entire triangulation was calculated and divided by the sliveriness ratio for an equilateral triangle. A lower sliveriness value represents a better triangulation.

- minimum angle: the minimum angle for each triangle was measured, then the average of these angles for the entire triangulation was calculated.

Table 5.8 compares the mesh quality measures for the sample of 24 random USGS DEMs. The results demonstrate the high quality meshes produced with our method.
The use of Delaunay triangulation significantly contributed to reduce the number of very thin triangles (slivers).

<table>
<thead>
<tr>
<th>USGS DEMs (1201×1201)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEM File</strong></td>
</tr>
<tr>
<td>Aberdeen E</td>
</tr>
<tr>
<td>Baker E</td>
</tr>
<tr>
<td>Caliente E</td>
</tr>
<tr>
<td>Dalhart E</td>
</tr>
<tr>
<td>Eagle Pass E</td>
</tr>
<tr>
<td>Fairmont E</td>
</tr>
<tr>
<td>Gadsden E</td>
</tr>
<tr>
<td>Hailey E</td>
</tr>
<tr>
<td>Idaho Falls E</td>
</tr>
<tr>
<td>Jackson E</td>
</tr>
<tr>
<td>Kalispell E</td>
</tr>
<tr>
<td>La Crosse E</td>
</tr>
<tr>
<td>Macon E</td>
</tr>
<tr>
<td>Nashville E</td>
</tr>
<tr>
<td>O’Neill E</td>
</tr>
<tr>
<td>Paducah E</td>
</tr>
<tr>
<td>Quebec E</td>
</tr>
<tr>
<td>Racine E</td>
</tr>
<tr>
<td>Sacramento E</td>
</tr>
<tr>
<td>Tallahassee E</td>
</tr>
<tr>
<td>Ukiah E</td>
</tr>
<tr>
<td>Valdosta E</td>
</tr>
<tr>
<td>Waco E</td>
</tr>
<tr>
<td>Yakima E</td>
</tr>
</tbody>
</table>

Table 5.8: Mesh quality measures for sample of 24 random USGS DEMs.
Figure 5.15 shows the RMS error for Lake Champlain using linear, quadratic, and cubic interpolation. It can be observed that the approximation error is not reduced with the use of polynomial interpolation. Compared to the linear interpolation, the approximate terrain models obtained using quadratic and cubic interpolation presented a more uniform distribution of the approximation error (measured as the difference between the actual elevation and the interpolated one) along each triangular patch. Therefore, although a number of points in the linear interpolation have higher deviation, it still presented a lower overall error measurement. Histograms illustrating the point-by-point differences in elevation for the Lake Champlain and Crater Lake using each interpolation method are given in Figures 5.16 - 5.17, respectively. Approximations of the Crater Lake and their corresponding contour maps are shown in Figure 5.18.

![Figure 5.15: Approximation error for Lake Champlain DEM using each interpolation method.](image-url)
Figure 5.16: Differences in elevation for Lake Champlain DEM (using 1000 points). (a) linear; (b) quadratic; (c) cubic.
Figure 5.17: Differences in elevation for Crater Lake DEM (using 1000 points). (a) linear; (b) quadratic; (c) cubic.
Figure 5.18: Approximations of the Crater Lake DEM (using 1000 points). (a) linear; (b) quadratic; (c) cubic.
The area that is visible from a particular location (viewshed) was calculated at five different observer points using the results obtained from each interpolation method. The triangulation was constructed using 16% of the original points. The viewshed calculation assumed an observer height of 70 meters above the surface. The results, given in Table 5.9, show that the linear interpolation provided the best match in terms of the number of visible points.

<table>
<thead>
<tr>
<th>Observer Location (x, y)</th>
<th>Interpolation Method</th>
<th>Number of Visible Points</th>
<th>Matching Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>(118,755)</td>
<td>Linear</td>
<td>265,897</td>
<td>257,921</td>
</tr>
<tr>
<td></td>
<td>Quadratic</td>
<td>262,233</td>
<td>254,458</td>
</tr>
<tr>
<td></td>
<td>Cubic</td>
<td>261,347</td>
<td>253,170</td>
</tr>
<tr>
<td>(198,249)</td>
<td>Linear</td>
<td>417,812</td>
<td>405,805</td>
</tr>
<tr>
<td></td>
<td>Quadratic</td>
<td>413,244</td>
<td>401,405</td>
</tr>
<tr>
<td></td>
<td>Cubic</td>
<td>411,846</td>
<td>399,587</td>
</tr>
<tr>
<td>(221,1012)</td>
<td>Linear</td>
<td>334,236</td>
<td>324,210</td>
</tr>
<tr>
<td></td>
<td>Quadratic</td>
<td>333,927</td>
<td>323,101</td>
</tr>
<tr>
<td></td>
<td>Cubic</td>
<td>332,389</td>
<td>322,165</td>
</tr>
<tr>
<td>(1012,946)</td>
<td>Linear</td>
<td>262,434</td>
<td>254,560</td>
</tr>
<tr>
<td></td>
<td>Quadratic</td>
<td>261,348</td>
<td>253,510</td>
</tr>
<tr>
<td></td>
<td>Cubic</td>
<td>260,675</td>
<td>252,852</td>
</tr>
<tr>
<td>(1080,295)</td>
<td>Linear</td>
<td>425,830</td>
<td>413,331</td>
</tr>
<tr>
<td></td>
<td>Quadratic</td>
<td>424,788</td>
<td>411,583</td>
</tr>
<tr>
<td></td>
<td>Cubic</td>
<td>424,089</td>
<td>411,054</td>
</tr>
</tbody>
</table>

Table 5.9: Viewshed determination at five locations of the Lake Champlain DEM.

Examples of viewsheds computed at the point (412, 268) for the Lake Champlain using each interpolation method are shown in Figure 5.19. White pixels represent the areas that are visible from the viewpoint.
Figure 5.19: Viewsheds computed at point $x = 412$ and $y = 268$ for Lake Champlain DEM. (a) original; (b) linear; (c) quadratic; (d) cubic. White pixels represent the areas that are visible from the viewpoint.
5.3 Computational Time

We have tested the performance of our algorithms on an SGI O2 workstation (IRIX 6.5, R5000 with a 200MHz MIPS processor and 64 Mbytes of main memory).

Figure 5.20 compares the running time of the linear interpolation algorithm on three terrain models. In the refinement step, the algorithm is able to select 55,000 points in approximately 60 seconds.

To select new vertices of a triangle more quickly during the mesh construction, a heap is used to implement a priority queue, which maintains the candidate points and their corresponding approximation errors. Each triangle can have only one candidate vertex (or zero if the triangle has no input points in its interior). For each modification to the mesh, it is necessary to update the approximation error and the candidates only for those triangles affected by the local retriangulation. This is done by applying a scan-conversion technique to each affected triangle. This idea was also explored by Garland and Heckbert [125].

5.4 Memory Requirements

The quad-edge data structure was used to represent the mesh by means of an interconnected graph consisting of vertices, directed edges, and triangles. The data structure currently uses 16 bytes to store each vertex position, 68 bytes per edge, and 24 bytes per triangle. Since the number of edges in a TIN is approximately three times the number of its vertices, and the number of triangles is about twice the number of vertices, the triangulation therefore requires $16n + 204n + 48n = 268n$ bytes, where $n$ is the number of vertices in the mesh.

As described in the previous section, a heap was used to maintain the candidate vertices and their corresponding errors during the construction of the mesh. The heap uses 16 bytes per vertex, and its size is proportional to the number of triangles in the mesh. Since a heap is used for the refinement process and other for the decimation
process, then about $64n$ bytes are required to maintain the heap nodes.

Therefore, our current implementation uses a total of $332n$ bytes to store the data structures. The memory requirements used to store the final meshes can be significantly reduced through the use of efficient techniques for compressing geometry and topology of triangular meshes. For a detailed description of mesh compression methods, see Section 3.1.2 on page 54.

5.5 Implementation Issues

Our algorithms were implemented in C++ programming language on UNIX workstations. Although our methods have been evaluated on SGI workstations, the programs also run on on SUN and IBM platforms.

The incremental Delaunay triangulation algorithm is adapted from Lischinski [198], described in Graphics Gems IV. OpenGL was used to incorporate several 2D and 3D graphics functions, including geometric transformations, color, lighting,
smooth shading, and texturing. Three examples of shaded images created using our program are shown in Figures 5.21 - 5.23.

Figure 5.21: Original Lake Champlain DEM.

Our programs support two different data input types, USGS DEM data and portable graymap (PGM) file format. The resulting terrain approximations can be stored as a list of triangular faces indexed by $X$, $Y$, and $Z$ vertex vectors or converted into equivalent DEM format.

The results have demonstrated a good balance between speed and ability to process large terrain data sets, even though the current version of our algorithm is still unoptimized in many aspects.
Figure 5.22: Crater Lake DEM (10,000 points).

Figure 5.23: Martian Terrain (100,000 points).
CHAPTER 6
CONCLUSIONS

6.1 Summary and Contributions

As an alternative to regular grid digital elevation models (DEMs), we have presented an adaptive method based on triangulated irregular models (TINs). The method offers storage efficiency, flexibility, support for multiresolution modeling, and the ability to incorporate terrain features to the triangulated model. Several quantitative and qualitative measures are proposed to characterize the fidelity of the terrain approximations.

The mesh construction is performed by a hybrid refinement and decimation approach, incrementally determining a better distribution of the data points. The resulting approximations preserve relevant topographic features, while satisfying a specified error tolerance. A constrained Delaunay triangulation is used to maintain the topology of the data points, whose vertices lie at a subset of the input data.

A new local error metric is used to select points to be inserted into the triangulation, which is based on the maximum vertical error weighted by the standard deviation calculated in a neighborhood of the candidate point. Conversely, a measure of angle between surface normals is used to determine whether a vertex should be removed from the triangulation. Approximations produced by our combined refinement/decimation method are significantly better than those generated by conventional triangulation algorithms, containing fewer triangles for a given level of accuracy.

Although no explicit optimization technique has been applied to the resulting triangulations, the meshes maintain a good balance between element shape and node valence. The use of Delaunay triangulation contributed to improve the quality of the
TINs, particularly relevant when subdivision schemes are used to generate curved surfaces over the triangulated network.

Mesh compression techniques can also be applied to encode geometric and topological information associated to the meshes, reducing storage requirements and transmission time.

Another characteristic of our method is the possibility of locally changing the level of resolution in different regions of the surface, allowing a coarser approximation of less relevant areas and a finer approximation within specific regions of interest.

Automatic procedures for construction of smooth terrain surfaces defined as a network of curved patches were also investigated. In particular, quadratic and cubic surfaces were constructed over the triangular meshes. Although smooth surfaces can produce superior results for rendering purposes, our experiments have demonstrated that such polynomial surfaces not necessarily reduce the approximation error, defined as the root mean square error (RMSE).

We have implemented a portable software package in C++ programming language, running on UNIX workstations. OpenGL is used to render the final terrain approximations. The method has been tested on a number of real terrain data sets, producing encouraging results.

6.2 Future Directions

Although the problem of surface modeling has been extensively investigated in the literature, terrain modeling particularly offers a rich area in possibilities for investigation. Some areas that require further work are listed below.

- the use of additional sources of information such as road networks, drainage networks, ridge lines, and other features could be used to guide the triangulation process. Indeed, preservation of these properties in the resulting surface is more important than just minimizing the maximum error;
• algorithms and implementations can be improved in several ways. Recomputation of error during the steps of refinement and decimation is still slow. The calculation of derivatives at each vertex and the construction of curved surfaces over the triangular meshes can be implemented on parallel architectures;

• the extraction of surface representations at different levels of resolution requires additional mechanisms to select and combine multiple levels of detail, particularly relevant for fast terrain rendering;

• the polynomial methods described here depend on precise values for derivatives. Our approach is based on the calculation of normal vectors for each triangular face sharing a vertex. More sophisticated techniques using geomorphologic information may produce better results;

• experimentation with range data in order to create 3D models that are useful for other applications, such as reverse engineering, computer graphics, and computer vision.
REFERENCES


138


151


APPENDIX A

TRIANGULATION ALGORITHMS

A triangulation of a set of data points in the plane can be defined in terms of a
planar graph in which pairs of vertices are connected by edges intersected only at
their endpoints, forming triangular faces. For a triangulated planar graph with \(N\)
vertices, \(N_b\) being the boundary points (i.e., points which lie on the convex hull), \(E\)
edges, and \(T\) triangles, then

\[
T = 2(N - 1) - N_b
\]

\[
E = T + N - 1 = 3(N - 1) - N_b
\]

The expression \(E = T + N - 1\) is a special case of Euler’s formula which holds
for \(T\) representing the number of finite regions in any connected planar graph.

The topology of the triangulation can be chosen either using only the \(xy\) projec-
tions of the data points or using the \(z\) elevations of the points as well. The latter
approach is called \textit{data-dependent triangulation} [87, 251].

A.1 Delaunay Triangulation

The most common triangulation method that uses only the \(xy\) projections is the
\textit{Delaunay triangulation} [74]. The Delaunay triangulation has the property that the
circumcircle of any triangle in the triangulation contains no other data points in its
interior. This property, known as \textit{circle property}, is illustrated in Figure A.1.

The Delaunay triangulation generates the triangulation that maximizes the min-
umum angle of all triangles. This property is known as \textit{max-min angle property}. If
the diagonal of any strictly convex quadrilateral formed by two adjacent triangles is
replaced with the opposite one, the minimum of the six internal angles of the result-
Figure A.1: The circle property. (a) Triangle $T_1$ is not a Delaunay triangle since a data point $P$ lies inside its circumcircle; (b) triangle $T_2$ is a Delaunay triangle.

Figure A.2: The max-min angle property. The diagonal $e_A$ in (a) is replaced with diagonal $e_B$, as in (b), in order to increase the minimum of the six internal angles.

The equivalence between the circle property and the max-min angle property has been shown by Lawson [184] and Sibson [291]. In a Delaunay triangulation, most of its triangles are nearly equiangular, which helps to minimize the occurrence of thin and long triangles since they can lead to undesirable behavior, affecting numerical stability and producing visual artifacts. Moreover, the Delaunay triangulation is unique if no four or more points are cocircular in the Euclidean plane.
Based on max-min angle property, Lawson [183, 184] proposes a local optimization procedure (LOP) to swap the diagonals of a strictly convex quadrilateral to achieve a most equiangular triangulation. An important consequence of this procedure is that the successive application of the local optimization procedure to all internal edges of an arbitrary triangulation produces a Delaunay triangulation in which all the edges are locally optimal. Therefore, after some finite number of swaps, a triangulation $T$ will be reached such that each internal edge in $T$ will not be swapped when tested by the local optimization procedure.

Another interesting property is that Delaunay triangles define nearest natural neighbors in the sense that the data points at the vertices are closer to their circumcenter than is any other data point. These circumcenters are the positions of vertices in the geometrically dual Voronoi diagram, also known as Dirichlet, Thiessen or Wigner-Seitz tessellation. The concept is very old and includes interest in several scientific disciplines. Dirichlet [78] used it for studying the theory of quadratic forms in 1850. In 1908, Voronoi [320] provided a generalization to higher dimensions. The climatologist Thiessen [309] used the concept of proximity polygons for improving the estimation of precipitation averages over large areas. Wigner and Seitz [328] used such diagrams in metallurgy in 1933.

The Voronoi diagram of a set $V = \{P_1, P_2, \ldots, P_n\}$ of points in the plane is a partitioning of the plane into $n$ convex regions, one per point. Each Voronoi region $R_i$ represents the space partition closest to a point $P_i$. The Delaunay triangulation can be obtained by connecting every pair of points of $V$ whose Voronoi regions share a common edge. An example of a Delaunay triangulation and its dual Voronoi diagram for a given set of points is shown in Figure A.3.

Preparata and Shamos [249], O'Rourke [231], and Aurenhammer [9] provide a number of definitions and applications of Delaunay triangulations and Voronoi diagrams. Several algorithms for constructing Voronoi diagrams in the plane and higher
dimensions have been proposed [27, 111, 134, 141, 142, 185, 187, 286, 323].

A.1.1 Constrained Delaunay Triangulation

In some practical applications, a triangulation must force to include a given set of edges, for example, linear features (breaklines [41]) such as ridges and valleys, or boundaries such as shorelines and buildings. Moreover, it may be important to combine the possibility of including prespecified edges with the advantages of a Delaunay triangulation.

The constrained Delaunay triangulation is a triangulation that includes a given set of edges, and it is as close as possible to the Delaunay triangulation. An alternative characterization of constrained triangulation can be given through the notion of visibility. Suppose a set $V$ of data points in the plane and a set $S$ of line segments with their endpoints in $V$. Two points $P_i$ and $P_j$ in $V$ are called mutually visible with respect to $S$ if and only if they can be joined by a line segment without intersecting any constraint segment in $S$. The graph containing the segments in $S$ is a subgraph of the visibility graph. A constrained triangulation is shown in Figure A.4.

A constrained Delaunay triangulation of a set $V$ of data points with respect to a set $S$ of constraint segments is a constrained triangulation satisfying the following version of the circle property. A triangle is a constrained Delaunay triangle if and
only if there does not exist any other point in \( V \) inside its circumcircle which is visible from all the three vertices of the triangle. An example is illustrated in Figure A.5. A standard Delaunay triangulation is obtained if the set \( S \) of constraint segments is empty.

The properties of a Delaunay triangulation can be extended to a constrained Delaunay triangulation with some modifications due to the set of constraint segments. The max-min angle property can be reformulated for a constrained Delaunay triangulation by considering only these convex quadrilaterals formed by two adjacent triangles which do not share a constraint segment.

Chew \cite{44, 45} presents a divide-and-conquer algorithm for constructing a constrained Delaunay triangulation in \( O(n \log n) \), the same time required to build a stan-
standard Delaunay triangulation. Line segment endpoints are sorted by $x$-coordinates, and split into vertical lines. Constrained Delaunay triangulations are calculated for each subset and then merged together.

### A.1.2 Conforming Delaunay Triangulation

A problem in using a constrained Delaunay triangulation in surface approximation is that the forced inclusion of constraint edges may produce thin triangles. An alternative approach is provided by the notion of conforming Delaunay triangulation. The idea is to split the constraint segments into shorter segments by adding new vertices, known as Steiner points, such that each constraint segment is the union of edges of the triangulation.

Figure A.6 shows a conforming Delaunay triangulation constructed from the example given in Figure A.4. Three extra points are added on the constraint edges.

![Figure A.6: A conforming Delaunay triangulation.](image)

Edelsbrunner and Tan [89] prove that a conforming Delaunay triangulation can be achieved in $O(m^2 n)$ upper bound, where $m$ is the number of edges and $n$ is the number of data points. Algorithms proposed by Boissonnat [26], Nackman and Srinivasan [224], Oloufa [230], and Saalfeld [269] are based on the idea of adding sufficiently many points on the edges of the constraint graph such that the Delaunay triangulation conforms to all input edges. However, these algorithms do not use a minimum number of points to compute a conforming Delaunay triangulation.
A.2 Delaunay Triangulation Algorithms

Several different algorithms for constructing a Delaunay triangulation have been proposed in the literature. This section describes a number of algorithms for the generation of TINs based on the Delaunay triangulation in two dimensions. The emphasis is on the approximation of terrain surfaces.

The triangulation algorithms are classified into the following four categories:

- **divide-and-conquer algorithms**: a Delaunay triangulation is constructed by recursively splitting the set of data points into two almost equally sized subsets until elementary primitives (triangles) are obtained and then merging the resulting tessellations in pairs;

- **growth algorithms**: a Delaunay triangulation is successively constructed by expanding the tessellation when new points are added to it;

- **incremental algorithms**: a Delaunay triangulation is constructed by iteratively adding data points, while maintaining the triangulation updated after each point is added;

- **sweepline algorithms**: a Delaunay triangulation (or its geometric dual, the Voronoi diagram) is calculated for a set of points by applying a geometric transformation in such a way that the algorithm needs to consider the region of a point only when the point is intersected by the sweepline.

A.2.1 Divide-and-Conquer Algorithms

The construction of a Voronoi diagram using a divide-and-conquer algorithm in optimal $O(n \log n)$ worst-case time was first presented by Shamos and Hoey [286]. Lee and Schachter [186] also describe an asymptotically optimal Delaunay triangulation algorithm, running in $O(n \log n)$ time in the worst case. A similar algorithm
using a different data structure is presented by Guibas and Stolfi [142], also running in $O(n \log n)$.

The idea is based on a recursive division of the set of data points into two almost equally sized disjoint subsets, and on the merging of these subsets separately computed. First, the points of the set $V$ are preliminarily sorted in lexicographically ascending order. Then, the set $V$ is divided into halves $V_L$ and $V_R$, such that $V_L$ contains the first half of the points and $V_R$ contains the second. The Delaunay triangulations of $V_L$ and $V_R$ are recursively constructed on each half and then merged together to form the Delaunay triangulation of $V$.

The merge step of $V_L$ and $V_R$ starts with the computation of the convex hull of $V = V_L \cup V_R$, which is the domain of the Delaunay triangulation of $V$. This corresponds to determine the lower and upper common tangents of the convex hulls of $V_L$ and $V_R$. These two tangents are illustrated in Figure A.7.

![Diagram showing upper and lower common tangents of two convex hulls.](image)

Figure A.7: Illustration of the upper and lower common tangents of two convex hulls.

The merge procedure deletes the edges which are not Delaunay edges by determining if the circumcircle of a triangle contains any other points in its interior. It also adds new edges to the triangulation, starting with the lower common tangent, zigzagging upward until the upper common tangent is reached. An illustration of the merge step in the divide-and-conquer algorithm is shown in Figure A.8.

Dwyer [85] shows that a simple modification of the divide-and-conquer algorithm runs in $O(n \log \log n)$ expected time on uniformly distributed points. Dwyer’s algo-
Figure A.8: Merge step of two triangulations in the divide-and-conquer algorithm. The dashed edges have to be removed.

The algorithm splits the set of points into $\sqrt{n / \log n}$ subsets by horizontal lines, constructs the Delaunay triangulation of each line by merging along vertical lines, and then merges the lines together.

### A.2.2 Growth Algorithms

Green and Sibson [134] describe a recursive method for computing Dirichlet tessellations in the plane by scanning the points in turn, recursively modifying the contiguities as each point is added. Each Dirichlet region has an associated list of contiguities, which is modified as new points are inserted around that region. The algorithm generates a Dirichlet tessellation by repeatedly adding a point to the current tessellation. Green and Sibson point out that the computational cost of growing a tile around a new point can be calculated in constant time and, therefore, a mesh of $n$ points would give a cost of $O(n)$. The cost of the step to locate the nearest neighbor of a new point is crucial for the overall complexity of the algorithm. A naive search would have a cost of $O(n)$ per point, giving a total cost of $O(n^2)$. An improvement on this is to take advantage of the constructed tessellations as a guide when seeking the nearest neighbor of a new point by iteratively traversing the mesh from an arbitrary starting point. If points are assumed to be added at random locations, then it is reasonable to start from a point near the center of the data points,
resulting in $O(n^{1/2})$ for each point, and a total of $O(n^{3/2})$. The authors suggest that the overall complexity could be reduced to $O(n \log n)$ if the nearest neighbor search was performed over several *generations* of tessellations, where generations are meshes saved after the addition of a certain number of points, at a cost in storage.

McCullagh and Ross [212] describe an algorithm consisting of the following steps:

* create an initial line, named *base line*, by connecting an arbitrary point to its closest point;

* search for a third point by applying the circle property. This point can be found by calculating the angle $\alpha$ for the candidate points, as shown in Figure A.9. The point which forms the largest $\alpha$ is the Delaunay neighbor of the line;

* construct the Delaunay triangle and use the other two new edges as new base lines;

* repeat steps (i) and (ii) until all base lines are considered.

Mirante and Weingarten [218] describe a *radial sweep algorithm* to construct a triangulation from a set of data points. The steps in the algorithm include the following:

* find a point which lies closest to the centroid of the set;
ii. compute the distances and headings from the central point to all other points in the set by using a radiating line (see Figure A.10(a));

iii. sort all other points in ascending order by heading, distance, and pitch;

iv. radially sweep and form the radiating triangles by connecting the central point to all other points and connecting any two consecutive points. Points with the same heading are used to form a pair of triangles on each side of the common line (see Figure A.10(b));

v. fill the concavities by connecting the points that are on the boundary of the radiating triangles, then forming the convex hull of the set (see Figure A.10(c)). The triangles formed are contiguous nonoverlapping triangles, however, they may have undesirable shapes (i.e., very thin triangles);

vi. successively optimize the triangulation by testing each triangle against each neighbor to determine whether their geometry may be improved. These two neighboring triangles define a quadrilateral. The shortest distance between two opposite vertices in the quadrilateral is chosen as the diagonal in order to improve the shape of the triangles. This process is repeated until an entire pass through the data set produces no changes (see Figure A.10(d)).

This algorithm usually does not result in a Delaunay triangulation since that the use of the shortest diagonal does not satisfy the circle property. On the other hand, a Delaunay triangulation can be achieved by using the max-min angle property.

A.2.3 Incremental Algorithms

Incremental algorithms typically construct a Delaunay triangulation by iteratively adding new points to the tessellation, updating it after each point is added.
Lee and Schachter [186] describe an iterative algorithm that requires $O(n^2)$ time in the worst case. McLain [213] presents an algorithm that starts by sorting all the points according to their Euclidean distance from a fixed origin and then constructs the triangulation such that each triangle belongs to the final tessellation. Fang and Piegl [95, 96], and Bowyer [27] present algorithms for constructing Delaunay triangulations in two or more dimensions. De Floriani et al. [62] describe a surface representation based on an incremental Delaunay triangulation algorithm where points are selected until the maximum deviation between the current triangulation and the original data satisfies a given error tolerance.

An algorithm described by Watson [323] locates the triangle containing the new point. Starting from this triangle, the algorithm deletes all the triangles whose circumcircle contains the new point. Such triangles form a region called the influence
polygon, containing the points in its interior. The new Delaunay triangulation is obtained by simply connecting the point to all vertices of the influence polygon. Figure A.11 illustrates these steps. Watson’s algorithm requires \( O(n^{3/2}) \) average-case time and \( O(n^2) \) worst-case time.

![Figure A.11: Illustration of the steps of Watson’s algorithm.](image)

Guibas and Stolfi [142] describes an algorithm that uses two basic steps. The first locates the triangle containing the new point. The second step updates the tessellation by flipping edges until all edges invalidated by the new point satisfy the circle property.

Figure A.12 illustrates the steps of this algorithm. The data points are assumed to be strictly enclosed by a large convex polygon, for instance, a triangle. To insert a new point \( P \), its containing triangle is located (Figure A.12(a)), or if \( P \) lies on an existing edge, that edge is deleted and \( P \) is connected to the four vertices of the containing quadrilateral. New edges are created to connect \( P \) to the vertices of the containing polygon (Figure A.12(b)). All edges defining the containing polygon are checked to verify whether they satisfy the circle property. If the property is satisfied (Figure A.12(c)), the edge remains unchanged. If it is violated (Figure A.12(d)), the edge is swapped with the other diagonal of its quadrilateral. In this case, two more edges become candidates for inspection (Figure A.12(e)). The process continues until no more candidate edges remain, resulting a Delaunay triangulation (Figure A.12(f)).

In the worst case, the insertion of a point can require \( O(n) \) edges to be swapped, giving therefore \( O(n^2) \) time for the insertion of all \( n \) points. However, if the points are inserted in random order, Guibas et al. [141] show that the expected number of
Figure A.12: Insertion of a point $P$ into the triangulation. Dashed lines indicate edges that need to be checked by the algorithm.

The edge flips is linear.

To locate the containing triangle, Guibas and Stolfi’s algorithm starts at a random edge in the current tessellation and walks through the diagram in the direction of the new point until the correct triangle is found, taking expected time $O(n^{1/2})$. Guibas et al. propose a tree-based data structure where internal nodes are triangles that have been deleted or subdivided during the construction, and the current triangulation is stored at the leaves. The total expected cost of the search step is $O(n \log n)$ time. Ohya et al. [229] use a bucketing algorithm and process the points using a breadth-first traversal of the tree. They claim that the algorithm runs in expected linear time on points that are uniformly distributed, and they provide experimental evidence for this fact.

Su and Drysdale [302, 303] present a variant of the search step that seems to have better performance than those mentioned when the input is uniformly distributed.
They use a bucketing algorithm similar to the one that Bentley *et al.* [18] used for finding the nearest neighbor of a query point. Their algorithm takes $O(n)$ time.

**A.2.4 Sweepline Algorithms**

Fortune [111] proposes a scheme for constructing the Voronoi diagram (or its geometric dual, the Delaunay triangulation) using a *sweepline* algorithm. The algorithm sweeps a horizontal line across the plane, recording the regions intersected by the line as it moves. Since it is difficult to compute the Voronoi diagram directly with a sweepline technique, the algorithm uses a geometric transformation\(^1\) of it. The transformed Voronoi diagram has the property that the lowest point of the transformed Voronoi region appears at the point itself. A Voronoi edge is mapped to a hyperbola segment and a Voronoi vertex is mapped to an intersection of three (or more) hyperbolae. Therefore, the sweepline algorithm needs to consider the Voronoi region of a point only when the vertex is intersected by the sweepline. Figure A.13 shows the Voronoi diagram of six points and its geometric transformation.

![Figure A.13](https://example.com/figure.jpg)

Figure A.13: (a) Original Voronoi diagram; (b) transformed Voronoi diagram.

A horizontal line is swept across the plane from below. At any point in time, the portion of diagram below the sweepline is complete. As the sweepline moves upward, the cross section has to be updated either by starting a new region when the line reaches a point or by starting a new edge when the line reaches a vertex that comes

\(^1\)A geometric interpretation of this mapping using a three-dimensional version of the Voronoi diagram is given in Fortune's paper.
from intersecting two deformed separators.

The sweepline algorithm maintains a priority queue containing two type of events. *Site events* occur when the sweepline reaches a site, and *circle events* occur when it reaches the top of a circle defined by three consecutive vertices on the boundary of the triangulation. The algorithm sweeps a line upward in the $y$ direction, processing each event that it encounters.

Fortune's algorithm requires optimal $O(n \log n)$ worst-case time to compute the Voronoi diagram of $n$ points. A sweepline algorithm based on a different geometric transformation is presented in [73]. The sweepline algorithm is competitive in simplicity with the incremental algorithms since it avoids the merge step used by the divide-and-conquer algorithms.

### A.3 Data-Dependent Triangulation

Algorithms for constructing Delaunay triangulations are based only on the distribution of the data points in two dimensions, ignoring the height of the vertices being approximated. The main purpose of these algorithms is to avoid thin and long triangles, which are traditionally undesirable since they can affect numerical stability and produce visual artifacts.

A different approach, called *data-dependent triangulation*, is based on the three-dimensional approximation of the surface to the input data points. A more accurate approximation to the underlying surface is possible using algorithms discussed by Rippa [257, 258], Dyn *et al.* [87], Quak and Schumaker [251], Hamann and Chen [145], and Brown [30].

The incremental Delaunay triangulation algorithm described in Section A.2.3 tests candidate edges using simply two-dimensional geometric criteria. Instead of checking the validity of edges with the circle property, data-dependent triangulation adopts other criteria for optimizing the triangulation. A first criterion tries to minimize the sum of square errors between the approximating function and the set of
data points. A second criterion tries to minimize the angle between normal vectors defined in two adjacent triangles.

Data-dependent triangulations can produce long and thin triangles. Using the error bounds obtained by Gregory [135], Rippa [258] shows that such triangles are suitable for approximating a function which has preferred direction, that is, having large second-directional derivatives in one of the directions. In such case, triangles should be thin in directions where the magnitude of the second directional derivatives is large and should be long in directions of low curvature.

A.4 Optimization Criteria for Generating Meshes

A problem of particular interest in computational geometry is to find an optimal triangulation of a point set in two dimensions, which represents the best partition of the planar domain according to some measure of quality. Optimization criteria include maximizing the minimum angle (achieved by the Delaunay triangulation), minimizing the maximum angle, minimizing the sum of edge lengths, minimizing the maximum containing circle [57], and maximizing the minimum height of a triangle.

Thacker [307] and Shephard [289] review a number of heuristics and methods for mesh generation. Ho-Le [158] presents a review and classification of mesh generation methods.

Babuška and Aziz [11] show that the maximum angle is an appropriate quality measure for meshes in finite element methods. Edelsbrunner et al. [90] show that an edge insertion is a generalization of a local edge flip. Edge insertion can be used to find a triangulation minimizing the maximum angle in $O(n^2 \log n)$.

An open problem in computational geometry is to find the triangulation such that the total length of all the edges is minimized. This problem is also known as minimum weight triangulation (MWT). It has not been shown that a polynomial time exists for it, nor if it is known to be NP-hard. Eppstein [91], and Levcopoulos and Krznaric [192] describe a triangulation that has length within a constant factor
of the minimum.

Edelsbrunner and Tan [89] show that the triangulation minimizing the maximum edge length contains the edges of the minimum spanning tree. Their method first computes the minimum spanning tree and then triangulates each remaining polygon using dynamic programming, running in \(O(n^2)\) time.

Bern et al. [20] present algorithms that approximately maximizes the minimum height and minimize the maximum angle. Mitchell [219] gives an algorithm for constructing a triangulation whose maximum angle is as small as possible. Bern et al. [22] give an algorithm for constructing a nonobtuse triangulation of polygonal regions with holes. The number of triangles in the triangulation is \(O(n)\) and the worst-case running time is \(O(n \log^2 n)\).

Bern et al. [21] present a quadtree-based algorithm for constructing a triangulation with no small and no obtuse angles. Chew [46] presents a method for generating high quality triangular meshes in which all angles are between 30° and 120°. The method is also extended to deal with meshes on curved surfaces. Bern et al. [23] extend work on two-dimensional bounded-angle triangulations to meshes in higher dimensions.

The minimum height of a triangle relates to the quality of a curved-surface approximation [129], and to the construction of a three-dimensional mesh generation algorithm [220]. Ruppert [268] presents a Delaunay refinement algorithm for generating quality meshes in two dimensions. All triangles have a bounded aspect ratio and the the number of triangles is within a constant factor of optimal. The aspect ratio of a triangle is defined as the length of the longest edge divided by the length of the shortest height.

The greedy triangulation [204] is obtained by repeatedly selecting the shortest edge that does not intersect any of the previously chosen edges and adding it to the triangulation. The greedy triangulation can be found by simply considering all
possible edges in increasing order of edge length [325]. This approach is not efficient for large point sets since the edges must be sorted, which requires $O(n^2 \log n^2)$. Unlike the minimum weight triangulation (MWT), which is globally optimal in the sense of minimizing the sum of edge lengths, the greedy triangulation provides a local optimization. For arbitrary point sets, the greedy triangulation can be computed in $O(n^2)$ time by dynamic maintenance of a constrained Delaunay triangulation [193].

A number of approaches have been used to improve the quality of finite element meshes. A common technique, called mesh smoothing, determines new geometric locations of mesh vertices, while retaining the same connectivity structure [7, 120, 121, 122, 123, 200]. Laplacian smoothing [38, 102, 121, 147, 199], the most frequently used scheme of smoothing, iteratively moves each interior point to the centroid of its neighbors. Freitag et al. [120] propose an optimization-based alternative to Laplacian smoothing which computes a new point placement that maximizes the minimum angle in adjacent triangles instead of using the centroid of the polygon.
APPENDIX B
DATA STRUCTURES FOR TINs

This appendix focus on a number of representations for TIN models. The choice of a particular data structure is important since it affects the representational fidelity to the data and the performance of the algorithms.

A complete representation must include not only the geometric data (e.g. physical dimensions and location in space), but also the mesh topology, which describes the adjacency between vertices, edges, and faces. The computation of gradient, for instance, may require visiting adjacent vertices. Therefore, a data structure that maintains the topology of these surface components explicitly can significantly improve the efficiency of the triangulation algorithms.

The topology of a mesh can be represented by an embedded graph, that is, a graph that has been mapped to a surface such that no two edges intersect, except at a vertex [146]. The graph nodes correspond to mesh vertices, edges correspond to element boundaries, and faces correspond to surface elements.

Several research areas have contributed to the development of data structures for the representation of spatial data, such as geometric modeling [12, 100], surface modeling [98, 169], and solid modeling [205, 222, 256]. A variety of data structures have been proposed for representing polyhedral models [15] and curved surface models [108, 326].

The following sections describe the most frequently used data structures for TINs.

B.1 Point-Based Data Structures

A simple data structure for a TIN consists of the vertices and their neighbors. A list of triangles contains pointers to the vertices that define the triangle (see Figure B.1).
B.2 Edge-Based Data Structures

a) Dual-edge data structure: the dual-edge structure was developed by Heller [155]. Each edge record stores a pointer to its endpoint, to its next edge in the triangle, and to its twin edge. The twin edge is geometrically the same as the current edge, however, it points at the opposite endpoint. Figure B.2 illustrates the dual-edge data structure.

Figure B.2: The dual-edge data structure.

b) Winged-edge data structure: the winged-edge representation was originally developed by Baumgart [15] to model polyhedra in computer vision. The topological information for each edge is composed of the adjacencies of the given edge with
other edges, vertices, and faces. The term *winged-edge* results from the arrangement representing the reference edge and its adjacent faces (see Figure B.4).

![Figure B.4: The winged-edge data structure.](image)

An edge record consists of links to its two vertices, to its two adjacent faces, and to its four neighboring edges, clockwise and counterclockwise. The vertex and face records contain links to an edge connected to them.

c) **Quad-edge data structure:** the quad-edge structure, developed by Guibas and Stolfi [142], was designed for representing subdivisions on any two-dimensional
manifolds\(^1\). It simultaneously represents both the subdivision and its dual, for instance, the Voronoi diagram and the Delaunay triangulation. Another advantage is that the quad-edge requires only two operators to build and modify arbitrary subdivisions.

Each quad-edge record [198] contains four directed edges corresponding to an undirected edge in the subdivision and to its dual edge (see Figure B.5). Each directed edge has two pointers: a pointer to the next counterclockwise edge around its origin, and a pointer to geometric and other nontopological information (such as the coordinates of its origin).

![Figure B.5: The quad-edge data structure.](image)

Figure B.6 shows a planar graph and its corresponding quad-edge structure. Each edge record is represented by a cross. Dashed lines represent the cycles of pointers corresponding to vertices, and solid lines represent the cycles of pointers corresponding to faces.

### B.3 Triangle-Based Data Structures

**a)** A triangle-based data structure proposed by Lawson [184] is shown in Figure B.7, where each triangle contains three pointers to its neighboring triangles, and three points to its vertices.

\(^1\)Dobkin and Lázló [80, 81] propose the facet-edge data structure to represent subdivisions of 3-manifolds, which is analogous to the quad-edge data structure. Brisson [29] proposes the cell-tuple data structure for representation of manifolds of arbitrary dimensions.
Figure B.6: (a) A subdivision and (b) its corresponding quad-edge data structure.

Figure B.7: (a) A subdivision and (b) its corresponding triangle-based data structure.
b) A list of triangles stores pointers to the vertices of the current triangle, and pointers to adjacent triangles. There is also a list of vertices containing x, y, and z coordinates for each vertex (see Figure B.8).

<table>
<thead>
<tr>
<th>#</th>
<th>X, Y, Z</th>
<th>#</th>
<th>NEIGHBORING VERTICES</th>
<th>ADJACENT TRIANGLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X₁ Y₁ Z₁</td>
<td>1</td>
<td>1 2 8</td>
<td>2 8</td>
</tr>
<tr>
<td>2</td>
<td>X₂ Y₂ Z₂</td>
<td>2</td>
<td>2 3 8</td>
<td>1 3</td>
</tr>
<tr>
<td>3</td>
<td>X₃ Y₃ Z₃</td>
<td>3</td>
<td>3 4 8</td>
<td>2 4</td>
</tr>
<tr>
<td>4</td>
<td>X₄ Y₄ Z₄</td>
<td>4</td>
<td>4 7 8</td>
<td>3 5 8</td>
</tr>
<tr>
<td>5</td>
<td>X₅ Y₅ Z₅</td>
<td>5</td>
<td>4 5 7</td>
<td>4 6</td>
</tr>
<tr>
<td>6</td>
<td>X₆ Y₆ Z₆</td>
<td>6</td>
<td>5 6 7</td>
<td>5 7</td>
</tr>
<tr>
<td>7</td>
<td>X₇ Y₇ Z₇</td>
<td>7</td>
<td>1 6 7</td>
<td>6 8</td>
</tr>
<tr>
<td>8</td>
<td>X₈ Y₈ Z₈</td>
<td>8</td>
<td>1 7 8</td>
<td>1 7</td>
</tr>
</tbody>
</table>

Figure B.8: Triangle-based data structure.