Fast polygonal Approximation of Height Fields and Application to Interactive Visualization and Watershed Segmentation

by

Yan CHEN

A thesis submitted to the
Department of Computer Science
in conformity with the requirements for
the degree of Master of Science

Bishop's University
Sherbrooke, Quebec, Canada
July 2011

Abstract

Height fields constitute an important modeling and visualization tool in many applications, and their exploration requires their display at interactive frame rates. However, it is hard to achieve even with high performance graphics computers due to their inherent geometric complexity. Thus, a fast polygonal approximation method is introduced in this work. It takes an image as input, typically a rectangular grid of elevation data $z = H(x, y)$, and approximates it with a mesh of triangles, also known as a triangulated irregular network, or TIN. In particular, greedy insertion algorithm is the most widely used refinement method to approximate height fields. We use greedy insertion algorithm with three importance measures: local error, global error and curvature. Thereby, the input point(s) with highest error(s) or highest curvature in the current approximation is(are) inserted as a vertex(vertices) in the triangulation to achieve any desired level of detail. All the meshes are constructed from the vertices using Delaunay triangulation.

The second part of the dissertation is about the study of a new method for the watershed segmentation. First, we present a review of the classical watershed transform definition and the existing watershed segmentation algorithms. Then, we propose a novel watershed algorithm based on the topological properties of the TIN associated with the input images. Since initial markers play a tremendous role in flooding simulation of watershed segmentation, the topological features, such as critical points, can be used to extract meaningful markers. The idea allows us to extract critical points (pits and peaks) as initial markers, and classify them into two categories: set of minimum markers and set of maximum markers.

The design, implementation, and comparison discussion of the critical-points based watershed algorithm are described in this thesis. Experiments are implemented to demonstrate practicability and flexibility by comparing with two classical algorithms. The experimental results demonstrate that the algorithm performs very well with different kinds of images.

Finally, we outline a new method for watershed segmentation that uses the polygonal approximation of the height field associated to the input image. The intuition behind the method is that in the approximation, only significant critical points are displayed. This allows the watershed segmentation to start from significant markers, which decreases the over-segmentation effet that characterizes this type of method. Experiments demonstrate that our method provides a good segmentation method when coupled with appropriate preprocessing techniques.

Acknowledgments

I would first like to thank my advisors, Dr. Madjid Allili and Dr. Layachi Bentabet, for their guidance throughout my master studies in Bishop's University. They gave me an opportunity to pursue a master degree in Computer Science, patiently taught me about the field of computer vision and image processing, and guided me towards exciting research questions. I appreciate all their contributions of time, ideas, and funding to make my Master experience productive and stimulating. The joy and enthusiasm they have for their research was contagious and motivational for me, even during tough times in the master pursuit. I am also thankful for the concerning, encouragement and support they have provided during tough times in Canada. They also have been tremendous role models for me as successful scholars and professors throughout my studies.

During my time at Bishop's University, I have had the good fortune to work and interact with several other exceptional people. In particular, I would like to thank Scosha Merovitz, Guodong Gao, Hui Zhang, and Xiaojun Zheng for helping me along the way and making my studies and research more enjoyable over these past two years. I would especially like to thank Scosha Merovitz, for being a supportive teacher and a considerate friend throughout hours working in Math Help Center. I had so much fun and also practised my oral English by working in Math Help Center.

Last, but not least, I would like to thank my family, professor Dazhi Meng and friends for their unwavering support. In particular, I have consistently drawn strength from my respectable parents and my little brother. And to my dear father, to whom this thesis is

dedicated, I am infinitely grateful. You are my sources of inspiration and any achievement, and my greatest teacher. Thank you, my dear Daddy.

Contents

List of Figures

Chapter 1

Introduction

This dissertation investigates the challenge of approximating height fields for interactive visualization, efficient ways of deciding markers of watershed segmentation, and a new method for watershed segmentation based on critical points extraction. These topics can be explained as follows:

First, we explore the use of height fields model and Delaunay triangulation. Our focus is fast polygonal approximation methods, which help generate simplified models of a height field from the original model. Next we describe greedy insertion algorithm, and three criteria to measure the accuracy of the approximation. To visualize efficiently and realistically, Phong shading, Grouraud shading and adaptive Phong shading are used. In the second part of the dissertation, we propose a novel watershed segmentation algorithm based on critical points extraction. Two triangulation selection approaches are introduced to ensure critical points satisfy maintain the topological properties of the interpolated surface. Critical points (pits and peaks) are used as initial markers to simulate an upward-and-downward flooding process. Finally, we also argue that height fields approximation and watershed segmentation can be linked together for three-dimensional mesh segmentation.

The remainder of this chapter provides the motivation and an overview of the thesis structure for the conducted research.

1.1 Motivation

A height field is a set of height samples over a planar domain. Terrain data, a common type of height field, is used in many applications, including flight simulators, ground vehicle simulators, and computer graphics for entertainment. Our primary motivation is to render height field data rapidly and with high fidelity. However, for terrains of any significant size, rendering the full model by brute force method is prohibitively expensive. For example, the 2,000,000 triangles in a $1,000 \times 1,000$ grid take about seven seconds to render on current graphics workstations, which can display roughly 10,000 triangles in real time (every 30th of a second). Typical workstations and personal computers are considerably slower than graphics workstations and will remain so for the foreseeable future. More fundamentally, the detail of the full model is highly redundant when it is viewed from a distance, and its use in such cases is unnecessary and wasteful. Many terrains have large, nearly planar regions which are well approximated by large polygons. Ideally, we would like to render models of arbitrary height fields with just enough detail for visual accuracy. Additionally, in systems which are highly constrained, we would like to use a less detailed model in order to conserve memory, disk space, or network bandwidth.

To render a height field quickly, we can use multi-resolution modeling, preprocessing it to construct approximations of the surface at various levels of detail [1]. When rendering the height field, we can choose an approximation with an appropriate level of detail and use it in place of the original. The various levels of detail can be combined into a hierarchical triangulation [2, 3].

On the other side, the intuitive idea underlying watershed segmentation comes from geography: it is that of a landscape or topographic relief which is flooded by water, watersheds being the divide lines of the domains of attraction of rain falling over the region. When simulating this process for image segmentation, it is easy for us to think of a 3-dimensional height field. Obviously, the idea behind the watershed segmentation is to consider the gray

levels of the image as a height field, and imagine placing a drop of water onto each pixel. The water will follow the terrain slope until it reaches the bottom of a valley. The segmentation is done by grouping pixels whose water drop end up in the same basin, that is, those that belong in the same watershed. The idea just described of following the terrain slope is one method of implementing the watershed algorithm. A second way is to flood the image terrain with water starting at the lowest point. As the water rises, new basins will begin to flood and watersheds will merge. The user can control at what point to stop the flooding and thus where to end the segmentation.

One of the main drawbacks of the flooding watershed algorithm is that it tends to create basins that only correspond to minima markers. According to the theory of differential topology, critical points on a smooth surface satisfy the topological integrity, i.e. the Euler formula. Thus, a solution for this could be to get critical points (pits and peaks) of this height field as markers. Then we can segment the minima regions and the maxima regions, obtaining watershed lines by starting at the lowest points and the highest points. Additionally, the traditional watershed flooding transform does not work well for different kinds of markers. Thus, it is necessary to design a new watershed algorithm based on critical points extraction. Furthermore, the watershed segmentation can be associated to the greedy insertion process of the height field approximation. Our idea is to segment the approximated height field from the topological perspective. We believe that the new method will help solve the problem of over-segmentation.

In this thesis, we will focus on height fields approximation and visualization, watershed segmentation based on critical points extraction, and a new watershed segmentation method on approximated height fields, as generally illustrated in Figure 1.1. It is an attractive application to image segmentation and approximated height fields visualization.

1.2 Thesis Outline

Based on our motivation and objectives (Section 1.1), this thesis is organized as follows:

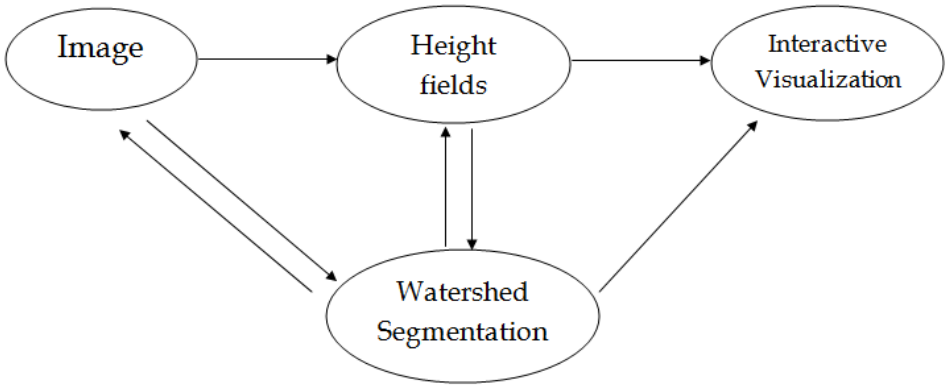


Figure 1.1: Overview of Research Work

In Chapter 2, we introduce the terrain height fields model, the concepts of triangulation, especially Delaunay triangulation, refinement simplification method, greedy insertion algorithm and three important selection measures. Regarding interactive visualization of fast polygonal approximations, our contribution is to combine the Computational Geometry Algorithms Library (CGAL) [4] with this research, and focuses on the effect of interactive visualization, such as human input, response time and shading. The section about importance measures and shading methods also describes some novel work carried out in this research.

In Chapter 3, we propose a new watershed segmentation based on critical points extraction. In particular, the conceptual watershed as a flooding simulation is studied. Two classical watershed methods are presented, implemented, and improved by image processing tools, such as noise reduction and points merging. A detailed description of critical points extraction techniques is presented. Additionally, we propose two triangulation selection approaches for ensuring critical points to satisfy topological integrity and eliminating ambiguities. Furthermore, the classical flooding algorithm is improved to adapt to two different types of markers for watershed segmentation. Experimental results demonstrate that the novel watershed segmentation algorithm outperforms other classical methods.

In Chapter 4, we propose a way of segmenting approximated height fields based on our new watershed algorithm. By comparing the segmentation effects from approximated height fields and the original image, it is obvious to see that watershed segmentation on height fields approximation produces better results. Experimental results demonstrate that it is a simple and most efficient method to segment large size images.

In Chapter 5, we summarize the primary findings of this work and provide an outlook for future work.

Each chapter ends with synopsis concluding the addressed aspects.

Chapter 2

Height Fields Approximation and Interactive Visualization

Our goal in this chapter is to approximate and visualize height fields in a simple application from raster images.

2.1 Preliminaries

In this Chapter, we introduce the most common rapid polygonal techniques for height fields approximation and implement interactive visualization using OpenGL. To obtain simplified representations of height field models, three importance measures in greedy insertion simplification method are used to reduce its natural complexity and size. The content of this chapter is primarily based on the results presented in Michael et al [5]. We begin by introducing basic definitions and algorithms of height fields and Delaunay Triangulation.

2.1.1 Height Fields

Height fields are recognized as terrain models that belong to a class of objects which have very complex geometry. The concept of height field also is the most simple and intuitive form to model a terrain. A height field is a bidimensional set of elevation values which can be expressed mathematically by a function $H : D \in R^2 \rightarrow R$, called elevation function (Figure 2.1).

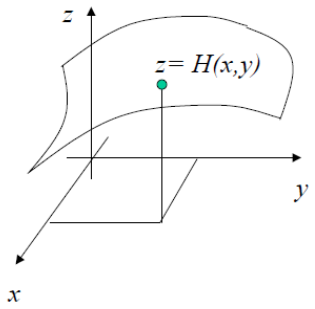


Figure 2.1: Terrain Height Field

Generally, a height field problem is characterized by a grid of points in x and y each with an associated z height. Suppose that the heights are given by z through a function $z = H(x, y)$, where x and y are the points on a two-dimensional surface such as a rectangle. Thus, for each x, y, we get exactly one z. Such three-dimensional surfaces are called height fields. Although not all surfaces can be represented this way, height fields still have many applications. For example, if we use an x, y coordinate system to give positions on the raster image, then we can use such a function to represent the grey value as the height at each pixel location. In this situation, such a function H is known only discretely and is represented as a set of grey level data of the form $z_{i,j} = H(x_i, y_j)$. The height field has led to a series of applications in image processing.

However, there is one drawback to height fields: they take a large amount of memory, as each height must be represented for a significant size. In an exploration, the application needs to render height fields data fast and with high fidelity, so the user can interact and make any desired changes in real time. Ideally, we can preprocess it to construct approximations of the surface at an appropriate level of detail and use it in place of the original. That is to say, simplified models (Figure 2.2) of height fields would need to be created from the original model with just enough details for visual accuracy.

Additionally, we would like to use height fields to devise an image segmentation algorithm and interactive visualization by extracting correct topological features.

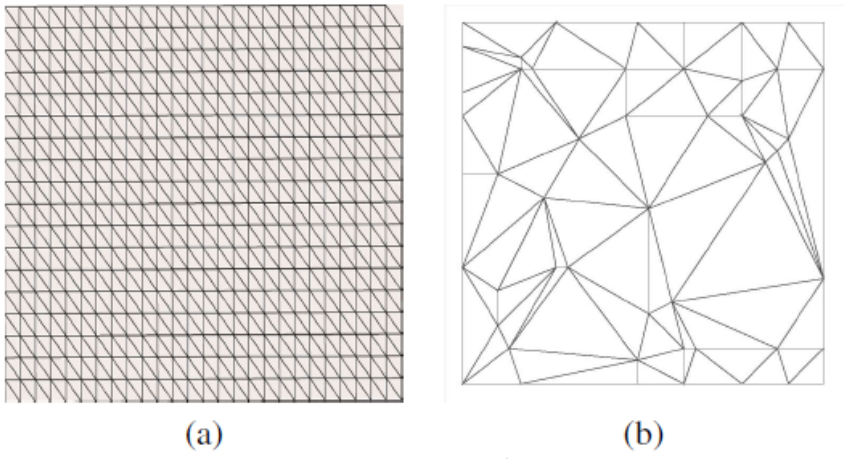


Figure 2.2: (a) Full model of a height field. (b) Simplified model of a height field.

2.1.2 Delaunay Triangulation

Most polygonal surface simplification methods employ triangles as their approximating elements when constructing a surface [6]. For height fields and parametric surfaces, there is a natural two-dimensional parameterization of the surface. Basic triangulation methods are described in a two-dimensional domain, or in a three-dimensional domain where the height z is a function of x and y.

The most popular triangulation method that does not use height values is Delaunay triangulation. The Delaunay triangulation algorithm is under the advanced computational geometry domain, and it was invented by Boris Delaunay. A Delaunay triangulation [7] (DT) which is used in generating well-shaped triangulations for a set P of points in the plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation, tending to avoid skinny triangles (Figure 2.3).

In this thesis, our focus is to use two-dimensional incremental Delaunay triangulation method to insert a vertex A with the largest error, locate its containing triangle, or, if it lies on an edge, delete that edge and find its containing quadrilateral, as illustrated in

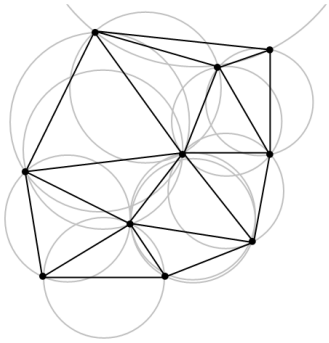


Figure 2.3: A Delaunay Triangulation in the Plane with Circumcircles

(Figure 2.4) [8, 9]. Add spoke edges from A to the vertices of this containing polygon. All perimeter edges of the containing polygon are suspect and their validity must be checked. An edge is valid if and only if it passes the circle test: if A lies outside the circumcircle of the triangle that is on the opposite side of the edge from A. All invalid edges must be swapped with the other diagonal of the quadrilateral containing them, at which point the containing polygon acquires two new suspect edges. The process continues until no suspect edges remain. The resulting is incremental Delaunay triangulation.

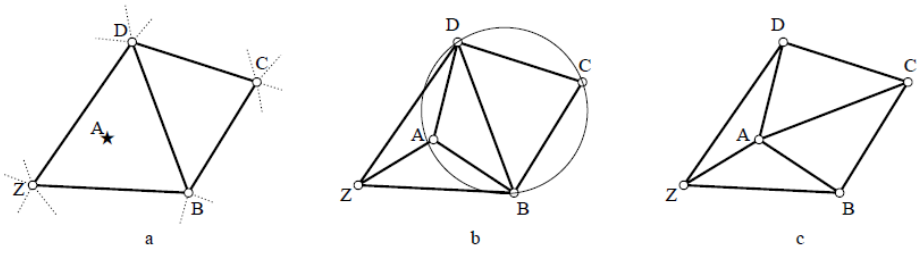


Figure 2.4: Incremental Delaunay triangulation: a) Point A is about to be inserted. Spoke edges from A to the containing polygon ZBD are added. b) The quadrilateral around suspect edge BD is checked using the circle test. The circumcircle of BCD contains A, so edge BD is invalid. c) After swapping edge BD for AC, edges BC and CD become suspect. The polygon ZBCD is the only area of the mesh that has changed.

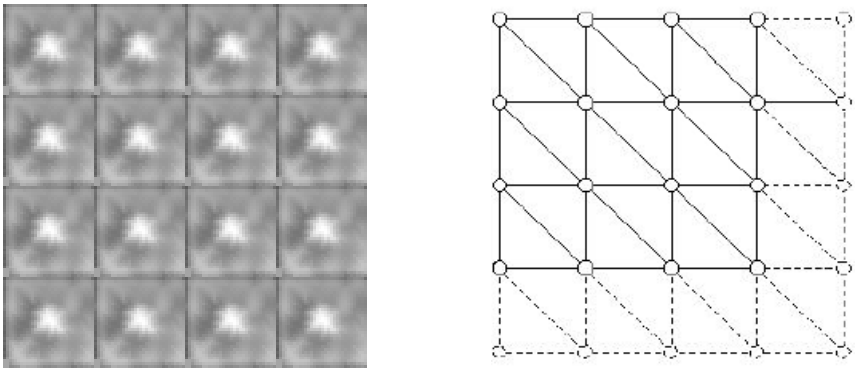
Thereby, we turn to CGAL Open Source Project for easily access to efficient and reliable Delaunay Triangulation. The class $Delaunay\_triangulation\_<Traits, Tds>$ of CAGL in

the form of a C++ library is designed to represent the Delaunay triangulation of a set of data points in the plane. It provides a data structure to store a two-dimensional triangulation that has the topology of a two-dimensional sphere, acts as a container for the vertices and faces of Delaunay triangulation and provides basic combinatorial operations on the triangulation.

2.2 Brute Force Method

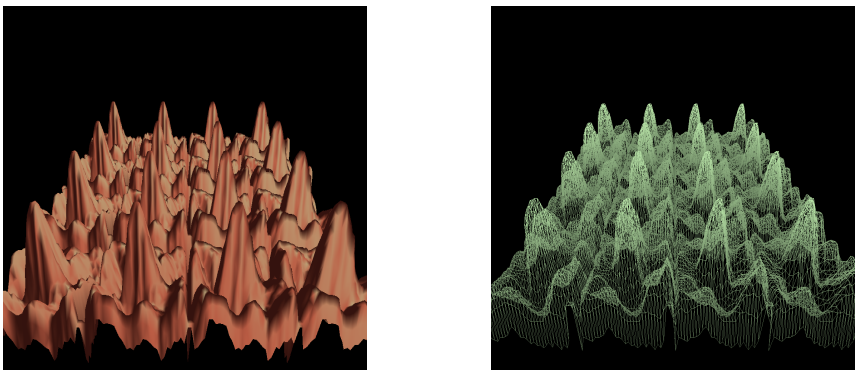
The first approach and the easiest way to build a height field is using a brute force method [10] by selecting all pixels from the raster image. The algorithm calls a method $SelectAllVertex()$ that goes through every pixel of the image and creates the mesh where every four vertices represent two triangles (Figure 2.5).

The method of brute force consumes a lot of computing time and memory although it is very easy to implement. Figure 2.5 is an example of an input raster image and its mesh of triangles by brute force method. In Figure 2.6, we generate a height field using a brute force method. The image file is a 128×128 image, and the result is a mesh of 32258 triangles (Figure 2.6).



(a) (b)

Figure 2.5: (a) Example of a raster image. (b) Mesh of Triangles by the Brute Force method.



(a) (b)

Figure 2.6: (a) Height Field representation of Figure 2.5. (b) Mesh of Triangles by the Brute Force method.

2.3 Fast Polygonal Approximation Method

To obtain a fast polygonal approximation of height fields, simplification methods are some of the techniques used to reduce its natural complexity. The intuitive idea behind these methods basically is to achieve the goal of a coarse-to-fine polygonal construction scheme by building progressively toward a more accurate approximation by increasing the number of vertices. In order to choose which points to add to the approximation, refinement methods rank the available input points using some importance measures.

2.3.1 Approach

Let $H : D \in R^2 \rightarrow R$ be a height field and $\epsilon \geq 0$ a tolerance value. The refinement problem consists of determining a subset $Q = \{q_1, q_2, \ldots, q_n\} \in D$ that decreases the functional $d(H - TS_Q) < \epsilon$, where d is a distance that measures the error between the original height field H and the approximated surface TS_Q determined by triangulating the set of points Q. Moreover, in mathematical point of view, given a discrete two-dimensional set of samples H of some underlying surface on a rectangular grid at integer coordinates, a height field will be reconstructed from H by triangulating its points. Then, a simplified model of a height field should accurately approximate the original model, using as few triangles

as possible, and the process of simplification should be as rapid as possible. Abstractly, the reconstruction operator T maps a function defined over a scattered set of points in a continuous domain (a function such as $z = H(x, y)$) to a function defined at every point in the domain. Hence, it accomplishes this by building a triangulation of the sample points and using this triangulated surface to give values to all points which are not part of the grid. If S is some subset of input points, then TS is the reconstructed surface, and $(TS)(x, y)$ is the value of the surface at point (x, y). Our goal is to find a subset S of H which, when triangulated, approximates H using as few points as possible, and allowing to compute the triangulation as quickly as possible.

2.3.2 Greedy Insertion

Among all of refinement methods, greedy insertion algorithms are the most widely used simplification method found in the literature [11]. The algorithms insert the point(s) of highest error and make irrevocable decisions as they go on each pass [12]. Methods that insert a single point in each pass are called sequential greedy insertion; and methods that insert multiple points in parallel on each pass are called parallel greedy insertion. The words "sequential" and "parallel" here refer to the selection and re-evaluation process, not to the architecture of the machine. The greedy insertion algorithm we now use is a brute force implementation of sequential greedy insertion and parallel greedy insertion with Delaunay triangulation. Tests in [5] showed constant threshold parallel insertion to be poorer than sequential insertion. When an insertion causes a small triangle to be created, it leads to a local change in the density of candidates. With the sequential method, smaller triangles are statistically less likely to have their candidates selected, because they will typically have smaller errors. In the parallel method, if the small triangles' candidate is over threshold, it will be selected, and a snowballing effect can occur, causing excessive subdivision in that area. Even on a simple surface like a paraboloid, which is optimally approximated by a uniform grid, the sequential method is better. On all tests we have run, sequential greedy

insertion yields better approximations than parallel greedy insertion.

The greedy insertion algorithm begins with some basic functions that query the Delaunay mesh and perform incremental Delaunay triangulation. The core of the algorithm is sequential greedy insertion, and is simple and unoptimized, which can be summarized by the following steps:

Step 1. Build an initial approximation of two triangles with the height field grid corners as vertices;

Step 2. Scan the unused points to find the one with the largest error, and call $Insert$ to add it to the current approximation;

Step 3. Recalculate errors at grid points, and repeat Step 2 until the termination conditions are stated as a function $Goal\_Met$.

2.3.3 Important Selection Measures

Within the basic framework outlined above, selection criterion is very important to the quality of approximation desired, and directly responsible for the order in which the points are selected to build the approximating subset. This is accomplished through the use of a measure that evaluates the contribution of a candidate point to the approximation error. Therefore, a good importance measure should be simple and fast, produce good results on arbitrary height fields, and use only local information. Here, three importance measures: local error, global error, and curvature, are explored and used in interactive visualization implementation.

Local Error Selection Measure

The first measure which we explored is simple vertical error. The importance of a point $H(x, y)$ is measured as the difference between the actual function and the interpolated approximation

$$|H(x, y) - (TS)(x, y)| \tag{2.1}$$

at that point. This difference is a measure of local error. Intuitively, we would expect that eliminating such local errors would yield high quality approximations, and it generally does. This measure also meets the other criteria suggested earlier: it is simple, fast, and uses only local information.

According to Michael et al [5], the local error measure yields the best results among all three measures. Nevertheless it presents shortcomings. They become evident when we apply this selection criterion to a terrain which has regions with different levels of variability.

Global Error Selection Measure

The global error, or sum of errors over all points, also is as an importance measure. At every point, we compute the global resultant error of a new approximation formed by adding that point to the current approximation, measured as

$$\sum_{x,y} |H(x,y) - (TS)(x,y)|. \tag{2.2}$$

Then the point that produces the smallest global error are merely selected. We expected that this "more intelligent" error measure would yield higher quality results than the local error measure, but at a penalty in speed. As matter of fact, greedy insertion with vertical local error measure is considered to be the one that yields best results [5].

Curvature Selection Measure

Intuitively, curvature is the amount by which a geometric object deviates from being flat or straight in different cases. Peaks, pits, ridges, and valleys are known for their high curvature and visual significance. These observations suggest that curvature would be a measure of importance. Since H is a discrete function, we estimate its derivatives numerically using central differences (sobel or robert operators). Due to the fact that this measure of importance is independent of the current approximation, it is essentially a feature method [6]. Hence, it lends itself to an one pass approach: compute values for $|H''|$ at all points

and select the m points with the highest values. The Laplacian,

$$\frac{\partial^2 H}{\partial x^2} + \frac{\partial^2 H}{\partial y^2} \tag{2.3}$$

would be a good measure of curvature for functions of two variables. However, the Laplacian(2.3) turns out to be a poor measure, as it sums the curvatures in the x and y directions, and there could cancel each other, for example in a saddle. Luckily, the sum of the squares of the principle curvatures [13] has been explored to be a better measure. Therefore, it is used to measure curvature, which can be computed as the square of the Frobenius norm of the Hessian matrix(2.4):

$$(\frac{\partial^2 H}{\partial x^2})^2 + 2(\frac{\partial^2 H}{\partial x \partial y})^2 + (\frac{\partial^2 H}{\partial y^2})^2. \tag{2.4}$$

The algorithm is the fastest technique in height fields approximation.

2.4 Interactive Visualization

Now we could combine our understanding of projections and modeling to build an interactive application for height fields.

2.4.1 Data Structure

This project needs special data structures to keep a representation of a height field in a dynamic memory. The Computational Geometry Algorithms Library (CGAL), offers packages to build and handle various triangulations, such as 2-dimensional Delaunay triangulation. The package acts as a nest container for faces and vertices of Delaunay triangulation and provides basic combinatorial operation on the triangulation [4]. In our project,

$CGAL :: Cartesian < double >:: Point\_2\, Point$;

$CGAL :: Delaunay\_triangulation\_2<K> Delaunay$;

$CGAL :: Delaunay\_triangulation\_2<K> :: Face\_iterator\, Face\_iterator$; and

$CGAL :: Delaunay\_triangulation\_2<K> :: Vertex\_iterator\, Vertex\_iterator$.

are included to achieve our implementation goal.

Each triangular face gives access to its three incident vertices and three adjacent faces. Each vertex gives access to one of its incident faces and through that face to the circular list of its incident faces. The triangulation data structure provides the types Vertex and Face for the vertices and faces of the Delaunay triangulations, the type $Vertex\_handle$ and $Face\_handle$ which are models of the concept $Handle$ and through which the vertices and faces are accessed, iterators and circulators to visit all the vertices, edges and faces incident to a given vertex. The triangulation data structure also is responsible for the creation and removal of faces and vertices. It provides a function that gives the number of faces, edges and vertices of the triangulation.

The triangulation data structure also provides member functions to perform the following combinatorial transformations on triangulations [2]: flipping of two adjacent faces, addition of a new vertex splitting a given face (see Figure 2.7), addition of a new vertex splitting a given edge and a new vertex raising by one the dimension of a degenerate lower dimensional triangulation, removal of a vertex incident to three faces and a vertex lowering the dimension of the triangulation.

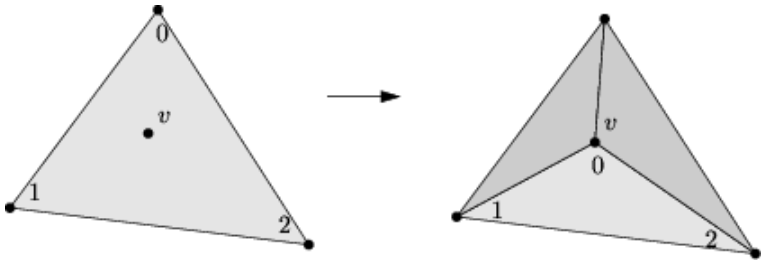


Figure 2.7: Insertion of a new vertex, splitting a face

Another vertex structure is defined to contain the three coordinates in three dimensional space and the normal vector at this point in the implementation. Calculating Normal vectors is explained in section 2.3.2.

2.4.2 Shading

Shading is a technique performed during height field rendering. With shading, it is possible to alter a color based on its angle to lights and its distance from lights to create a photo realistic effect. Surface shading depends not only on the position of the triangles and the light source, but the observer's viewpoint.

Phong Shading

The OpenGL library uses Phong shading model by default [14]. Phong shading linearly interpolates a normal vector across the surface of the triangle. The normal vector is calculated from the normals of the three given vertices. In the rendering, the final pixel color is calculated with the surface normal interpolated and normalized at each pixel, and then used in the Phong reflection model [15]. Since the Phong reflection model is more complicated, then we focus on the calculation of the normal vector at each vertex.

The calculation of the normal vector for every vertex is essential for the shading model explained above. A height field is a surface S over a two-dimensional domain. We may assume that it is a graph of a differentiable function $H = f(x, y)$, where the function H represents the gray value associated to the point (x, y) in the planar domain. Normal vector n at point P is the normal vector to the tangent plane of the surface at this point. A tangent plane at any given point (x, y, z) of the surface S is spanned by the tangent vector $V_x(1, 0, \frac{\partial H}{\partial x})$ and $V_y(0, 1, \frac{\partial H}{\partial y})$ (In Figure 2.8, V_x and V_y are shown as r_u and r_v). Therefore, an outer normal vector n to the tangent plane at (x, y, z) is given by the cross product of V_x and V_y, that is

$$n = V_x \times V_y = \big(-\frac{\partial H}{\partial x}, -\frac{\partial H}{\partial y}, 1\big). \tag{2.5}$$

In the context of height fields, we only know sampled values of x, y, and z. Therefore, we use classical derivative operators in image processing to compute values of $\frac{\partial H}{\partial x}$ and $\frac{\partial H}{\partial y}$,

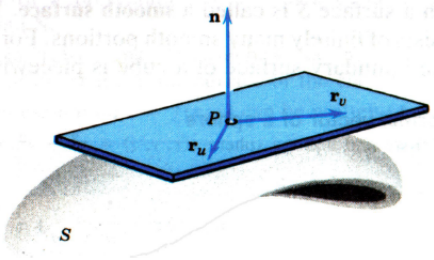


Figure 2.8: Normal vector of a tangent plane.

that is

$$\frac{\partial H}{\partial x} \simeq H(x+1,y) - H(x,y), \tag{2.6}$$

$$\frac{\partial H}{\partial y} \simeq H(x,y+1) - H(x,y). \tag{2.7}$$

In addition, based on various approximations of the two-dimensional gradient, first-order derivatives of a digital image also can employ Roberts cross-gradient operators and Sobel operators, which are

$$\frac{\partial H}{\partial x} \simeq H(x+1,y+1) - H(x,y), \tag{2.8}$$

$$\frac{\partial H}{\partial y} \simeq H(x+1,y) - H(x,y+1). \tag{2.9}$$

$$\begin{aligned}\frac{\partial H}{\partial x} \simeq\ & H(x+1,y-1) + 2*H(x+1,y) + H(x+1,y+1) \\ & -(H(x-1,y-1) + 2*H(x-1,y) + H(x-1,y+1)),\end{aligned} \tag{2.10}$$

$$\begin{aligned}\frac{\partial H}{\partial y} \simeq\ & H(x-1,y+1) + 2*H(x,y+1) + H(x+1,y+1) \\ & -(H(x-1,y-1) + 2*H(x,y-1) + H(x+1,y-1)).\end{aligned} \tag{2.11}$$

Before computing the derivatives, the image is smoothed using a Gaussian kernel to decrease the effect of noise.

Gouraud shading

From an OpenGL perspective, Gouraud shading is deceptively simple. Gouraud shading defines the normal at a vertex to be the normalized average of the normals of the triangles that share the vertex [15]. We need to set the vertex normals correctly and apply our lighting model. In our implementation, the vertex normal is given by:

$$n = \frac{n_1 + n_2 + \ldots + n_i}{|n_1 + n_2 + \ldots + n_i|}. \tag{2.12}$$

Here, the normalized normal vector (n_1, n_2, ..., n_i) of the triangle plane is given by:

$$n_i = \frac{\overrightarrow{AB} \times \overrightarrow{BC}}{\| \overrightarrow{AB} \times \overrightarrow{BC} \|}. \tag{2.13}$$

Note that A, B, and C are the vertices in the neighboring triangle plane in the counter-clockwise direction.

Adaptive Phong Shading

For the approximated height field in our implementation, most vertices have different approximated heights from the original ones. Accordingly, shading effect should work on an approximated height field surface. The actual intensity pixel for most vertices should be replaced by an approximated pixel value by getting the actual height. Therefore, to adapt to the Phong shading, a new shading way of calculating the normal vector for each vertex is to use the approximated height to compute the first derivative.

2.4.3 Experimental Results

The algorithm and shading model described in the previous sections are used to implement height fields approximation. This section shows its user interface and experimental results of three input images.

Figure 2.9 shows a dialog with buttons that could change vertex insert type, select importance selection measure, display shading styles and quit. Vertex insert type buttons include One Point, 5% Points, and 15% Points. Importance measure buttons include local

error and curvature. Display shading styles cover Gouraud shading, adaptive Phong shading (namely, Image concept 1), and traditional Phong shading (namely, Image Concept 2). The edit control in Figure 2.10 can set a threshold value and control the maximum local error difference when approximating a height field using local error as its importance measure.

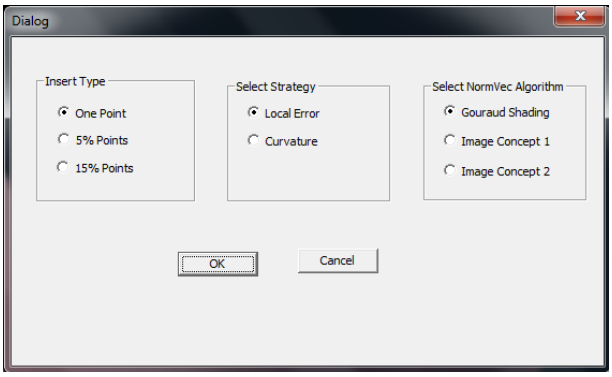


Figure 2.9: Height fields Interactive Visualization Dialog

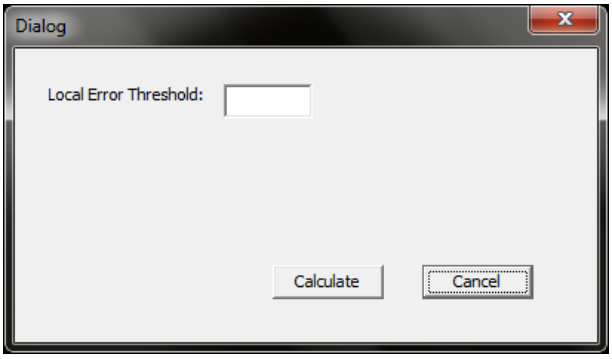


Figure 2.10: Height fields Interactive Visualization threshold Dialog

The first example is a picture of 32 $\times$ 32 pixels, as shown in Figure 2.11(c). Brute force method generates a height field of 1922 triangles for this image, as illustrated in Figure 2.11(a) and 2.11(b). Approximations in Figure 2.11(d) and Figure 2.11(e) reach a local maximum error 10. Figure 2.11(f) and Figure 2.11(g) use 15% vertices with highest curvature. Figure 2.11(h) and Figure 2.11(i) are the second approximation by using 15% vertices with highest curvature.

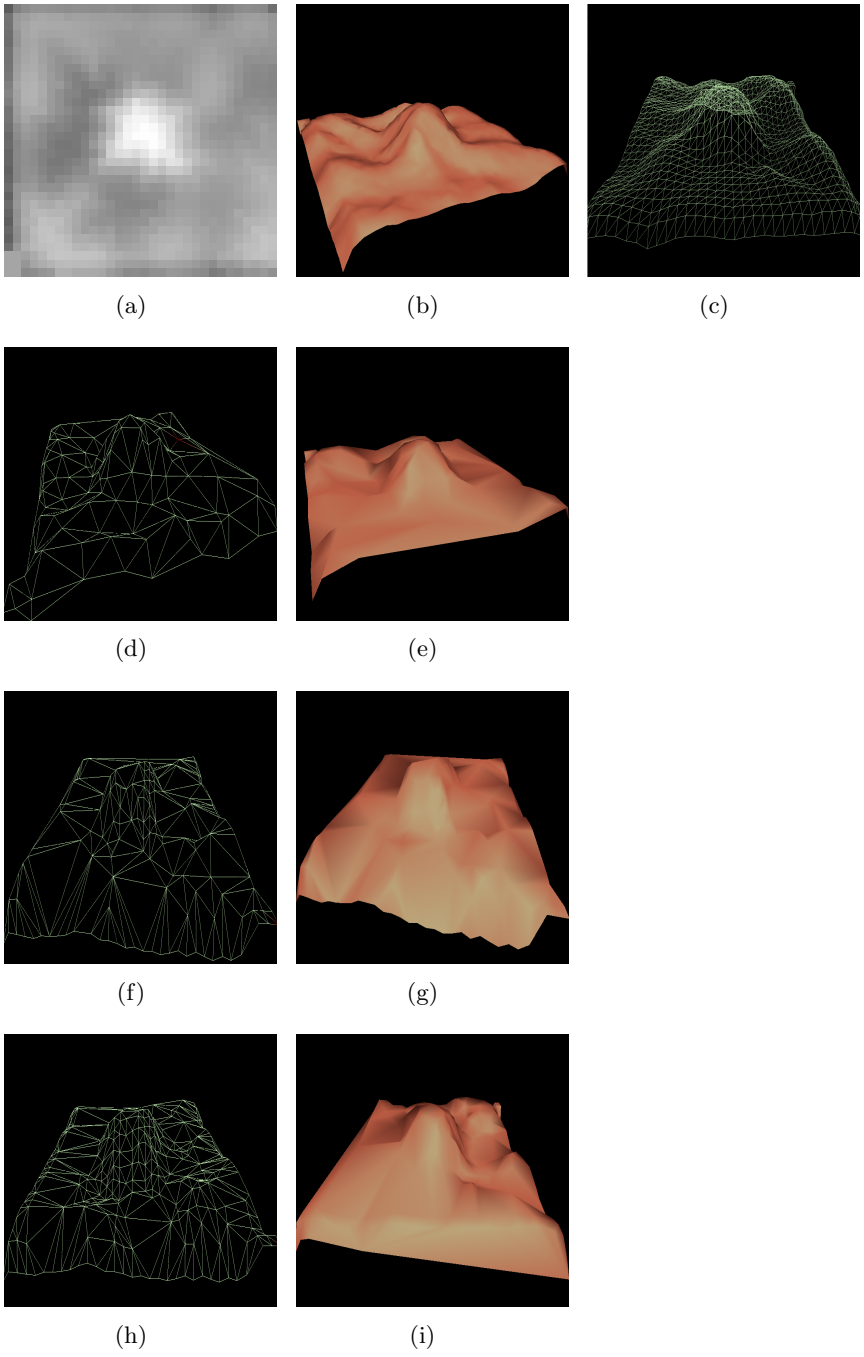
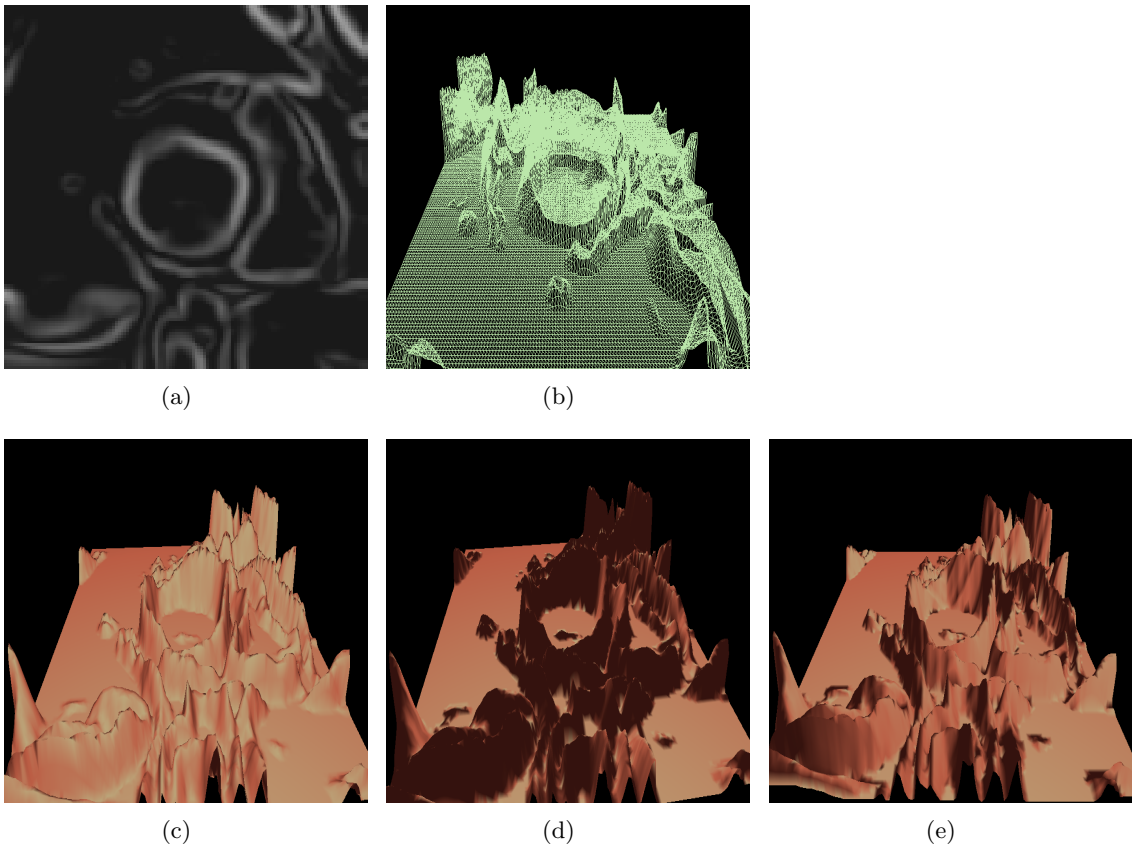


Figure 2.11: Approximation of a 32 × 32 image and their associated height fields displays. (a) Original image. (b) Full triangulation. (c) Wire-frame image of its triangulation. (d),(e) Approximation with local maximum error 10. (f),(g) First approximation using 15% vertices with highest curvature. (h),(i) Second approximation using 15% vertices with highest curvature.

The Second image is shown in Figure 2.12(a), which is like a tube. It is a picture of 128 $\times$ 128 pixels. Brute force method generates a height field of 32258 triangles as illustrated in Figure 2.12(b). The height fields are rendered in Gourand shading, adaptive Phong shading and traditional shading respectively. The results are presented in Figure 2.12(c), 2.12(d) and 2.12(e). The experimental results demonstrate that Gourand shading and traditional shading could yield better shading effects than the adaptive shading.



(a) (b)

(c) (d) (e)

Figure 2.12: A height field approximation visualization
(a),(b) Original image. (c) Approximation of the height field with Gouraud shading. (d) Approximation with adaptive Phong shading. (e) Approximation with Phong shading.

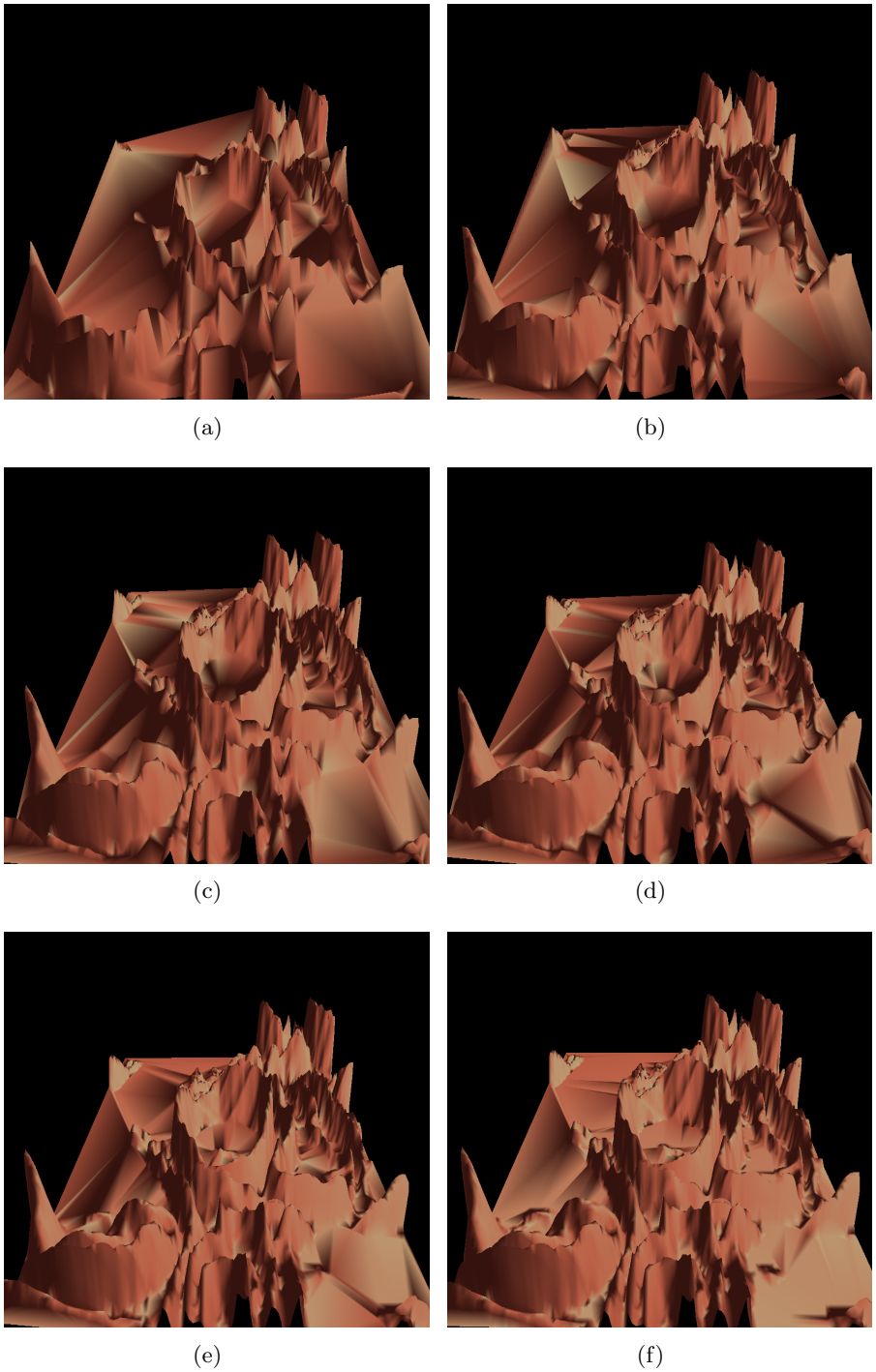
The display in Figure 2.13 employs Parallel Greedy Insertion method and curvature selection measure by inserting 15% vertexes five times. Height fields visualization are rendered in traditional shading. Experiment results show that the first approximation (Figure

2.13(a)) uses 2462 vertices and reaches local maximum error 267; second approximation (Figure 2.13(b)) uses 4554 vertices and reaches local maximum error 100; third approximation (Figure 2.13(c)) uses 6326 vertices and reaches local maximum error 43; fourth approximation (Figure 2.13(d)) uses 7835 vertices and reaches local maximum error 17; fifth approximation (Figure 2.13(e)) uses 9117 vertices and reaches local maximum error 4; sixth approximation (Figure 2.13(f)) uses 10206 vertices and reaches local maximum error 0. Obviously, the sixth height field gets a zero error approximation. The advantage of parallel greedy insertion is that it approximates height field quickly and efficiently.

The third image file is shown in Figure 2.14(a), which also is an experimental image in Chapter 3. Brute force method generates a height field of 32258 triangles. It is an image of 128 $\times$ 128 pixels with speckle noise (salt and pepper noise). The observation is that such images with noise yield inaccurate approximation by implementation directly in this project. Fortunately, there are many methods to reduce the speckle noise in image processing techniques. We expect to see the results that the input image is subjected to noise reduction before using our method. For the noise reduction, we could apply Median filter to the specific image many times. Just as expected, the noise reduction decreases the number of vertices needed for the approximation. In Figure 2.14(e), an approximation of original image uses 2730 vertexes to reach a desired local maximum error 19. However, using Median filter twice, an approximation that reach the same desired error only uses 343 vertices as illustrated in Figure 2.14(e). By contrast, this test shows that noise reduction helps yield better approximation. The height field with noise reduction needs much less vertices than the height field without noise reduction, and it consumes less time as well.

2.5 Summary

This chapter presents and compares the design of height fields approximation using brute force method and refinement simplification method. Fast polygonal approximation of height fields reduces the number of geometric primitives. Starting from a rough approximation, a

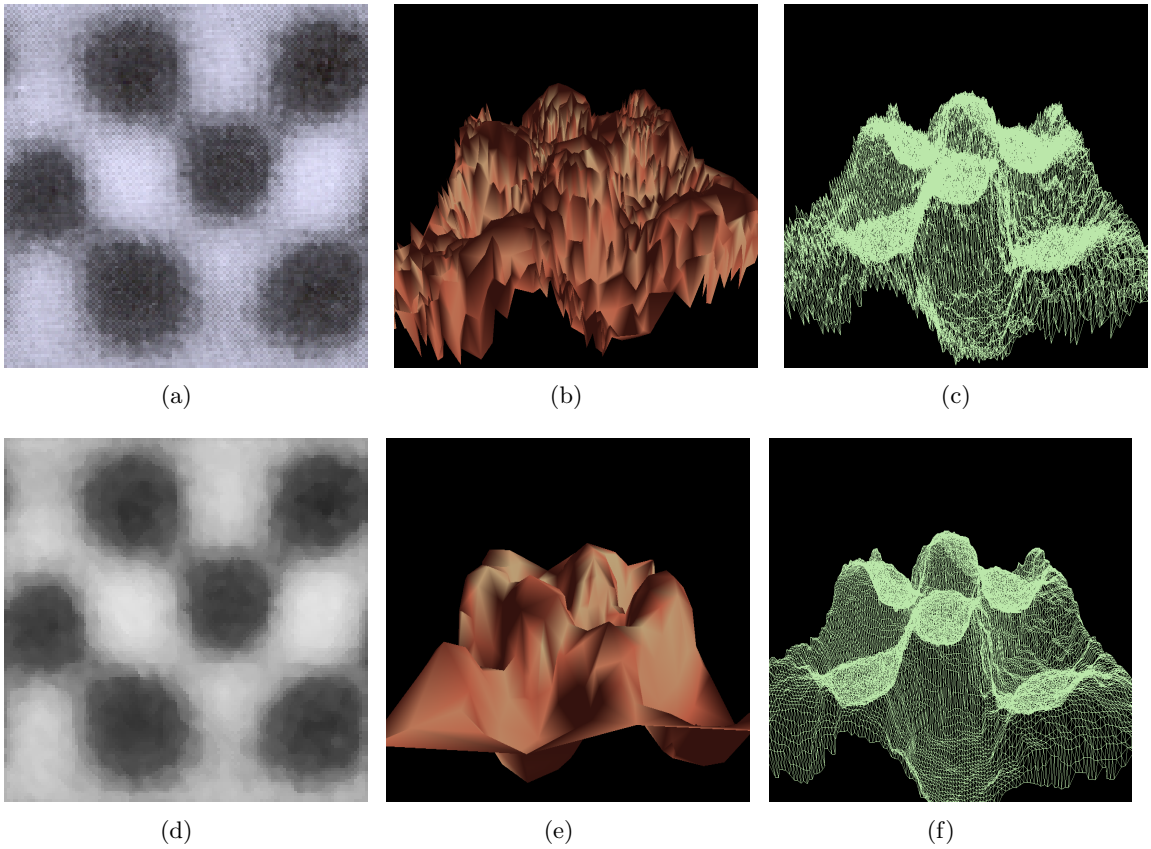


(a) (b)

(c) (d)

(e) (f)

Figure 2.13: Approximation by Parallel Greedy Insertion
(a) First approximation by inserting 15% vertices. (b) Second approximation. (c) Third approximation. (d) Fourth approximation. (e) Fifth approximation. (f) Sixth approximation.



(a) (b) (c)

(d) (e) (f)

Figure 2.14: Comparison of approximation between original and smoothed image (a), (c) Original image. (b) Approximation of the height field with local maximum error 19. (d), (f) Image smoothed using the median filter. (e) Approximation of the smoothed height field.

refinement process is operated until a desired level of detail is reached. So when rendering a height field, we can choose an approximation with an appropriate level of detail and use it in place of the original. Of course, various selection measures could contribute to an approximation jointly. In addition, both sequential and parallel greedy insertion are tested in previous work. Experimental results suggest that constant parallel insertion is poorer than sequential insertion at a certain error level. However, parallel method implementation could achieve a faster approximation than the sequential method. In the part of interactive visualization, three shading methods are tested, which are Gouraud shading, classical shading and adaptive shading. Gouraud shading is found by averaging the surface normals of polygons which meet at each vertex. The idea of the adaptive shading comes up from local error measure and classical shading calculation. Unfortunately, test results show that the adaptive shading could not achieve a better shading result with fidelity as the other two shading methods.

Chapter 3

Watershed Segmentation on Images

3.1 Background

Segmentation means breaking down an existing structure into meaningful connected subcomponents [16]. Watershed segmentation is a common way of automatically separating or cutting apart regions with different color properties in image processing. It is based on the concept of topographic representation of image intensity. The major idea of watershed segmentation was originally proposed by Digabel and Lantuejoul [17] and later improved by Beucher and Lantuejoul [18] in the late 70's.

Watershed segmentation can be classified as a region-based segmentation approach [19]. The intuitive idea underlying this method comes from geography: it is that of a landscape or a topographic surface which is flooded by water, watersheds being the divide lines of the domains of attraction of rain falling over the region [20]. An alternative approach is to imagine the landscape being immersed in a lake, with holes pierced in local minima. Catchment basins will fill up with water starting at these local minima, and, at points where water coming from different basins would meet, dams are built [21]. When the water level has reached the highest peak in the landscape, the process is stopped. As a result, the landscape is partitioned into regions or basins separated by dams, called watershed lines. That is to say, a gray level image could be segmented into its constituent regions or objects

by following the watershed transform.

Thereby, segmenting an image by the watershed transform could be considered as a two-step process: first finds basins, then second watersheds by taking a set complement. In this chapter, we mainly focus on watershed segmentation as a flooding simulation. We evaluate and compare the performance of watershed segmentation with different ways of finding markers. Essentially, we consider the definitions of critical points on a continuous surface, together with their topological properties. We propose to extract the correct critical points (such as pits, peaks and saddles) of the corresponding height field to define markers for watershed segmentation. To be consistent with watershed transform, we present a modified watershed algorithm that fits the markers definition. Several images are used to evaluate the modified watershed segmentation algorithm.

3.2 Watershed as a Flooding Simulation

3.2.1 Basic Algorithm Definition

There are many watershed algorithms in the literature [21, 22, 23]. One of the best and most intuitive definitions of the watershed transform is the one based on the flooding simulation. Generally speaking, if we flood a surface from its minima and, if we prevent the merging of the waters coming from different sources, we partition the image into two different sets: the catchment basins and the watershed lines, as shown in Figure 3.1.

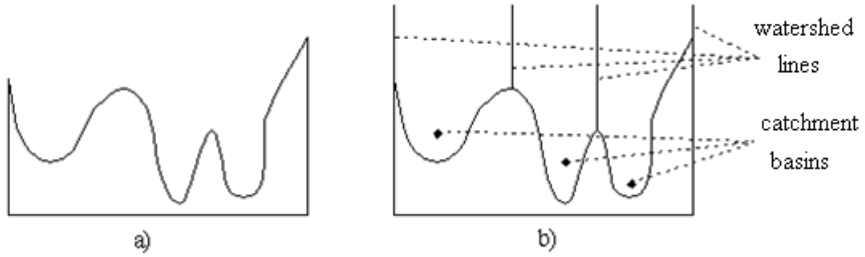


Figure 3.1: One-dimensional example of watershed segmentation. a) Gray level profile of image data. b) Watershed segmentation local minima of gray level (altitude) yield catchment basins and the watershed lines.

An algorithmic definition of the watershed transform by simulated flooding was given by Vincent and Soille [23]. Let $f : D \to \mathbb{N}$ be a digital grey value image, with h_{min} and h_{max} being the minimum and maximum value of f. Define a recursion with the grey level h increasing from h_{min} to h_{max}, in which the basins associated with the minima of f are successively expanded. Let X_h denote the union of the set of basins computed at level h. A connected component of the threshold set T_{h+1} at level $h+1$ can be either a new minimum, or an extension of a basin in X_h: in the latter case one computes the geodesic influence zone $IZ_{T_{h+1}}(T_h)$ of X_h within T_{h+1}, resulting in an update X_{h+1}. Let MIN_h denote the union of all regional minima at altitude h.

Definition 3.2.1 (Watershed by Flooding)**.** Let us define the following recursion:

$$\begin{cases} X_{h_{min}} = \{p \in D | f(p) = h_{min}\} = T_{h_{min}} \\ X_{h+1} = \text{MIN}_h \cup IZ_{T_{h+1}}(T_h), \qquad h \in [h_{min}, h_{max}) \end{cases}$$

The watershed $Wshed(f)$ of f is the complement of $X_{h_{max}}$ in D:

$$Wshed(f) = D \setminus X_{h_{max}}.$$

According to Definition 3.2.1, pixels with grey value $h' \leq h$, which are not yet part of a basin after processing level h, are merged with some basin at the higher level $h + 1$. Pixels which in a given iteration are equidistant to at least two nearest basins may be provisionally labeled as 'watershed pixels' by assigning them the label W. For an example of the watershed transform according to flooding simulation, see Figure 3.2, in which A, B, C, D are labels of basins, and W is used to denote watershed pixels (minima pixels in the input image are indicated in bold, as to be markers). Note the dependence on the connectivity. The example given in Figure 3.2, is for a 7×7 discrete image on the square grid with 8-connectivity. There are four local minima (the zeroes), so there will be four basins whose pixels are labeled A, B, C, D.

0	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	**0**	1	2	3
2	1	5	1	4	1	2
3	2	1	**0**	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	**0**	5

A	W	B	B	B	B	B
W	W	B	B	B	B	B
B	B	B	B	B	B	B
B	W	W	W	W	W	W
B	W	C	C	C	C	C
B	W	C	C	W	W	W
B	W	C	C	W	D	D

Figure 3.2: Watershed transform on the square grid with 8-connectivity, showing thick watersheds. (a): original image; (b): result according to flooding.

3.2.2 Meyer's Flooding Algorithm

On the basis of the classical watershed transform, Meyer and Beucher [24] introduced an algorithmic inter-pixel flooding definition of the watershed in the early 90's. The algorithm works on a gray scale image. During the successive flooding of the grey value relief, watersheds with adjacent catchment basins are constructed. This flooding process is performed on the gradient image, i.e. the basins should emerge along the edges.

Starting from a grey scale image F and a set M of markers with different labels (in our case, these will be the minima of F), it expands as much as possible the set M, while preserving the number of connected components of M. The watershed process can be summarized by the following steps:

Step 1. Select a set of markers M, where the flooding shall start. Each marker is given a different label;

Step 2. Insert every neighboring pixel x of each marked area in a priority queue, with a priority level corresponding to the gray level $F(x)$ of the pixel. Note that a point cannot be inserted twice in the queue;

Step 3. Extract a pixel x with the highest priority level from the priority queue. If the neighborhood $\Gamma(x)$ of the extracted pixel x that have already been labeled all have the same label, then the pixel x is labeled with their label. All non-marked neighbors that are not yet in the priority queue are put into the priority queue;

Step 4. Redo Step 3 until the priority queue is empty.

The non-labeled pixels are the watershed lines. The watershed lines set is the complement of the set of labeled points. Note that this algorithm does neither label nor propagate watershed pixels, which "stop" the flooding. Thus, the watershed lines produced by Meyer's algorithm are always thinner than lines produced by other watershed algorithms [25]. Normally this will lead to an over-segmentation of the image, especially for noisy images. Either the image must be pre-processed or the regions must be merged on the basis of a similarity criterion afterwards.

An example illustrating Meyer's flooding algorithm is given in Figure 3.3, where there are four minima, all with value 0. See Figure 3.3(b-l), in which A, B, C and D are labels of basins, and W is used to denote watershed pixels (in this and other figures to follow, markers in the input image are indicated in bold).

3.2.3 Watershed using Gradients

In the Meyer's flooding algorithm described above, the set of markers M where the flooding starts has a significant impact on the segmentation result. Traditionally, local minima of the image may be chosen as markers. However, it is difficult to automatically determine appropriate gray level threshold values under which a pixel is considered a minimum. In particular, if we flood the surface from its local image minima, the watershed lines would not be the final contours of the objects, but partitioning lines for different objects. Therefore, it is not possible to segment the image by flooding it starting from its local minima.

Intuitively, a very powerful gray-scale segmentation methodology can result from applying the watershed to the morphological gradient of the image to be segmented. The

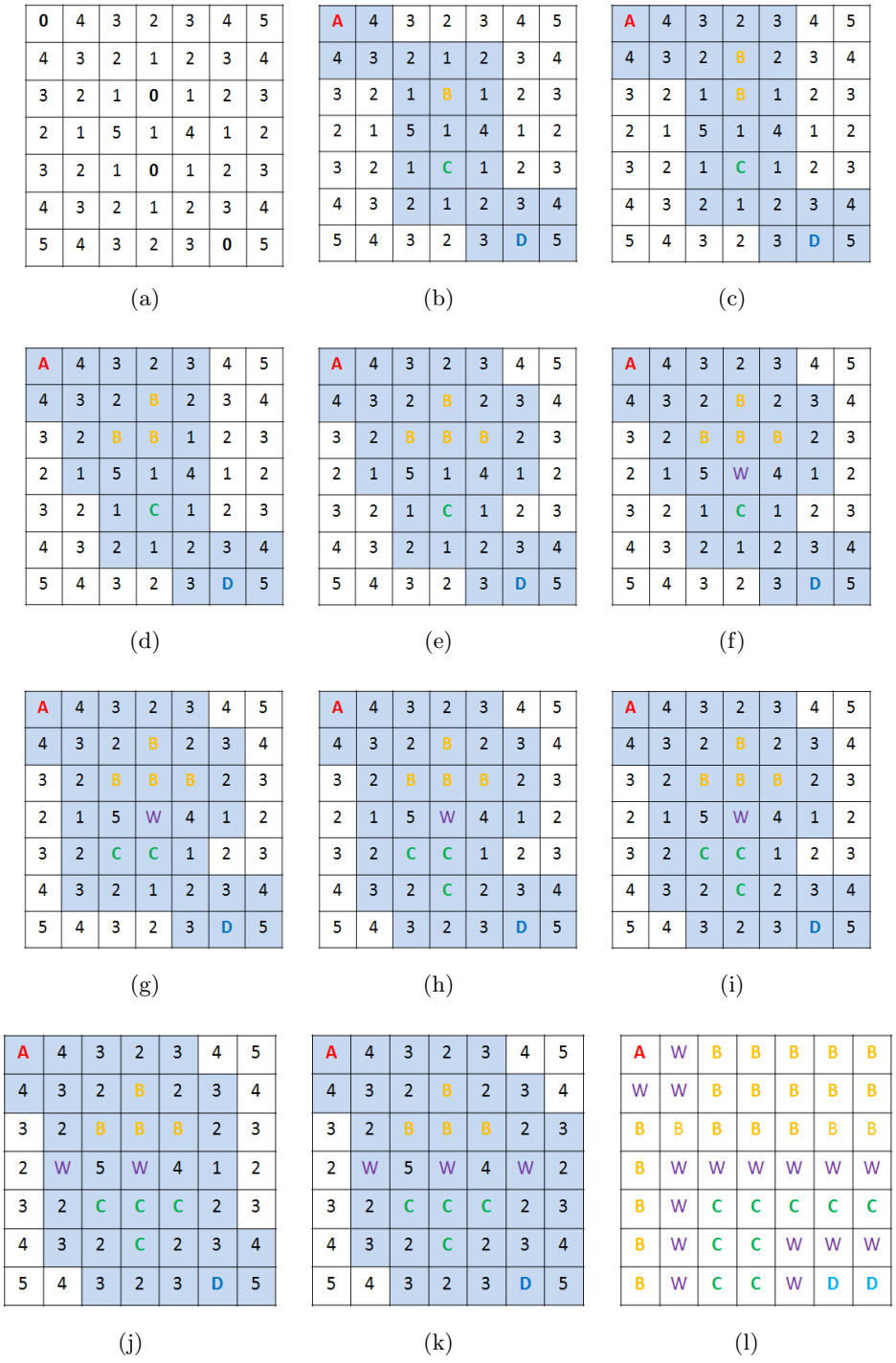


Figure 3.3: Watershed transform by Meyer's flooding algorithm on the 8-connectivity grid. (a) Original image. (b-k) Some labelling steps. (l) Final result.

gradient magnitude of a scalar function $f(x_1, x_2, x_3, \cdots, x_n)$ is denoted ∇f or grad(f). The gradient of f is defined to be the vector field whose components are the partial derivatives of f:

$$\nabla f = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n}). \tag{3.1}$$

From the above definition, gradient magnitude image has high gray level values along object edges, and low gray level values everywhere else. Ideally, the watershed transform would result in watershed ridge lines along abject edges by setting gradient minima as markers. Thus, the gradient magnitude is used often to preprocess a gray-scale image and to help get fine markers prior to using the watershed transform for segmentation.

In the implementation, we start by computing the gradient magnitude (3.1) of each pixel in the image to substitute for its gray level. Each pixel of a gradient image measures the change in intensity of that same point in the original image. Either the linear filtering methods, i.e. Roberts, Prewitt and Sobel methods, or a morphological gradient [26] can be used to get the gradient magnitude image. The gradient magnitude image defines a new height field on which watershed segmentation will be applied. A threshold value is computed from the gradient image and used to extract markers from the gradient minima. Each regional minimum originates a catchment basin of the final watershed transform. When we apply watershed transformation on the image gradient, the catchment basins will theoretically correspond to the homogeneous grey level regions of this image. Furthermore, the watershed ridge lines would be shown in the image by Meyer's flooding algorithm.

3.2.4 Limitations of Watershed Segmentation using Gradient Magnitude

Although the use of gradient images, described in Section 3.2.3, helps get the markers for the image regions, it produces an increased number of local minima. Consequently, too many watershed ridge lines do not correspond to the objects in which we are interested. This phenomenon is undesirable and it is referred to an over-segmentation in the literature [27]. Thereby, an over-segmentation is produced by this gradients algorithm. In particular,

it is noteworthy that gradient images amplify the noise contained in the original images [26]. Therefore, the watershed transform produces an important over-segmentation due to noise or local irregularities in the gradient image. In practice, although some improvement could be achieved by smoothing images or merging segmented regions, there are still some extraneous ridge lines, and it can be difficult to determine which catchment basins are actually associated with the objects of interest. The results obtained using watershed segmentation on gradient images will be presented in Section 3.4.2.

Solving the problems discussed above requires additional consideration and ideas. The next section describes a novel watershed segmentation technique based on critical points extraction to deal with over-segmentation.

3.3 Critical Points Extraction

3.3.1 Critical Point Analysis

In this section, we will develop an algorithm to find the critical points of a height field. The methods of extracting critical points from discrete elevation data have been extensively studied by researchers in the fields of computer graphics and geographical information system ($GISs$) [28, 29]. By critical points we mean peak (maximum), pit (minimum) and saddle points. As one moves higher, contours bounding a local maximum, called a peak, become smaller and smaller, ultimately becoming a point. Likewise, as one moves lower, contours bounding local minimum, called a pit, become smaller and smaller, ultimately becoming a point as well. However, in some cases, as one smoothly changes elevation, two contours meet at a single point, forming a single self-intersecting contour topologically [29]. That point, called a pass or a saddle, is neither a local minimum nor a local maximum. In general, peaks, pits, and passes are isolated from each other.

3.3.2 Critical Points Extraction Algorithm

This section explains how to extract the critical points from the discrete terrain data with topological integrity. Before going into details, we give the mathematical definitions of critical points [30] and Euler's formula [31, 32].

Assume that the terrain surface is represented by a single-valued function $z = f(x, y)$, where z is height in the Cartesian coordinate system formed by the $x-$, $y-$, and $z-$axes. $z = f(x, y)$ is a height function which gives the height of each point on a terrain surface. A point p of the function f is called a critical point of f if $\text{grad}f(p) = 0$, i.e.$\frac{\partial f}{\partial x} = 0$ and $\frac{\partial f}{\partial y} = 0$. Note that the topology of the cross-sectional contours changes at p while scanning the critical level $f^{-1}(f(p))$ from its upper side to its lower side. If a new contour appears at p, the critical point p is called a peak. If an existing contour disappears at p, the critical point p is called a pit. (A peak (pit) is higher (lower) than all other points in its neighborhood.) If a contour is divided or two contours are merged at p, the critical point p is called a pass. The critical point p is called non-degenerate if one and only one of the above topological changes occurs in the contour containing p.

To ensure that critical points extraction criteria have no ambiguities and maintain the topological integrity, it should follow from the theory of differential topology that critical points on a smooth surface satisfy the Euler formula. The Euler formula represents a topological invariant of smooth surfaces. Suppose that the terrain surface is a part of a sphere as illustrated in Figure 3.4. We consider a terrain surface with a virtual pit that is the local minimum of the terrain surface at the bottom of the sphere [33]. Then, the Euler formula states that the number of peaks $\#\{peak\}$, the number of passes $\#\{pass\}$, and the number of pits $\#\{pits\}$ satisfy the relation:

$$\#\{peak\} - \#\{pass\} + \#\{pit\} = 2. \tag{3.2}$$

The Euler formula (3.2) is consistent with the critical points and contour changes with respect to the height when we consider the virtual pit. To ensure that the extracted critical

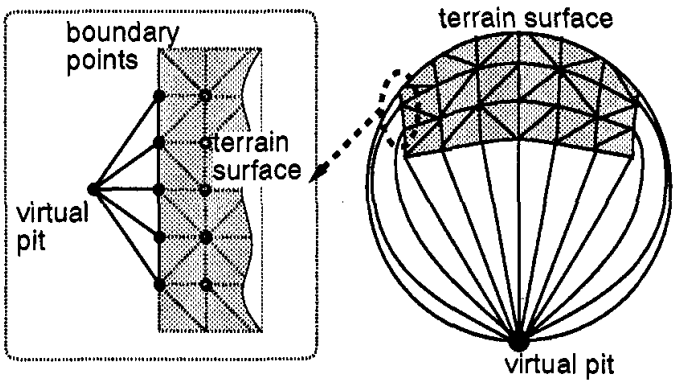


Figure 3.4: A terrain surface and a virtual pit on a sphere.

points satisfy the Euler formula, it is necessary to determine the contour changes according to the height. This means that we have to determine the interpolated surface from the sample points. For this purpose, we use triangulation. The contour changes depend on the manner in which we triangulate the sample points. In the case of grid points, the influence of the previous eight neighbors is reduced by choosing either of the two diagonals in each square of the chess board pattern.

To handle the boundary sample points and a virtual pit, a virtual pit is assumed to be a point of the height $-\infty$. After triangulating the sample points, the virtual pit is inserted to the circular list of the boundary points so that the virtual pit and two adjacent boundary points form a triangle. In this process, the virtual pit is considered as a point exterior to the sample region with respect to the $(x, y)-$ coordinates.

Suppose that all critical points are non-degenerate. Then, the neighbors of the point p are the points that are adjacent to p in the triangulated sample points. In our implementation, each point p has a circular list of neighbors in counter-clockwise order around p with respect to the (z, y)-coordinates. Each of the neighbors $p_i (i = 1, 2, ..., n)$ is scanned to see whether it satisfies the conditions of the critical points. When all the critical points are non-degenerate, the criteria of the critical points are as follows:

Peak	$\|\Delta_+\| = 0, \|\Delta_-\| > 0,$	$N_c = 0$
Pit	$\|\Delta_-\| = 0, \|\Delta_+\| > 0,$	$N_c = 0$
Pass	$\|\Delta_+\| + \|\Delta_-\| > 0,$	$N_c = 4$

Here, n is the number of the neighbors of p, Δ_i is the height difference between $p_i (i = 1, 2, ..., n)$ and p, Δ_+ is the sum of all positive $\Delta_i (i = 1, 2, ..., n)$, Δ_- is the sum of all negative $\Delta_i (i = 1, 2, ..., n)$, and N_c is the number of sign changes in the sequence $\Delta_1, \Delta_2, ..., \Delta_n, \Delta_1$.

3.3.3 Triangulation Selection

From the above section, triangulation can be used to ensure that the extracted critical points satisfy Euler's formula and maintain the topological properties of the interpolated surface. Triangulation of the lattice is equivalent to reconstructing a continuous surface from the lattice points from [34]. For each rectangle on the lattice, two triangular planar facets are created by joining of the rectangle's diagonals. It is that interpolation method which introduces the least number of new critical points on the reconstructed continuous surface. The fact that triangulation is equivalent to a particular reconstructed continuous surface is the reason any arbitrary triangulation of the lattice is consistent topologically.

Suppose that the grid point is labeled with the points clockwise around the rectangle A, B, C and D as shown in Figure 3.5, and the heights of these four points are denoted as $H(A)$, $H(B)$, $H(C)$ and $H(D)$ respectively. To triangulate the regular grid points, it is desirable to use a method similar to the Delaunay triangulation. Each square is then divided into two triangles with either of the two diagonals $A - C$ and $B - D$. Obviously, each rectangle use that diagonal which should produce the "flattest" pair of facets. Here, the obvious idea for us is to choose the diagonal so that the two divided triangles constitute the flatter surface. In this section, we introduce two different ways of constructing triangulation configuration for the sample points.

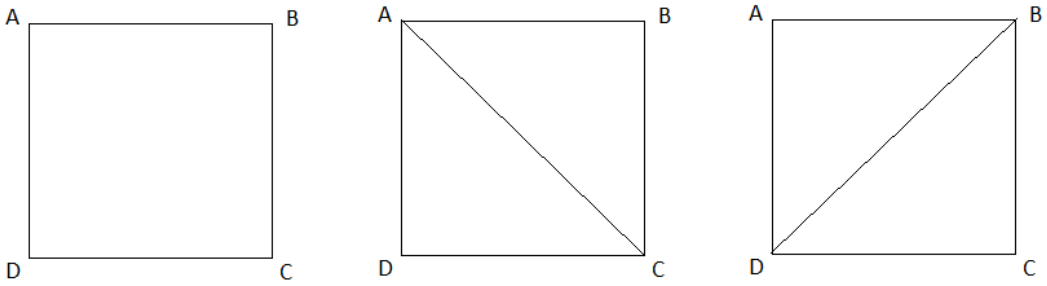


Figure 3.5: Rectangle and possible joined diagonal.

First approach: Triangulation to produce flattest surface

Takahashi [33] suggested to choose the diagonal that makes the smallest absolute angle between the normals to the triangles divided by the diagonal. Assume that N_{ABC}, N_{ACD}, N_{ABD} and N_{BCD} denote the normals of the triangles ABC, ACD, ABD and BCD respectively. Then, absolute angle θ_1 between the normals to the triangles divided by the diagonal $A-C$ is equal to:

$$\theta_1 = \arccos\left(\frac{N_{ABC} \cdot N_{ACD}}{|N_{ABC}| \cdot |N_{ACD}|}\right). \tag{3.3}$$

Absolute angle θ_2 between the normals to the triangles divided by the diagonal $B-D$ is equal to:

$$\theta_2 = \arccos\left(\frac{N_{ABD} \cdot N_{BCD}}{|N_{ABD}| \cdot |N_{BCD}|}\right). \tag{3.4}$$

Accordingly, the criteria that produces the flattest surface is:

If $\theta_1 < \theta_2$, select diagonal $A-C$;

If $\theta_1 > \theta_2$, select diagonal $B-D$;

If $\theta_1 = \theta_2$, select diagonal $A-C$ or $B-D$.

Second approach: Triangulation that minimizes the height curvature

In the second approach, we compute the curvature of the height to decide which configuration is flatter. According to the previous chapter, curvature is a measure of describing how sharply a curve bend. Therefore, it is wise to consider to choose the diagonal that makes

the smallest absolute curvature between the triangles divided by the diagonal. Laplacian curvature formula,

$$\frac{\partial^2 H}{\partial x^2} + \frac{\partial^2 H}{\partial y^2} \tag{3.5}$$

can be a good measure of curvature for continuous surfaces. Here, H is the height. In the following, we propose a novel method of selecting the two triangles to constitute the flatter surface based on the idea of curvature. The numerical curvature for the discrete lattice based on the Laplacian can be computed as follows:

If the lattice is divided by diagonal $A - C$, which is named Configuration 1, then, the curvature $Curv_1$ for Configuration 1 can be expressed by:

$$Curv_1 = |H(B) + H(D) - 2 * H(A)| + |H(B) + H(D) - 2 * H(C)|. \tag{3.6}$$

The expression above is the sum of the absolute curvatures along the two possible paths that go from triangle 1 to triangle 2 or vice versa. These paths start from a non-shared vertex and ends at a non-shared vertex. They both pass by a vertex that belongs to the shared edge. The curvature is minimum if

$$H(A) = H(C) = \frac{H(B) + H(D)}{2}, \tag{3.7}$$

which means that A, B, C, D lie on the same plane.

In a similar way, if the lattice is divided by diagonal $B-D$, then it is called Configuration 2. And, curvature $Curv_2$ for Configuration 2 could be expressed by:

$$Curv_2 = |H(A) + H(C) - 2 * H(B)| + |H(A) + H(C) - 2 * H(D)|. \tag{3.8}$$

The criteria that produces the configuration with minimum curvature is:

1. If $Curv_1 > Curv_2$, Select diagonal $B - D$;
2. If $Curv_1 < Curv_2$, Select diagonal $A - C$;
3. If $Curv_1 = Curv_2$, Then:

If $(|d-a| > |b-c|)$, Select diagonal $A-C$;

If $(|d-a| < |b-c|)$, Select diagonal $B-D$;

If $(|d-a| = |b-c|)$, Select $A-C$ or $B-D$ randomly.

Let us take an example of sample data to illustrate the critical points extraction detected by both triangulation selection criteria, as is shown in Figure 3.6.

From the example illustrated in Figure 3.6, we can get the same triangulation results from both triangulation criterion. Note that the number of peaks $\#\{peak\} = 1$, the number of passes $\#\{pass\} = 1$, and the number of pits $\#\{pits\} = 2$, beside a virtual pit. Thus, $\#\{peak\} - \#\{pass\} + \#\{pits\} = 1 - 1 + 2 = 2$ satisfy the Euler formula. Meanwhile, it is worth mentioning that the second triangulation selection criteria has more details when we encounter that both triangulation configurations have the same curvature. Unlike Takahashi's [33] approach which suggests to choose randomly a configuration of $\theta_1 = \theta_2$, the second approach proposes to use the slope to choose between the two configurations. As a result, it produces a flatter surface.

3.4 Critical-Points based Watershed Segmentation

The watershed requires a set of markers. Each marker must be placed on a sample region of the object to segment. In this case, we need critical points (pits and peaks) as two types of markers to segment different regions: one for the minima regions and the other for the maxima regions, see Figure 3.7. The selection of the markers in watershed segmentation is one of the most crucial steps for a successful solution. The design of correct critical points extraction as markers has been presented in Section 3.3. Thus, we will propose critical-points based approach for watershed segmentation and present the experimental results and discussion in next sections.

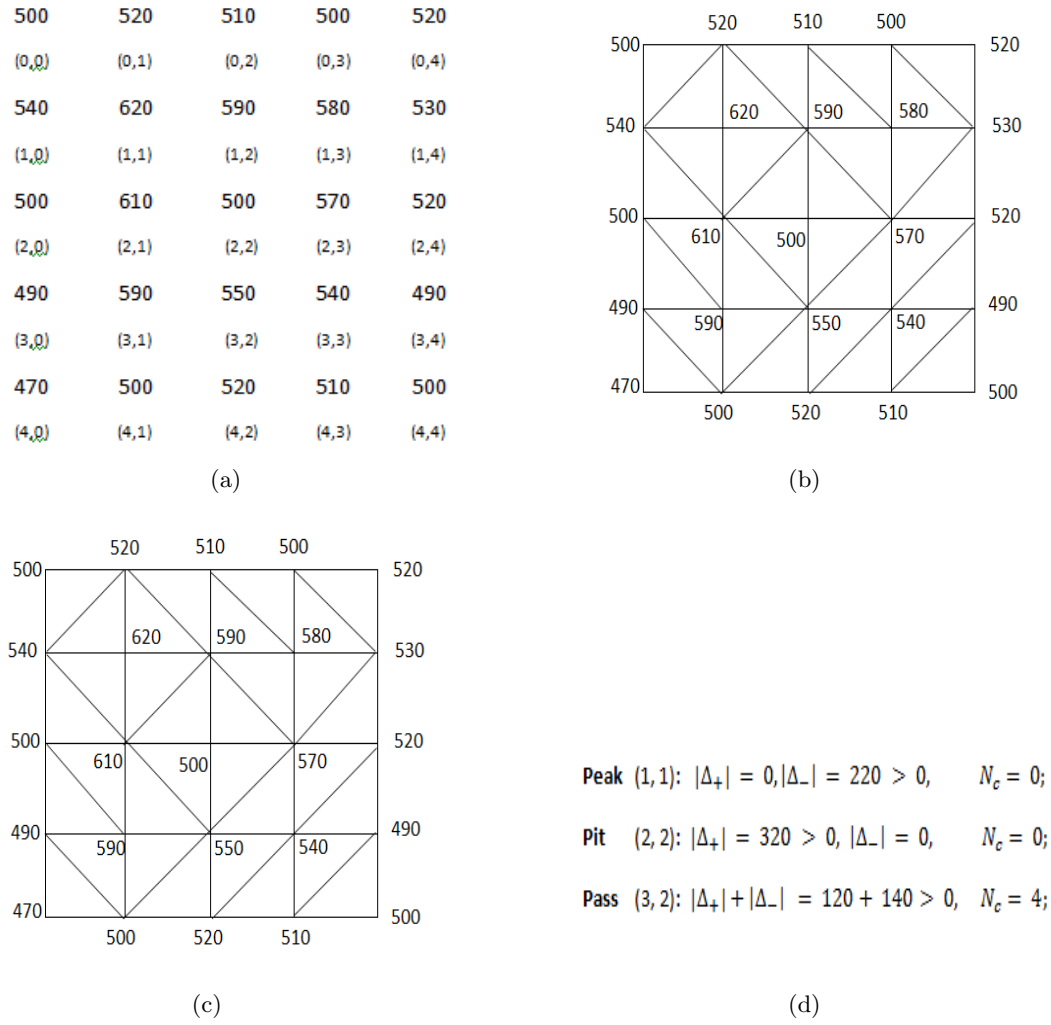


Figure 3.6: An example of sample data: both triangulation method to extract correct critical points from this example.
(a) Height values of the sample data. (b) First triangulation to the "flattest" criteria. (c) Second Triangulation to the "biggest" Curvature Criteria. (d) the values in the criteria of both triangulation criteria.

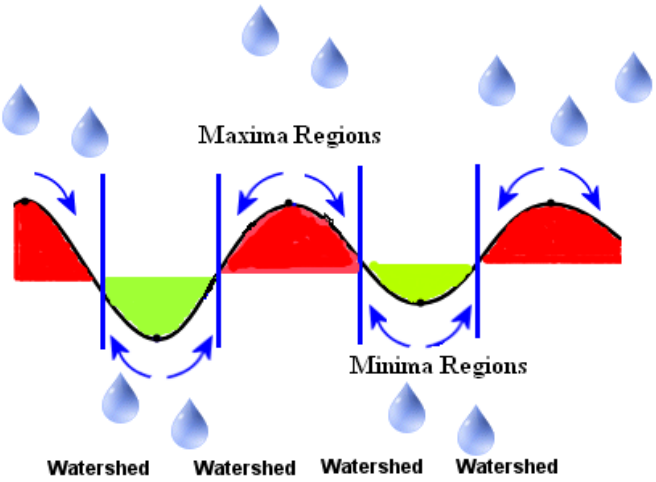


Figure 3.7: One dimensional illustration of the critical-points based watershed segmentation. Minima regions are plotted in Green, and maxima regions are plotted in red. Critical points (Peaks and Pits) are plotted in black. When two different floods meet, a dam is built up. Those dams are the resulting contours of the segmentation.

3.4.1 Critical-Points based Watershed Algorithm

Unlike the algorithm described in section 3.2.3, which uses the gradient image, the critical-points based watershed algorithm works on the original gray scale image. During the successive flooding of the grey value relief, watersheds with adjacent catchment basins are constructed.

The critical-points based watershed starts from a grey scale image F and two sets of markers M_1 and M_2 with different labels, namely, minimum (pits) markers and maximum (peaks) markers. In this case, these will be minimum and maximum of F. Both sets, M_1 and M_2, expand as much as possible, while preserving the number of connected components of M_1 and M_2. Let Γ be the neighborhood of the extracted pixels. The algorithm steps are:

Step 1. Select a set of minimum markers M_1 and a set of maximum markers M_2 according to the critical points extraction algorithm presented in Section 3.3. The upward flooding shall start from the pixels belonging to the minimum markers, and the downward flooding

shall start from the pixels belonging to the maximum markers. Each marker is given a different label;

Step 2. Insert every neighboring pixel x of each marked minimum area into a minimum priority queue, with a priority level corresponding to the lower gray level $F(x)$ of the pixel. Sequentially, insert every neighboring pixel y of each marked maximum area into a maximum priority queue, with a priority level corresponding to the higher gray level $F(y)$ of the pixel. Note that a point cannot be inserted twice in the queues;

Step 3. Extract a pixel x with the highest priority level from the minimum priority queue. If the neighborhood $\Gamma(x)$ of the extracted pixel x that have already been labeled all have the same label, then the pixel x is labeled with their label. And all non-marked neighbors that are not yet in both priority queues are put into the minimum priority queue;

Step 4. Likewise, extract a pixel y with highest priority level from the maximum priority queue. If the neighborhood $\Gamma(y)$ of the extracted pixel y that have already been labeled all have the same label, then the pixel y is labeled with their label. All non-marked neighbors that are not yet in both priority queues are put into the maximum priority queue;

Step 5. Redo Step 3 and Step 4 sequentially until both of the priority queues are empty.

Note that the non-labeled pixels are the watershed lines. The watershed lines set is the complement of the set of labeled points. Generally speaking, the algorithm presented above includes two steps: (1) Select a set of minimum markers and a set of maximum markers by critical points extraction algorithm presented in Section 3.3; (2) two opposite flooding steps: upward flooding and downward flooding, preceding level by level and starting from the minimum and maximum markers respectively. In the flooding process, minimum and maximum priority queues work sequentially until both of the queues are empty.

To illustrate the critical-points watershed segmentation algorithm, it is necessary to take an example for demonstration. Here, an example for the critical-points based watershed segmentation algorithm with 8-connectivity is presented as follows. Figure 3.8(a) is an original image with 7×7 pixels. By critical points extraction algorithm proposed in Section

3.3, the minimum marker set 2, 71 and the maximum markers set 158, 48 could be extracted from the original image. Accordingly, 2 is labeled as A1 and 71 is labeled as A2 in the minimum markers set. Then we insert their neighbors into the minimum priority queue. Sequentially, we are starting to process the maximum markers set, labeling 158 as B1 and 48 as B2. In addition, the neighbors of all maximum markers 158 and 48 are inserted into the maximum priority queue. Noticeably, each pixel can just be treated only once in the critical-points based watershed segmentation algorithm. This step, called "h = 0", is illustrated in Figure 3.8(b). Then, the next step as shown in 3.8(c) is to look for the lowest value pixel in the minimum priority queue and label it according to its neighbors' label. A pixel which is adjacent to only one label, and then initially gets the same label; a pixel which is adjacent to two different labels, and therefore initially gets labeled "W". It is obvious to see that 6 in (1,6) is the pixel with the smallest gray value in the current minimum priority queue. Therefore, 6 in (1,6) has the highest priority level to be processed. On the basis of its neighbors' label, 6 in (1,6) should be labeled as A1, and then be removed from the top of the minimum priority queue. In the meantime, its neighbors are inserted into the minimum priority queue if they have not been processed yet. Oppositely, in the maximum priority queue, 111 in (4,2) is the pixel with the highest priority level to be processed. Still according to the label situation of its neighbors, 111 in (4,2) is labeled as B1 and removed from the maximum priority queue. And its neighbors 50, 66 and 79 that have not been processed are inserted into the maximum priority queue. Thereby, first step "h = 1" is done. Labeled basins are propagated inside the set of labeled pixels until both of queues are empty. Then, final result is shown in Figure 3.8(d).

3.4.2 Experimental Results and Discussion

This section provides experimental results obtained by the three various watershed algorithms, i.e. classical Meyer's flooding algorithm, watershed using gradients, and critical-points based watershed algorithm. We also compare the performance of various watershed

86	43	58	41	15	6	17
88	46	68	36	10	2	19
50	71	105	60	31	20	8
66	111	158	50	43	30	15
79	130	98	140	41	47	16
90	81	71	76	46	48	17
86	84	83	80	31	20	30

(a)

86	43	58	41	15	6	17
88	46	68	36	10	A1	19
50	71	105	60	31	20	8
66	111	B1	50	43	30	15
79	130	98	140	41	47	16
90	81	A2	76	46	B2	17
86	84	83	80	31	20	30

(b) h = 0

86	43	58	41	15	A1	17
88	46	68	36	10	A1	19
50	71	105	60	31	20	8
66	B1	B1	50	43	30	15
79	130	98	140	41	47	16
90	81	A2	76	46	B2	17
86	84	83	80	31	20	30

(c) h = 1

B1	B1	B1	W	A1	A1	A1
B1	B1	B1	W	A1	A1	A1
B1	B1	B1	W	A1	A1	A1
B1	B1	B1	W	A1	A1	A1
B1	W	W	W	W	W	W
B1	W	A2	A2	W	B2	B2
B1	W	A2	A2	W	B2	B2

(d) Final result according to critical-points based watershed algorithm

Figure 3.8: Watershed transform by critical-points based watershed algorithm on the 8-connected grid, showing relabeling of 'watershed' pixels
(a) Original image. (b-c) Labeling steps. (d) Final result according to critical-points based watershed algorithm.

segmentation techniques for four different input images.

In what follows, we use a flow chart to represent the critical-points based watershed algorithm, showing the steps as boxes of various kinds processes in advance. The general flow chart is illustrated in Figure 3.9.

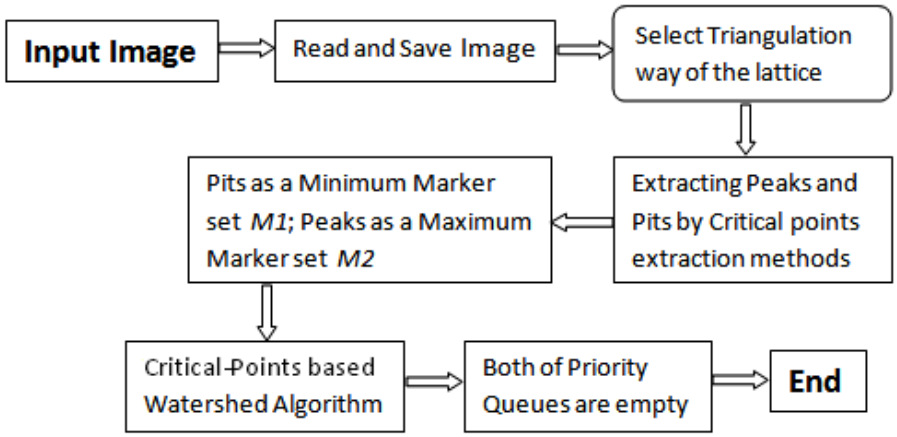
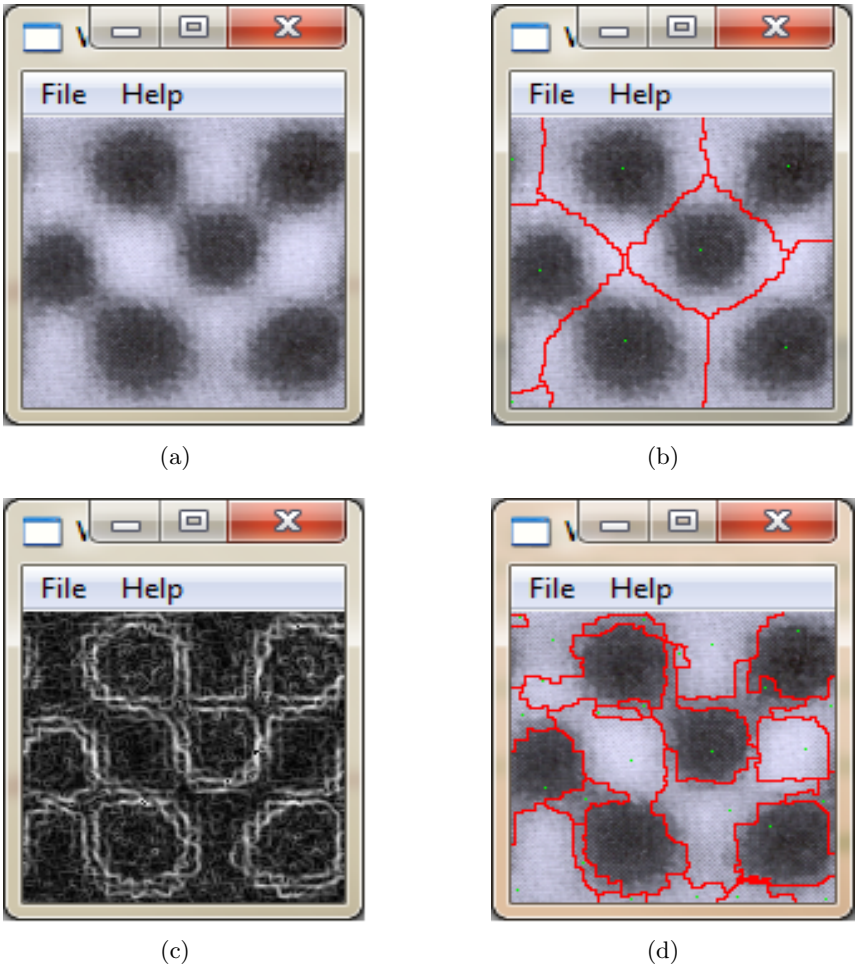


Figure 3.9: Flow Chart of Critical-Points based watershed algorithm

The experimental examples are illustrated in Figure 3.10, Figure 3.11, Figure 3.12, Figure 3.13 and Figure 3.14. These experiments are aimed at evaluating the difference between the critical-points based watershed algorithm and the other two watershed algorithms.

The first image file in Figure 3.10(a) is a picture of 128×128 pixels. The implementation of the first image in Figure 3.10 has shown that it is possible to get a better segmentation for images with noise. That is to say, to avoid over-segmentation, it is recommended to smooth the image before starting the watershed segmentation with a median filter. Then, implementation for the first image after smoothing it by median filter is presented in Figure 3.11, which shows the critical points based watershed algorithm could achieve a better result than the other two algorithms.

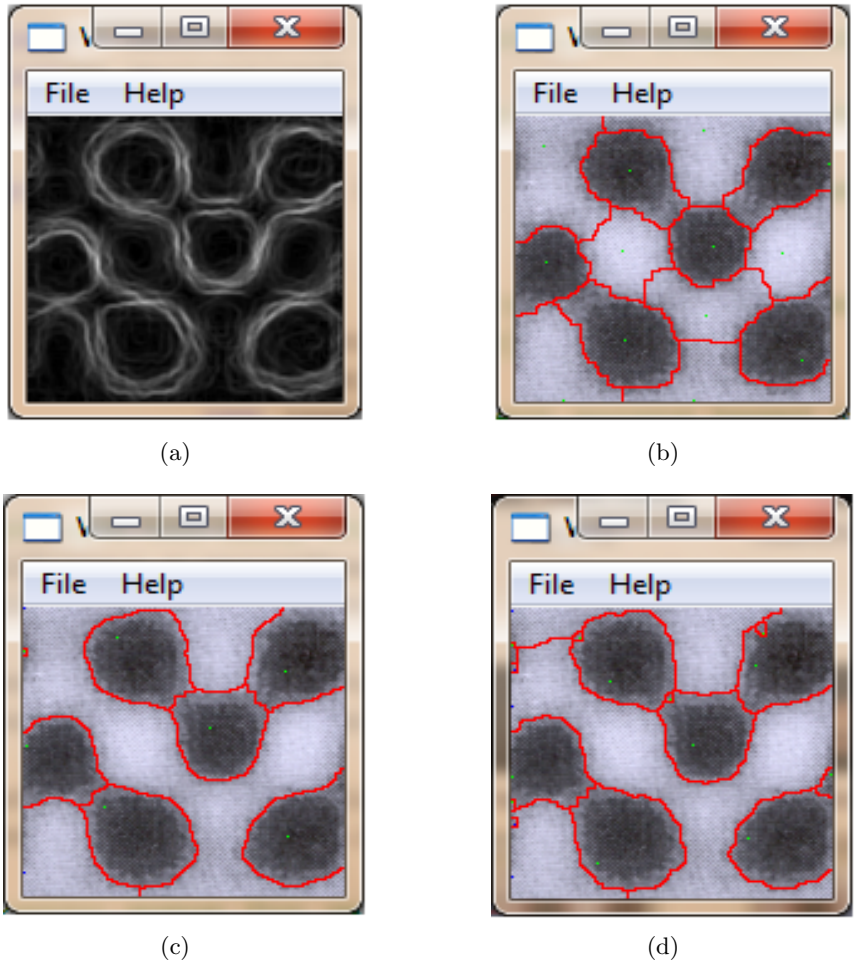
The second image file in Figure 3.12(a) is a picture of 171×171 pixels. The gradient watershed algorithm over-segments this image into three different kinds of regions. The test presents that the image has three kinds of regions, i.e. peaks region, pits region, and saddle



(a) (b)

(c) (d)

Figure 3.10: Watershed Segmentation
(a) Original image. (b) Watershed transform on image. (c) Gradient image. (d) Watershed transform on gradient image.



(a) (b)

(c) (d)

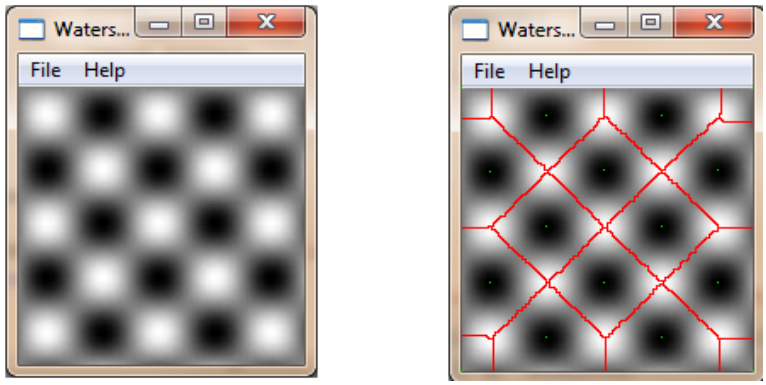
Figure 3.11: Example 1 of Watershed Segmentation after smoothing the image 3.8(a) (a) Smoothed gradient image. (b) Result using smoothed gradients watershed. (c)Result using critical-points watershed by angle between normal vectors. (d) Result using critical-points based watershed by curvature.

region. Actually, there are only black regions and white regions in this image. However, from Figure 3.12(e) and Figure 3.12(f), two kinds of regions have been segmented correctly by the critical-points based watershed algorithm. Thus, an obvious conclusion is that the critical-points based watershed algorithm works well for the image with ambiguous saddle regions.

The third image file in Figure 3.13(a) is a cell-colony picture of 256×254 pixels. First of all, there are lots of irregular tiny objects in the picture, which increases the difficulty of segmentation. Classical Meyer's watershed segmentation just segments the image into many different sections, but does not sketch the outline of those small cells. In addition, it is easy to see that gradient watershed algorithm has no effect on the large image with too many small objects. However, the critical-points based watershed algorithm works well for this kind of complicated image, as illustrated in Figure 3.13(e) and Figure 3.13(f). Nearly all small cells' contour has been sketched completely. Thus, the critical-points based watershed algorithm is able to produce a better experimental result.

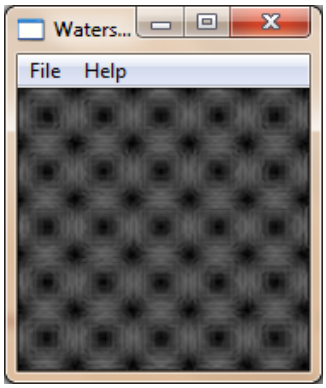
The fourth image file in Figure 3.14(a) is a gradient image of heart magnetic resonance imaging (MRI). It is a picture of 216×216 pixels. The image with snake-like plateaus spread over different sub-domains. Normally, this kind of image is supposed to be difficult to segment by using common segmentation methods. Figure 3.14 shows that critical-points based watershed algorithm almost clearly segment the circular contour as well as extract scattered irregular parts. By comparison, Meyer's watershed and watershed using gradients only get a general shape for the snake-like objects in the image.

In all, the fact is that the critical-points based watershed algorithm outperforms Meyer's and gradients methods in the above experiments. Critical-points based watershed algorithm allows to combine topological features with upward-downward flooding process, making it possible to keep the important topological properties and necessary contour in the segmented images. Therefore, the critical-points based watershed algorithm may become an

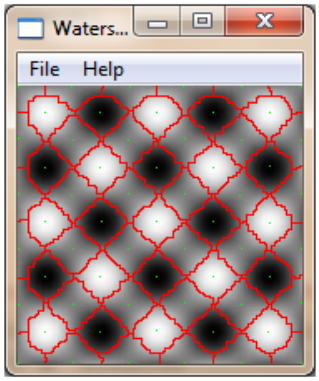


(a)

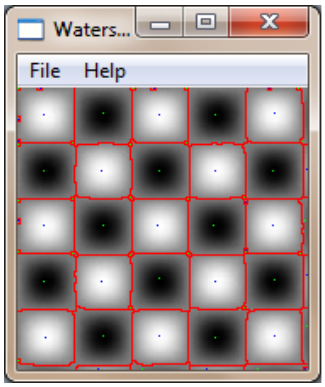
(b)



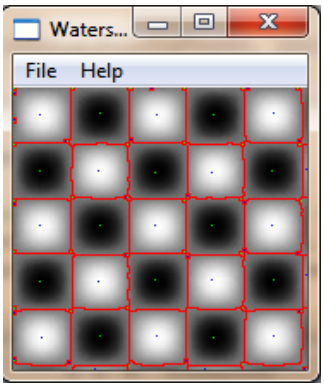
(c)



(d)



(e)



(f)

Figure 3.12: Example 2 of Watershed Segmentation
(a) Original image. (b) Result on image. (c) Gradient image. (d) Result on gradients image. (e) Result using critical-points based watershed by angle between normal vectors. (f) Result using critical-points based watershed by curvature.

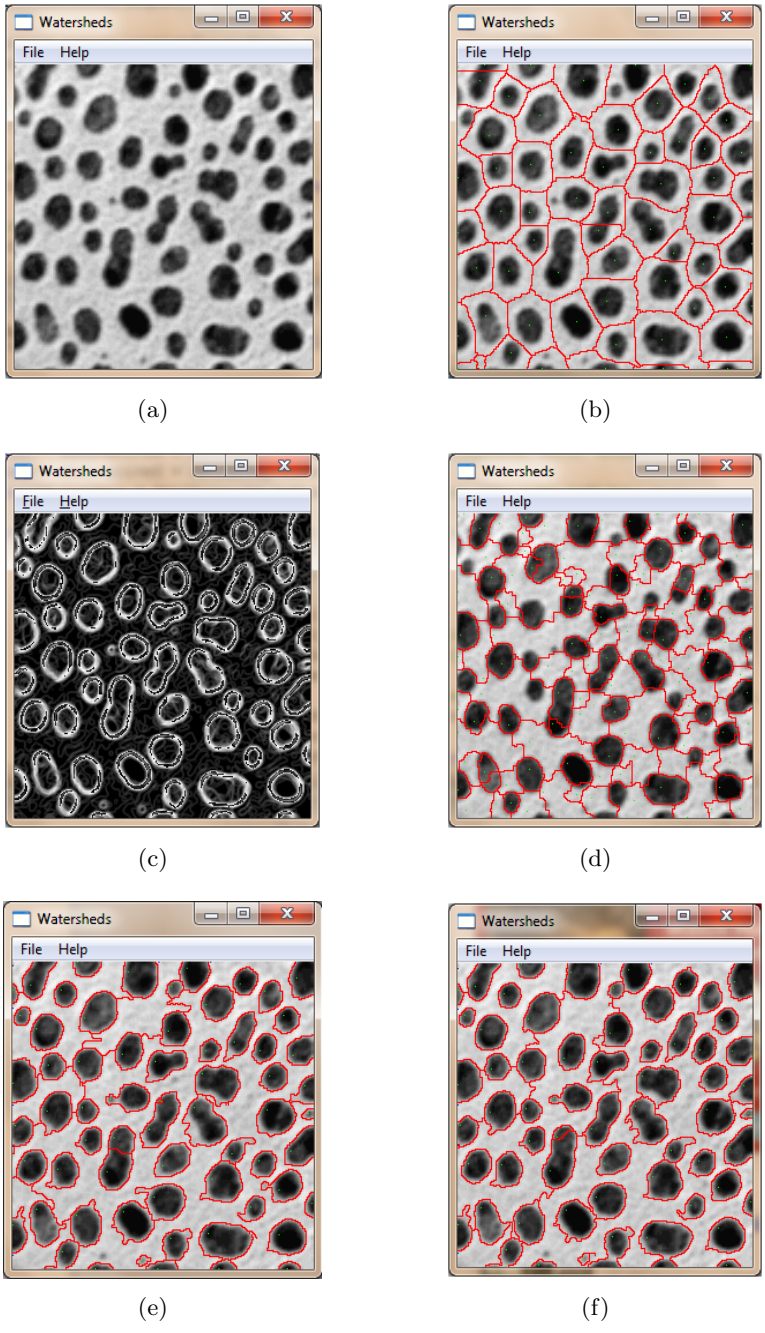


Figure 3.13: Example 3 of Watershed Segmentation
(a) Original image. (b) Result on image. (c) Gradient image. (d) Result on gradients image. (e) Result using critical-points based watershed by angle between normal vectors. (f) Result using critical-points based watershed by curvature.

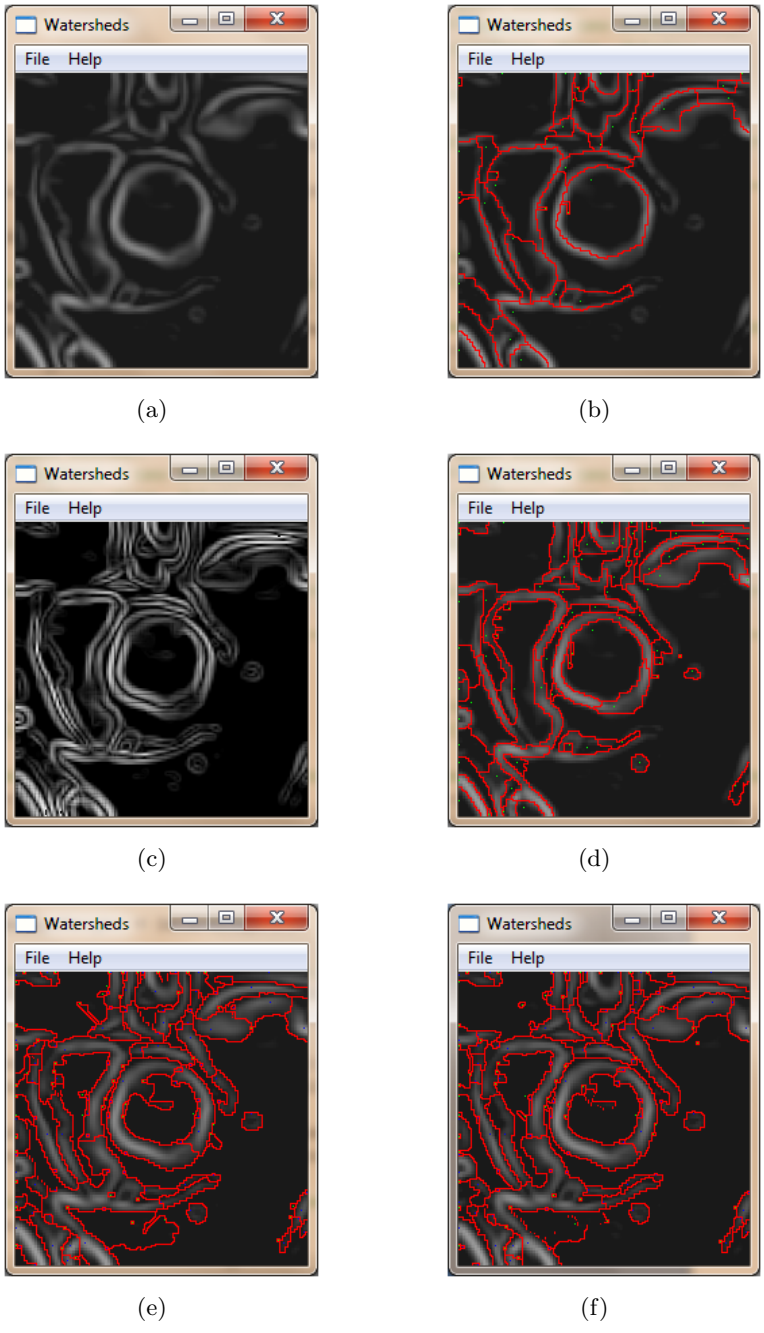


Figure 3.14: Example 4 of Watershed Segmentation
(a) Original image. (b) Result on image. (c) Gradient image. (d) Result on gradients image. (e) Result using critical-points based watershed by angle between normal vectors. (f) Result using critical-points based watershed by curvature.

interesting solution in segmenting complicated images in the perspective of differential topology. Meanwhile, to achieve an accurate segmentation, the strategies of image smoothing and markers merging could be combined with the proposed watershed method. Further, we could deploy the critical-points based watershed algorithm on a three-dimensional height field from an image, and do segmentation visualization on a surface. We will focus on watershed segmentation and visualization in Chapter 4.

3.5 Summary

Summarizing the performance results for the various watershed segmentation implementations discussed in previous sections, it was found that the critical-points based watershed method has the best segmentation results, especially for images containing large or snake-like plateaus spread over different sub-domains. Three watershed segmentation algorithms are applied to four different images. Those images with too much noise are in particular not suitable for watershed algorithms due to too many local minima and maxima. The reason is that noise tends to produce too many regions. We have solved this problem by using average filter or median filter, and merging those markers that are close to each other.

The improvement of flooding simulation over minima priority queue and maxima priority queue is of utmost importance. They make sure the implementation of upward and downward flooding processes. Additionally, the design of the two priority queues and critical points extraction by triangulation altogether guarantee the prevalence of image segmentation.

Triangulation is used to ensure that the extracted critical points satisfy the Euler formula and maintain the topological properties. Except for choosing the diagonal that makes the smallest absolute angle between the normals to the triangles divided by the diagonal, the curvature configuration method is a relatively fast, and less complexity way of triangulating regular grid points. Concerning the two types of priority queues, the queue for the minimum markers set works as the classical approach, where the lowest pixel always has the highest

priority to be processed. Moreover, those maximum markers are inserted into the queue for the set of maximum markers, where the highest level pixels always has the highest priority to be processed. Then, the watershed dam will be formed in the middle of the two different regions.

Chapter 4

Watershed Segmentation on Approximated Height Field

4.1 Watershed Segmentation on Approximated Height Field

In Chapter 2 and 3, we present height fields and explore their visualization and watershed segmentation on original images. In this chapter, we explore an integrated image segmentation visualization application. In brief, our idea is to segment the approximated height fields by the critical-points based watershed algorithm. And then, we evaluate whether an incremented number of vertices could render a better segmentation result on the original image.

Our approach starts with an approximated height field $H_1 = f_1(x, y)$ constructed from an image F with an initial set of vertices (for instance 20% of all the pixels). Afterwards, we derive an approximated height for each pixel, which is also called the approximated gray level value. This field is used to extract critical points and to perform watershed segmentation using the critical-points based watershed algorithm. Height fields approximation is produced using the techniques described in Chapter 2, where vertices are inserted one by one accordingly to a local error measure or by a given percentage (20% for instance) at once using the curvature measure.

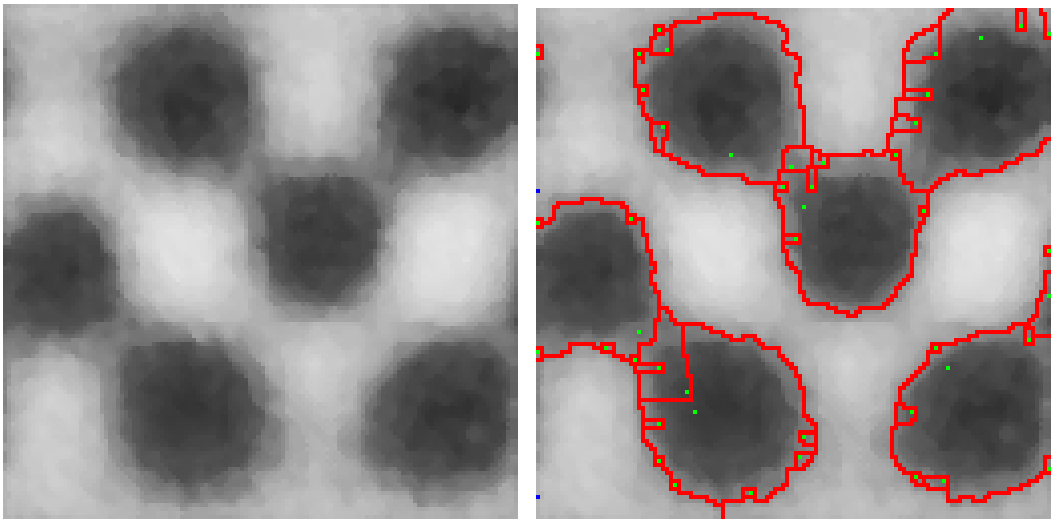
The segmentation results obtained from the original image are illustrated in Figure 4.1. In this chapter, the proposed watershed segmentation is tested with both local error and

curvature measure based approximation techniques. The results for inserting vertices one by one with local error measure are presented in Figure 4.2. In Figure 4.2(a), the maximal local error of the height field is 23 with 248 vertices and its global error is 90728. On the other hand, the results for inserting 20% vertices one time with curvature measure are presented in Figure 4.3.

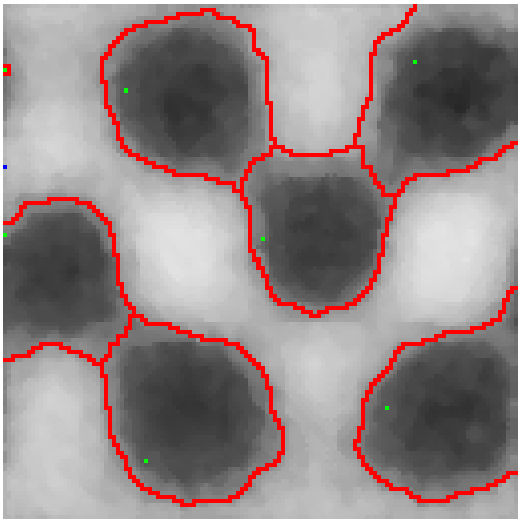
From the above experimental results, we can conclude that parallel greedy insertion with curvature selection measure approximates the height field much faster and better than local error selection measure. Also, by comparing the above results to the original image segmentation, our method of segmenting the approximated height field with only 20% vertices without using smoothing pre-process gives a better segmentation result. And, the fact is that our new method for images with noise generates nearly the same good segmentation effects as the original method using smoothing pre-process. Therefore, the new method for watershed segmentation that uses the polygonal approximation of the height field is a fast and an efficient way of segmenting large images.

4.2 Summary

A key aspect to segment the approximated height fields is the approximation accuracy with which we choose a subset of critical vertices. In this chapter, we present the critical-points based watershed segmentation on the approximated height fields, and discuss the segmentation effects on both approximation ways. The intuition behind the method is that watershed segmentation can start from significant critical points as markers, which decreases the over-segmentation that characterizes this type of method. Experiments demonstrates that the new method can generate a favorable segmentation without pre-process.

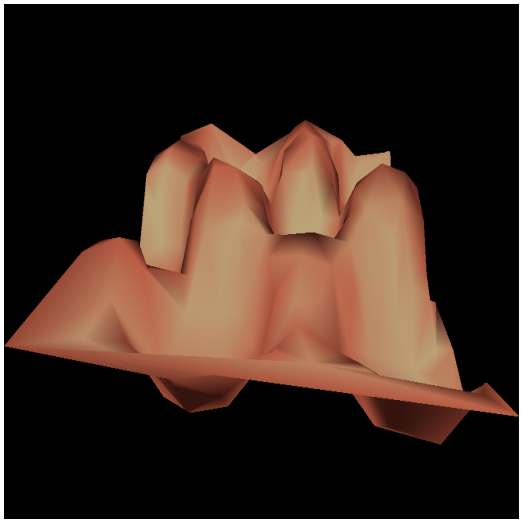


(a) (b)

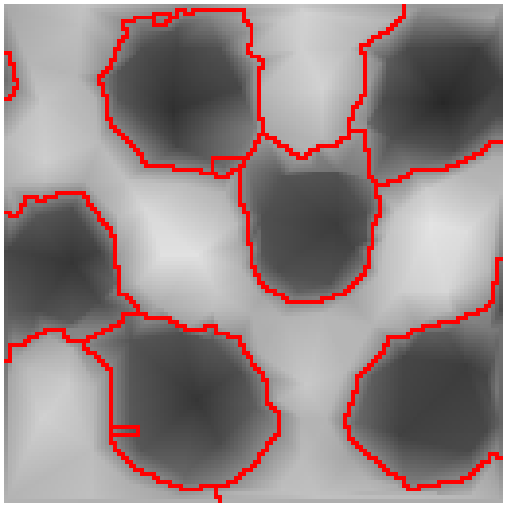


(c)

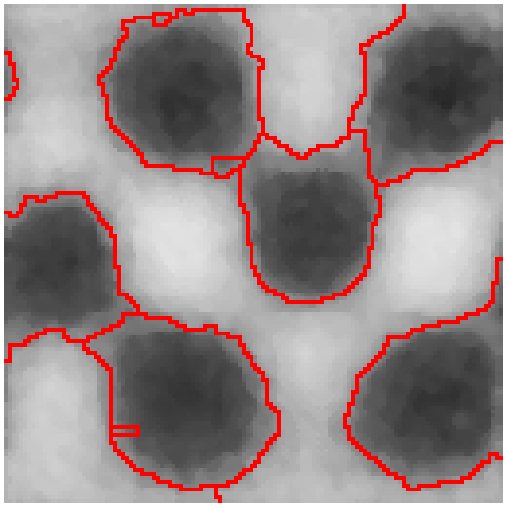
Figure 4.1: Critical-Points based Watershed Segmentation on original image (a) Original image. (b) Result on image without pre-process. (c) Result on image with median filter pre-process.



(a)



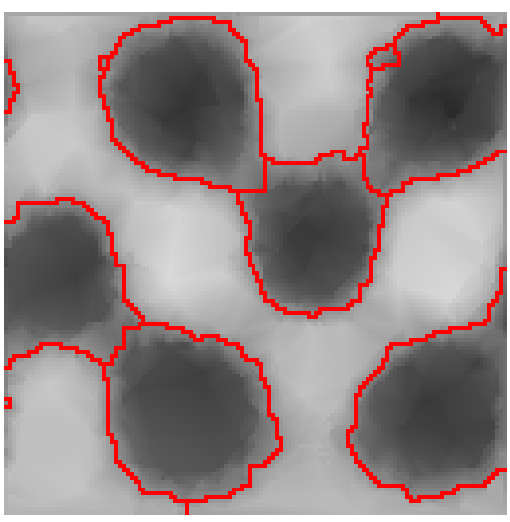
(b)



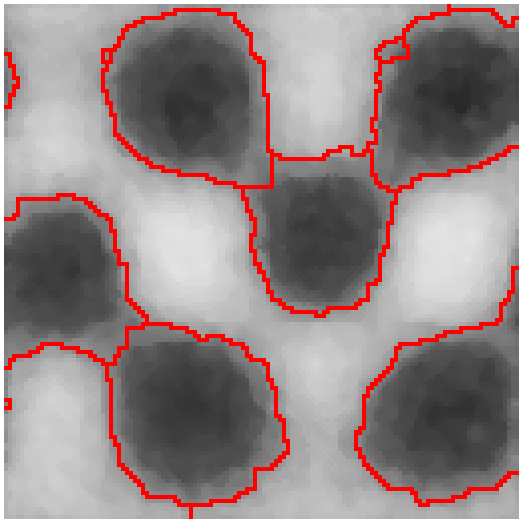
(c)

Figure 4.2: Critical-Points based Watershed Segmentation on approximated height field with local error
(a) Approximated height field with maximal local error 23. (b) Result visualized on the approximated image. (c) Result visualized on the original image.

(a)



(b)



(c)

Figure 4.3: Critical-Points based Watershed Segmentation on approximated height field with curvature measure
(a) Approximated height field with 20% vertices. (b) Result on the approximated image. (c) Result on the original image.

Chapter 5

Conclusion and Outlook

This chapter summarizes the findings of the thesis and discusses recommendations for future work.

5.1 Conclusion

The thesis investigates the main research questions of "*Height fields approximation and watershed segmentation*". It advances the research on how to approximate height fields, visualize the height field from an arbitrary image, and perform watershed segmentation based on critical points extraction in the perspective of topological properties.

We successfully implemented height fields approximation and its visualization using the CGAL library under C++. The greedy insertion algorithm is used to approximate height fields. It is a fast and flexible method that can produce high quality approximations. It takes a height field as input and produces a triangulated mesh approximating that height field as output. The algorithm starts with a minimal approximation consisting of two triangles and repeatedly inserts a vertex with the greatest error or more than a vertex each time into the approximation. The process is terminated either when a given number of vertices is reached, or when the error drops below a given error tolerance. Three important selection measures are discussed in this work including local error, global error and curvature. All have been examined in the implementation, although our focus was on local error and curvature. In

particular, in order for the visualization results to be more realistic, we use three different ways to shade the approximated height fields. Gouraud shading and classical shading have better visualization effects than the adaptive method.

Regarding watershed segmentation, I devoted part of my work to studying and fabricating a new watershed algorithm and critical points extraction criteria. Additionally, two classical watershed algorithms based on flooding have been reviewed. It is notable that watershed algorithm based on critical points extraction is a sequential algorithm for computing the watershed lines according to both priority queues that were described in Chapter 3. In practice, the watershed algorithm by flooding is hard to parallelize because of its inherently sequential nature. And, we use minima markers and maxima markers to start an upward-downward flooding process. Strategies for extracting critical points are discussed based on differential topological features. Critical point are extracted from discrete data in the same way as from the continuous data. Additionally, triangulation is used to determine the interpolated surface from the sample points and ensure that the extracted critical points satisfy the Euler formula. Triangulation by angles and by curvature configuration achieve almost the same results while implementing our new watershed segmentation algorithm based on critical points. Experimental examples are presented to demonstrate the capability of the new watershed method.

Most sigificantly, the implementation of the critical points-based watershed segmentation on approximated height fields opens up a new research domain of watershed segmentation and visualization. Segmentation performance on approximated critical points-based watershed algorithm could produce a more informative and favorable watershed result on surface and images without pre-process. Thus, an attempted and exploratory job is presented in Chapter 4.

5.2 Recommendations for Future Work

Based on the discussion in the previous sections, some recommendations for future work can be given:

1. **Watershed on Height Field.**

Watersheds are line-like features in two-dimensional scalar field. Watersheds describe ridges/valleys of a height field $s(x)$: integrate the gradient field $\nabla s(x)$ (backward/forward), starting at saddle points. The watersheds provide a segmentation of the domain into so-called Morse-Smale complexes [29]. In addition, most segmentation methods decompose three-dimensional objects into parts based on curvature analysis. Segmentation is one of the main areas of three-dimensional object modeling. Therefore, it is sensible to partition a two-dimensional height field model into meaningful parts based on critical points extraction. This is the idea of the critical-points based watershed segmentation method introduced in Chapter 3 and Chapter 4. Then, this representation of watershed segmentation for images is useful for three-dimensional interactive visualization. If the critical-points based watershed segmentation algorithm is introduced into the three-dimensional mesh area, the watershed-based height field algorithm would become of more interest. Thus, we have attempted watershed-based height fields implementation and visualization in Chapter 4. However detailed research on height fields based watershed segmentation has not been carried out so far due to the time limits. The research prospects on visualization is considerably broad and bright. We expect that watershed segmentation result could be visualized in a three-dimensional interactive surface. Therefore, visualization for critical-points based watershed segmentation on height fields still has a long way to go.

2. **Applications in Computer Vision.**

The approximation algorithm can be used for the simplification of range data in computer vision. The output data of many stereo and laser range scanners is in the form of

a height field or z-buffer in a perspective space, so it can be fed into the algorithm described here with little or no modification. When the triangles output by the algorithm are perspective-transformed to world space, their planarity is preserved, so the approximation in that space is also valid.

Some range scanners acquire data in a cylindrical format, outputting radius as a function of azimuth and height: $r(\theta; z)$. The algorithms here could be modified to generate triangulated approximation to these surfaces, but since the transform between cylindrical space and world space is not a perspective one, nonlinear interpolation would be needed for best results.

Bibliography

[1] P. S. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In Proceedings of Graphics Interface '94, pp. 1-8, 1994.

[2] L. D. Floriani. A pyramidal data structure for triangle-based surface description. IEEE Computer Graphics and Applications, 9(2):67-78, March 1989.

[3] M. D. Berg and K. T. G. Dobrindt. On levels of detail in terrains. Technical Report UU-CS-1995-12, Department of Computer Science, Utrecht University, June 1995.

[4] CGAL Open Source Project. CGAL-3.6.1 User and Reference Manual. 2010. http://www.cgal.org

[5] M. Garland and P. S. Heckbert. Fast polygonal approximation of terrains and height fields. Technical report CMUCS-95-181, CS Dept., Carnegie Mellon U., Sept., 1995.

[6] P. S. Heckbert and M. Garland. Survey of surface approximation algorithms. Technical report CMU-CS-95-194, CS Dept., Carnegie Mellon U., 1995.

[7] M. d. Berg, M. v. Kreveld, M. Overmars, and O. Schwarzkopf. Computational geometry: Algorithms and applications, Second Edition, Springer-Verlag, 2000.

[8] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. ACM Transactions on Graphics, 4(2):75-123, 1985.

[9] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, ed., Mathematical Software III, pp 161-194. Academic Press, NY, 1977.

[10] M. Allili, D. Corriveau, and A. Villares. Exploring Height Fields: Interactive Visualization and Applications. Proceedings of the SPIE, Volume 7868, pp. 78680B-78680B-13, 2011.

[11] L. D. Floriani, B. Falcidieno, and C. Pienovi. A Delaunay-based method for surface approximation. Proc. Eurographics '83, pp. 333-350, 1983.

[12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms. The MIT Press, 1990.

[13] W. Welch. Serious Putty: Topological Design for Variational Curves and Surfaces. PhD thesis, CS Dept., Carnegie Mellon U., 1995.

[14] M. Woo, J. Neider, and T. Davis, Opengl programming guide, Addison-Wesley, 2007.

[15] E. Angel. A top-Down Approach Using OpengGL, Fifth Edition. Pearson Education, Inc., Addison Weley, 2005.

[16] R. M. Haralick and L. G. Shapiro. Survey: image segmentation techniques. Comput. Vision Graphics Image Process. 29:100-132, 1985.

[17] H. Digabel and C. Lantuejoul. Interactive algorithms. Proc. of 2nd European Symposium on Quantitative Analysis of Microstructures in Material Science, J-L. Chermant Ed.:85-99, 1978.

[18] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. In Proc. Int. Workshop image processing, Real-Time Edge and Motion detection, pp. 17-21, 1979.

[19] J. B. T. M. Roerdink and A. Meijster. The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. Fundamenta Informaticae, vol. 41, pp. 187-228, 2000.

[20] J. Serra. Image Analysis and Mathematical Morphology. London: Academic Press, 1982.

[21] F. Meyer and S. Beucher. Morphological segmentation. J. Visual Commun. and Image Repres. 1(1):21-45, 1990.

[22] F. Meyer. Topographic distance and watershed lines. Signal Processing 38, pp. 113-125, 1994.

[23] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Trans. Patt. Anal. Mach. Intell. 13(6):583-598, 1991.

[24] S. Beucher and Meyer F. The morphological approach to segmentation: the watershed transformation. In Mathematical Morphology in Image Processing, E. R. Dougherty, Ed. Marcel Dekker, New York, ch. 12, pp. 433-481, 1993.

[25] L. Najman and M. Couprie. Watershed algorithms and contrast preservation. Proc. DGCI, LNCS, Springer, Vol. 2886, pp. 62-71, 2003.

[26] R. C. Gonzalez, R. E. Woods, and Eddins S. L. Digital Image Processing Using matlab. Prentice Hall, Upper Saddle River, NJ, 2003.

[27] J. M. Gauch. Image Segmenation and Analysis via Multiscale Gradient Watershed Hierarchies. IEEE Transactions on Image Processing, VOL. 8, No., 1, January 1999.

[28] M. Allili, D. Corriveau, S. Deriviere, T. Kaczynski, and A. Trahan, Discrete dynamical system framework for construction of connections between critical regions in lattice height data, Journal of Math. Imaging and Vision 28(2):99C111, 2007.

[29] H. Edelsbrunner, J. Harer, and A.J Zomorodian, Hierarchical Morse-Smale complexes for Piecewise Linear 2-Manifolds, Discrete & Compu. Geom. 30, pp. 87C107, 2003.

[30] J. W. Milnor. Morse Theory. Princeton University Press, 1963.

[31] W. S. Massey. A Basic Course in Algebraic Topology. Springer-Verlag, 1991.

[32] H. B. Griffiths, Surfaces. Cambridge University Press, 2nd ed., 1981.

[33] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithm for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. Eurographics '95, 14:C-181-C-192, 1995.

[34] P. J. Scott. An Algorithm to Extract Critical Points from Lattice Height Data. International Journal of Machine Tools and Manufacture, Vol. 41, Iss.13-14, pp. 1889-1897, 2001.