THE UNIVERSITY OF CALGARY

Synthesizing Techniques based on Multiresolution

by

Lakin Christopher Wecker

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

January, 2007

Your file Votre référence
ISBN: 978-0-494-25680-0
Our file Notre référence
ISBN: 978-0-494-25680-0

Abstract

Digital models can be created by gathering a set of measurements from geometric objects. For various reasons, these models may be incomplete representations of the objects. Incomplete models fail to meet the requirements defined by their potential applications.

In this thesis we develop a multiresolution approach to synthesizing digital models. Multiresolution can reduce the number of data points in a model to produce a coarse approximation. The coarse approximation will lack some features that are present in the original data. These features can be captured and are usually called the *details*. We consider the *details* to be characteristics of the model and we consider the coarse approximation to be the overall structure of the model. Using MR, we can iteratively decompose the model to form a hierarchy of components. We selectively combine components from existing models to synthesize new models with the appropriate characteristics.

Our technique can synthesize models with characteristics similar to those found in existing models. The technique is general and can be adapted for many application areas. As a demonstration we provide two sample applications. First, it is used to synthesize patches for the voids commonly found in digital elevation models (DEM). Secondly, it is used to augment existing iris image databases by synthesizing iris images. The results in both applications demonstrate that our approach is effective at synthesizing realistic models.

Acknowledgements

First and foremost I would like to especially thank my wife, Anita Wecker. She believes in me, supports me, and most of all she pushes me to be all that I can be.

Special thanks go to Dr. Samavati, who gave me a chance. Without him, I never would have applied for admission, let alone finished writing my thesis. He consistently goes above and beyond the call of duty as a Supervisor. He has always had the time to talk, whether it be about my work, or about life. In addition, this work is inspired by many discussions with him and he has provided invaluable insight and advice. I'd also like to thank Dr. Gavrilova who accepted me for co-supervision. She has provided me with much insight and valuable advice over the past two years.

Special thanks also go out to Mike Simoens, who is a invaluable ally in the great big game called life. Not only has he helped me immensely in writing my thesis, but he has provided support and many grand ideas along the way.

I'd also like to thank my parents, Barry and Alberta Wecker. They instilled in me a desire to learn, a desire to explore and a desire to do the best that I can in everything I do. They have always supported me in every endeavor and continue to do so.

Last but not least, I'd like to thank my colleagues in the Jungle: Luke, John, Erwin, Adam and Mik, who made my time here enjoyable, funny and (un)productive (as the case may be). You guys rock.

Portions of the research in this paper use the CASIA iris image database collected by Institute of Automation, Chinese Academy of Sciences

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A multiresolution approach to synthesizing data is presented in this thesis. The approach is adapted and implemented for two application areas. First, it is used to repair the voids commonly found in digital elevation models (DEM). Secondly, it is used to augment existing iris image databases by synthesizing iris images.

1.1 Motivation

Digital models can be created by gathering a set of measurements from geometric objects. For various reasons, these models may be incomplete representations of the objects. For some cases, the measurement gathering process may fail in some areas. In other cases, there may be too few sample objects to produce a sufficiently large database. Incomplete models often fail to meet the requirements defined by their potential applications. This provides us with motivation to augment incomplete models.

1.2 Goals

Our aim is to augment models to meet the requirements of their potential applications. We plan to achieve this by synthesizing new data from existing models. Geometric objects display a wide variety of characteristics, as do the models that represent them. Our synthesized data should exhibit the characteristic properties of the original objects.

1.3 Methodology

We propose to use multiresolution (MR) to capture the characteristics from existing models for the synthesis process. MR is an approach that can reduce the number of data points in a model to produce a coarse approximation. The coarse approximation will lack some of the features that are present in the original data. Therefore, approaches to MR typically store this information to facilitate a return to the original model. The extra information is usually called the *details*.

Using MR, we can iteratively reduce the resolution of the coarse approximation and capture more *details*. The coarse approximation and the set of *details* form a hierarchy of components for that model. Traditional MR applications focus only on the coarse approximations of the original data. In our case, we focus on utilizing the *details* as characteristics of the model. In addition, we can use the coarse approximation as the overall structure of the model. We selectively combine components from the hierarchy of existing models to synthesize new data with the appropriate characteristics. We use reverse Chaikin filters [60] to provide a MR representation.

1.3.1 Terrain Repair Methodology

Satellite gathered digital elevation models (DEM) are one example of incomplete models. The automated sampling process that is used to gather the elevations often fails to produce accurate measurements. Missing areas in the DEM are referred to as voids and inhibit the use of the DEM for visualization and analysis. We use a two

Figure 1.1: Examples of data-models synthesized with our approach for digital elevation models.

step approach to repair these voids. For the first step, a smooth patch that follows the structure of the nearby terrain is produced. In the next step, we extract the characteristics of the surrounding terrain and add them to the smooth patch. For both steps, we can take advantage of the structure that a DEM provides, by looking at each row and column of data individually.

We use a visual process to validate the patched DEM. To facilitate the validation, we propose two improvements for 3D rendering of DEMs. We introduce a method of triangulating the terrain to improve the accuracy of the rendering. Additionally, we suggest a visualization approach that emphasizes the characteristics of the DEM.

1.3.2 Iris Image Synthesis Methodology

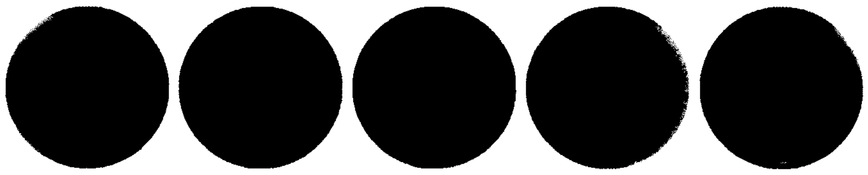


Figure 1.2: Examples of synthetic iris images generated with our synthesis technique.

Freely available iris image databases are small, and do not contain enough sample objects for testing biometric algorithms [73, 78]. In this application of our approach, our goal is to augment existing iris image databases. We use a novel iris image synthesis technique that has two distinct components. First, a number of preprocessing steps based on similar iris recognition methods are used to rid the image of extraneous (non-iris) information. For the synthesis stage, we capture and combine the characteristics of real images to compose new images. Generally speaking, this is similar to genetic inheritances, in which the irises of the children resemble those of the parents. For the purposes of combination, we explore a classification of iris images that increases the compatibility of the selected components.

In the case of iris images, we use two methods to validate synthetic images. As a first step, we use a visual process to ensure the images exhibit realistic characteristics. In the second step, we use a well known iris recognition algorithm on both the original database and a database augmented with our synthetic images. The goal of the second step is to verify that our augmented database exhibits the same characteristics as the original database.

1.4 Contributions

This thesis makes the following contributions:

- A multiresolution synthesis technique that uses components extracted from existing digital models. Use of multiresolution (MR) details as characteristics of models, and the MR coarse approximation as overall model structure.

- Method for patching voids in digital elevation models that produces highly realistic results.

- Novel approach to synthesizing human iris images from example irises.

In addition, this thesis makes the following improvements:

- Method of 3D rendering that improves the accuracy of the rendered terrain.
- Method of 3D rendering that emphasizes the details of the terrain.

1.5 Organization of Thesis

The thesis is organized in the following manner. Chapter 2 provides the necessary background information for multiresolution analysis and other techniques used throughout the work. Chapter 3 is a review of the reverse subdivision approach to multiresolution analysis. The following two chapters present the applications of the RS details: DEM void repair in Chapter 4 and the iris image synthesis in Chapter 5. Chapter 6 presents our results, and Chapter 7 presents our conclusions and a discussion of future work.

Chapter 2

Background

This chapter provides the necessary background information for techniques used in this work. Our focus is on using multiresolution to provide an approach for data synthesis. Therefore, there are two main areas of related works: multiresolution (Section 2.1) and data synthesis (Section 2.3).

Multiresolution (MR) and wavelets are introduced from a historical perspective in Section 2.1. Then, in Section 2.2, we introduce the theory behind wavelets and discuss some of the methods used to derive them. We introduce subdivision methods in Section 2.2.1, and its extensions for multiresolution methods in Section 2.2.2.

Then, we discuss works related to data synthesis (DS) in Section 2.3. Computer graphics has a significant body of works using DS for 3D model repair (Section 2.3.1), image completion (Section 2.3.2), texture synthesis (Section 2.3.3), and domain specific rendering (Section 2.4.1). Biometric researchers are turning their attention to synthesis techniques for the purpose of generating test data (Section 2.4).

We wrap up the Chapter by introducing two techniques that are used in our synthesis method: Hermite curves (Section 2.5.1) and bilinear interpolation (Section 2.5.2).

2.1 Multiresolution Background

A motivation for multiresolution methods comes from the fact that geometric objects can be observed and represented at different resolutions. We can view a hill from a distance and observe its overall structure, or we can view it up-close and observe its details. Unlike geometric objects, a digital model has a resolution that is determined by the distance between its measurements. Multiresolution (MR) is a broad term encompassing all methods that change the resolution of digital models.

Although, it was not labelled as such, an early form of MR is the *image pyramid* approaches used for image processing. They are motivated by the human visual system which can simultaneously view all resolutions of an image [40, 41]. The underlying approach is to construct an image at multiple resolution levels, resulting in a pyramid like structure [23, 13, 39].

Wavelets are a "mathematical tool for hierarchically decomposing functions", [63]. In our case, the digital model obtained from a geometric object represents the function. This means that wavelets can decompose a model into a coarse approximation and some detail components. The detail components represent the characteristics for our synthesis method, and subsequently wavelets play an integral role in our work. Conversely, Fourier transforms decompose a function into its frequency components, and *pyramids methods* only provide the coarse representation of the image.

In 1985, Morlet [36] showed that arbitrary signals could be represented with scales and translations of a single wavelet. Mallat and Meyer [50, 53] further developed this notion into the now classical form of MR. Later, Mallat [51] demonstrated that wavelets are equivalent to *image pyramids.* Stollinitz et al. showed that wavelet theory is closely related to recursive subdivision [63].

In the following Section (2.2), we introduce the high-level concepts behind wavelets using the notation from [63]. The intimate relationship with subdivision prompts many works on wavelets to formulate their notation and derivation in terms of subdivision matrices [32]. We adopt this approach and, in Section 2.2.1, subdivision is introduced from a high level. Then, in Section (2.2.2), we extend the subdivision notation to a complete MR representation.

2.2 Wavelet Theory

In wavelet theory f^k denotes a signal at resolution level k which exists in the linear function space V^k, where its dimension is denoted by $v(k)$. V^k is one member of a nested set of function spaces, $V^0 \subset V^1 \subset V^2 \subset \ldots$, where the resolution increases with k. Each of these function spaces has a set of basis functions $\Phi^k = \left[\phi_0^k, \phi_1^k, \ldots, \phi_{v(k)-1}^k\right]$, also denoted as the *scaling* functions for V^k. Each $f^k \in V^k$ can be written as a linear combination of the *scaling* functions. Therefore, f^k can be represented by C^k that is a column vector containing the coefficients of Φ^k: $\left[c_0^k, c_1^k, \ldots, c_{n-1}^k\right]^T$. There exists a relationship between the nested spaces that is captured by the matrix $\mathbf{P}^k$ where [63]:

$$\Phi^{k-1} = \Phi^k \mathbf{P}^k. \tag{2.1}$$

In other words, each of the lower resolution *scaling* functions Φ^{k-1} can be represented as a linear combination of the higher resolution *scaling* functions Φ^k.

This relationship is enough to define the process which refines the coefficients of C^k into a new set of coefficients C^{k+1} that are in the higher resolution space, V^{k+1}. The process is also known as recursive subdivision. The next step for multiresolution is to define the wavelets space, W^k where the dimension is denoted by $w(k)$. The term *wavelets* is commonly used for wavelet systems in general, but it more accurately refers to the basis functions of W^k: $\Psi^k = \left[\psi_0^k, \psi_1^k, \ldots, \psi_{w(k)-1}^k\right]$. W^k is chosen such that it is the complement space of V^k in V^{k+1}; this means that $V^k \cup W^k = V^{k+1}$. Therefore, any function in V^{k+1} can be written as the sum of a unique function in V^k and a unique function in W^k. This choice of a wavelet space provides another important relationship that is captured by the matrix $\mathbf{Q}^k$:

$$\Psi^{k-1} = \Phi^k \mathbf{Q}^k. \tag{2.2}$$

In other words, each of the lower resolution *wavelet* functions Ψ^{k-1} can be represented as a linear combination of the higher resolution *scaling* functions Φ^k.

The matrices $\mathbf{P}^k$ and $\mathbf{Q}^k$ define the transition from level k to $k+1$, an increase in resolution. For multiresolution, we are also interested in the opposite transition from k to $k-1$, a decrease in resolution. In order to perform this transition, we need to decompose Φ^k into complementary basis functions, Φ^{k-1} and Ψ^{k-1}. The decomposition operation is usually represented by two matrices, $\mathbf{A}^k$ and $\mathbf{B}^k$, such that:

$$\left[\begin{array}{c|c} \Phi^{k-1} & \Psi^{k-1} \end{array}\right] \left[\begin{array}{c} \mathbf{A}^k \\ \hline \mathbf{B}^k \end{array}\right] = \Phi^k. \tag{2.3}$$

It is important to note that, for wavelet theory, C^k denotes a function or a signal. However, for this work we assume that a geometric object has a functional representation. Thus, the digital model contains the coefficients sampled from the object's function and is equivalent to C^k.

Haar wavelets are the best known and have a history of usage in many scientific domains, including engineering, signal processing [63] and biometrics [60]. They are a set of step functions that correspond to first order B-Splines. The matrices $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{A}$ and $\mathbf{B}$ are sparse and allow for efficient operations. However, the scaling and wavelet functions are non-continuous and smooth objects are best represented with higher order B-Splines [60].

Chui and Quack [49], Quack and Weyrich [18], and Lyche and Mørken [57] further investigated B-splines wavelets. Their work included both the bounded and unbounded cases and used the inner product on the L_2 space. In 1994, Finkelstein and Salesin [32] showed how wavelets for cubic B-splines can be used to obtain a multiresolution curve representation. Unser et. al. [1] used least squares data fitting for B-spline wavelets on the infinite line. In contrast to [49, 18, 32, 57], Unser et. al. looked at using the inner product on the more general space ℓ_2.

In 1999 Samavati and Bartels [59] used a purely linear algebra approach to deriving the MR matrices by reversing subdivisions. The main idea behind their work is to use a global least-squares data fitting technique. Similar to [1], they use the inner product on ℓ_2, however their method was not restricted to B-spline scaling functions and treated data of finite extent. By using the inner product on ℓ_2, they were able to produce wavelets that are more compact than traditional wavelets. The resulting matrices $\mathbf{P}^k$ and $\mathbf{Q}^k$ were banded and their operations could be implemented in linear time. However, due to the nature of the wavelets, they had to appeal to matrix factorization to achieve an efficient implementation of the $\mathbf{A}^k$ and $\mathbf{B}^k$ matrices. Later, Bartels and Samavati [6] further improved this method by using a local least-squares approach. In this work, all four resulting matrices are banded and consequently they have a very fast, linear time implementation. Particularly pleasing are their results in reversing the subdivision rules for Chaikin subdivision; the values contained in the matrices are simple fractions that mimic the ones used in the Chaikin subdivision scheme, see Sections 3.1.2 and 3.1.2.

2.2.1 Subdivision

In terms of MR theory, subdivision is a process that refines the coefficients of C^k into a new set of coefficients C^{k+1} that are in the higher resolution space, V^{k+1} [63]. More practically for our work, subdivision is the process of increasing the resolution of a model. Recall that the matrix $\mathbf{P}^{k+1}$ represents the relationship between Φ^k and Φ^{k+1}. Also recall that C^k represents the coefficients of Φ^k. Essentially, this means that $\mathbf{P}^{k+1}$ represents a conversion of C^k's coefficient terms into a new set of coefficients C^{k+1} that are in V^{k+1}. As a result, we can apply $\mathbf{P}^{k+1}$ to C^k to increase its resolution:

$$\mathbf{P}^{k+1}C^k = C^{k+1}. \tag{2.4}$$

The set of nested subspaces that include V^k are infinite and we can further increase the resolution of C^{k+1} by applying the subdivision matrix, P^{k+2}. In many situations, including ours, we can find a continuous representation(piecewise linear curve) for C^k by connecting the values of C^k with lines. C^k represents the coefficients of a piecewise linear curve, see Figure 2.1 (a). The limit curve of subdivision schemes is usually a smooth curve such as the one in Figure 2.1 (d).

The first subdivision scheme was introduced by Chaikin [17] in 1974. His algo-

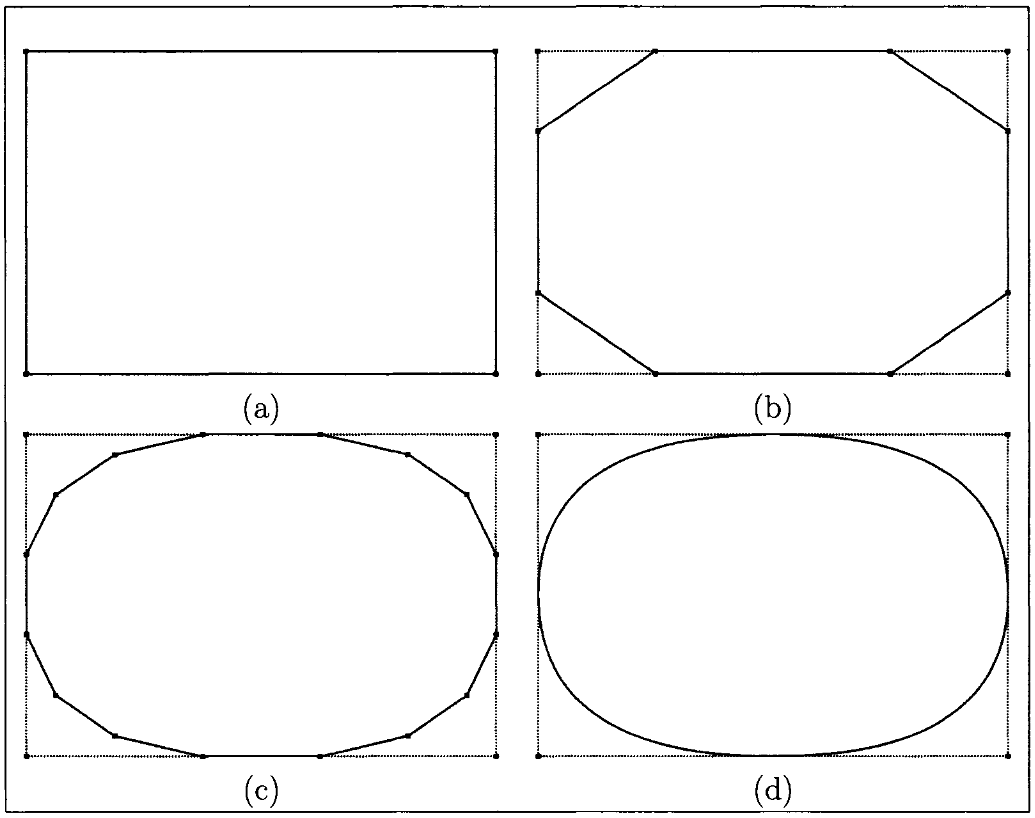


Figure 2.1: Chaikin subdivision example. (a) Original model consisting of four points. (b) After one level of subdivision. (c) After two levels of subdivision. (d) The limit curve.

rithm is equivalent to replacing every point, c_i^k with two new points, one at $\frac{3}{4}c_i^k+\frac{1}{4}c_{i-1}^k$, and one at $\frac{3}{4}c_i^k+\frac{1}{4}c_{i+1}^k$, Figure 2.1 (b) and (c). Later, Lane and Riesenfeld [42] showed that his "corner cutting" algorithm actually generated quadratic B-splines curves as their limit curve. Expressed in matrix terms the first step from the example in Figure 2.1 is equivalent to:

$$\begin{bmatrix} \frac{3}{4} & 0 & 0 & \frac{1}{4} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & \frac{3}{4} \end{bmatrix} \begin{bmatrix} c_0^k \\ c_1^k \\ c_2^k \\ c_3^k \end{bmatrix} = \begin{bmatrix} c_0^{k+1} \\ c_1^{k+1} \\ \vdots \\ c_7^{k+1} \end{bmatrix}$$

For our work, we only consider Chaikin subdivision. This is due, in part, to the good behavior of B-splines such as affine invariance, local control and the convex hull property, [4]. In addition, the Chaikin subdivision filters from [6] are simple and have an easy implementation with a fast, linear running time.

Subdivision research typically focuses on methods for both curves and arbitrary topology surfaces. The curve case can easily be extended to tensor product surfaces such as DEMs and iris images. Therefore, we do not cover subdivision methods for arbitrary topology surfaces.

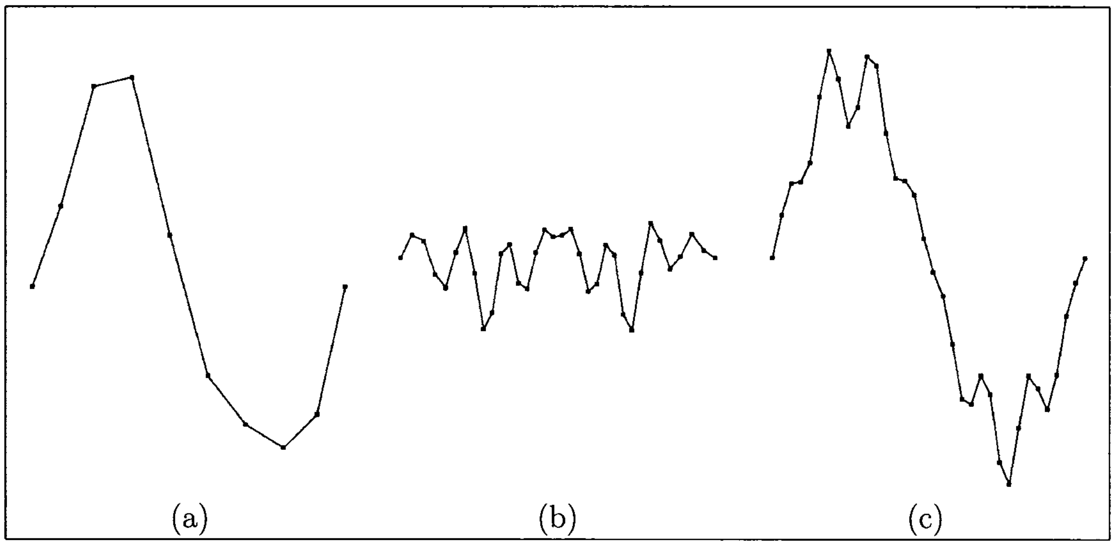


Figure 2.2: Multiresolution representation of a curve. (a) A smooth underlying curve. (b) High frequency characteristics (c) Original curve.

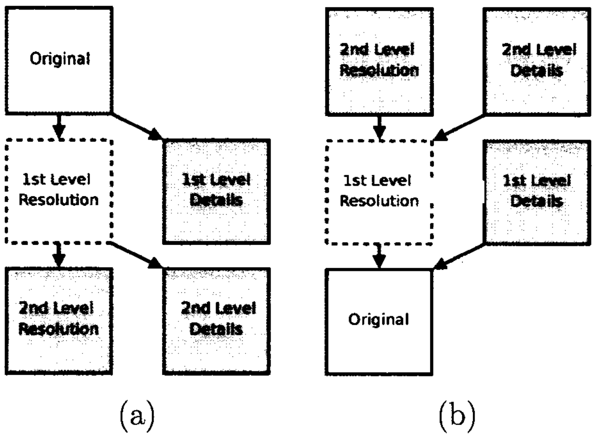


Figure 2.3: (a) Multiresolution decomposition process. (b) Multiresolution reconstruction process. The reconstruction process only requires the low resolution approximation and each level of details extracted during the decomposition process in order to fully recreate the original object.

2.2.2 Multiresolution Notation

Subdivision provides the facility to increase the resolution of models. We now discuss mathematical notation used when extending subdivision to a full multiresolution tool. Multiresolution (MR) is the more general process; it is concerned with increasing and reducing the resolution of some model C^k.

Reducing the resolution of the C^k occurs in two steps, denoted as the *decomposition* process. In the first step, the coefficients of C^k are coarsened into a new set of coefficients C^{k-1} that are in the lower resolution space V^{k-1}. The coarse approximation C^{k-1} will not contain some of the information that is present in the original model C^k. Therefore, a second step is performed on C^k that extracts the missing information D^{k-1} that is in the lower resolution space W^{k-1}.

Increasing the resolution of C^{k-1} occurs in three steps, denoted as the *reconstruction* process. In the first step, the coefficients of C^{k-1} are refined into a new set of coefficients $C^{k'}$ that are in the higher resolution space V^k. Then, the detail coefficients D^{k-1} are refined into a new set of coefficients $D^{k'}$ also in the higher resolution space V^k. The final step in the reconstruction process combines the two components $C^{k'}$ and $D^{k'}$ into the original model C^k.

Decomposition

Recall that the matrix $\mathbf{A}^k$ represents the relationship between Φ^k and Φ^{k-1}. Essentially, this means that $\mathbf{A}^k$ represents a conversion of C^k's coefficient terms into a new set of coefficients C^{k-1} that are in V^{k-1}: a resolution decrease. As a result, we can apply $\mathbf{A}^k$ to C^k to decrease its resolution:

$$C^{k-1} = \mathbf{A}^k C^k. \tag{2.5}$$

The low resolution coefficients C^{k-1} contains a coarse approximation of the original model in V^{k-1} and can be considered equivalent to the overall structure of the model.

The original model C^k exists in the higher resolution space V^k. Therefore C^{k-1} will not contain some of the information that is present in the original model. These differences, denoted as the *details*, should be stored such that the original model can be reconstructed. In MR terms, these differences will exist in the space complementary to V^{k-1}: the wavelet space W^{k-1}. Recall that $\mathbf{B}^k$ represents the relationship between Φ^k and Ψ^{k-1}. Essentially, this means that $\mathbf{B}^k$ represents a conversion of C^k's coefficient terms into a new set of *detail* coefficients D^{k-1} that are in W^{k-1}. Therefore we can use $\mathbf{B}^k$ to extract the *details* from C^k:

$$D^{k-1} = \mathbf{B}^k C^k. \tag{2.6}$$

D^{k-1} contain the *detail* information extracted from the original model, and can be considered equivalent to the characteristics, or features of the model.

Reconstruction

Finally, when we want to reconstruct the original model, we should recombine the *details* and the low resolution model. The subdivision operations discussed in Section 2.2.1 will suffice to convert C^{k-1} back into the higher resolution space. However, we must also include the details in order to exactly reproduce the original model. Recall that the matrix $\mathbf{Q}^k$ represents the relationship between Ψ^{k-1} and Φ^k. Essentially this means that $\mathbf{Q}^k$ represents a conversion of D^k's detail coefficients from W^{k-1} to V^k. Therefore we use both $\mathbf{P}^k$ and $\mathbf{Q}^k$ to reconstruct the original model:

$$C^k = \mathbf{P}^k C^{k-1} + \mathbf{Q}^k D^{k-1}. \tag{2.7}$$

Matrix Properties

There are a number of properties of $\mathbf{A}^k$, $\mathbf{B}^k$, $\mathbf{P}^k$, and $\mathbf{Q}^k$ that are desirable. The most important property is that, for the original points C^k be exactly reconstructible from C^{k-1} and D^{k-1}, we must have [63]:

$$\left[\begin{array}{c}\mathbf{A}^k \\ \hline \mathbf{B}^k\end{array}\right]\left[\begin{array}{c|c}\mathbf{P}^k & \mathbf{Q}^k\end{array}\right] = I. \tag{2.8}$$

Although it is not strictly required, it is desirable that the matrices have a banded, regular structure in which each column is a shifted version of the previous column. As a result, it is straightforward to provide an implementation that avoids matrix multiplication and has linear running time [60]. In addition, applying $\mathbf{A}^k$ must produce a coarse approximation C^{k-1} that is a good approximation for the original points C^k. Or, equivalently, the impact of the details must be minimized. Additionally it is desired that the storage required for coarse approximation C^{k-1} and the details D^{k-1} is not more than the storage requirements for the original model C^k.

2.2.3 Multiresolution Applications

Multiresolution methods and wavelets have been applied in biometric and other image processing areas for iris identification [25, 38, 75, 81], for fingerprint edge detection and matching [72, 70], for document contour extraction [77], as well as for handwritten numeral recognition [58]. Recently, Chaikin reverse subdivision (RS) filters have been proposed as alternatives to Haar wavelets for biometric applications [60]. RS methods have been used successfully in a number of areas; polygonal silhouette error correction [33], visualization of clinical volume data [65]. More closely related to this work, RS methods have been used in a synthesis capacity. Brosz modeled terrain by example, using MR to match the resolution of the terrains and provide the finer details needed for synthetic terrains of higher resolution [10, 11]. In order to extract characteristics of line drawings, Brunn used MR to decompose the drawings which allowed him to carry the characteristics to other line drawings [12].

2.3 Synthesis Background

Data synthesis refers to the creation of new data to meet some intended purpose. In this section we review some of the areas of data synthesis that relate to our approach. For our work, data synthesis refers to the combining of components to form new digital models. We use a multiresolution process to decompose example models into re-usable components for the synthesis process. In some cases, we use the synthesized data to repair missing areas in existing models. The approaches to 3D model repair (Section 2.3.1) and image completion (Section 2.3.2) are similarly motivated to fill in missing areas in their data. Our approach can be described as a context sensitive approach, because the synthesis components are extracted from data near to the missing areas. In other cases, the synthesis process is used to produce new models which fill a need for more sample data. Approaches to texture synthesis (Section 2.3.3), domain specific rendering (Section 2.4.1) and biometric synthesis (Section 2.4) are also motivated to produce more data with the certain characteristics.

2.3.1 3D Model Repair

In the computer graphics area, the general inconsistencies in automatically scanned data pose an ubiquitous problem for data processing [15]. Due to the varied nature of the data and the voids contained therein, the approaches are also significantly varied. Traditional approaches focus on providing a smooth patch for the missing areas including: the missing areas of polyhedron produce by CAD software [3], the voids and scarred areas of 3D models [19, 26], and the voids in unstructured triangle meshes [46].

Similar to our method, Sharf et. al. [62] introduced a context sensitive approach for patching the holes in 3D models. They were motivated by holes that are large compared with the characteristics of the surface. As a result, a smooth patch would appear out of place when compared with the surrounding characteristics.

2.3.2 Image Completion

Image completion and texture inpainting are two methods that focus on repairing the missing portions of images that can occur if the image is damaged or some large foreground object has been removed. The surrounding areas of the image are used as training for the inpainting methods, followed by a filling phase that attempts to plausibly patch the missing areas [7, 29, 44]. Extensions to these techniques include using inpainting methods on surface models [68], inpainting based on similarity, [22, 55] and inpainting based on Markov Random Fields [79]. Further research investigates structure propagation, by selectively using both texture synthesis and texture transfer [8] and by using user input to direct structure propagation [64].

2.3.3 Texture Synthesis and Texture Transfer

Over the past decade, researchers have had significant interest in texture synthesis and texture transfer. In texture synthesis, the most common recent approach is to provide a sample texture that is used to produce a new texture that displays similar qualities. The methods vary from a pyramid-algorithms [9] to Markov random fields [31, 74]. Additional techniques use patch based sampling, [45], and some are real time approaches, [74, 45]. Improvements have been made in domain specific areas, [2] and further improvements are made in [80].

Texture transfer uses two sample input images and produces a new image that contains similar features to both input textures. Typically, one of the input images provides the texture and the other provides the features for the output image. Approaches to texture transfer and texture synthesis include Wang tiles, [20], image quilting [30], and image analogies [37].

2.4 Biometric Synthesis Background

Biometrics conventionally involves the analysis of biometric data for identification purposes. Due to logistical and privacy issues with collecting and organizing large amounts of biometric data, a new direction of biometric research concentrates on the synthesis of biometric information. One of the primary goals of the synthesis of biometric data is to provide databases on which the classical biometric algorithms can be tested [78, 35].

Cappelli et al. describes a complete system for human fingerprint image generation [14]. Their system uses fingerprint classifications to generate the global shape of each fingerprint. This global shape then undergoes a series of deformations and random variations in order to achieve the master ridge patterns. Once the master ridge patterns are known, their system can generate multiple samples of the original image using combinations of deformations throughout the synthesis process. They validate their results by exploiting the ubiquity of fingerprint identification and the subsequent availability of fingerprint experts and competitions.

Recently, Luo and Gavrilova proposed an approach for facial synthesis and expression modeling based on the underlying mesh modification. Selection of control points in their method is guided by the 3D Voronoi diagram [48]. The general overview of utilization of geometric algorithms in real-time terrain rendering and facial expression modeling can be found in Plenary lecture delivered by Gavrilova at 3IA'2006, Limoges, France [34].

Cui et al. proposed a method to synthesize iris images, based on principal component analysis (PCA) and super-resolution [24]. Using their experimental results, they conclude that a 75 dimensional PCA global feature vector provides the most benefit to the performance of the PCA recognition method. The synthesis method uses the same size feature vector, and the problem is constrained to a search in a limited high dimensional space. Once a feature vector has been synthesized with this search, they use super-resolution to obtain an iris image of the desired size. Using their method they generated a database of ten thousand iris classes with fifty one iris images in each class.

2.4.1 Domain Specific Rendering

Further afield Lefohn et al. introduced a method to synthesize human irises for use in computer graphics applications [43]. The method makes use of the domain knowledge of ocularists to obtain results that take on a high level of realism. Their approach uses 30-70 layers of painted textures, scanned into the program using a conventional flat bed scanner.

2.5 Additional Background

Although our method concentrates on data synthesis with multiresolution, we use a number of other techniques that are fairly well known. This section provides the reader with the necessary information to understand our application of these methods.

2.5.1 Hermite Interpolation

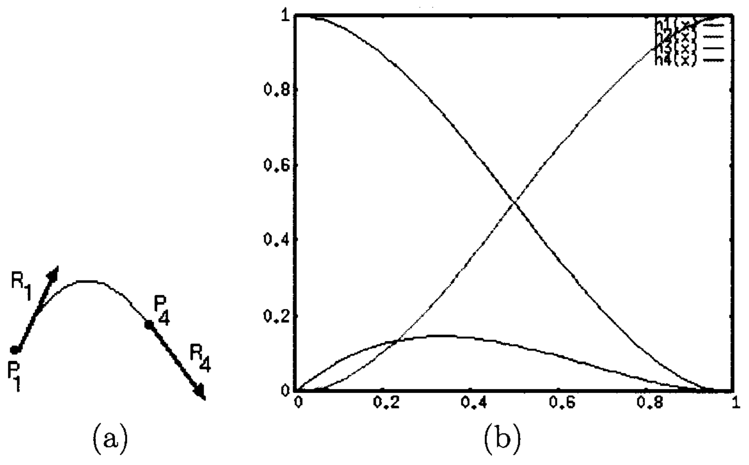


Figure 2.4: (a) Hermite curve. (b) Hermite Basis functions.

Hermite interpolation is a method of building a smooth curve that interpolates a set of points [4]. Typically, it achieves the overall curve by focusing on providing a single cubic polynomial interpolating two given points p_i and p_{i+1} such that the segment has G^1 continuity with the surrounding segments. The definition of the cubic equation is given by:

$$Q(u) = a_1 + a_2 u + a_3 u^2 + a_4 u^3 \tag{2.9}$$

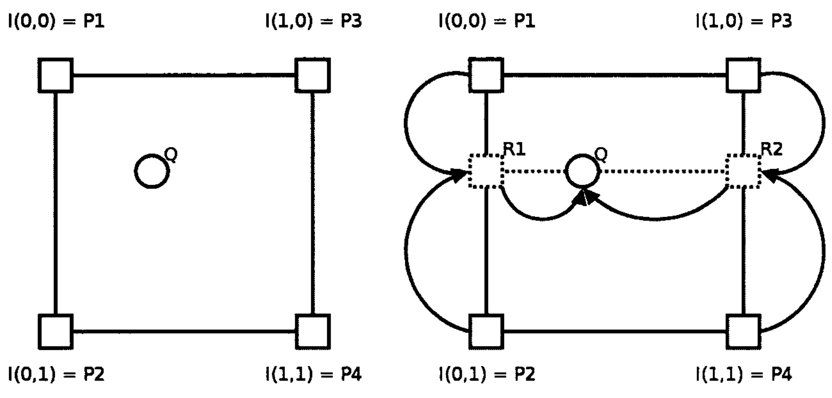
Assuming that we are given the derivative of $Q(u)$ at $Q\prime(p_i) = r_i$ and $Q\prime(p_{i+1}) = r_{i+1}$, then we can symbolically solve for a_1, a_2, a_3 and a_4:

$$\begin{aligned} a_1 &= p_i & (2.10) \\ a_2 &= d_i & (2.11) \\ a_3 &= 3(p_{i+1} - p_i) - 2r_i - r_{i+1} & (2.12) \\ a_4 &= 2(p_i - p_{i+1}) + r_i + r_{i+1}) & (2.13) \end{aligned}$$

Using these equations, we can easily find the value of the Hermite Curve at u, by performing the arithmetic from 2.9.

2.5.2 Bilinear Interpolation

Bilinear interpolation is a commonly used technique for sampling a texture when the desired value falls in between four known values from the texture. Consider, Figure 2.5 (a), where the values for of $I(x, y)$ are known at $I(0,0) = P_1$, $I(0,1) = P_2$, $I(1,0) = P_3$ and $I(1,1) = P_4$, but we are interested in finding $Q = I(x_1, y_1)$. Given a setup like this, it is straightforward to find Q using two linear interpolations. In the first step we find R_1 by linearly combining P_1 and P_2 and then find R_2 similarly



(a) (b)

Figure 2.5: Bilinear interpolation: (a) Assuming that P_1, P_2, P_3, and P_4 are given, and Q is wanted. (b) The interpolation is performed as two separate linear interpolation steps, hence the name.

using P_3 and P_4. Then, in the second step we linearly combine R_1, and R_2 to find Q, resulting in a final combination of:

$$Q = I(x_1, y_1) = \begin{bmatrix} (1 - x_1) & x_1 \end{bmatrix} \begin{bmatrix} P_1 & P_2 \\ P_3 & P_4 \end{bmatrix} \begin{bmatrix} (1 - y_1) \\ y_1 \end{bmatrix} \tag{2.14}$$

Chapter 3

The Multiresolution Synthesis System

In this thesis, we explore the use of Reverse Subdivision (RS) details as a synthesis tool. This chapter provides an overview of the synthesis process using RS. First, we cover RS as it applies to a curve case in Section 3.1.1. Then we extend this to digital elevation models (DEM) and images in Section 3.1.3. Section 3.2 provides a discussion of the details as characteristics. We then discuss how to visualize details and wrap this chapter up with some illustrations of the decomposition of both iris images and terrain.

3.1 Reverse Subdivision

Conventionally, wavelets are employed to obtain an efficient and compact representation of the original data [63]. Samavati and Bartels introduced a method for multiresolution curve representation [5] that has local multiresolution filters produced from subdivision methods.

In this work we have used multiresolution filters of reverse Chaikin subdivision. Chaikin subdivision is a discrete approximation of quadratic B-Spline while Haar wavelets are based on zero degree B-Splines (piecewise constant). Although Haar filters are simple, their scaling and wavelet functions are not continuous. Smooth objects are, therefore, better represented by higher degree B-Splines and preferred for our purposes [66, 59]. Chaikin reverse subdivision filters have been proposed as alternatives to the traditional Haar wavelets for applications in biometrics [60]. We chose these filters as they satisfy all of the properties needed for Multiresolution and they provide the banded regular structure with shifted filter values for efficient implementation. In addition, the filters obtained from this work are simple fractions that minimize roundoff errors. In addition, they exhibit bands that are more narrow than the bands produced by conventional wavelets [59]. Therefore, the filters are more compact (see Figure 3.1). This compactness allows the filters to be used locally; that is, they only consider the data in a local neighborhood for the decomposition process, that is desirable for many applications, including ours.

$$[0,\ \ldots,\ 0,\ -\tfrac{1}{303},\ \tfrac{29}{303},\ -\tfrac{147}{303},\ \tfrac{303}{303},\ -\tfrac{303}{303},\ \tfrac{147}{303},\ -\tfrac{29}{303},\ \tfrac{1}{303},\ 0,\ \ldots,\ 0\ ]^T$$

$$[\ 0,\ \ldots,\ 0,\ -\tfrac{1}{4},\ -\tfrac{3}{4},\ \tfrac{3}{4},\ \tfrac{1}{4},\ 0,\ \ldots,\ 0\ ]^T$$

Figure 3.1: Comparison of the filters from the **Q** matrix (from Equation 2.7). On the top are the filters derived using the traditional wavelet approach [63]. On the bottom are the filters derived using the reverse subdivision approach [59].

A curve can have a non-unique parametric representation $Q(u)$ that is defined by a set of control points $p_1, p_2, \ldots, p_n$. The curve itself is constructed by evaluating over the parametric range (usually $[0,1]$). Tensor-product surfaces $Q(u,v)$ are extended from the parametric form of curves and are defined by an $m \times n$ set of control vertices:

$$\begin{bmatrix} p_{1,1} & p_{2,1} & \cdots & p_{n,1} \\ p_{1,2} & p_{2,2} & \cdots & p_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,m} & p_{2,m} & \cdots & p_{n,m} \end{bmatrix}$$

with two independent parameters: u and v that correspond to the rows and columns of the model. The surface itself is constructed by evaluating over the parametric range for one of the parameters, and then the other. If we fix the u parameter at a specific value and vary the v parameter, we obtain the parametric form of a single curve denoted as a u curve. v curves are similarly defined by fixing the v parameter and varying u. We can use the multiresolution representation for curves to define a multiresolution representation for tensor-product surfaces.

Multiresolution methods have also been defined for arbitrary-topology surfaces. However, in contrast to tensor-product surfaces, arbitrary-topology surfaces do not have a restriction on their representation. As a result, multiresolution methods for arbitrary-topology surfaces are more complex than those for tensor-product surfaces [47, 61]. We do not investigate arbitrary-topology surfaces in this research, as reverse subdivision for tensor product surfaces is sufficient for our applications. This is not a limitation of our framework as it can be applied to arbitrary-topology surfaces provided they have a local multiresolution representation.

3.1.1 Reverse Subdivision for Curves

Using the notation and equations from Section 2.2.2, assume that a high resolution curve, C^{k+1}, is given and that we are interested in finding a lower resolution approximation C^k. Although it can be convenient to assume that C^{k+1} was produced by subdividing the lower resolution curve, C^k, this is not necessarily the case. Rather, Samavati and Bartels approach this problem using least squares data fitting to search for a C^k such that $||C^{k+1} - \mathbf{P}^{k+1}C^k||_2^2$ is minimized [59, 6]. In their method, the details vector D^k is computed from the residual vector using:

$$C^{k+1} - \mathbf{P}^{k+1}C^k = \mathbf{Q}^{k+1}D^k \tag{3.1}$$

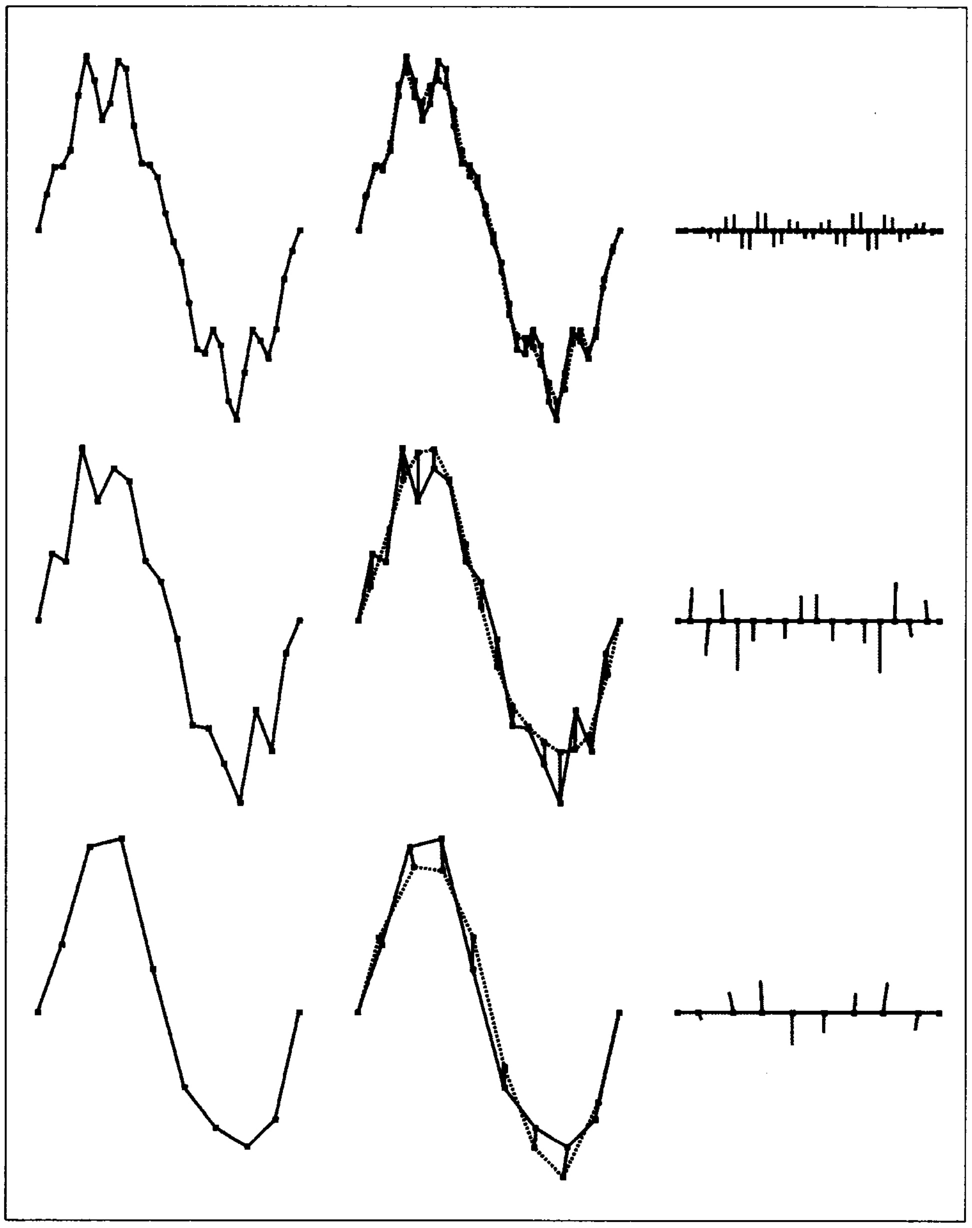


Figure 3.2: Decomposition of a curve. C^i is on the left and in the middle(*blue*). In the middle the subdivided coarse data $\mathbf{P}^i C^{i-1}$ (*dashed-black*) with the residuals(*red*): $C^i - \mathbf{P}^i C^{i-1}$. On the right, residuals are displayed relative to a flat curve.

where Q^{k+1} is the orthogonal complement of P^{k+1} (see Equation 2.7). To provide an intuition of this details vector, consider Figure 3.2. In this figure, the original curve is on the left, and the middle column contains both the original curve and $\mathbf{P}^{k+1}C^k$, the curve that minimizes $||C^{k+1} - \mathbf{P}^{k+1}C^k||_2^2$ which are rendered using a grey dashed line and a blue line respectively. The red lines illustrate the residual vectors, $\mathbf{Q}^{k+1}D^k$.

Using this setting they derive all four filter matrices $\mathbf{A}^n$, $\mathbf{B}^n$, $\mathbf{P}^n$, and $\mathbf{Q}^n$. These, along with Equations 2.4, 2.5, 2.6, and 2.7 provide a complete multiresolution representation for curves.

3.1.2 Reverse Chaikin Multiresolution Filters

For some applications, including the iris image synthesis, the data can be viewed as periodic (circular curve as u and radial lines as v curves). In this setting, the derivation of the RS filter matrices can be done in a uniform manner with no special case conditions. However, in other applications such as the terrain repair application, the data is non-periodic. For our terrain application, the end points of each curve should line-up with the surrounding terrain. To accommodate this, RS filter matrices have special starting and ending conditions that preserve the end points.

Periodic Setting

In the periodic setting, the filter matrices for a small system are:

$$\mathbf{A}^k = \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 & -\frac{1}{4} & \frac{3}{4} \\ -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} \end{bmatrix}$$

$$\mathbf{B}_r^k = \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{4} \\ \frac{1}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{1}{4} \end{bmatrix}$$

$$\mathbf{P}_r^k = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & \frac{3}{4} \\ \frac{3}{4} & 0 & 0 & \frac{1}{4} \end{bmatrix}$$

$$\mathbf{Q}_r^k = \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & 0 & 0 \\ \frac{1}{4} & -\frac{3}{4} & 0 & 0 \\ 0 & \frac{3}{4} & -\frac{1}{4} & 0 \\ 0 & \frac{1}{4} & -\frac{3}{4} & 0 \\ 0 & 0 & \frac{3}{4} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{4} & -\frac{3}{4} \\ -\frac{1}{4} & 0 & 0 & \frac{3}{4} \\ -\frac{3}{4} & 0 & 0 & \frac{1}{4} \end{bmatrix}$$

Notice that the filter matrices are banded. Each column in the matrix consists of the same set of filter values shifted by two rows. We can make use of this regular structure to create a linear implementation. Rather than perform matrix multiplication, a set of filter values from each matrix are used for the decomposition and reconstruction steps: $\left[-\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, -\frac{1}{4}\right]$, $\left[\frac{1}{4}, -\frac{3}{4}, \frac{3}{4}, -\frac{1}{4}\right]$, $\left[\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}\right]^T$, $\left[-\frac{1}{4}, -\frac{3}{4}, \frac{3}{4}, \frac{1}{4}\right]^T$, are the filter values for the matrices $\mathbf{A}^n$, $\mathbf{B}^n$, $\mathbf{P}^n$, $\mathbf{Q}^n$ respectively [6].

In order to demonstrate the decomposition steps, consider a coefficient vector of eight points: $C^k = \left[C_1^k, C_2^k, C_3^k, C_4^k, C_5^k, C_6^k, C_7^k, C_8^k,\right]$. Using (Equation 2.5), the initial coarse point C_1^{k-1} is calculated from $\frac{1}{4}C_7^k + \frac{3}{4}C_8^k + \frac{3}{4}C_1^k + \frac{1}{4}C_2^k$. Subsequent coarse points are calculated using: $C_i^{k-1} = \frac{1}{4}C_{2i-3}^k + \frac{3}{4}C_{2i-2}^k + \frac{3}{4}C_{2i-1}^k + \frac{1}{4}C_{2i}^k$ where $i \in [2, 3, 4]$. This can be performed in an iterative fashion, leading to a general algorithm with a single beginning case and a single loop resulting in a linear time implementation. The process of calculating D^{k-1} from C^k is identical, but uses the filter values from B^k.

The reconstruction steps for C^{k-1} and D^{k-1} are similar. The first set of points in

C^k are calculated using two equations: $C_i^k = \frac{3}{4}C_i^{k-1} + \frac{1}{4}C_{i+1}^{k-1} + \frac{3}{4}D_i^{k-1} + -\frac{1}{4}D_{i+1}^{k-1}$ and $C_{i+1}^k = \frac{1}{4}C_i^{k-1} + \frac{3}{4}C_{i+1}^{k-1} + \frac{1}{4}D_i^{k-1} + -\frac{3}{4}D_{i+1}^{k-1}$ where $i \in [1, 2, 3]$. The final two points in C^k are also calculated using two equations: $C_7^k = \frac{3}{4}C_4^{k-1} + \frac{1}{4}C_0^{k-1} + \frac{3}{4}D_4^{k-1} + -\frac{1}{4}D_0^{k-1}$ and $C_8^k = \frac{1}{4}C_4^{k-1} + \frac{3}{4}C_0^{k-1} + \frac{1}{4}D_4^{k-1} + -\frac{3}{4}D_0^{k-1}$. As in the decomposition, these steps can be represented by a single loop with two final cases resulting in a linear time implementation.

Endpoint Interpolating Setting

For the endpoint interpolated setting, the filter matrices $\mathbf{A}^n$, $\mathbf{B}^n$, $\mathbf{P}^n$, and $\mathbf{Q}^n$ will still contain the single set of repeating, shifted filter values as in Section 3.1.2. However, they will also contain special starting and ending arrangements that differ from the repeated values. For RS, it is common to denote the matrices using a block-matrix notation (where the subscripts s, r, and e represent the starting condition, the repeating values and the ending condition respectively):

$$\mathbf{A}^k = \begin{bmatrix} \mathbf{A}_s^k \\ \mathbf{A}_r^k \\ \mathbf{A}_e^k \end{bmatrix}, \quad \mathbf{B}^k = \begin{bmatrix} \mathbf{B}_s^k \\ \mathbf{B}_r^k \\ \mathbf{B}_e^k \end{bmatrix}, \quad \mathbf{P}^k = \begin{bmatrix} \mathbf{P}_s^k \\ \mathbf{P}_r^k \\ \mathbf{P}_e^k \end{bmatrix}, \quad \mathbf{Q}^k = \begin{bmatrix} \mathbf{Q}_s^k \\ \mathbf{Q}_r^k \\ \mathbf{Q}_e^k \end{bmatrix},$$

where the actual matrices for the equations are:

$$\mathbf{A}_s^k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ -\frac{1}{2} & 1 & \frac{3}{4} & 0 & 0 & 0 & \cdots \end{bmatrix}$$

$$\mathbf{A}_r^k = \begin{bmatrix} 0 & 0 & -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & -\frac{1}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & \cdots \\ & & & & & & & & \ddots \end{bmatrix}$$

$$\mathbf{A}_e^k = \begin{bmatrix} \cdots & 0 & 0 & -\frac{1}{4} & \frac{3}{4} & 1 & -\frac{1}{2} \\ \cdots & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}_s^k = \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{3}{4} & \frac{1}{4} & 0 & 0 & 0 & \cdots \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{3}{4} & \frac{1}{4} & 0 & \cdots \end{bmatrix}$$

$$\mathbf{B}_r^k = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{4} & \frac{3}{4} & -\frac{1}{4} & \cdots \\ & & & & & & & & & & \ddots \end{bmatrix}$$

$$\mathbf{B}_e^k = \begin{bmatrix} \cdots & 0 & 0 & \frac{1}{4} & -\frac{3}{4} & 1 & -\frac{1}{2} \end{bmatrix}$$

$$\mathbf{P}_s^k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & \cdots \end{bmatrix}$$

$$\mathbf{P}_r^k = \begin{bmatrix} 0 & \frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots \\ 0 & \frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots \\ 0 & 0 & \frac{3}{4} & \frac{1}{4} & 0 & \cdots \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} & 0 & \cdots \\ 0 & 0 & 0 & \frac{3}{4} & \frac{3}{4} & \cdots \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \cdots \\ & & & & & \ddots \end{bmatrix}$$

$$\mathbf{P}_e^k = \begin{bmatrix} \cdots & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \cdots & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Q}_s^k = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ \frac{1}{2} & 0 & 0 & 0 & \cdots \\ -\frac{3}{4} & \frac{1}{4} & 0 & 0 & \cdots \\ -\frac{1}{4} & \frac{3}{4} & 0 & 0 & \cdots \\ 0 & -\frac{3}{4} & \frac{1}{4} & 0 & \cdots \\ 0 & -\frac{1}{4} & \frac{3}{4} & 0 & \cdots \end{bmatrix}$$

$$\mathbf{Q}_r^k = \begin{bmatrix} 0 & 0 & \frac{3}{4} & -\frac{1}{4} & 0 & 0 & \cdots \\ 0 & 0 & \frac{1}{4} & -\frac{3}{4} & 0 & 0 & \cdots \\ 0 & 0 & 0 & \frac{3}{4} & -\frac{1}{4} & 0 & \cdots \\ 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{4} & 0 & \cdots \\ 0 & 0 & 0 & 0 & \frac{3}{4} & -\frac{1}{4} & \cdots \\ 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{4} & \cdots \\ & & & & & & \ddots \end{bmatrix}$$

$$\mathbf{Q}_e^k = \begin{bmatrix} \cdots & 0 & 0 & 0 & \frac{1}{2} \\ \cdots & 0 & 0 & 0 & 0 \end{bmatrix}$$

Again, the decomposition and reconstruction algorithms can be implemented in linear time. The general process is similar to the periodic setting, but the starting and ending cases vary slightly. For a detailed explanation of the implementation we refer the reader to [60].

Size Constraints on the Data

Due to the nature of B-Spline scaling functions, the RS filters have size restrictions imposed on the data [60]. For non-periodic Chaikin subdivision, if we would like to perform l levels of decomposition, we should have curves with $2^l\kappa + 2$ samples where $\kappa > 2$. In the case of periodic Chaikin subdivision, each curve should have $2^l\kappa$ samples. When the size of the data does not match with this restriction, a number of approaches are used. In some cases, the data can be re-sampled such that it contains the appropriate number of points. In other cases there is more data on either side of the target areas and we can simply ensure that the target area is of the appropriate size.

3.1.3 Extending Curves to DEMs and Images

In this section we extend the curve case to images I^{k+1}. We consider images as tensor-product surface and apply the filters in a two step process. For the first step, the filters are simultaneously applied to all of the columns of the image. Denote column j of the image at resolution level k as I_j^k , and the corresponding details of column j as D_j^k. We can define column operations for decomposition using the same equations 2.5, and 2.6, but substituting I for C, giving us:

$$I_j^{k-1} = \mathbf{A}^k I_j^k, \tag{3.2}$$

$$D_j^{k-1} = \mathbf{B}^k I_j^k. \tag{3.3}$$

These operations result in both an intermediate coarse representation of the image from equation 3.2 and the corresponding details obtained from reducing the resolution of each column using equation 3.3. Due to the properties of RS, we can store both sets of data in the same storage as the original image. The middle column in Figure 3.3 shows the resulting image where the coarse representation is on the left, and the details are visualized as in Section 3.2.1 on the right.

For the second step, the intermediate coarse representation is further decomposed

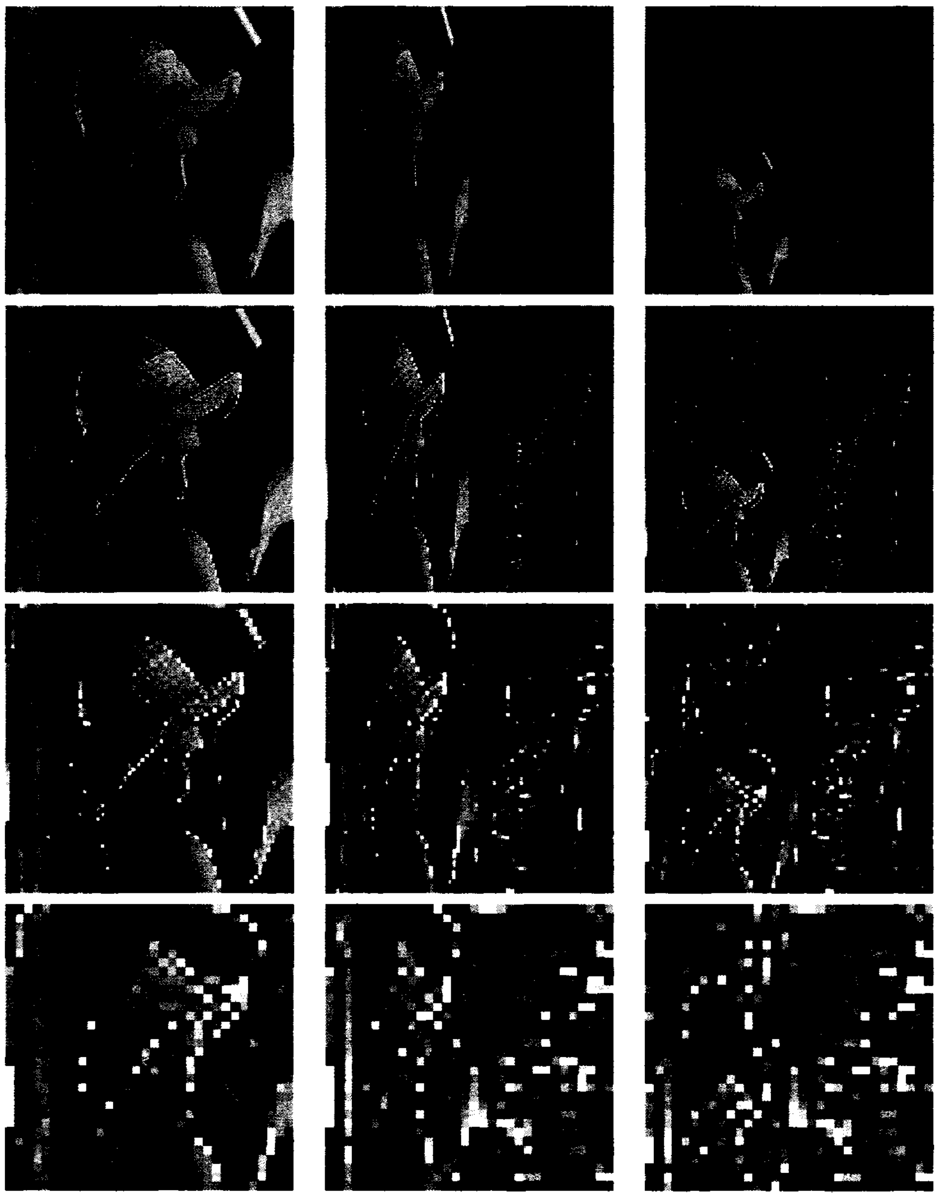


Figure 3.3: Chaikin decomposition of the standard image of Lena. The left column is the original and then subsequent coarse representations. The middle column is the image after reducing the resolution of the columns; the coarse representation on the left with the column-details on the right. The right column is the image after reducing the resolution of the rows; the coarse representation is in the bottom left, with the row-details in the upper left.

using equations 3.2 and 3.3, but applying them simultaneously to the rows rather than the columns. The final results of the decomposition are: a coarse approximation of the original image, and two sets of details corresponding to the columns and the rows respectively. The left column of Figure 3.3 shows the results of the decomposition after both the columns and the rows have been decomposed. Again, the final data has the same storage requirements as the original image. The two sets of details are stored and manipulated as one and are subsequently referred to as D^k throughout the rest of the work.

The process can be iteratively applied to the coarse approximations to further reduce the image's resolution. Rows two, three and four in Figure 3.3 are examples of further decomposition. Although the subsequent coarse representations and their details have the same storage requirements as their original data, they are displayed at the same resolution as the original image for illustration purposes.

The reconstruction process proceeds in a similar manner but with a reversed order for the columns and rows. Using the same notation as before to denote columns and rows, we can define the column operations for reconstruction using equation 2.7, but substituting I for C, giving us:

$$I_j^k = \mathbf{P}^k I_j^{k-1} + \mathbf{Q}^k D_j^{k-1}. \tag{3.4}$$

Again, the process is performed in two steps. For the first step, the filters are simultaneously applied to all the rows of coarse representation and the corresponding row-details. The results of this step are a fully reconstructed intermediate coarse representation. In the second step, the filters are simultaneously applied to all of the columns of the intermediate coarse representation and the corresponding column-details. The image obtained through the reconstruction process is an exact copy of the original image. Reverse subdivision for DEMs proceeds in an identical manner due to having the same data structure as images.

3.2 Details as Characteristics

In this section, we attempt to provide the reader with a better concept of the detail vectors that are extracted during RS. Throughout this work we will refer to the details as the characteristics of the original model. Therefore, it is important that the reader is familiar with this concept.

Consider Figure 3.2, that illustrates $\mathbf{P}^k C^{k-1}$, in blue, the high resolution curve C^k in dashed black, and the residual vectors $\mathbf{Q}^k D^{k-1}$ in red. In the right column, the detail vectors from each level of decomposition are displayed in red and relative to a straight line. From the mathematical description of the RS process, it is to be expected that the details represent the high-frequency characteristics of the curve that are lost during the decomposition. Confirmation of this can be made by observing the original curve in the figure. The curve, from left to right, goes in a general upwards direction followed by a general descent and then ending with another ascent. However, the high resolution characteristics are much more varied in their direction and represent higher frequency changes. Now observe the residual vectors in both the middle and left columns. They provide a good intuition for the high-frequency characteristics for each of the curves. Additionally, as we continue the decomposition, these vectors contain increasingly lower frequency information from the original data. This is to be expected, as each subsequent level of decomposition extracts increasingly lower level of resolution information. Recall that the residual vectors are represented by $\mathbf{Q}^k D^{k-1}$. $\mathbf{Q}^k$ is a constant defined by the multiresolution representation and it is therefore D^{k-1}, or the details, that contain the characteristics.

We now shift our focus from the curve case to some two-dimensional (2D) data. Consider Figure 3.3, where the left column contains an image and subsequent coarse approximations. The middle column shows the image after the rows have been decomposed; the coarse approximation is on the left and row-details are on the right. The right column shows the image after the full decomposition; the coarse approximation is in the bottom left, the column-details are in the top left and the row details are on the right. In the image case it is even more obvious that the details do a good job of extracting the characteristics of the original image. Lena's hair is the area of the greatest concentration of high frequency characteristics and the details contain the high-frequency part of these characteristics. In addition, the outline of Lena, her hat and the objects in the room can be clearly seen in the details. This is to be expected, as the outlines of objects in an image are usually defined by a change in color. The attributes that we call the characteristics of are the changes in color.

Now let us return to Figures 3.4, 3.5, 3.6, and 3.7. In each figure, the leftmost image is the original, and the columns represent increasingly lower resolution levels proceeding from left to right. Similar to the curve case, it is easy to visually verify the images to determine that the details represent increasingly lower frequency characteristics as the terrain is decomposed. For each of the figures, the columns contain the lower resolution coarse approximation on the top row, and the details that were extracted during the decomposition on the bottom row.

In general, the characteristics (details) extracted through the RS process will be the same (vectors) for all models. However, the way we interpret the characteristics depends on the nature of the original model. Therefore, we will discuss the characteristics for the iris images separately from the terrain data.

A visual inspection of iris images will quickly reveal that most of their information consists of high frequency data. As a result, the first level of details extracted already contains distinctive characteristics from the iris image, Figure 3.4. Each subsequent level of details contains further characteristics up to the fourth level, Figure 3.5. Further decomposition adds little to the characteristics, and it is easy to visually confirm that the coarse representations at level four and beyond contain only the color information from the original iris.

In the case of terrain data, there is a more even distribution of high frequency characteristics and the low frequency characteristics that describe the overall terrain structure. Interestingly, the first level of details extracted during the first decomposition step are not visible as characteristics, Figure 3.6. In addition, the coarse representation after this step is very similar to the original data. For these reasons, we can say that the first level of details represents noise present from the sampling process. The second and third level of details are good representations of the characteristics of the terrain (Figures 3.6, 3.7, 3.8, and 3.9). Unlike iris images, terrain has a distinct underlying structure that becomes increasingly more noticeable in the characteristics when the decomposition process proceeds past three levels (Figures 3.7, 3.9).

In our experiments, we found that three to four levels of decomposition were sufficient to extract all characteristics from the data. The exact number is specific to the data that is to be decomposed and the spacing between its measurements. In the case of an iris image, four levels of decomposition were required to fully extract the characteristics of the image, while only three are necessary for the terrain data. In either case, once the decomposition proceeds past this level, the detail vectors begin to include some of the overall structure of the original model. This is to be expected, as the frequency of the details will become increasingly lower as the decomposition process proceeds.

3.2.1 Visualizing Details

The details for all digital models are vectors. As such, care should be taken to render the vectors in a manner that maximizes the intuition of their meaning. We have adopted an approach which is commonly used for visualizing details. First, the model is decomposed such that the necessary number of details have been extracted. Then, the coarse approximation is replaced with a known data model. Usually an average of the coarse approximations samples is found and the coarse representation is replaced with a model where every sample is this average value. Then, the model is reconstructed using the new coarse approximation and the details. Visualizing the details in this manner provides a good intuition for the details and their level of meaning.

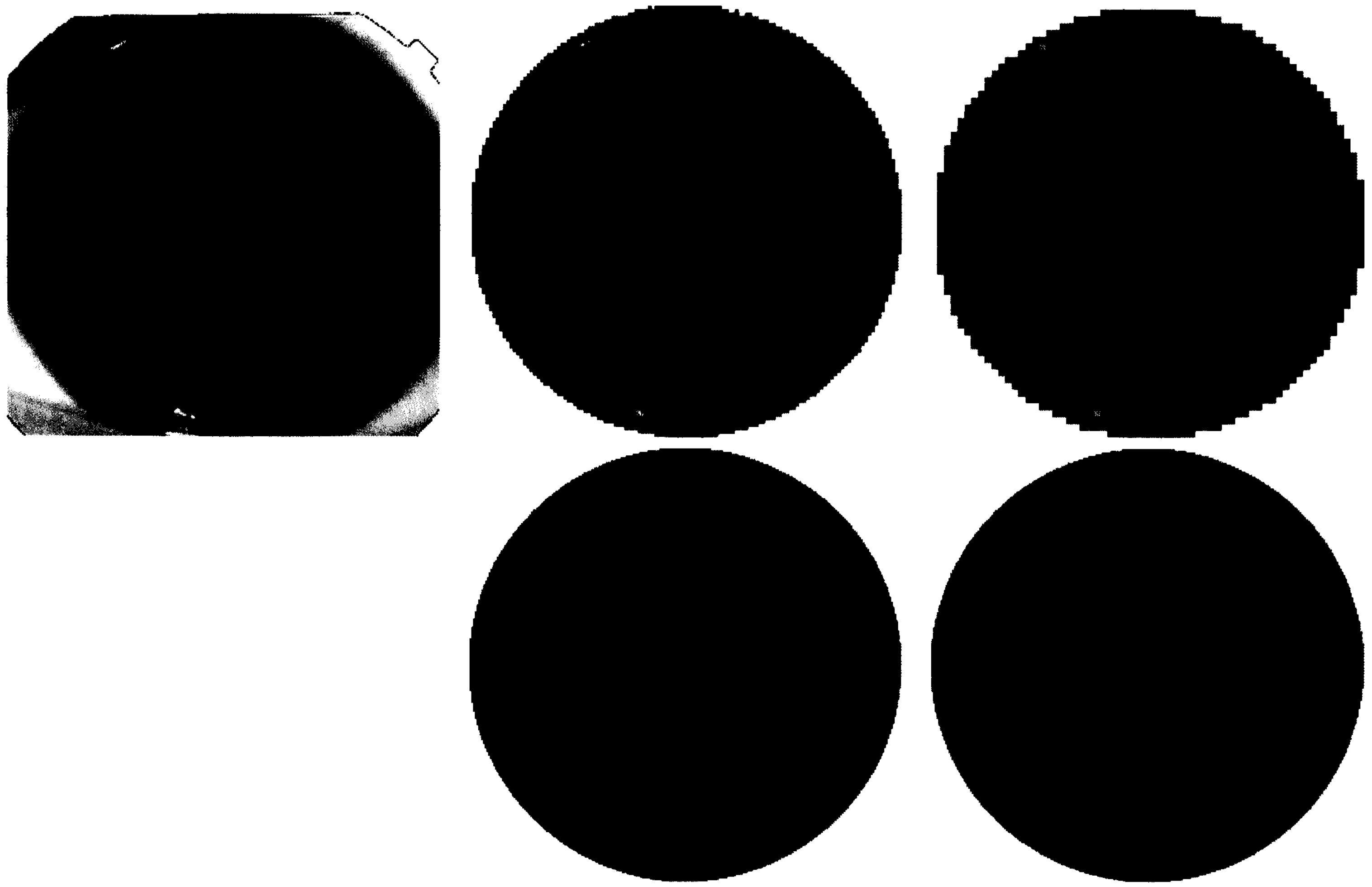


Figure 3.4: Chaikin decomposition of an iris image. The leftmost column is the iris image. The other columns represent increasing levels of decomposition with the coarse approximation in the top row and the details on the bottom row.

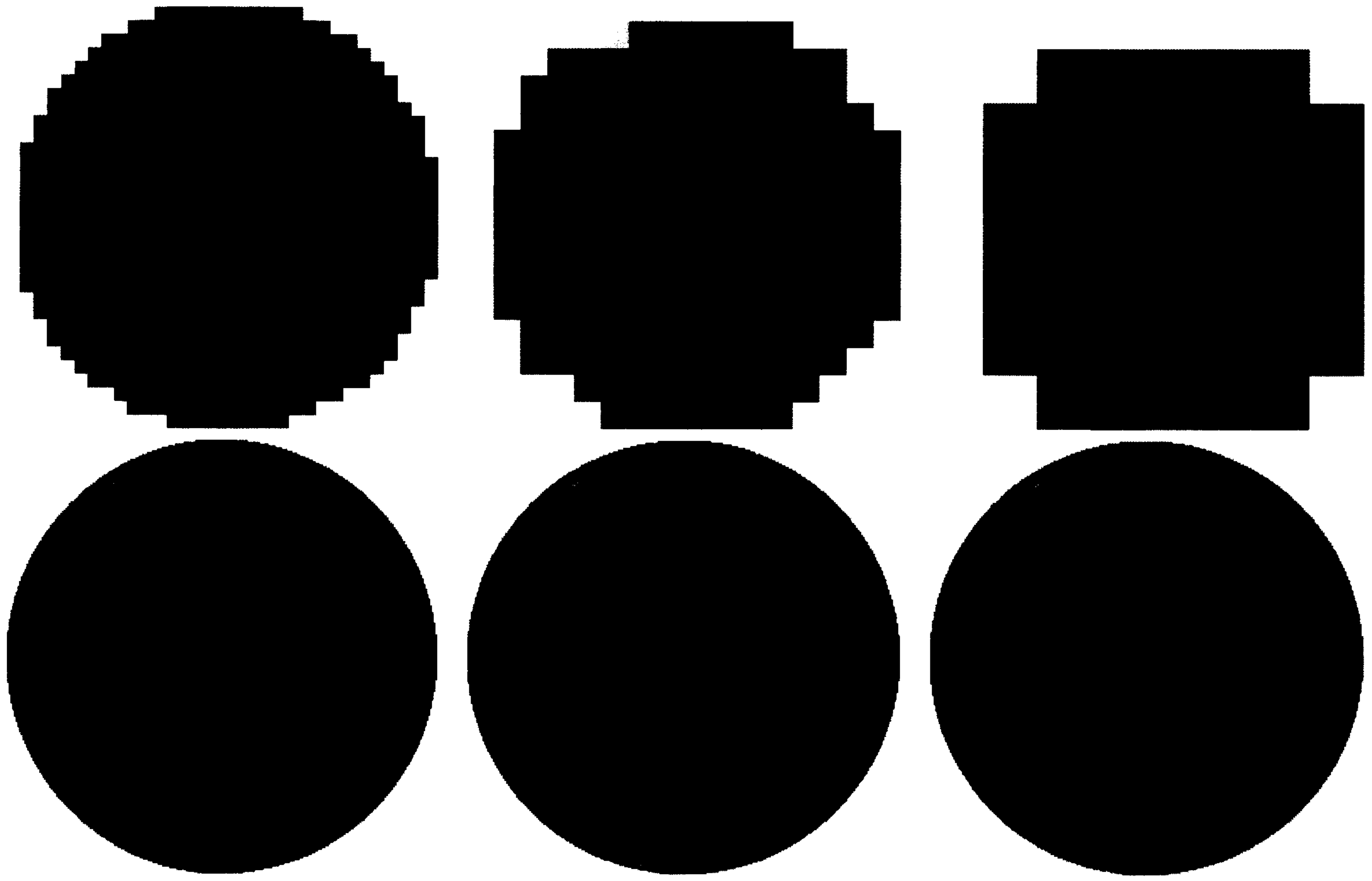


Figure 3.5: Continuation of Chaikin decomposition from Figure 3.4. Starting from the third coarse approximation, the columns represent the increasing levels of decomposition with the coarse approximation in the top row and the details on the bottom row.

Figure 3.6: Chaikin decomposition of terrain. The leftmost column is the original terrain. The other columns represent increasing levels of decomposition with the coarse approximation in the top row and the details on the bottom row.

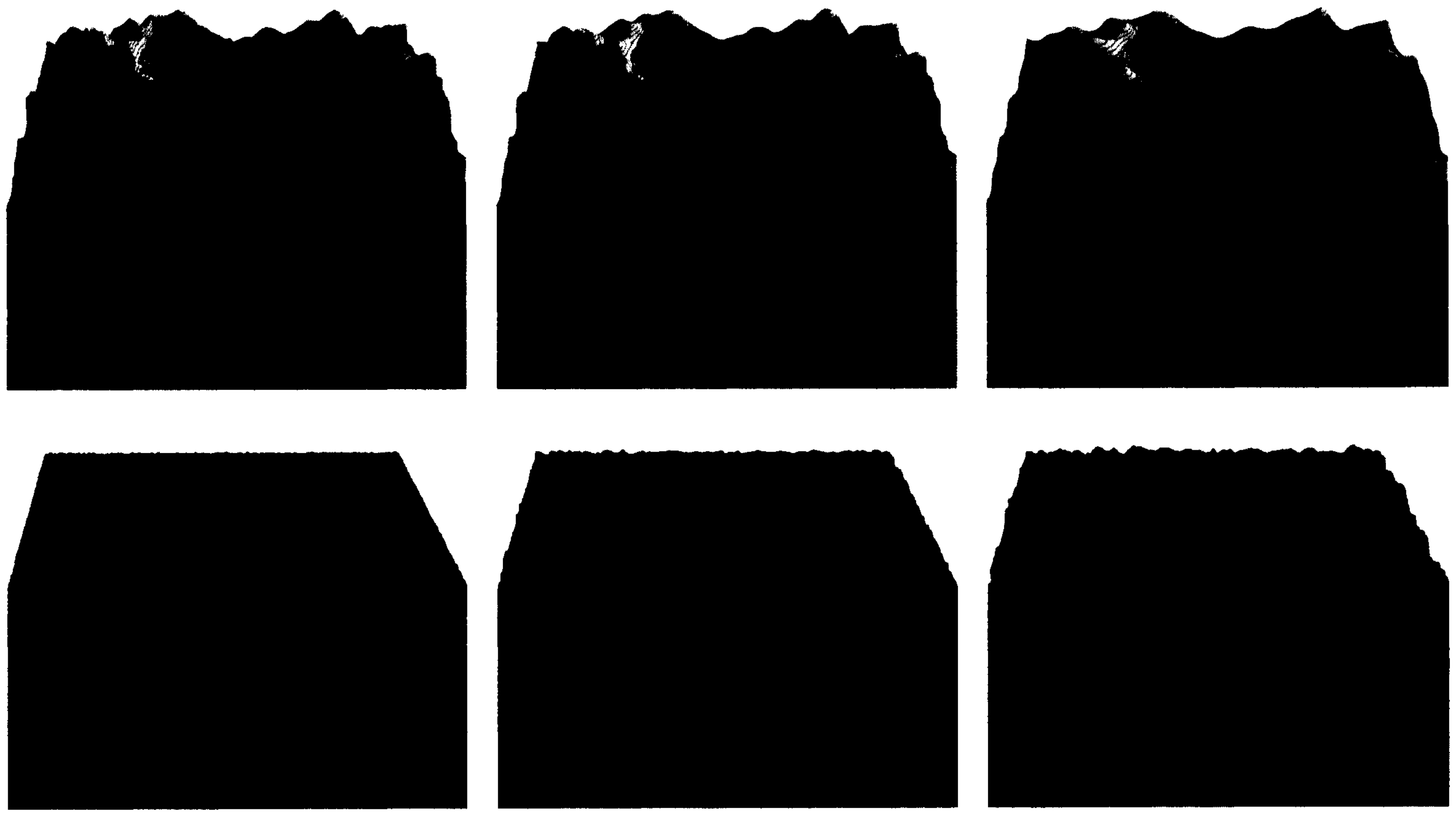


Figure 3.7: Continuation of Chaikin decomposition from Figure 3.6. Starting from the third coarse approximation, the columns represent increasing levels of decomposition with the coarse approximation in the top row and the details on the bottom row.

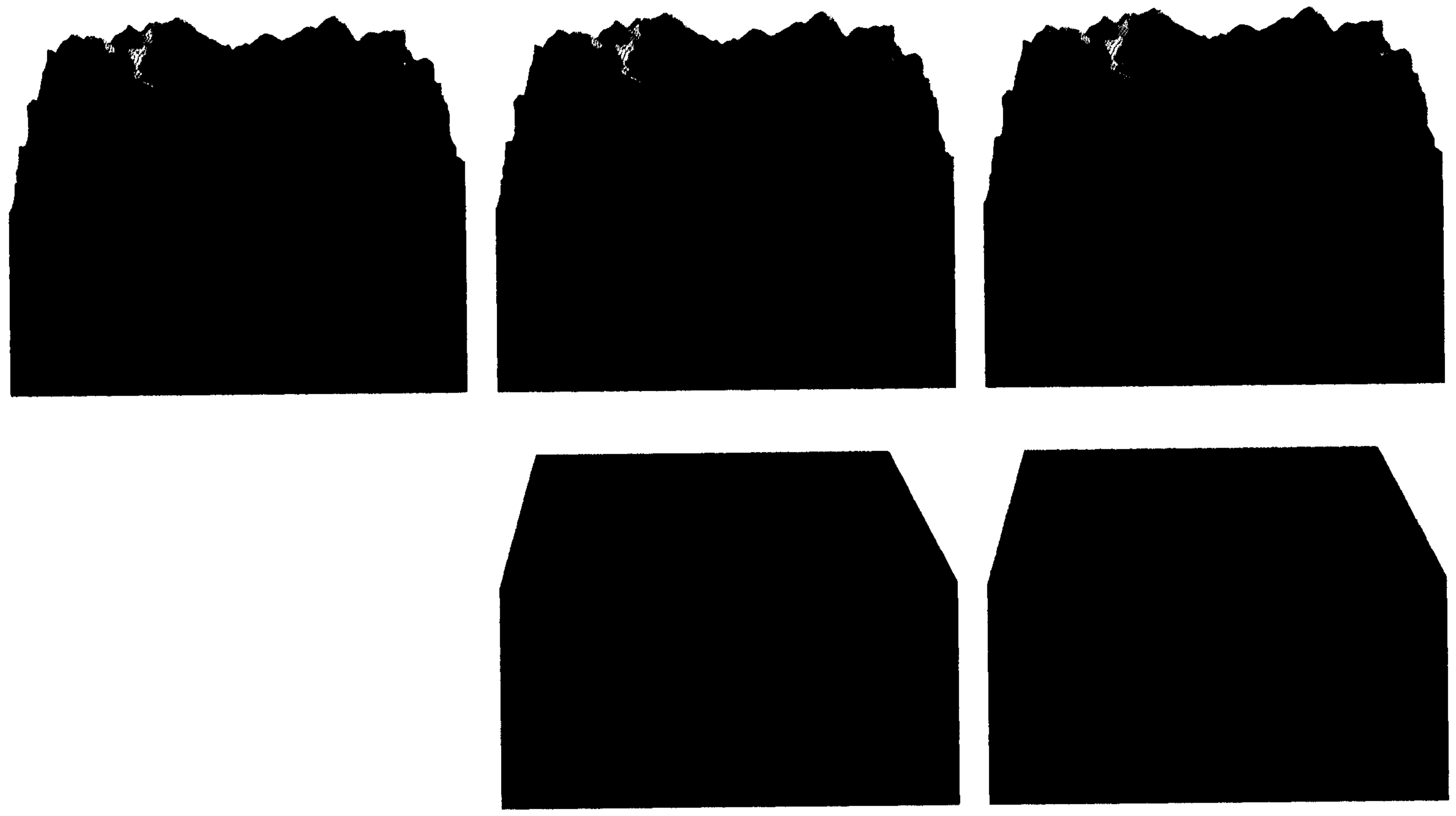


Figure 3.8: Haar decomposition of terrain. The leftmost column is the original terrain. The other columns represent increasing levels of decomposition with the coarse approximation in the top row and the details on the bottom row.

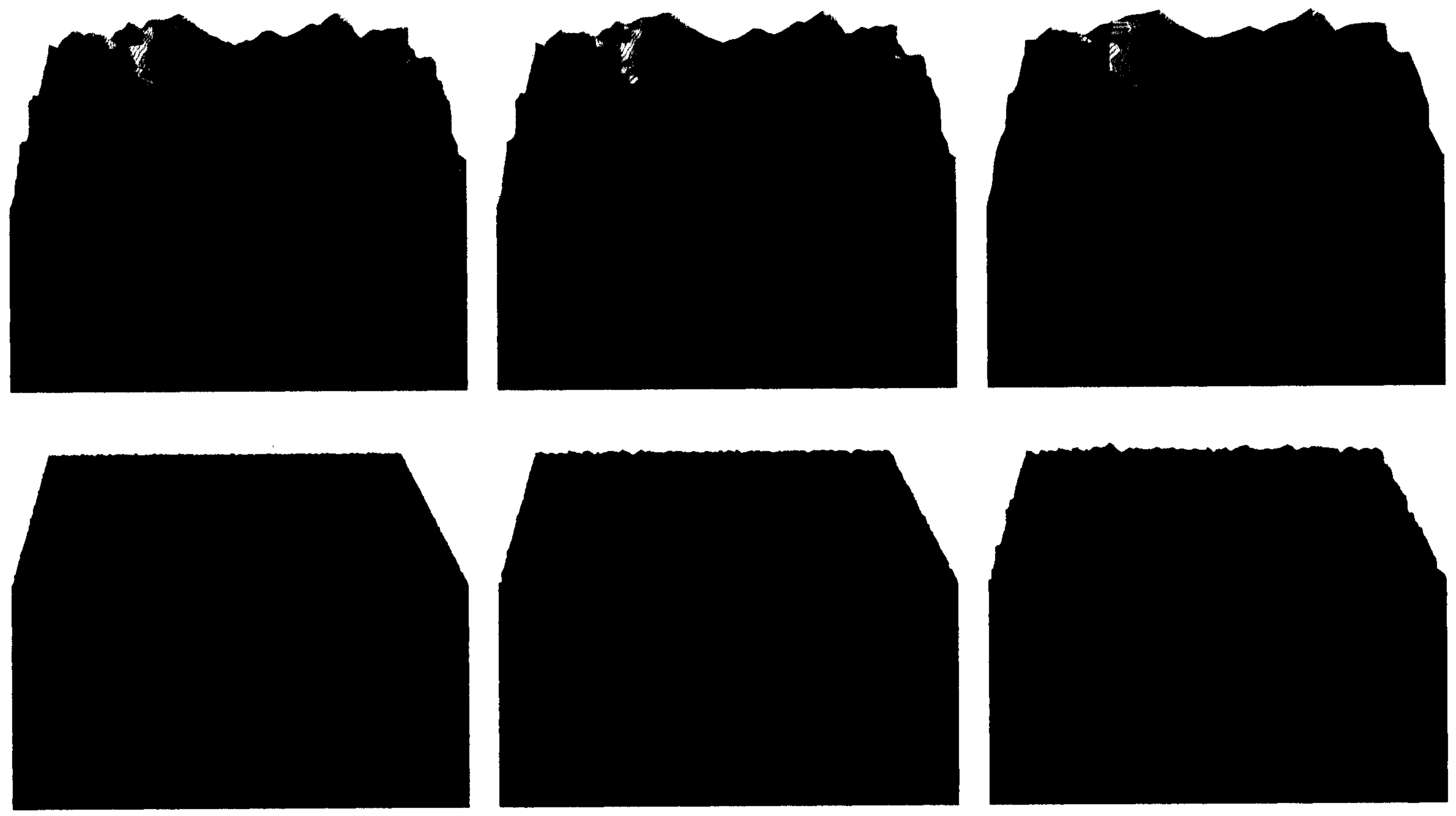


Figure 3.9: Continuation of Haar decomposition from Figure 3.6. Starting from the third coarse approximation, the columns represent increasing levels of decomposition with the coarse approximation in the top row and the details on the bottom row.

Chapter 4

Repairing Digital Elevation Models

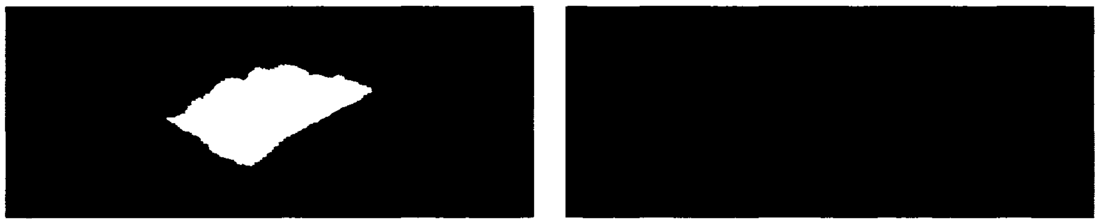


Figure 4.1: Terrain repair. The left illustrates the terrain with the samples missing. The right illustrates the results of our method.

In this chapter we present a technique for repairing digital elevation models. Digital elevation models (DEM) are used for terrain representation. DEM data is usually gathered through an automatic sampling of the elevation. The automated gathering process sometimes fails to obtain the elevation data in some areas. We refer to these incomplete areas as *voids.*

DEM is a commonly used digital model for terrain representation. They consist of an evenly spaced two dimensional grid of values where each value represents the elevation of the terrain at that location. One of the main reasons for their ubiquity is due to their simple structure which requires less storage than other model types.

This work provides a context sensitive method for repairing the voids found in DEMs. Specifically, we repair the DEMs provided by the Shuttle Radar Topography Mission (SRTM), flown in February 2000 [67]. The goal is to produce a patch with structure and characteristics similar to the surrounding terrain. An intuitive approach is to analyze the samples surrounding the *void* and find those that will provide the most useful information. To determine the usefulness of a real sample, we define *relevance* as: the closer a real sample is to the void the more *relevant* it is. *Relevance* is central to this work and will shape many of the algorithms discussed in this Chapter.

Our method consists of three steps. First, we must identify each void. Second, we use the overall shape and structure of the surrounding terrain to build a smooth patch for the void. Third, we extract the high frequency characteristics from the surrounding terrain and apply them to the patch.

In the rest of this Chapter we will cover the approach in more detail. To start, we introduce the data-set that is used in Section 4.1. Next, in Section 4.2, we make some observations about the data format and its impact on our methods. We then discuss the method in detail in Section 4.3. Finally, we provide a discussion of the results and draw some conclusions in Section 4.5. In addition to our method of repairing voids, we will make a brief discussion of the visualization methods that we employ for the DEMs in Section 4.4. Although, visualization is not the primary goal of this work, it is important as it will allow us to compare the results.

4.1 Introduction and Preliminaries

The SRTM DEMs are an evenly spaced 2D array of values, where each value is represents the height above sea level, in meters, of that location. They are canopy based with a vertical accuracy of 16m and a horizontal accuracy of 20m with 90% confidence [28, 67]. We use the SRTM data for the United States which is available at a 30m resolution. Unfortunately the gathering system for SRTM, Interferometric

Synthetic Aperture Radar (IFSAR), is not perfect: an estimated 1.2% of the samples are missing. The missing samples (voids) can occur due to geometric artifacts, specular reflection, and absorption [28]: The original data has been post-processed by a team from Boeing and Intermap to minimize the number of voids. Their approach involved removing spikes and wells [1], interpolating smaller voids[2], setting ocean samples to 0 meters, flattening lakes, stepping down the elevation along rivers and filling in data from other known datasets. The results of these efforts represent a final product called the SRTM Data Version 2. Version 2 is more complete than the initial raw data, but is still missing many samples. All non-water samples that are part of voids larger than 16 contiguous values have not been repaired. The second version of the data contains approximately 0.7% missing samples.

4.1.1 Related Work

Brosz used MR techniques in his work of *Synthesizing Terrain by Example* [10, 11]. For his system, a user should provide a terrain with the overall features and a terrain with the necessary characteristics. The system then creates an high resolution terrain that contains both the desired overall features and desired characteristics. Other mainstream techniques for synthesizing terrain are fractals, erosion simulations and multi fractal terrain generation. Brosz compared his system to each of these and he found that:

- fractals created homogeneous characteristics across the terrain which is not usually found in real terrain,

[1]Samples that exceeded a 100 meter difference when compared to the surrounding elevation.[28]
[2]Voids smaller than 16 contiguous samples were interpolated.[28]

- erosion simulations may not generate realistic terrain due to their empirical methods and they require significant computations,
- although multi-fractals are able to avoid homogeneity in the characteristics of the terrain, they cannot deal with high elevation valleys and plateaus.

In contrast, MR techniques were able to capture and maintain local features, including erosion features. Also, they are able to deal easily with terrain of any type including high elevation valleys and plateaus. In conclusion, MR provides a method for extracting, using and synthesizing terrain features in a simple, more stable, and more realistic manner.

4.2 Data Format and Initial Observations

The DEM structure is similar to a regular tensor product surface and allows us to define a simple parameterization of the grid. As discussed in Chapter 3, tensor product surfaces are an extension of parametric curves. Usually they are defined using the two dimensional uv parametric space. The two parameters for the space are denoted with u and v which correspond to the rows and columns of the data. Such a parameterization defines u curves as the set of points obtained when v is fixed and u varies (Figure 4.2 (a)). Similarly, v curves are the set of points obtained when u is fixed and v varies (Figure 4.2 (b)).

4.2.1 Alternative Directions

In general, there are many ways to parameterize a 2D heightmap. The approach described above is popular because the u curves are independent of the v curves.

For our purposes, independence is not necessary. The u and v parameters can be described as directions in which to traverse the terrain: they denote the rows and columns. Traversing the terrain using the two diagonal directions will give us two new ways to traverse the terrain (Figure 4.2(b), 4.2 (c)) Using these two new directions, we can define an alternative parameterization of the DEM. The new directions are denoted by the w and t parameters respectively (Figure 4.2(b), 4.2(c)). Rather, both w and t represent alternative directions in which the characteristics of the surrounding terrain can be captured.

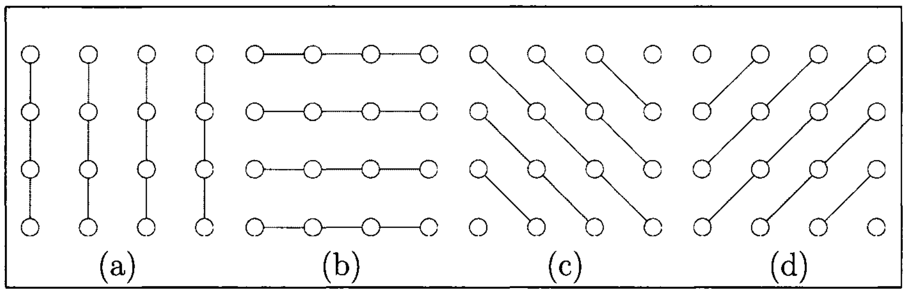


Figure 4.2: Illustration of the (a) u curve set for a 4x4 heightmap. (b) and the v curve set for a 4x4 heightmap. (c) and the w curve set for a 4x4 heightmap. (d) and the t curve set for a 4x4 heightmap.

4.2.2 Observations and General Approach

We approach this problem using the well known divide and conquer method. The first partitioning of the problem are the curve sets for the four parameter directions, each of which can be considered apart from the others. Additionally, each curve set can be further divided into each of the individual curves that compose the set. Considering each curve individually allows us to operate in a 1D parametric set which greatly simplifies the approach. Therefore, as an initial exploration of the problem, we will provide repair methods that iteratively solve individual curves.

The set of solutions for a curve set represent an intermediate, but complete, solution for that set. After each of the intermediate solutions for the curve sets are built, they are combined into a final solution. The combination uses a heuristic that prefers samples from solutions that are the most *relevant* to the missing sample. Experimental results show that finding characteristics separately for each curve in a void does not produce the best results. Consequently, we extend the method to find, extract and apply the characteristics to the entire void.

4.2.3 Data Structures

Due to the grid-like nature of the data, it is intuitive to store the terrain data in a 2D array of height values, called a *heightmap*. Each sample in the heightmap is denoted by: $\mu_{u,v}$. Given a particular curve j, in the u, v, w, or t direction, each sample i along the curve is denoted using: $\mu_i^{u,j}$, $\mu_i^{v,j}$, $\mu_i^{w,j}$, and $\mu_i^{t,j}$ respectively. Many algorithms in this work operate on a single curve, and require access to all samples along that curve. Curve data structures were built that support an iteration concept by providing access to any element of the curve and to discover its length. In addition to data structures that represent an entire curve, there are algorithms that need to operate only on a specific area of the curve. Sub-curve data structures were built to represent these partial curves and provide the same code-interface, allowing the two data structures to be interchangeable. All curve data structures implemented using well known indexing arithmetic, which allows them to be efficient both in terms of running time and memory usage.

4.3 Methodology

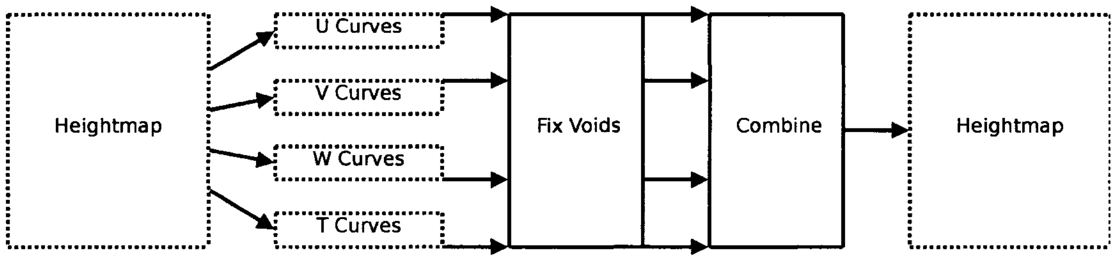


Figure 4.3: The method's high-level pipeline.

In this section, we will cover the approach that was developed for repairing the voids found in the SRTM data set. First, we will provide a high level summary of the pipeline (Figure 4.3). Then we will provide the necessary details of the algorithms that were used. In the case of the characteristics algorithms, we will describe two options that were explored. In the first approach, the characteristics are found individually for each curve in the void. For the final approach, the characteristics are found at once in a single step.

To start, each of the voids in the area are identified. For each void, a smooth patch is built using the overall directions of the surrounding area. Then, the characteristics from the surrounding area are extracted and applied to the smooth patch. In some cases, a number of intermediate solutions are built in this manner. For these cases a heuristic is provided that combines the solutions such that the most relevant results are preferred.

4.3.1 Building The Patch: Curve Case

To provide a solution for an individual curve, the following steps are taken:

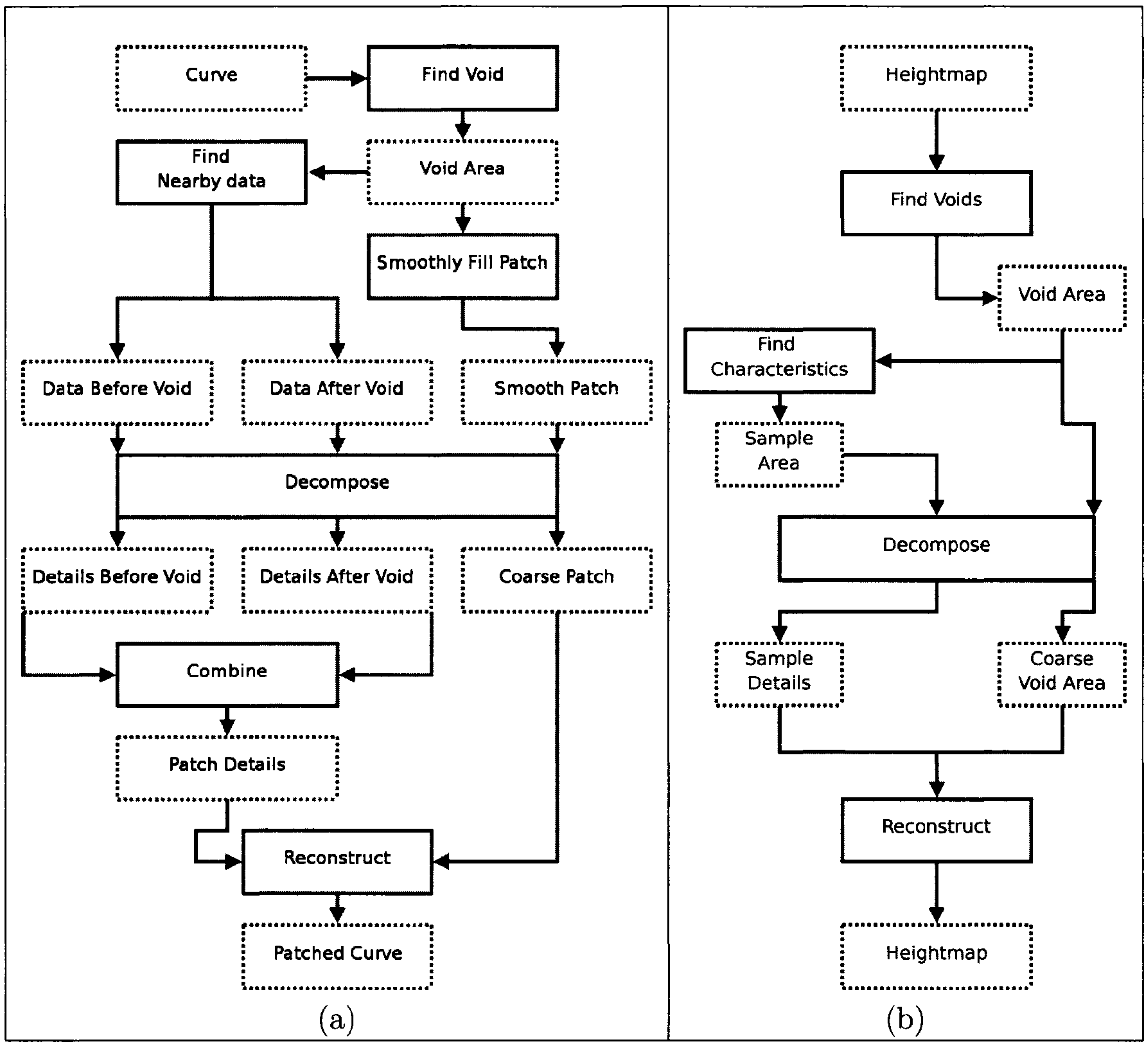


Figure 4.4: In these figures, the dotted lines indicate data structures, and the solid lines indicate algorithms that operate on those data structures. (a) The overall structure of the patching algorithm that operates on individual curves, i.e. the 1D case. (b) The overall structure of the patching algorithm that operates on the heightmap, i.e. the 2D case.

1. The boundary of each of the voids is identified (Section 4.3.2),
2. Using this boundary as a guide, the area surrounding each void is analyzed for its overall direction and a general patch is built to cover the void (Section 4.3.3),
3. Characteristics of the area surrounding each void are then extracted and applied to the general patch (Section 4.3.3).

4.3.2 Finding the Voids: Curve Case

The SRTM data format explicitly marks each missing sample which makes finding voids along a curve straightforward. A search is performed from the beginning of each curve until it finds the first missing sample μ_σ (Figure 4.5). The end of the void can be found by continuing the search from: $\mu_{\sigma+1}$ until the next non-missing sample $\mu_{\epsilon+1}$ is found. The last missing sample μ_ϵ is marked and the search for the next void continues along the curve from $\mu_{\epsilon+2}$. It is easy to see that the search visits each sample on a curve only once and results in a linear run time method.

4.3.3 Creating a Smooth Patch: Curve Case

In this step, we will create a smooth patch using the general direction of the surrounding terrain. Hermite curves are used to create the patch because they have the same level of continuity G^1 as the Chaikin RS (quadratic B-splines) [60, 4] that are used for the characteristic extraction. We will generate a single Hermite curve to fill the void (see Section 2.5.1 for the Hermite calculation). The edges of the surrounding terrain $\mu_{\sigma-1}$ and $\mu_{\epsilon+1}$ are used for the starting and ending points in the Hermite

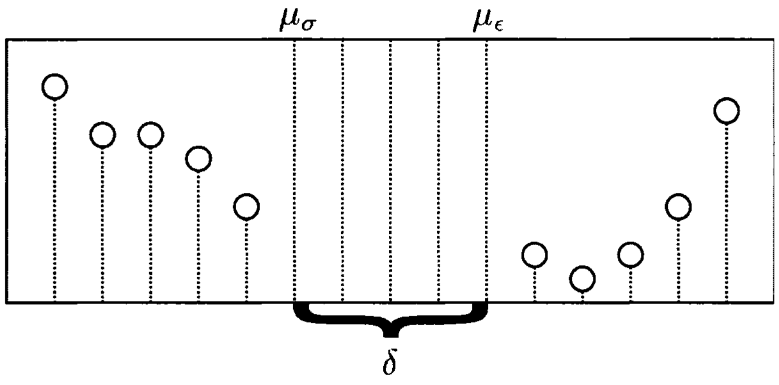


Figure 4.5: Illustration demonstrating the boundaries of a void, $\mu_{\sigma+1}$, μ_ϵ and the length of the void: δ.

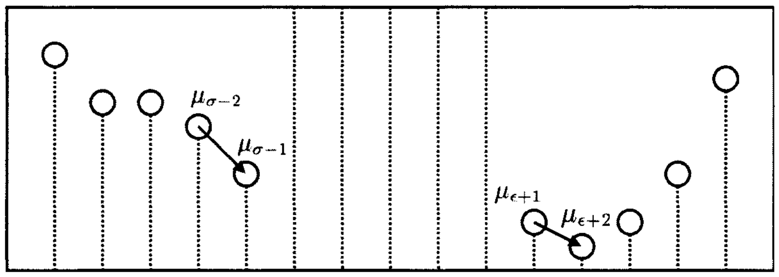


Figure 4.6: Illustration of the local terrain direction, Equation 4.1.

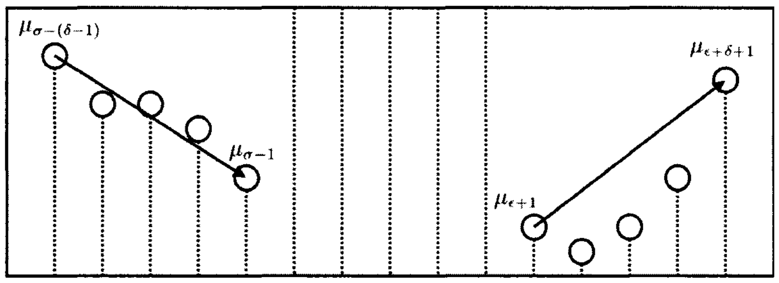


Figure 4.7: Illustration of the terrain direction looking in an area the same size as the void, Equation 4.4.

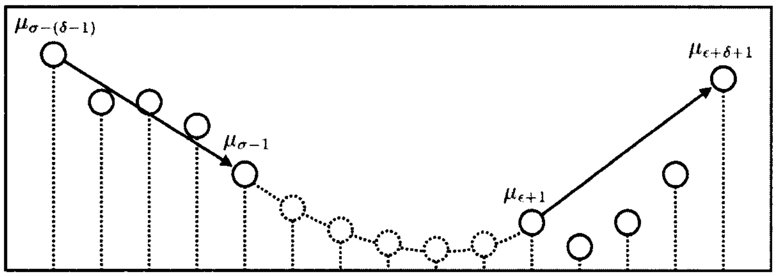


Figure 4.8: Illustration of providing a smooth patch using a Hermite curve.

calculation. The last items needed for the Hermite calculation are the tangents at either side of the terrain (Figure 4.8).

Capturing the Terrain Structure: Curve Case

To produce a patch that closely resembles the surrounding terrain, the direction of the terrain near to the void should be considered. In the curve case, the data on either side of the void will give us the best representation of the direction. An initial idea is to find the tangent of the terrain using the two samples closest to the edge of the void (Figure 4.6).

$$\tau_\sigma = \mu_{\sigma-1} - \mu_{\sigma-2}, \tag{4.1}$$

$$\tau_\epsilon = \mu_{\epsilon+2} - \mu_{\epsilon+1}. \tag{4.2}$$

This choice provides us with the local direction of the terrain near to the edge of the void. In many cases this local direction is too sensitive to small changes in the terrain (Figure 4.9). We would rather have the general direction of the terrain at the edge of the void. A possible approach is to use RS to reduce the resolution of the surrounding terrain and use its coarse representation. Another approach is to calculate the tangent using a larger area of surrounding terrain. Either approach will provide a more accurate indication of the general direction of the terrain and

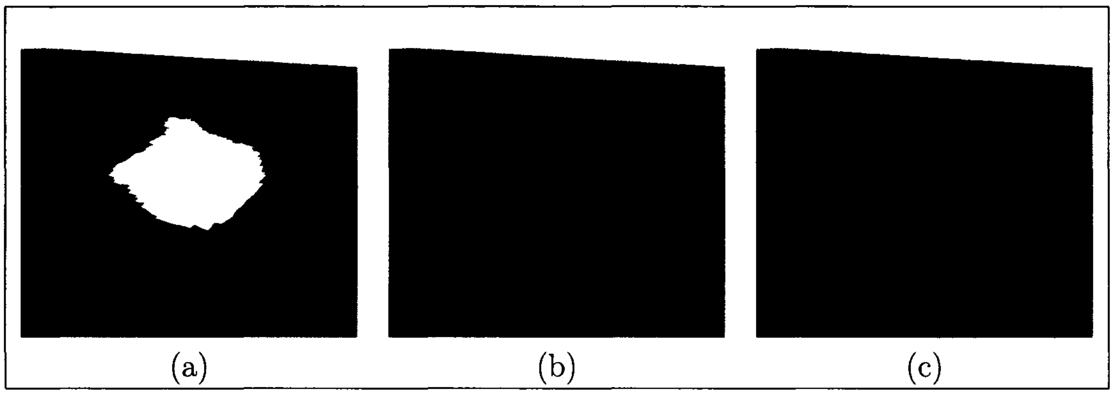


Figure 4.9: (a) An example void. (b) Smooth patch for the u curves using Equation 4.1 for calculating the nearby terrain direction. (c) Smooth patch for the u curves using Equation 4.4 for calculating the nearby terrain direction.

will less sensitive to small changes (Figure 4.7). In our implementation we use an area that is the same size as the void (Figure 4.7). This provides us with a better indication of the change in terrain height for areas which are a similar size to the void. Let δ be:

$$\delta = (\mu_\epsilon - \mu_\sigma) + 1. \tag{4.3}$$

The tangents on either side of the void can then be found using:

$$\tau_\sigma = \mu_{\sigma-1} - \mu_{\sigma-(\delta-1)}, \tag{4.4}$$

$$\tau_\epsilon = \mu_{\epsilon+\delta+1} - \mu_{\epsilon+1}. \tag{4.5}$$

4.3.4 Capturing and Applying Characteristics: Curve Case

The final step in the patching process involves finding and applying characteristics to the void. There are two interesting steps of this method and we shall cover each in turn. First we discuss how and where the characteristics are found. Second, we discuss how to apply the characteristics to the patch.

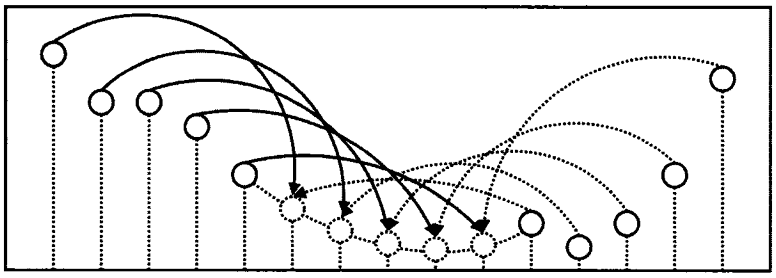


Figure 4.10: The source and destination of the extracted characteristics for the simple interpolation method.

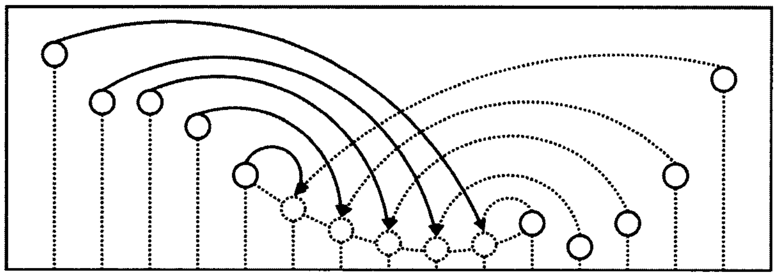


Figure 4.11: The source and destination of the extracted characteristics for the mirrored interpolation method.

A sub-curve from either side of the void is chosen to provide relevant characteristics. These two sub-curves have the same size as the void and are denoted at resolution level k using: μ_σ^k and μ_ϵ^k. Using the same iterative process as in Chapter 3, the resolution of the two curves is reduced:

$$\mu_\sigma^{k-1} = A^k \mu_\sigma^k, \tag{4.6}$$

$$\mu_\epsilon^{k-1} = A^k \mu_\epsilon^k. \tag{4.7}$$

During the process, we capture the detail vectors from the curves:

$$^{\sigma}D^{k-1} = B^k \mu_\sigma^k, \tag{4.8}$$

$$^{\epsilon}D^{k-1} = B^k \mu_\epsilon^k. \tag{4.9}$$

The number of levels of resolution used to extract the characteristics depends both to the application and the size of the void. Experimental results showed that no more than four levels were needed for most voids encountered in the SRTM dataset, and usually only three are needed. Using the notation from Chapter 3, the extracted characteristics at resolution level i are denoted with:

$$\begin{aligned} {}^{\sigma}D^i &= \left[{}^{\sigma}d^i_1, {}^{\sigma}d^i_2, \ldots, {}^{\sigma}d^i_m\right], & (4.10) \\ {}^{\epsilon}D^i &= \left[{}^{\epsilon}d^i_1, {}^{\epsilon}d^i_2, \ldots, {}^{\epsilon}d^i_m\right]. & (4.11) \end{aligned}$$

We now have two sets of characteristics that are near to the void. Before applying the characteristics to the patch, the two sets must be combined in a way that maximizes their relevance to the void. A simple solution is to use linear interpolation to combine the two sets of characteristics at resolution i as follows:

$$D^i = \begin{bmatrix} \left(\frac{m}{m+1}\right) {}^{\sigma}d^i_1 + \left(\frac{1}{m+1}\right) {}^{\epsilon}d^i_1 \\ \left(\frac{m-1}{m+1}\right) {}^{\sigma}d^i_2 + \left(\frac{2}{m+1}\right) {}^{\epsilon}d^i_2 \\ \vdots \\ \left(\frac{1}{m+1}\right) {}^{\sigma}d^i_m + \left(\frac{m}{m+1}\right) {}^{\epsilon}d^i_m \end{bmatrix}^T . \qquad (4.12)$$

Each row in D^i has the form $(\alpha){}^{\sigma}d^i_j + (1-\alpha){}^{\epsilon}d^i_j$ where the value of α is close to 1 when $j = 1$ and close to 0 when $j = m$. This solution ensures that most relevance is given to the the characteristics extracted from the curve closest to the respective void edges (Figure 4.10).

In order to ensure a smooth transition between the characteristics on the curve near to the edge of the void and the characteristics on the edge of the patch, a mirroring technique is also employed when combining the characteristics. The technique results in the following characteristic combination at resolution i:

$$D^i = \begin{bmatrix} \left(\frac{m}{m+1}\right) {}^{\sigma}d^i_m + \left(\frac{1}{m+1}\right) {}^{\epsilon}d^i_m \\ \left(\frac{m-1}{m+1}\right) {}^{\sigma}d^i_{m-1} + \left(\frac{2}{m+1}\right) {}^{\epsilon}d^i_{m-1} \\ \vdots \\ \left(\frac{1}{m+1}\right) {}^{\sigma}d^i_1 + \left(\frac{m}{m+1}\right) {}^{\epsilon}d^i_1 \end{bmatrix}^T . \tag{4.13}$$

This final set of characteristics is used to replace all detail vectors in the smooth patch (Figure 4.11). Essentially, this involves reducing the resolution of the smooth patch, and replacing each level of characteristics with those from Equation 4.13.

In some research a local frame relative to the tangent and normal of the curve is found when extracting detail vectors. Using this local frame, the researchers can re-orient the details as needed [32, 12]. In our case, the DEM has a fixed horizontal spacing between all of the samples. Using a local frame could result in samples that have a smaller horizontal spacing and the sample may overlap; a situation not supported by DEM.

4.3.5 Patching the Void Using Several Directions

Our experimental results, see Figure 4.12, show that applying the characteristics on an individual curve basis is not sufficient. The resulting patches tend to be more smooth than the surrounding terrain. In addition, there is a visible regularity that is introduced with the characteristics. A simple explanation is that the detail vectors found on an individual curve basis may not be compatible. That is, we found that blending the detail vectors from different areas can result in a smoothing of the high frequency information that is present. For these reasons, we investigated extracting characteristics for the entire void from the same 2D area of the terrain, rather than

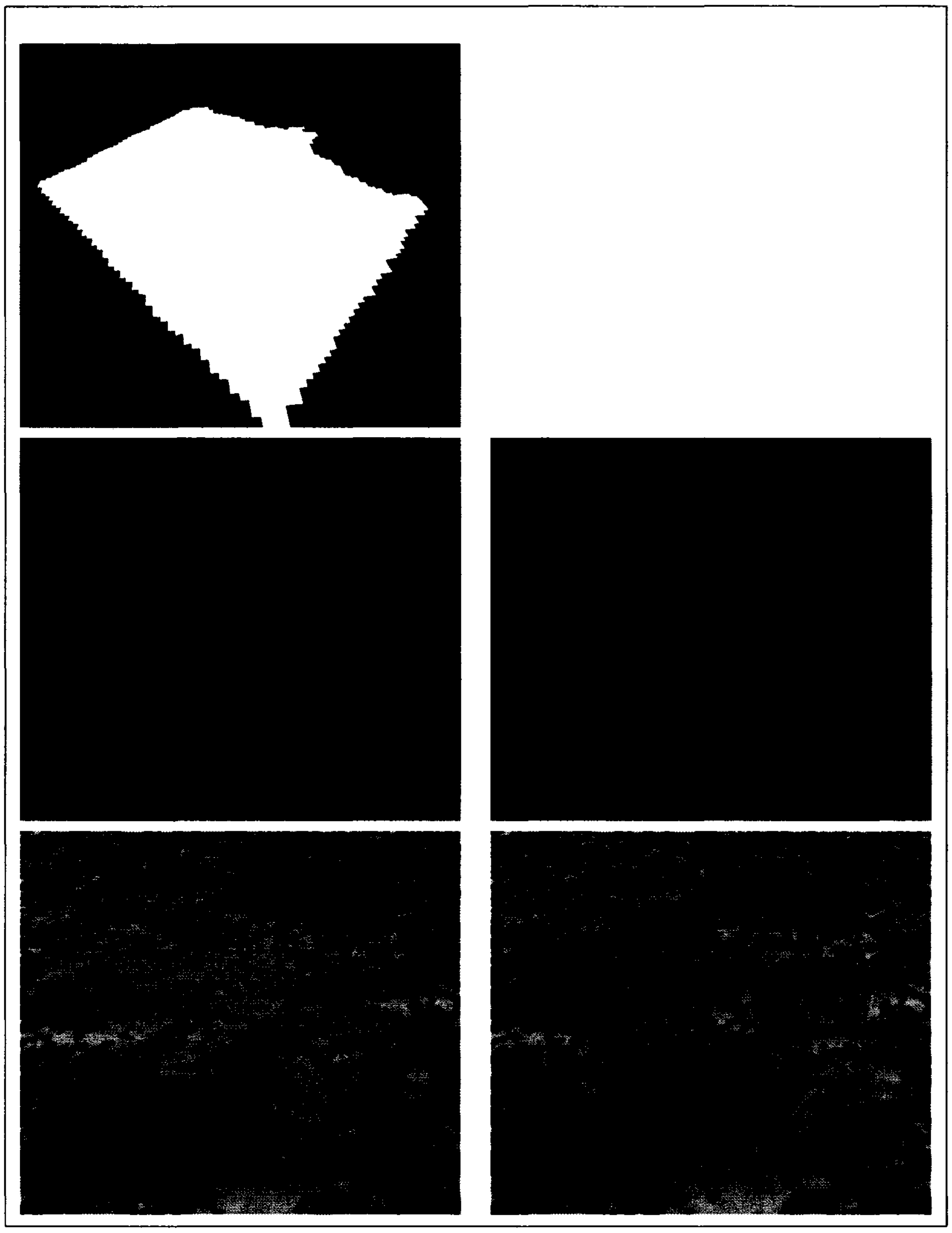


Figure 4.12: The left column illustrates the results obtained using the 1D (per-curve) method for characteristics extraction. The right column illustrates the results obtained using the 2D (patch-based) method for characteristics extraction. Notice how in both the Phong shaded rendering (2nd row) and the characteristics rendering (3rd row) the 2D approach generates better results.

from individual curves. Note that, for this section, we will assume that a smooth patch for the void has already been generated using Hermite curves.

The high-level approach is very similar to the curve case. First, we use a graph traversal algorithm to identify each of the voids. Next, we search the area surrounding the void to find the area that is the best representation of the terrain. Finally, we extract and apply the details using the approach described in Chapter 3 for image based RS. The following sections contain the necessary descriptions of the steps for this case.

4.3.6 Finding the Voids

A heightmap can be represented as a graph, where each sample is a vertex on the graph and is directly connected to the samples above, below, to the left and to the right of itself (Figure 4.13(a)). Using this representation, we can perform a breadth-first graph traversal to identify the missing samples of each void as a connected component (Figure 4.13(b)). In order to find the first missing sample for each void we iterate over the samples in the heightmap. This sample is given to the graph traversal algorithm which uses it as a starting point.

During the graph traversal an axis aligned bounding box is created such that it marks the boundaries for each void (Figure 4.13(c)). It is important to note that the bounding box will not necessarily have dimensions suitable for the RS approach (Section 3.1.2). We calculate a target width and height that are suitable for RS using the same approach as in Section 3.1.2. Then, we make a new bounding box that has the same center as the original box but with a slightly larger width and height. In the following sections we use the new bounding box for both the search and the

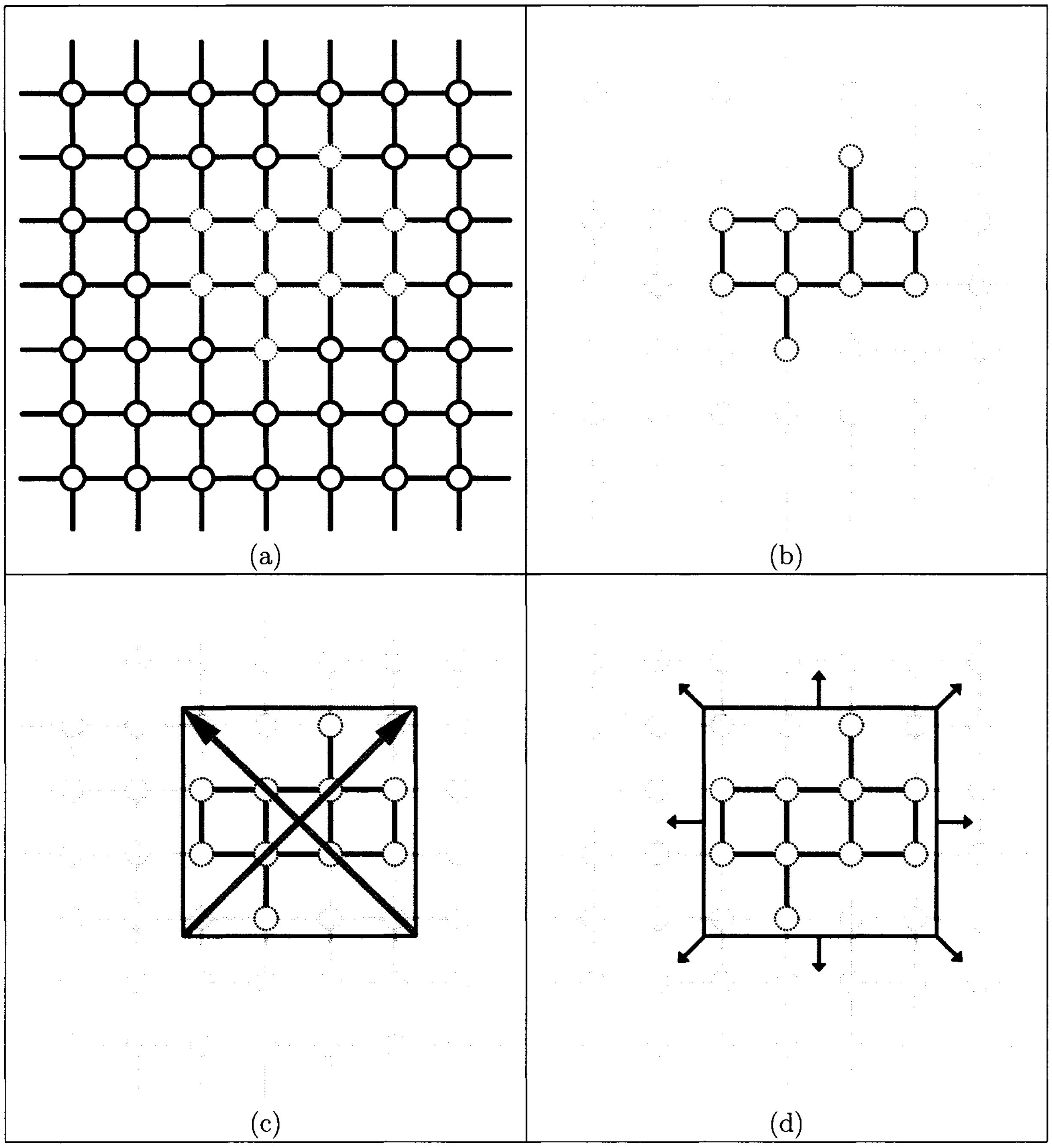


Figure 4.13: (a) Heightmap as a connected graph, the dotted circles represent missing samples. (b) A void as a connected component of the graph and (c) its axis aligned bounding box and the vectors used to calculate the normals of the area. (d) The eight possible search directions due to the u, v, w, and t parameter directions.

characteristics extraction.

4.3.7 Finding the Characteristics

After the boundary for a void has been determined, we find a nearby area that contains the most relevant characteristics for the void. In the 2D case, finding the most relevant information is more challenging than the 1D case as there are more than two directions in which to search. If we consider each of the parameter directions that we use for the curve case, we have a eight possible directions in which to search (Figure 4.13(d)). Each of these areas represents data that is equally relevant to the void, so we must make a choice of which one to use.

To determine which of the candidate areas is most suitable for the void, we use two criteria. The first is that any of the search areas may contain voids making it less suitable. So we count the number of missing samples in that area and use the area with the least missing samples. In some cases, there will be multiple candidate areas that have the minimum number of missing samples. In this situation we prefer to use an area of terrain that has a similar type to the terrain containing the void. For example, if the void is on the side of a hill we would like to extract characteristics from the side of a nearby hill. We use a simple geometric comparison to find the best match from the candidate areas. First, we calculate an approximate normal vector for the area that contains the void. Then we also calculate an approximate normal vector for each of the candidate areas. An approximation of the normal can be calculated using the cross product the two vectors that run diagonally across the bounding box (Figure 4.13(c)). These normal vectors provide us with the general orientation for each candidate area. We then compare the normals of the candidate areas with the normal for area surrounding the void. The candidate area that has the minimum angle between its normal and that of the void is chosen.

4.3.8 Applying the Characteristics

At this point, we know the bounding of the void and we have an area chosen which will provide the characteristics for the patch. The coarse approximation is then obtained from the void's smooth patch using equation 3.2. Then, we extract three levels of details from the target area using equation 3.3. To produce a final patch we use equation 3.4 to combine the characteristics and coarse representation.

The final patch is a rectangular area that fully encompasses the void. However, we should be careful to only use the samples from the patch that are missing in the original void. In addition, there are some cases where the normals at the edges of the void do not match up with the normals at the edges of the patch. The mismatching normals result in a visible discontinuity along the edge of the patched terrain (Figure 4.14(b)). To minimize these effects, we use a feathering technique for samples on the edge of the void: for each newly generated sample, we first check to see if there is a real sample within a small neighborhood of the point. If there is, an average of the new sample and the surrounding real samples is used (see Figure 4.14(c) for the effect of this feathering).

4.3.9 Combining Multiple Solutions and Relevance

Recall from Section 4.2.2 that we build four intermediate solutions: one for each of the curve directions u, v, w and t. Therefore, we have four possible solutions for each missing sample: one from each curve direction. In the final patch, we can

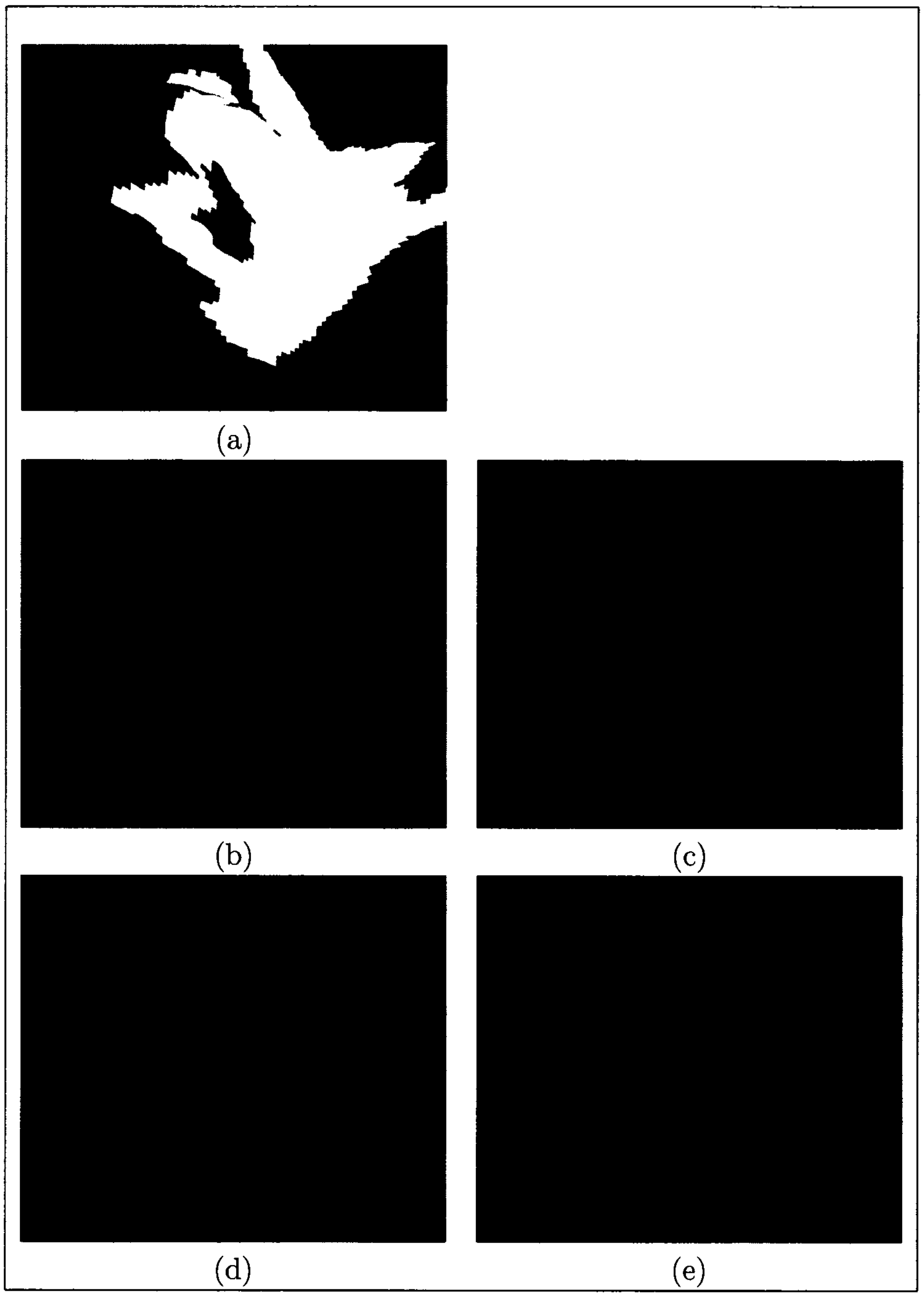


Figure 4.14: (a) An example void. (b) Results after repairing the void without feathering of the edges. (c) Results after repairing the void with feathering of the edges. (d) Same result from (b), but with the patch boundary outlined. (e) Same result from (c), but with the patch boundary outlined.

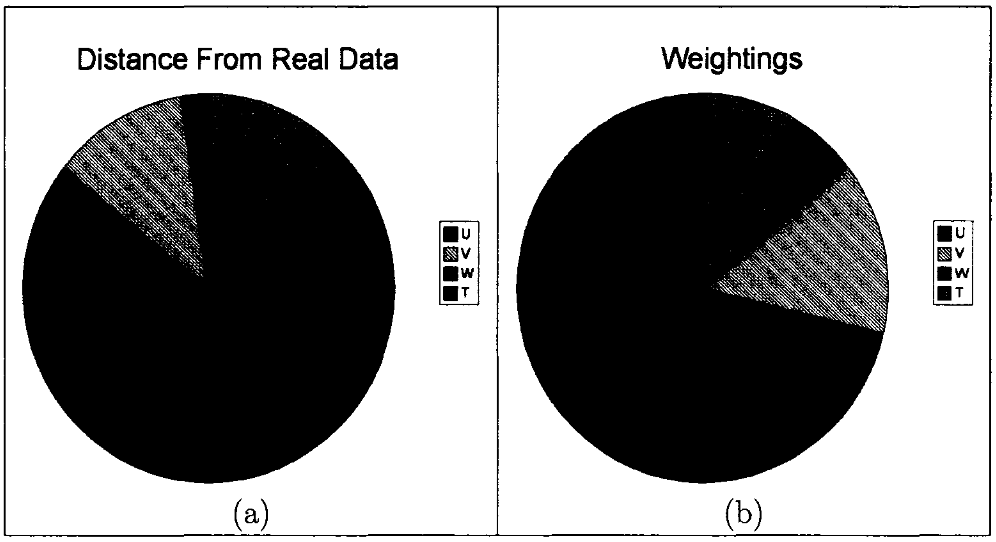


Figure 4.15: (a) Pie chart showing distances from real data for a the intermediate solutions for u, v, w, and t curve sets. (b) Pie chart showing the inverse weightings that are associated with the intermediate solutions from u, v, w, and t. Note how the weighting heavily weights the sample that is closest to real data.

only place one value into the missing sample and we should provide a value that best represents the surrounding terrain. Therefore we combine each of our solutions using a weighted averaging that prefers the more relevant solutions:

$$\mu_{final} = \sum_{i=1}^{n} \alpha_i \mu_i \tag{4.14}$$

where μ_i represents the solution from curve direction i, and α_i is the weighting for that solution. We only use four terrain directions in this work, however, equation 4.14 is general and can be applied for n directions. In addition, the combination should be Affine:

$$\sum_{i=1}^{n} \alpha_i = 1. \tag{4.15}$$

In order to provide the best solution for each missing sample, we should prefer the more *relevant* solution. For each missing sample, assume that we are given a set of intermediate solutions for each curve direction, $[\mu_1, \mu_2, \ldots, \mu_n]$. In addition we can find the distance between each missing sample and its closest real sample along that curve direction: $[d_1, d_2, \ldots, d_n]$. Then, we define the total distance as $\delta = \sum_{i=1}^{n} d_i$. Using the original distances and the total distance, we can define a set of inverse ratios, $[r_1, r_2, \ldots, r_n]$ as

$$r_i = \frac{\delta}{d_i}, \tag{4.16}$$

and a new total as:

$$\Delta = \sum_{i=1}^{n} r_i. \tag{4.17}$$

From this, we can easily normalize these ratios thereby defining weights for each of the solution values as:

$$\alpha_i = \frac{r_i}{\Delta}. \tag{4.18}$$

As a demonstration of the weighting, consider the pie charts in Figure 4.15. The chart on the left, shows the distances between a missing sample and its closest real sample for each curve direction: the largest portion represents the sample that is the least relevant. Conversely, the chart on the left shows the weightings that we have calculated: the largest portion represents the sample that is the most relevant.

4.4 Visualization of the Terrain

In this section we discuss the methods that we use to visualize DEMs. Visualization may not be the primary goal of this work, but it is important as it allows us to compare our results.

First, we discuss a two dimensional approach for introductory investigation of the voids (Figure 4.16 (a)). This method allows us to view the distribution of the voids, on the other hand, it can not visualize the 3D nature of the terrains. Consequently, we have also used the following three dimensional rendering methods:

- a wire frame approach (Figure 4.16(b)),
- a Phong shaded triangle approach (Figure 4.16(d)),
- a contour map approach ((Figure 4.16(e)).

The main concentration of this work is on using RS details for synthesis. None of the above rendering techniques facilitate the viewing of the details of the terrain. Therefore we also developed a new technique for rendering terrain that emphasizes the characteristics (Figure 4.16(f)).

4.4.1 2D Rendering

Rendering terrain to an image is a simple and straightforward process. Recall that the SRTM data is provided as a 2D array of height values, otherwise known as a heightmap (Section 4.1). The image width and height are the same as the width and height of the heightmap. In addition, the location of each sample in the heightmap is the same location for the image. Each height value is converted to a gray-scale intensity. White represents the highest possible height within the dataset, and dark gray is the lowest possible. Dark gray is chosen so that the voids, which are rendered in black, are distinguished from low elevations. This method provides an excellent overview of the terrain and the voids that are contained therein (Figure 4.17). However, it does not provide a perception of depth which is necessary for viewing 3D models and we do not use it to compare the results of our method.

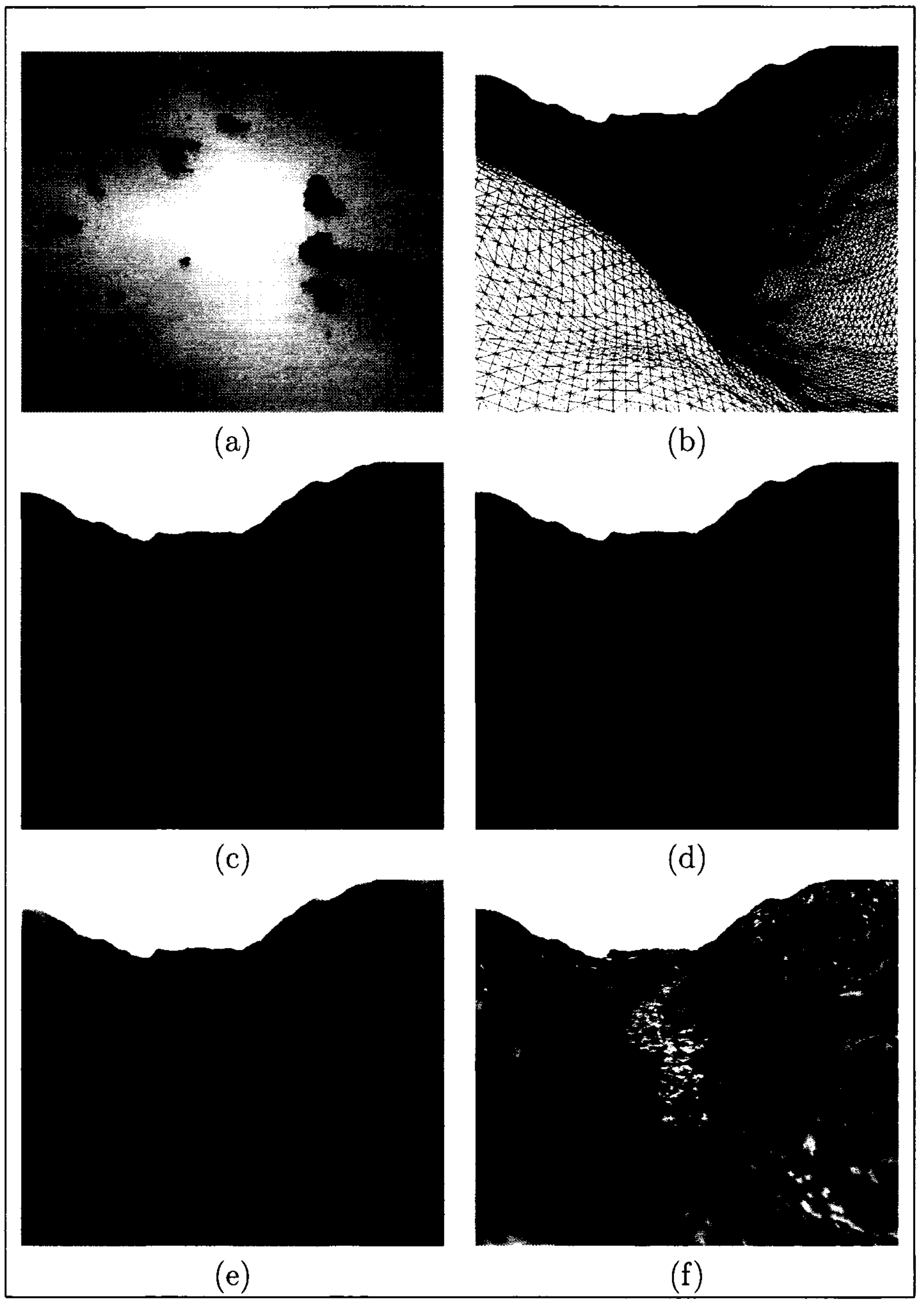


Figure 4.16: (a) 2D image rendering of a mountain taken capture in the SRTM data. (b) 3D wireframe rendering. (c) 3D Phong shaded rendering. (d) 3D Phong shaded rendering with wireframe. (e) 3D rendering where color is based on the height of the terrain, with contour lines. (f) 3D rendering based on the steepness of the third level of details.

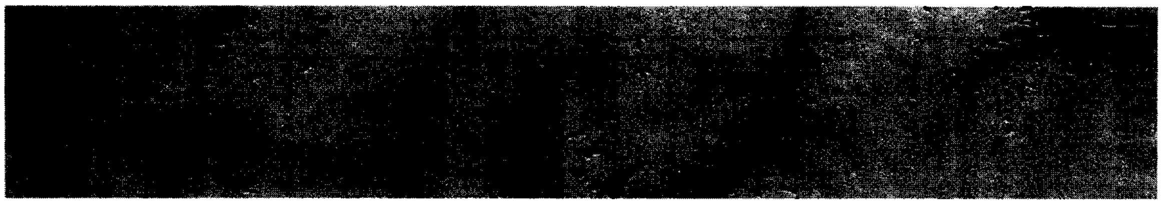


Figure 4.17: The 2D rendering provides an excellent view for the distribution of the voids with an area of terrain.

4.4.2 3D Triangle Based Approach

Terrain is not a discrete object, rather it is continuous. In order to render the areas in between each of the samples, an interpolation technique is required. It is common to use modern graphics hardware to render triangles between the points for 3D visualization of terrain. Current graphics hardware is heavily optimized to render these triangles. Additionally, it can automatically perform the necessary interpolation (shading) between the vertices of the triangles.

In computer graphics, the typical approaches to 3D visualization of terrain have a primary goal of increasing rendering speed. For our purposes rendering speed is not the primary goal, rather it is more important to have a high level of accuracy in the visualization.

Triangulating the Heightmap

There are a number of ways in which one can triangulate a terrain heightmap (Figure 4.18). The following rendering methods all make use of a triangle based approach, and the choice of the triangulation is important. When rendering terrain in three dimensions, there is a rectangular area in between the samples that results from the grid-like structure. Commonly, the goal is to reduce the number of triangles rendered and the rectangle is split into two triangles. It is easy to see that there are

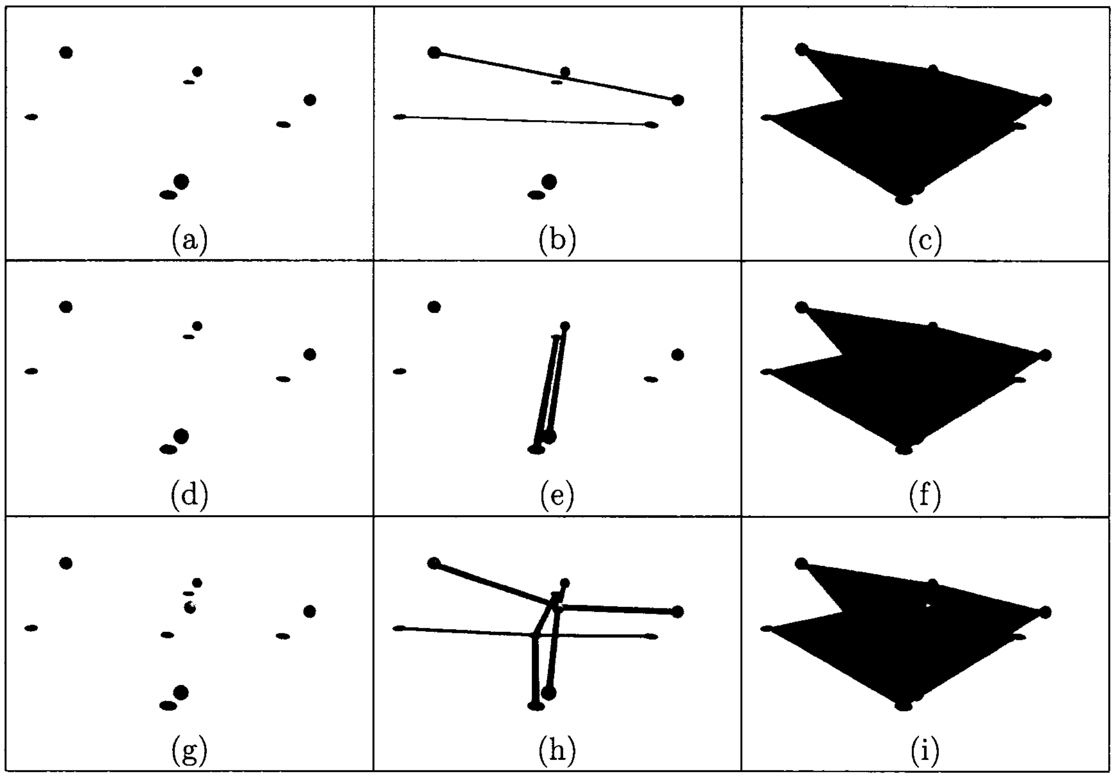


Figure 4.18: (a)(d) Four sample points from a heightmap. (a) The first choice for triangulating the four points and (c) the two triangles which will be rendered as a result. (e) The second choice for triangulating the four points and (f) the two triangles which will be rendered as a result. Note that this is not the same visual result as (c). (g) Our proposed method inserts an interpolated midpoint. (h) The connections of the samples with the new midpoint. (i) The four triangles that are rendered with our triangulation. This method provides a better compromise between the two possibilities.

two choices, and that they do not produce the same rendering of the terrain, (Figure 4.18).

The triangles represent the area in between the samples, and not the samples themselves. Therefore, without more data there is no way to conclusively show that one triangulation is more accurate than the other. We use a new approach that produces a more accurate interpolation (Figure 4.18 (g)). A midpoint is inserted which is equidistant from the four surrounding samples, and four triangles are used to render the space between each of the four original samples and the new midpoint. This triangulation method provides a more accurate visualization of the heightmap than either of the first two triangulation methods.

4.4.3 3D Wireframe Rendering

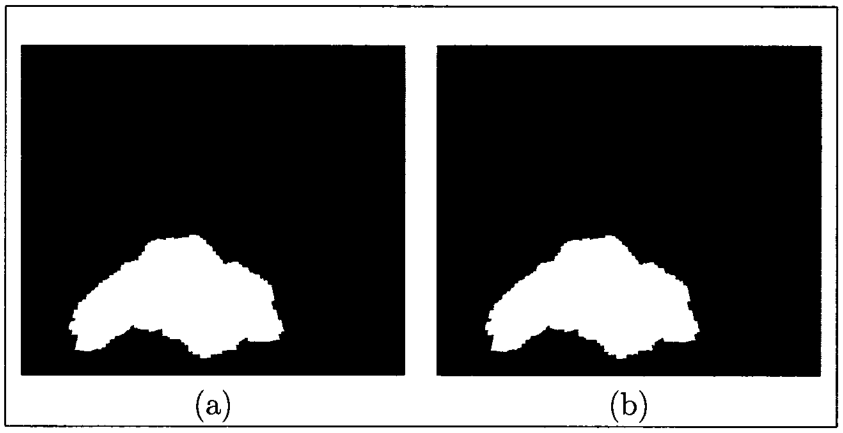


Figure 4.19: Wireframe rendering is ideal for viewing the size of the voids. (a) Without the wireframe, it is hard to determine the size of the void. (b) With the wireframe it is easier to see that the void is about 65 sample across, which is 65*30m or about 2km across.

The wireframe approach renders the outline of each triangle in the triangulated heightmap, see Figure 4.16 (a). This approach provides an explicit view of the size of the terrain and voids contained therein. From a distance, the wireframe approach provides enough detail to view the overall direction and some of the larger details of the terrain. However when viewing the terrain from a close distance it does not provide enough detail to view the intricate characteristics of the terrain, leaving only the overall direction.

4.4.4 Phong Shaded Rendering

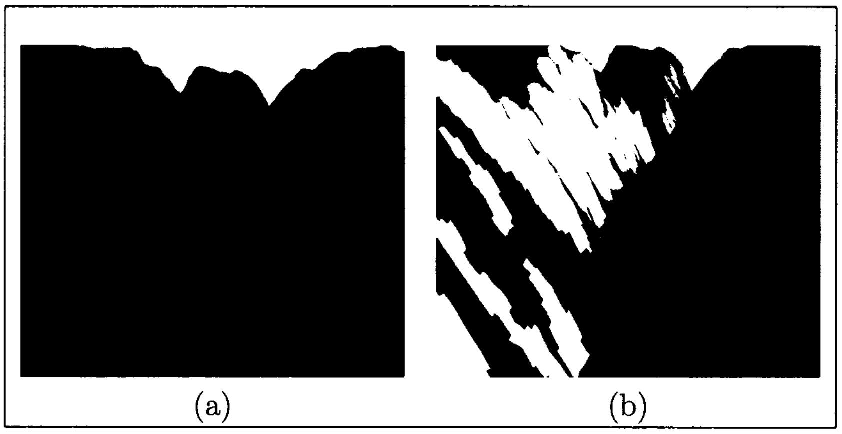


Figure 4.20: Phong rendering is the rendering that we use most often to display results. This is due to its ability to show both the overall direction of the terrain and the details. (a) The results of our synthesis method. (b) The original void.

We also investigate a Phong shaded triangle rendering of the terrain. Phong was chosen over Gouraud as it provides a better view of the intermediate structure of the terrain without losing the ability to view the details 4.21. Again, this approach triangulates the heightmap as discussed in Section 4.4.2. The general lighting equation that we used is [71]:

$$C_{output} = C_a I_a + C_d I_d \left(N \cdot L\right) + C_s I_s \left(R \cdot V\right)^k$$

where I_a, I_d, I_s represent the ambient, diffuse and specular intensity of the light

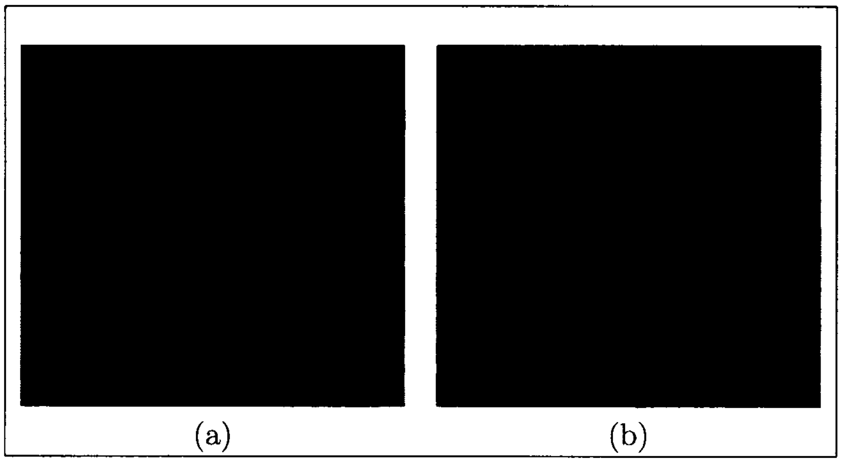


Figure 4.21: (a) Phong rendering of an area of terrain. (b) Gouraud rendering of an area of terrain.

respectively. The ambient, diffuse and specular colors of the terrain are represented by C_a, C_d, C_s respectively. L is the direction from the triangle to the light, N is the triangle's normal, and R is the reflected version of L with respect to N. Finally, k is a factor determining the size of the specular highlight.

For normal calculation, we prefer Phong shading over the Gouraud shading to increase smoothness [71]. Gouraud calculates a normal and subsequent color at each triangle vertex and interpolates these colors across the triangle. Conversely, Phong improves on this model by interpolating the normals from the vertices across the triangle, and calculates the color based on the interpolated normals.

To facilitate a Phong shading, we needed to calculate smooth normals. First, normalized normals are found for each triangle in the triangulation of the terrain. Then, for each vertex in the triangulated terrain we calculate its normal by averaging the normals from each of the surrounding triangles, and normalizing the result.

As can be seen in Figure 4.16, this approach provides enough detail to see most characteristics of the terrain from a distance. It also provides enough detail to view the intricate characteristics of the terrain when close up. For these reasons it is the primary method we use when presenting and comparing our results.

4.4.5 Contour Rendering

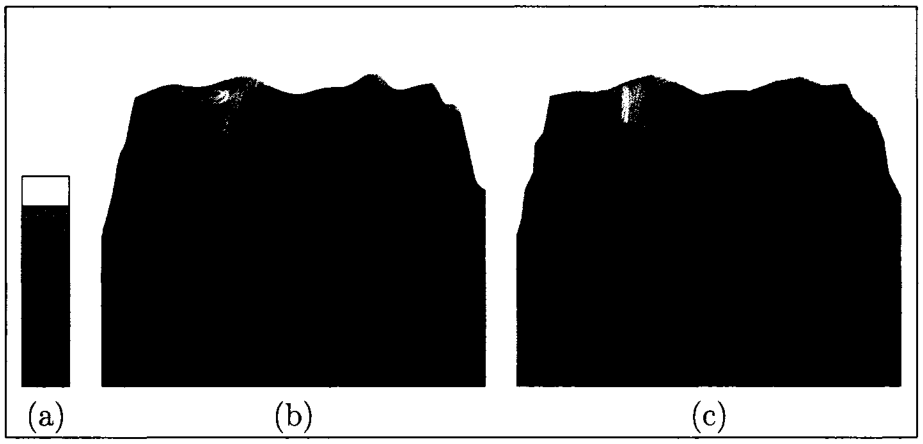


Figure 4.22: Contour rendering based on elevation aids the interpretation of the terrain. The elevation is split into seven equal parts and the colors from (a) are used. This is helpful when comparing the coarse approximations for Multiresolution methods such as (b) Chaikin and (c) Haar.

When viewing a large area of terrain from a distance, it can be difficult to properly interpret the elevation of the terrain. To aid in the interpretation of the terrain we have adopted a common approach: color the terrain based on elevation. We partition the terrain elevation into seven levels and use the color scheme from Figure 4.22 (a) for each level. When comparing the coarse representations obtained through the Multiresolution process, this rendering approach provides a good way to interpret the results (see Figure 4.22).

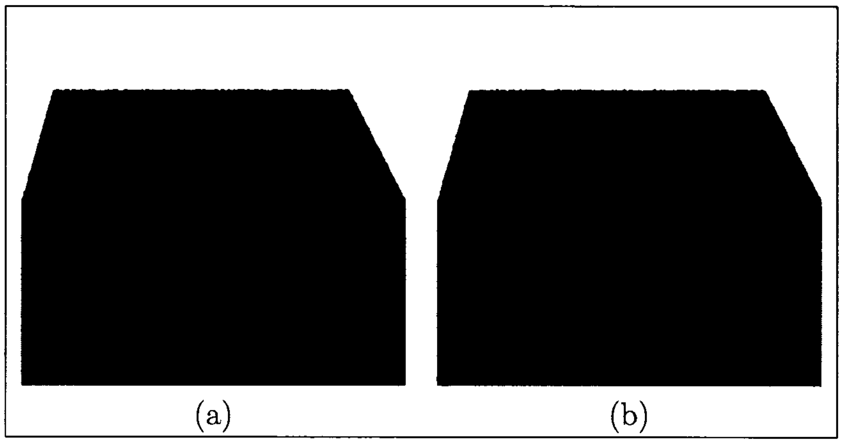


Figure 4.23: Rendering of the details in a manner that emphasizes it as a component of the terrain. Contour visualization as from Section 4.4.5 is used. This method is invaluable when comparing the details for Multiresolution methods such as (a) Chaikin and (b) Haar.

4.4.6 Rendering Details

In addition to visualizing the terrain, we are also interested in visualizing the details that are extracted from the terrain. The main goal of rendering the details is to illustrate them as the characteristics of the terrain. As we discussed in Section 3.2.1, directly rendering vectors will not accomplish this goal. Rather, we render the details in a manner that is similar to our terrain visualizations.

To start, the terrain is decomposed using the RS equations (Chapter 3). Then the average height of the coarse approximation is calculated. Each sample in the coarse representation is then replaced with this average, resulting in a plane. To finish, the plane and the details are used in the RS reconstruction process to create a terrain with just the high frequency characteristics. We can render the resulting terrain using the same techniques as for the original terrain (Figures 3.6,3.7,3.8,3.9).

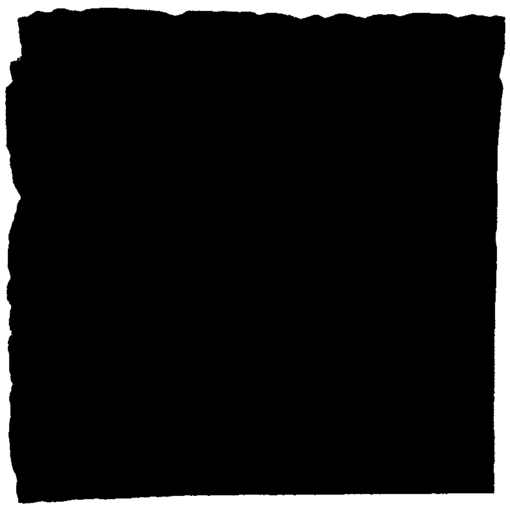
4.4.7 Detail Steepness Rendering

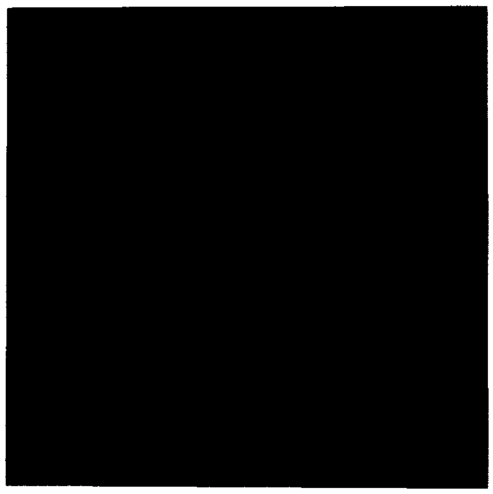
In this section we present a method for terrain visualization that emphasizes its characteristics. Geometrically, slope steepness is a measurement of the angle between the plane tangent to the terrain and the horizontal plane. More generally, it describes the maximum rate of elevation change for the terrain. Sousa et al [21] define slope steepness as:

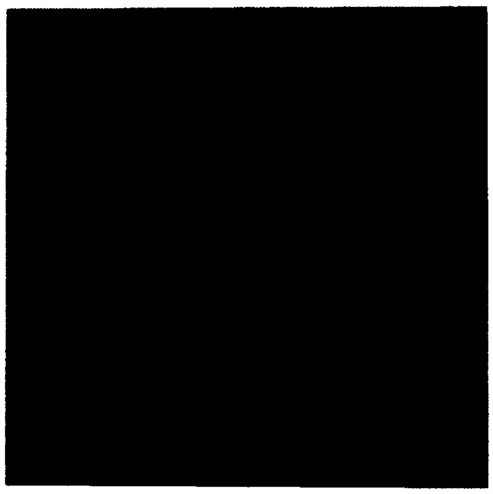
$$GA = arctan\left(\left(\frac{\delta z}{\delta x}\right)^2 + \left(\frac{\delta z}{\delta y}\right)^2\right)^{0.5} \tag{4.19}$$

Brosz [10] uses Equation 4.19 to render terrain in a manner that emphasizes the smoothness or bumpiness of the terrain. Consider the bumpiness of the terrain in Figure 4.24. In the first level of details nearly all of the terrain has the same bumpiness. Even for the second and third level of details there are large areas of terrain with similar bumpiness. When coloring the terrain based on the slope steepness of the original terrain it is not obvious that these areas have similar and in some cases identical characteristics. A simple explanation for this is: the general steepness of terrain greatly affects the chosen color which reduces the effects of the bumpiness for steep areas. For our purposes we would prefer that the visualization of the characteristics is not affected by the general steepness of the terrain.

Recall that we consider the coarse approximation equivalent to the overall structure (or direction) of the terrain. Therefore, if we remove the coarse approximation from the slope steepness calculation, we can calculate the slope steepness of the details. To achieve this, we decompose the terrain and replace the coarse approximation with a flat plain. We use the details and this plane to reconstruct a terrain with only the characteristics, similar to our detail rendering method (Section 4.4.6).


(a)


(b)


(c)

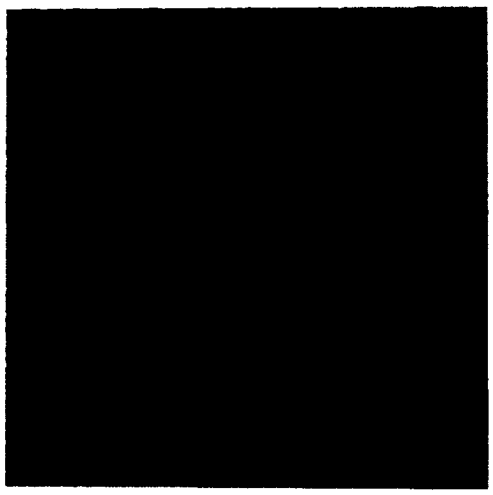

(d)

Figure 4.24: Slope steepness rendering as in Brosz [10]. (a) For the original terrain. (b) For one level of details. (c) For two levels of details. (d) For three levels of details. Note how there are much larger areas of similar bumpiness in the details than the original terrain.

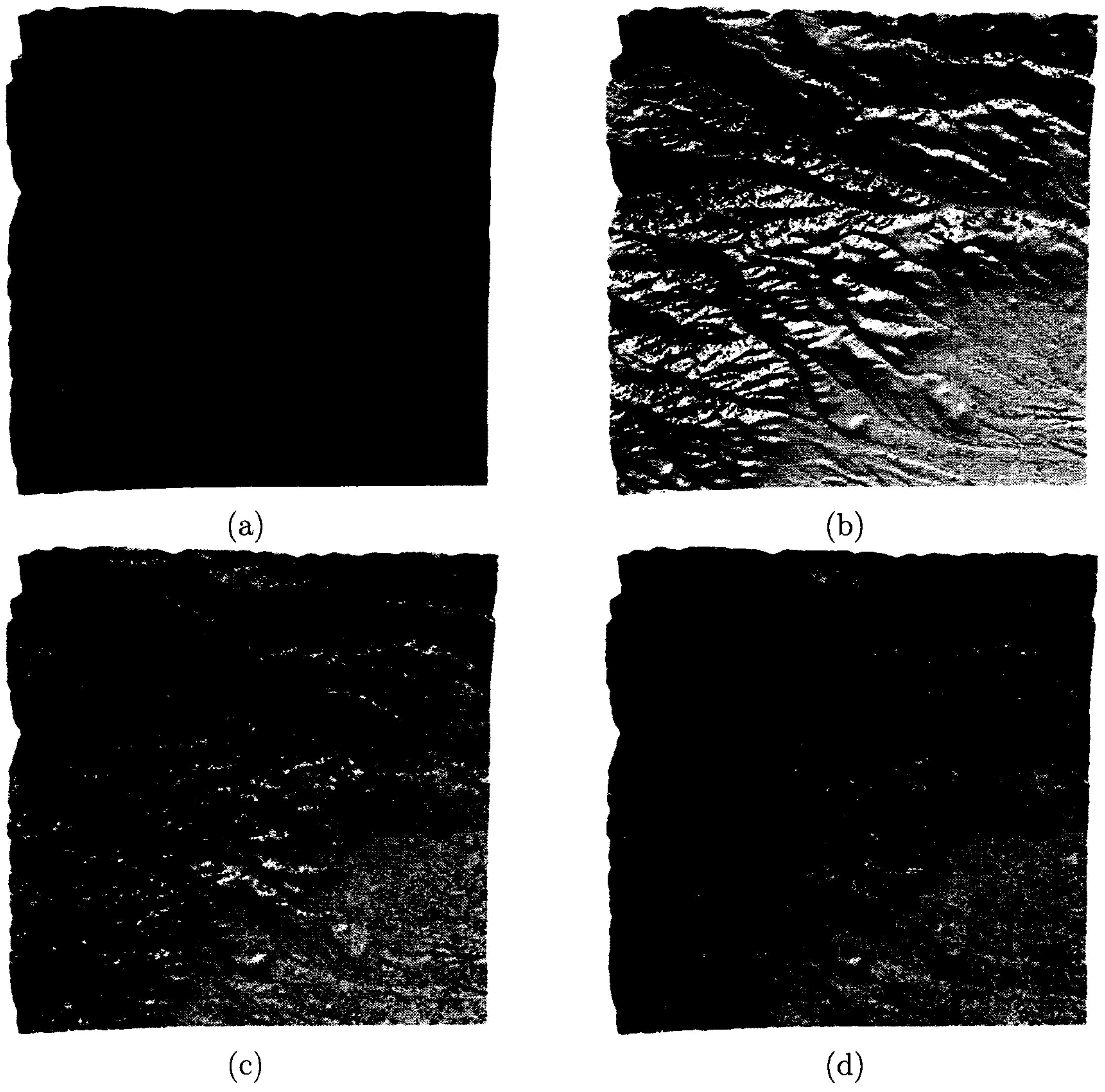


Figure 4.25: (a) Slope steepness rendering as in Brosz [10]. (b) Our details rendering for one level of details. (c) Our details rendering for two level of details. (d) Our details rendering for three level of details. Note how the areas of similar terrain characteristics are readily visible in our method.

Then we calculate the slope steepness of the resulting terrain using Equation 4.19. We use the results of the slope steepness calculation to choose the color for the original terrain (Figure 4.25 (b)(c)(d)). *Red* represents the highest level of steepness, *green* represents the next level of steepness, followed by *blue* and varying shades of gray and black where black represents no steepness.

We can use this visualization to show the different types of characteristics present in a terrain. In addition, it does not depend on the level of decomposition, so we can use it to display each level of details as needed.

4.5 Conclusion

In this chapter we have presented a novel framework for fixing the voids found in DEMs. The method described can provide a smooth patch for voids by analyzing the surrounding terrain in eight directions. In addition, we described a technique that can extract the high-frequency terrain characteristics from the surrounding terrain and apply them to the smooth patch.

We have also presented well-known visualization methods for terrain and discussed how they can help view the results of this work. As a final contribution we have presented a novel method for viewing the characteristics of the terrain, something that was lacking in the well-known methods.

Chapter 5

Applications to Iris Image Synthesis

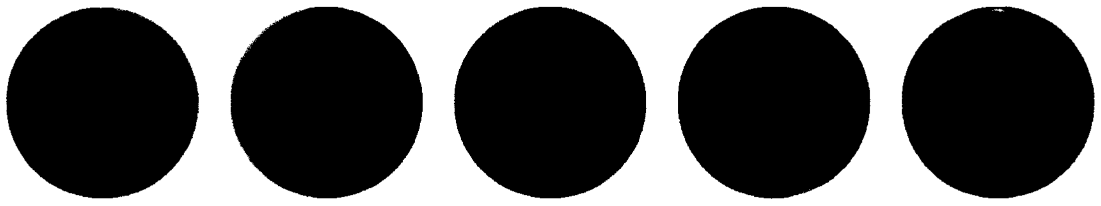


Figure 5.1: Synthetic irises generated with our algorithm.

In this chapter we present an application of our synthesis technique to an important problem in the area of biometrics. Biometrics is the science of using biological properties to identify individuals. The purpose of biometric identification algorithms is to determine if a given biometric identifies anyone within a database of known individuals. Images of an individual's irises are a biometric that can be reliably used for identification [75, 25]. Reasons include:

- irises form early in life and exhibit almost no change after childhood [76],
- irises exhibit high variation between individuals [76]
- images of the iris can be captured without active participation of the subject [76].

However, the development of new iris identification algorithms can be hindered by the lack of test databases with enough sample iris images [78]. At the time of this writing there are only three, freely available, public databases that can be used for testing iris recognition algorithms [16, 56, 27]. This motivates our interest in the development of a new technique for iris synthesis. We propose to use a multiresolution approach to synthesizing new iris images.

In this chapter, we introduce a method to augment existing iris image databases through a novel iris image generation technique. For our synthesis technique, we use multiresolution methods to decompose existing iris images into several components as the characteristics of the iris. We then reconstruct new iris images by combining components from different irises. By using components of real irises in our method we are able to synthesize iris images that are very realistic.

Our synthesis technique creates new, unique irises. Therefore, all comparisons using our synthesized irises will be inter-class comparisons (i.e. comparisons between images of different irises). Our goal is to synthesize iris images that exhibit expected inter-class differences.

The rest of the chapter is organized as follows. First, we describe the available iris image database in Section 5.1. Section 5.2 discusses pre-processing and post-processing steps that we employ when synthesizing new images. Finally, we present the core method utilized for iris synthesis in Section 5.3.

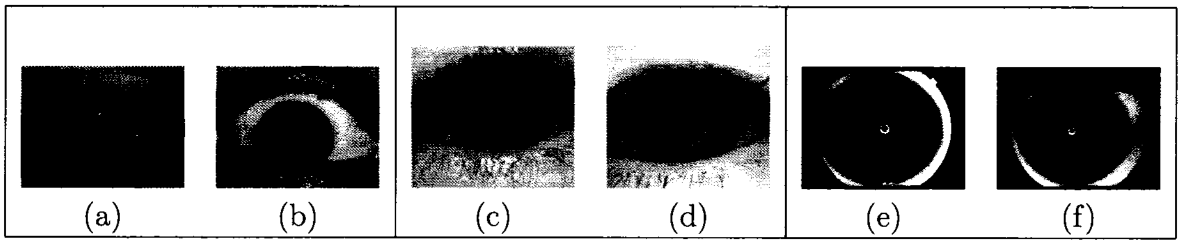


Figure 5.2: (a)(b) Example UBIRIS iris images. (c)(d) Example CASIA iris images. (e)(f) Example UPOL images.

5.1 Image Databases

Our synthesis technique requires a pre-existing set of iris images. There are currently three publicly available databases of iris images: CASIA [16], UPOL [27], and UBIRIS [56] (see Figure 5.2). Each of the three databases has distinct characteristics: CASIA contains black and white images of medium resolution, UPOL contains color images of high resolution, and UBIRIS contains color images of high resolution as well as noisy images. Our method will use the existing images in the database to synthesize new images. The synthesized images will be of the same color space and quality of the input iris images which, in turn, affects our decision of which iris database to use. Although UBIRIS is the largest public and freely available iris database, it purposefully contains many noisy iris images for testing purposes. We intend to extract characteristics from the existing iris images and noisy images will produce noisy characteristics which is unsuitable for our purposes. The CASIA database images are black and white making it difficult to visually inspect the synthesized images. Therefore we chose the UPOL database which contains high-quality, high-resolution iris images that aid visual inspection as well as remove the need to detect and avoid noisy input images.

5.2 Preprocessing: Isolating the iris

The images in the UPOL database may also contain the surrounding eye, eyelids and the pupil (see Figure 5.5). In order to rid the image of this extraneous information, we use a number of preprocessing steps (see Figure 5.5). The iris itself is a circular, pigmented portion of the eye that functions to regulate the amount of light allowed to pass through to the retina. The stroma is a fibrovascular tissue that is visible as the lines connecting the outer edge of the pupil with the outer edge of the iris. Additionally, irises have circular rings that are called the collarette. The stroma and collarette give the iris some structure. We would like to take advantage of this structure but reverse subdivision operates on the rows and columns of the image (Section 3.1.3). Therefore, we also use a preprocessing step that unwraps the iris image such that the stroma and collarette are aligned with the rows and columns of the image.

Before describing the details of our technique in Sections 5.2.1 and 5.2.2, we give a high-level overview of the method here. First, the iris is transformed into a polar coordinate system. The polar coordinate system is defined in terms of (r, θ) where θ is an angle from a polar axis and r is the distance from the origin. This effectively unwraps the circular iris information into columns and rows. Next, the pupil is removed such that only the iris information is left in the image, a necessary requirement of our synthesis method. Once a new iris image has been synthesized, we re-insert the pupil information and return the iris to its original coordinate system. The sequential organization of the preprocessing stages and subsequent synthesis framework are provided in Figure 5.3. We explain each of the preprocessing stages in detail in the following sections.

5.2.1 Polar Coordinate Representation: Unwrapping the Iris

The circular nature of the iris lends itself to an alternate coordinate system. We use a polar coordinate transform to *unwrap* the iris image into a rectangular shape, resulting in the image in Figure 5.5(c).

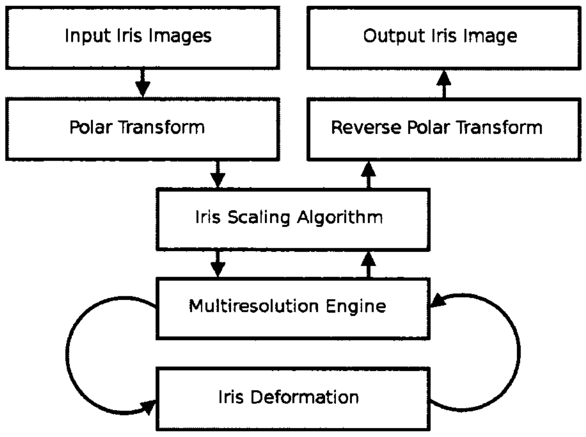


Figure 5.3: The preprocessing stages, and subsequent multiresolution engine organization.

Each input image is a uniform sampling of the light reflecting off of an iris. The image resulting from the polar coordinate transform represents a uniform sampling in the angle and radius axes. A standard polar coordinate transform produces an output image of the same width and height as the input image. However, the Cartesian and polar sampling rates are not equivalent. The outer edge of the iris is a higher resolution sampling of the iris in the angle dimension, than is the inner edge. To capture all of the details of the iris the resolution of the outer edge can be used.

For the unwrapped image, we use a resolution that is large enough to contain the unwrapped circumference of the iris. The circumference of any circle is π multiplied by the diameter. For our database, the diameter of the iris is approximately the width of the image. Therefore, the resulting image size must be at least πw_i, where w_i is the width of the input image. Digital images use integer sizes, hence we use the ceiling of $\pi = 4$ which results in a polar coordinate image with dimensions $4w_i \times w_i$. The inverse transform reverses this process, producing an iris image in the Cartesian coordinate system that is the same size as the original iris image.

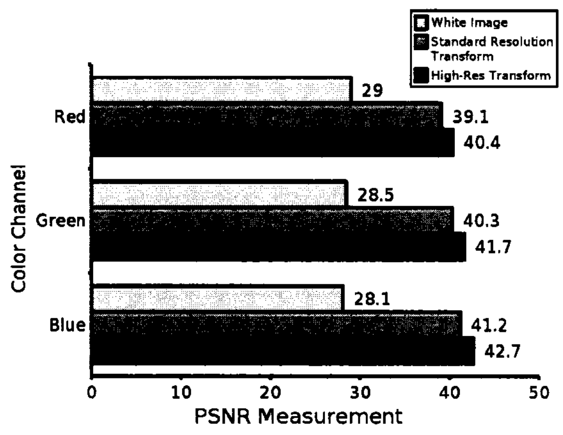


Figure 5.4: Chart of the PSNR measurements used to test the two polar coordinate transforms. *Standard transform* refers to a transform that results in an image size of $w_i x w_i$. *High-res transform* denotes our high resolution transform that results in an image size of $4w_i x w_i$

To test this high-resolution transform, we used the standard measure of peak signal to noise ratio (PSNR) [54]. PSNR compares an original signal to a reconstructed version and provides a measure of the similarity of the original signal to the reconstructed signal. We use the following definition of PSNR:

$$20.0 \log_{10}\left(\frac{255.0}{\sqrt{\frac{\sum\left([f(i,j)-F(i,j)]^2\right)}{p^2}}}\right) \tag{5.1}$$

where p is the number of pixels in the image, f is the original iris image and F is the image obtained by applying a polar coordinate transform followed by the inverse transform. We use our high resolution polar transform and compare its PSNR value with that of a standard resolution transform. Although it is straightforward to tell that an increase in the PSNR value represents a decrease in the noise present, PSNR is by nature a relative measurement. Consequently, we also compare the PSNR values with those from a known result. For this purpose we also compared a completely

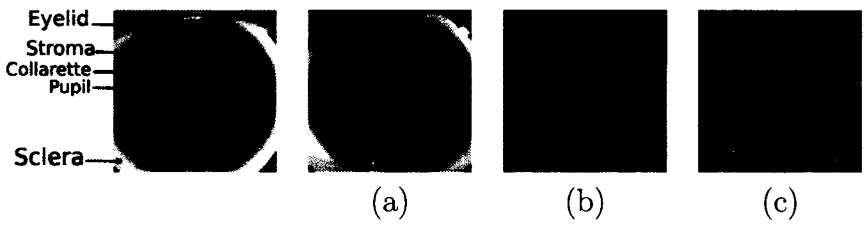


Figure 5.5: Iris image preprocessing steps, (a) the original image, (b) after being unwrapped with the polar coordinate transform, and (c) after being scaled using piecewise linear scaling algorithm.

white image, the equivalent of extreme noise, to the original iris image and recorded the resultant PSNR. The chart in Figure 5.4 demonstrates that the high resolution polar coordinate transform provides a significant improvement in the quality of the resulting image.

5.2.2 Removing The Pupil: Iris Scaling

The pupil is the central portion of the iris, where the light passes through to the retina. Its size is dependant upon the amount of light available. In addition, the pupil may not be centered in the iris image. Both of these conditions produce abnormalities in the polar coordinate images, see Figure 5.5(b). The variability of the pupil size causes a black rectangular area in the top of the image. There may also exist small parabolas along the bottom of the image due to non-centered pupils. In order to isolate the iris for our synthesis algorithms we should remove this extraneous information.

For this iris scaling algorithm, we needed an efficient technique that would allow us to both enlarge the iris and then shrink it with minimal distortion. In most cases our experimental results indicated that we would be increasing the size of the iris by about 133% and then shrinking it by 75% in the reverse process. We use a piecewise linear scaling algorithm [69]. This algorithm re-samples the column to produce an output column with the desired number of samples. Edges of the iris are found in each column by detecting the first pixel from the top, and the first pixel from the bottom which is not black. The detection may be sensitive to noise in the pupil. However, the noise can be removed by some averaging filters or the edge of the iris can be detected using line detection algorithms as is standard in iris recognition research [76, 25]. Have some problems if there is some noise in the pupil A threshold is used in situations where lighting and the color of the irises vary. The threshold can be found by analyzing the pupils and irises in the given database to determine their typical color. The re-sampling algorithm maps the index of each pixel in the scaled column into the domain of the original column using a standard parametric representation. The resulting parameter is a real number and a linear interpolation is used to obtain the desired value for the pixel in the scaled column. The image is then ready to be used in the combination algorithm to produce the new iris image. The resulting image is similar to the one in Figure 5.5(c).

Once the synthesis algorithm is finished, the new iris image will also appear similar to Figure 5.5(c). It is then post-processed to return it to the original shape and size of a real iris (see Figure 5.3). During the post-processing each column is scaled to a fraction of its original height. The size used during the post-processing step can be varied to achieve varying sized pupils. This allows our synthesis algorithm to produce images of the same iris with different pupil sizes.

The iris image obtained through the reconstruction process is an exact copy of the original iris image.

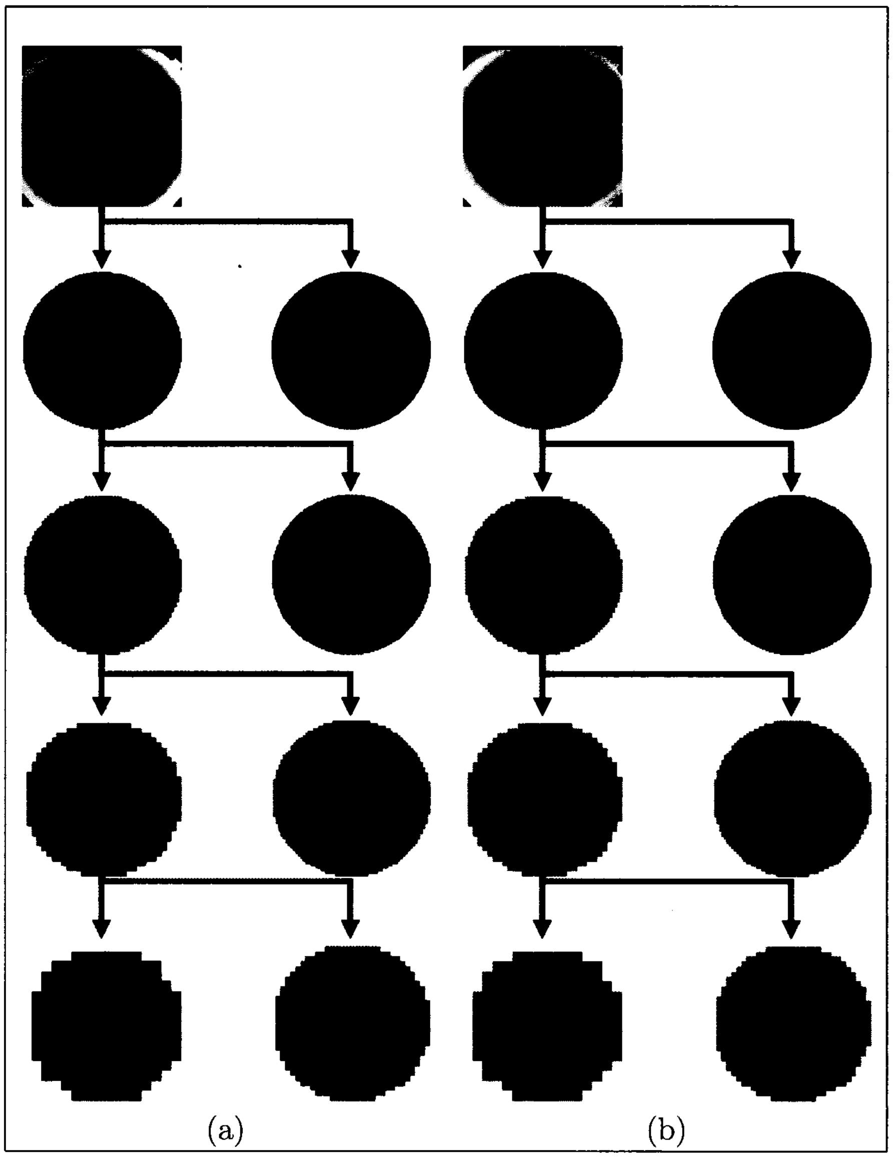


Figure 5.6: Iris image decomposition. The original iris image is on the top row. Each subsequent row contains a coarse representation after one level of decomposition and the details extracted during the process.

(a) (b) (c) (d) (e) (f)

Figure 5.7: (a) Original iris. Coarse approximation after (b) two, (c) three, (d) and four levels of reverse subdivision. The third level still contains pattern information, while the fourth level contains only color information. (e) Original iris image, compared with details (f) captured from four levels of reverse subdivision.

5.3 Iris Synthesis: Combining Multiple Iris Images

Our goal is to augment existing iris image databases through our image synthesis process. The synthesized images should display similar characteristics to real irises, yet be unique. To achieve the goal of realism, we introduce a method that uses characteristics of real iris images. For the goal of uniqueness our method uses characteristics extracted from multiple differing irises.

Looking at any iris image, it becomes obvious that most of its characteristics are made of high resolution data. Therefore, we employ the filters of reverse Chaikin subdivision (Section 3.1.2) to capture these details from the iris. Experimental results, shown in Figure 5.7, demonstrate that only four levels of decomposition are required to effectively capture all of the details from a 256x256 iris image. The low resolution approximation which is left after four levels of decomposition contains only the global color scheme of the given iris. To demonstrate this observation, we decomposed an iris image and replaced it's low resolution approximation with a solid grey image. We then restored it to its original resolution, resulting in the grey iris in Figure 5.7(f). The new iris is simply a grey-scale version of the original iris image.

We can therefore conclude that the details extracted during four levels of decomposition completely capture the characteristics of an iris image; an observation that is central to the combination algorithm. This observation allows us to completely decompose an iris image into the following five components: the base color of the image and the four sets of characteristics at each resolution level denoted by: I^{k-4}, D^{k-1}, D^{k-2}, D^{k-3}, and D^{k-4} respectively.

5.3.1 Selecting Combinations of Iris Images

Recall that any 256x256 iris image can be decomposed into five new components. The reverse is also true, a new 256x256 iris image can be created from five components, due to the properties of multiresolution. For each new synthesized iris image, we select the five necessary components from those available to us in the original database. As a way of synthesizing new unique iris images, each of the components can be selected from a different iris. Given a database of N input images, we can decompose each of these images into their five components. Therefore, when synthesizing an iris image, we have N choices available for each of the necessary components which allows us to create a total of N^5 possible combinations. The original N iris images are included in these new combinations, as they are recreated when each of the selected components is from the same iris image. Clearly, given even a reasonable small database to start with, our method can generate an exponential increase in size.

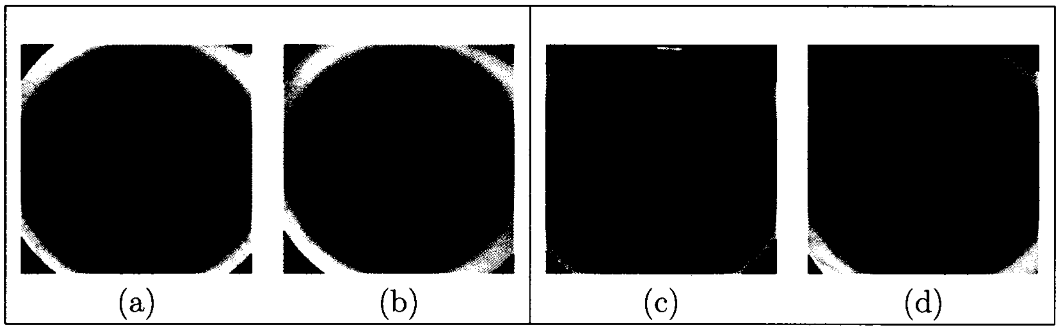


Figure 5.8: Classifications of iris images, (a)(b) with two rings of similar frequency characteristics and (c)(d) with one ring of similar frequency characteristics.

5.3.2 Validating the results

In this work we will use two approaches for validating our iris images. In the first approach, we will individually consider each synthetic image in terms of quality and uniqueness. For this approach to be tractable, we do not generate an exhaustive set of iris images. We restrict the method to a small number of irises which are a good distribution of the possible combinations Section 6.2.1. For the second approach, we will use an iris-recognition algorithm to test the individuality of the iris images when compared to the original iris images and our synthetic irises Section 6.2.2.

5.3.3 Classifying irises images

Fingerprints have well known classifications that allow the fingerprint generation algorithms to target a much smaller subset of the full range of possibilities. In the UPOL database, there are two distinct types of iris images and we propose a technique for classifying them. This allows us to separate the irises into groups of similar types and subsequently improve the realism of the results.

All of the irises contain visible lines that run from the outer edge in towards the

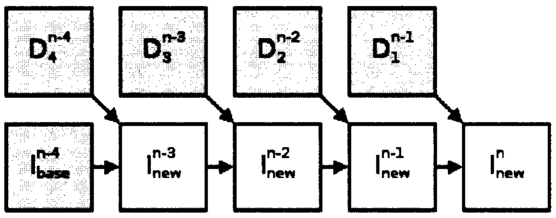


Figure 5.9: Structure of the combination algorithm.

pupil. In the first set, each iris will have two distinct areas: an outer ring where the lines appear thinner and closer together and an inner ring of where the lines appear thicker and further apart (Figure 5.8(c)(d)). In the second set of irises, the lines are uniform throughout the iris, although the color of the iris may change slightly (Figure 5.8(c)(d)). When choosing iris images that are to be combined, we organize the irises into these two sets. The UPOL Database does not contain any irises that do not fall into these groups. However, we expect that this classification can be applied to other databases and can be further generalized for other cases. An example is provided in Figure 5.8, where we demonstrate two different classes of irises.

Once one set of iris images have been classified, the possible combinations of irises that can be made from this small set is very large. If, for example, only 8 iris images are available, a possible $8^5 = 32768$ images can be created. Although it would be possible to automate the classifications, the combinatorial possibilities of this method make this automation a secondary goal. A user need only obtain a small number of similar iris images which can be chosen by any iris expert, and then the user is able to synthesize a database large enough to satisfy their needs.

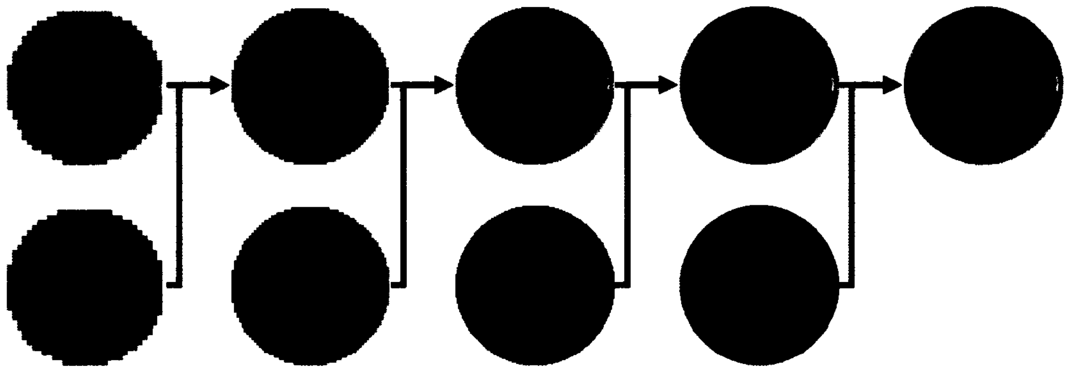


Figure 5.10: The components used to synthesize an iris and the final output iris. The left-most column is the initial set of details and coarse approximation extracted from the original irises. The subsequent columns have the results of the reconstruction process on the top and another set of details on the bottom. The right-most column is the final output iris image.

5.3.4 Combining the details to form a new iris

Our combination algorithm requires five images from the database as input; using the subscript notation to identify each of the five images we have: I_{base}, I_1, I_2, I_3, and I_4. From these images, we extract the five necessary components to build a complete iris image. First, the base color information for the new iris image I^{k-4}_{new}, is directly extracted from I_{base} by using the low resolution approximation of I_{base}: I^{k-4}_{base}. Then the characteristics, D^{k-4}_4, D^{k-3}_3, D^{k-2}_2, D^{k-1}_1 are each extracted from one of the other four images. The process starts by combining the base color information, I^{k-4}_{new}, with the lowest resolution set of details, D^{k-4}_4 that results in a new, intermediate resolution iris image, I^{k-3}_{new}. Similarly, we obtain the higher resolution images, I^{-2}_{new}, I^{k-1}_{new}, and I^{k}_{new} by iteratively applying (3.4). :

$$I^j_{new} = \mathbf{P}^j I^{j-1}_{new} + \mathbf{Q}^j D^{j-1}_{k+1-j} \quad j = n-2, n-1, n \tag{5.2}$$

Figure 5.9 provides a visual overview of the composition of the new iris image, and Figure 5.10 shows the construction of an actual iris from synthetic components. The use of details from multiple real iris images provides the synthesized iris with both a unique and realistic combination of characteristics as contained in the real irises.

5.4 Conclusions

In this chapter, we have presented a novel method to augment existing iris image database using our synthesis technique to generate new iris images. We have shown how an iris can be isolated from surrounding anatomy by using a polar transform and scaling technique. Using reverse subdivision, we have illustrated how individual iris images can be decomposed into components that represent iris characteristics. Our techniques allow characteristic components from multiple irises to be combined forming new unique iris images. We further increase the effectiveness of this technique by organizing the pre-existing irises into compatible groups resulting in synthetic images with greater realism.

Chapter 6

Experimentation

In this thesis, we have presented a synthesis technique based on multiresolution. The technique is based on extracting characteristics from existing digital models. These characteristics are used in the synthesis process to create digital models that have realistic properties. In many cases, including ours, the original and synthesized models have a visual nature, and the primary validation method is visual. Our main criteria for evaluating the success of our synthesis method is the realism of the results. In addition, we have provided two applications of our synthesis technique and each application has its own criteria for success.

6.1 Terrain Repair Experimental Results

In the first application of our synthesis method, we provided a technique to repair the voids that are commonly found in DEMs. The goal is to produce a patch that fits with the surrounding terrain. That is, when presented with the patched terrain a viewer should not be able to notice the areas that have been patched. To achieve the goal of realism, we used multiresolution to extract and use characteristics from the terrain surrounding the void. Therefore, we also evaluate the realism achieved by using multiresolution for extracting and applying the characteristics of existing models.

In section 4.1.1, we discussed Brosz's findings that multiresolution techniques perform better than other widely used terrain synthesis methods. Statistical measures of terrain are typically related to these synthesis methods. As we discussed, in our method the patches are generated in such a way that the variations from from the base curve follow the same measures as the surrounding terrain. Therefore our method inherently uses basic statistical plausibility to fill the voids, however we also solve the issue of visual artifacts(see Figure 4.12). In addition, consider visual artifacts such as those in Figure 4.12. The artifacts may not affect statistical evaluation of the patch but are detrimental to visual quality. Recall that our primary goal is to provide a patch that fits with the surrounding terrain. Thus, the visual inspection process provides the main validation of our patches.

Terrain does not have C^1 continuity and therefore may have large features in small areas. We cannot reproduce these features in areas where there is no evidence of the feature in the surrounding terrain. However, to provide an initial validation of our work, we have created an artificial void in an area of terrain where we have complete data. This allows us to compare our synthetic results with the real terrain to determine if our patch is indeed realistic. We provide two figures which illustrate the results: 6.5, 6.6. The patches in both illustrations have characteristics that fit with the surrounding terrain and fit with the original data for the void.

6.1.1 Realism of patches

Testing the realism of the patches is achieved by viewing the patches. We present the results of five tests on terrain that varies in both the general steepness and overall characteristics. They range from relatively flat areas (Figures 6.1, 6.3, and 6.4) to mountainous areas (Figures 6.2, 6.1.3, 6.8 and 6.9). In all of these cases, the patches

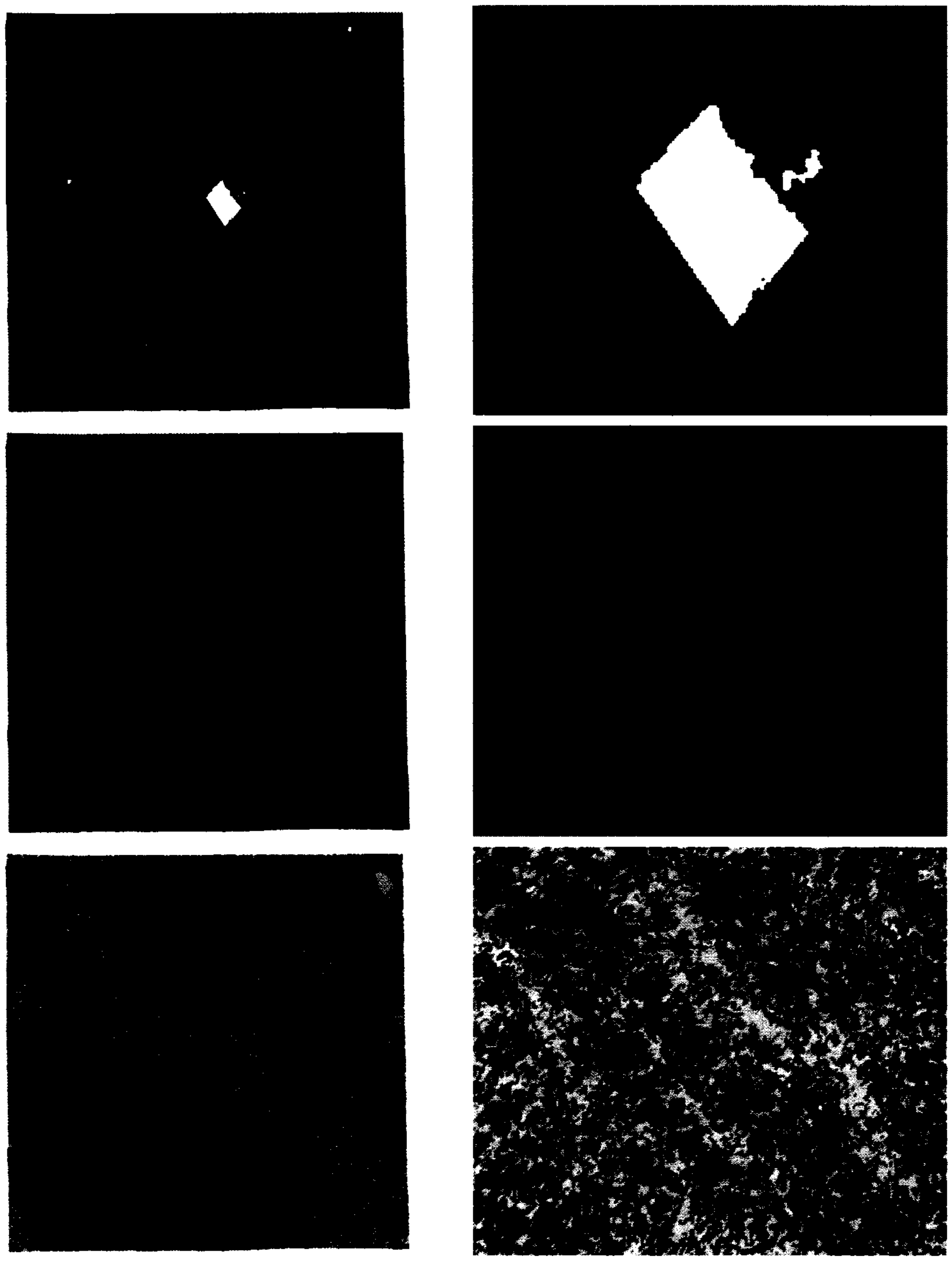


Figure 6.1: The left column represents a high level view of the void, and the right column shows a closeup. The first row illustrates the voids in the terrain. The second row illustrates the fixed voids with a phong shaded rendering. The third row illustrates the fixed voids with our characteristics rendering.

that are produced are an excellent match for the surrounding terrain. The areas of the void are undetectable without the aid of the photos that show the voids.

Due to the even spacing between the sample, DEMs cannot store vertical areas of terrain. In addition, there are very few samples with which to represent extremely steep portions of terrain. Our algorithm will fix these areas, but the resulting patch will not exhibit the same characteristics for extremely steep areas (Figure 6.10).

6.1.2 Characteristics extracted with Multiresolution

Recall that our method focuses on the characteristics of the patch. That is, each of the patches should have characteristics that are similar to the surrounding terrain. To illustrate our success in achieving this goal, we provide two types of results. Firstly, we use the rendering technique, introduced in Chapter 4, to visualize the terrain with characteristics emphasized. Using this style of visualization, we show that all of our patches contain characteristics that are very similar to the surrounding terrain. Secondly, we show our patches alongside smooth patches that do not use any of the surrounding characteristics (Figure 6.3). Although all of the figures support our results, Figure 6.3 emphasizes the usefulness of the characteristics for increasing the realism of the patches. In this figure, the smooth patch is clearly visible, and does not fit in with the surrounding terrain. Conversely, the patch produced by our method has characteristics that are identical to those in the surrounding terrain.

6.1.3 Characteristics Rendering

Finally, we should also discuss the success of the rendering technique that emphasizes the characteristics of the terrain. In each of the figures where this rendering style

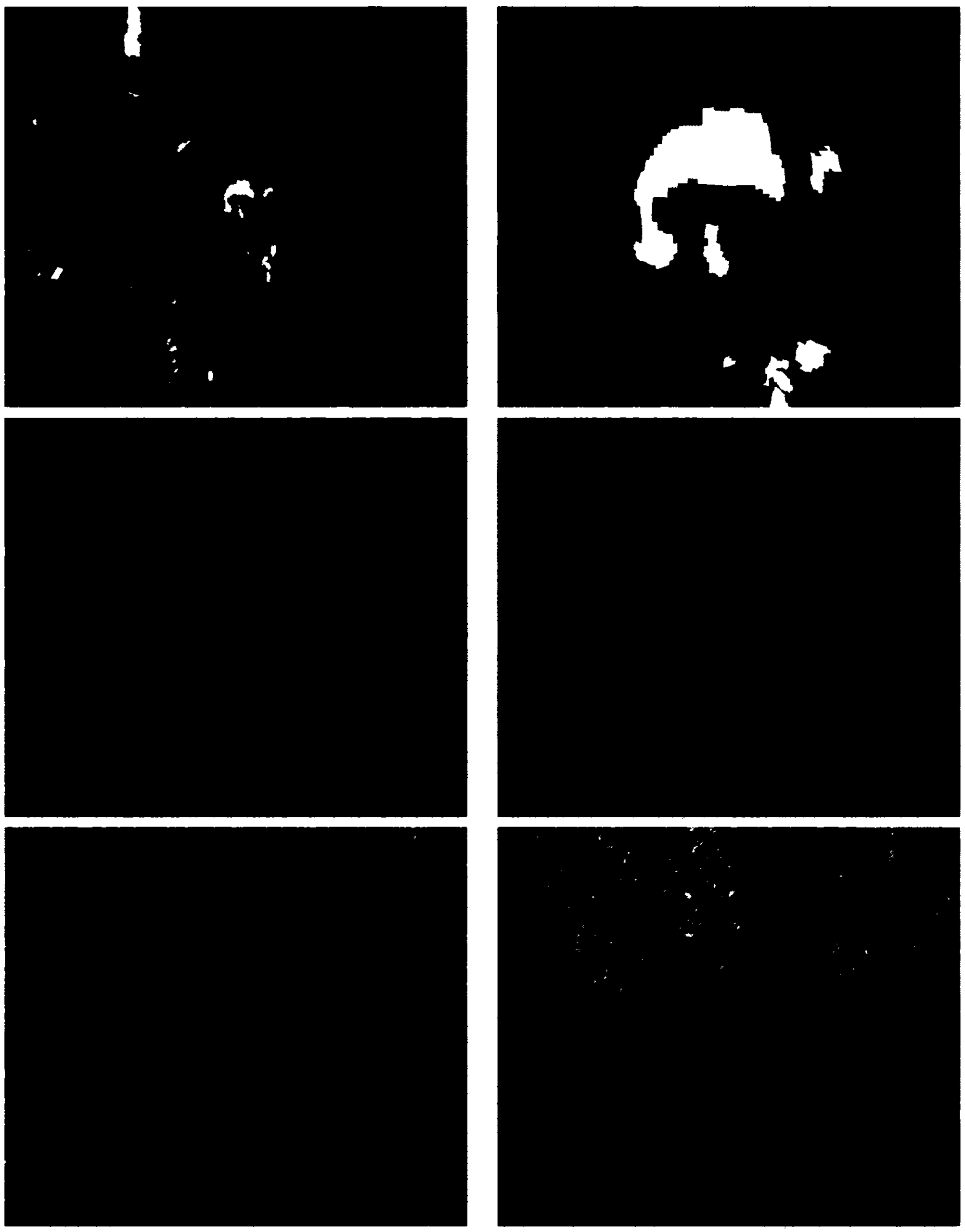


Figure 6.2: The first row illustrates the voids in the terrain. The second row illustrates the fixed voids with a phong shaded rendering. The third row illustrates the fixed voids with our characteristics rendering.

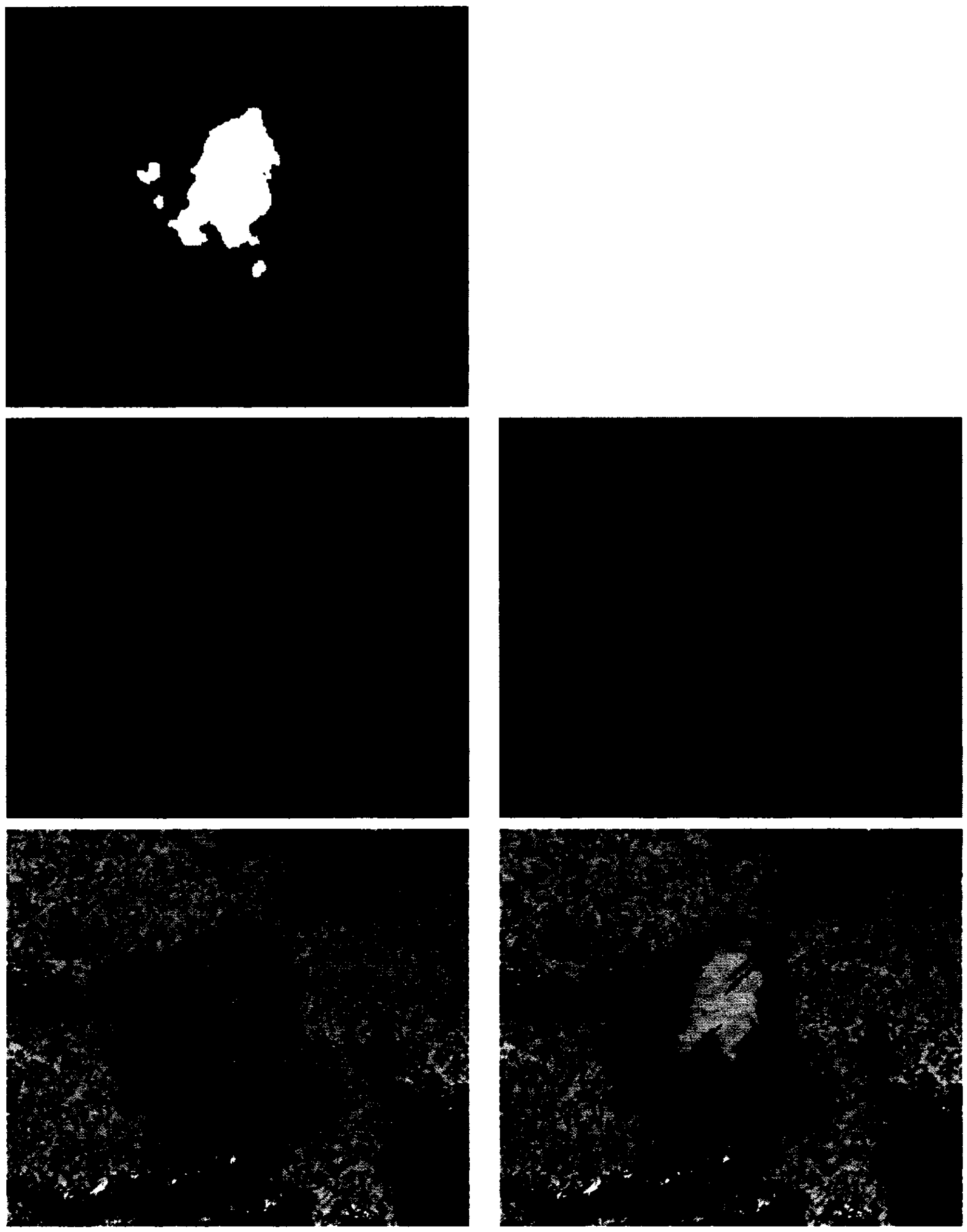


Figure 6.3: The first row illustrates the voids in the terrain. The second row illustrates the fixed voids with a phong shaded rendering. The third row illustrates the fixed voids with our characteristics rendering. The right column is a smooth patch, with no characteristics.

is used, it provides a good representation of the characteristics of the terrain. In addition, to illustrate its usefulness, consider the flat areas in Figures 6.3 and 6.4. In each of these figures, the flat area, which is devoid of characteristics, is visible in the illustrations.

6.2 Iris Synthesis Experimental Results

In the second application we provided a synthesis method for iris images. The goals of this method were to increase the size of existing iris image databases with realistic, synthetic irises. When considering an individual synthetic iris, the main goal is to create unique irises with a realistic appearance. In addition, the final goal of this application is to synthesize images that are suitable for testing iris recognition methods.

6.2.1 Database Augmentation

In the first test of our method, we selected two sample databases of irises from the existing UPOL database. The first database consisted of all of the UPOL images with duplicates removed; leaving a total of 128 iris images. We named this database *Complete*. The second database is a copy of the first database but is separated into two groups. The first group contains the irises that have two rings of distinctly different characteristics. The second group contains the irises that have one ring of characteristics. We called this second database *Grouped*; it also consists of 128 iris images. Our method was used to augment each of these two databases.

To fully test our method, we needed to consider each synthesized iris image

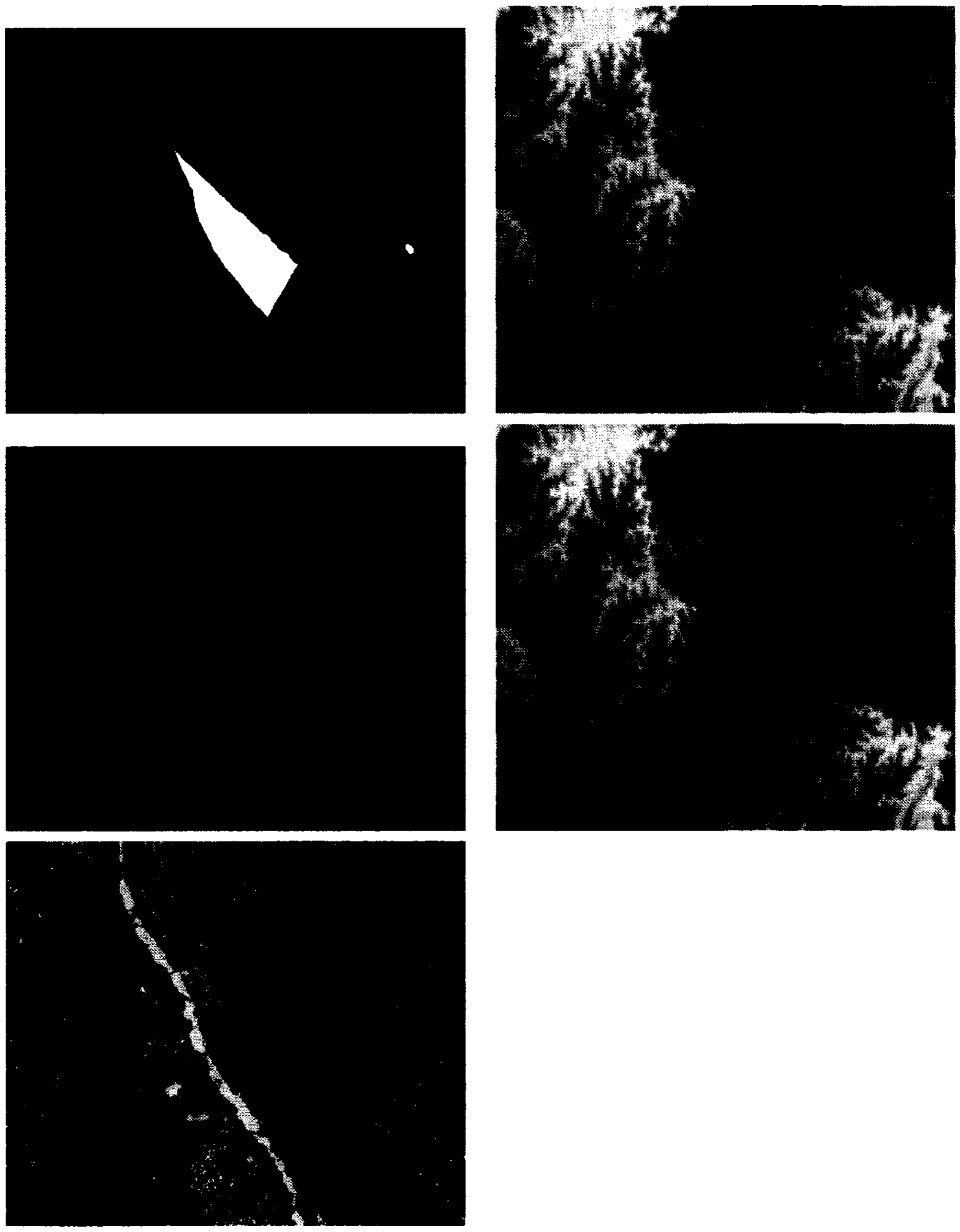


Figure 6.4: The left column illustrates this result using 3D rendering techniques, while the right column illustrates the result using the 2D image based visualization. The first row illustrates the void in the terrain. The second row illustrates the fixed void. The third row illustrates the fixed void with our characteristics rendering.

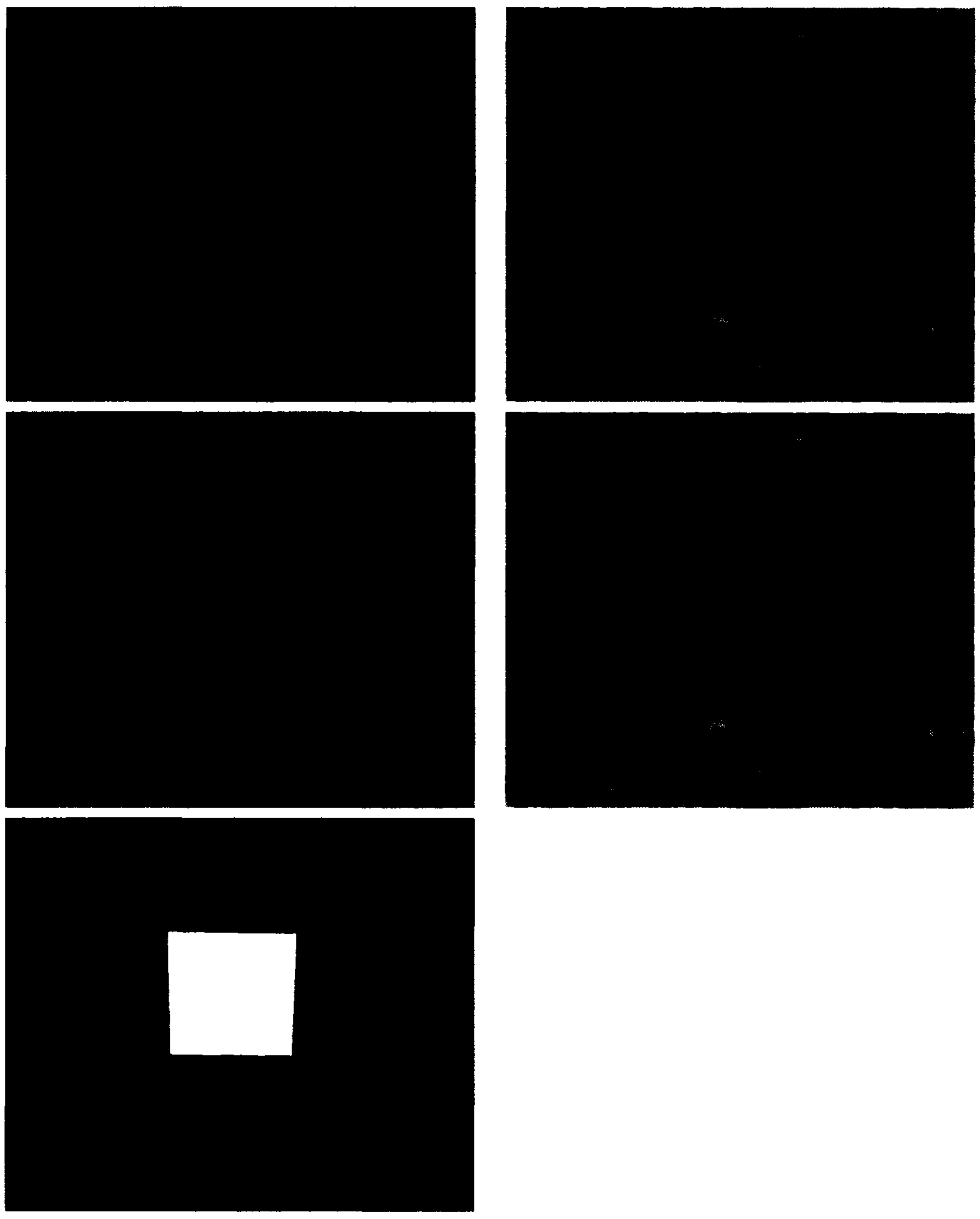


Figure 6.5: Illustrates the results of our patch for voids for which we have real data. In the first row is the original terrain in the phong and details rendering. The second row contains our fixed void in the both the phong and details rendering. The final picture illustrates the area of the void.

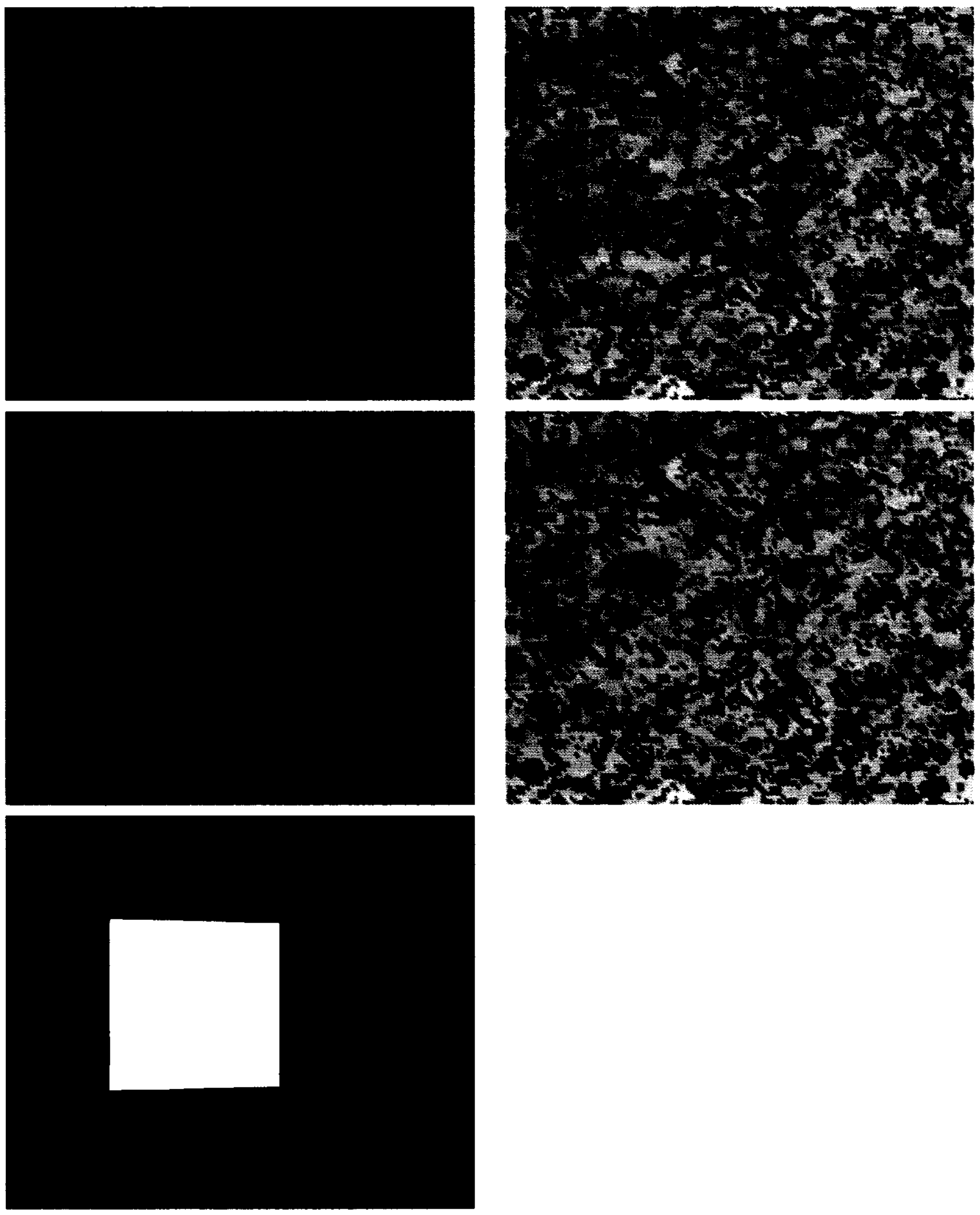


Figure 6.6: Illustrates the results of our patch for voids for which we have real data. In the first row is the original terrain in the phong and details rendering. The second row contains our fixed void in the both the phong and details rendering. The final picture illustrates the area of the void.

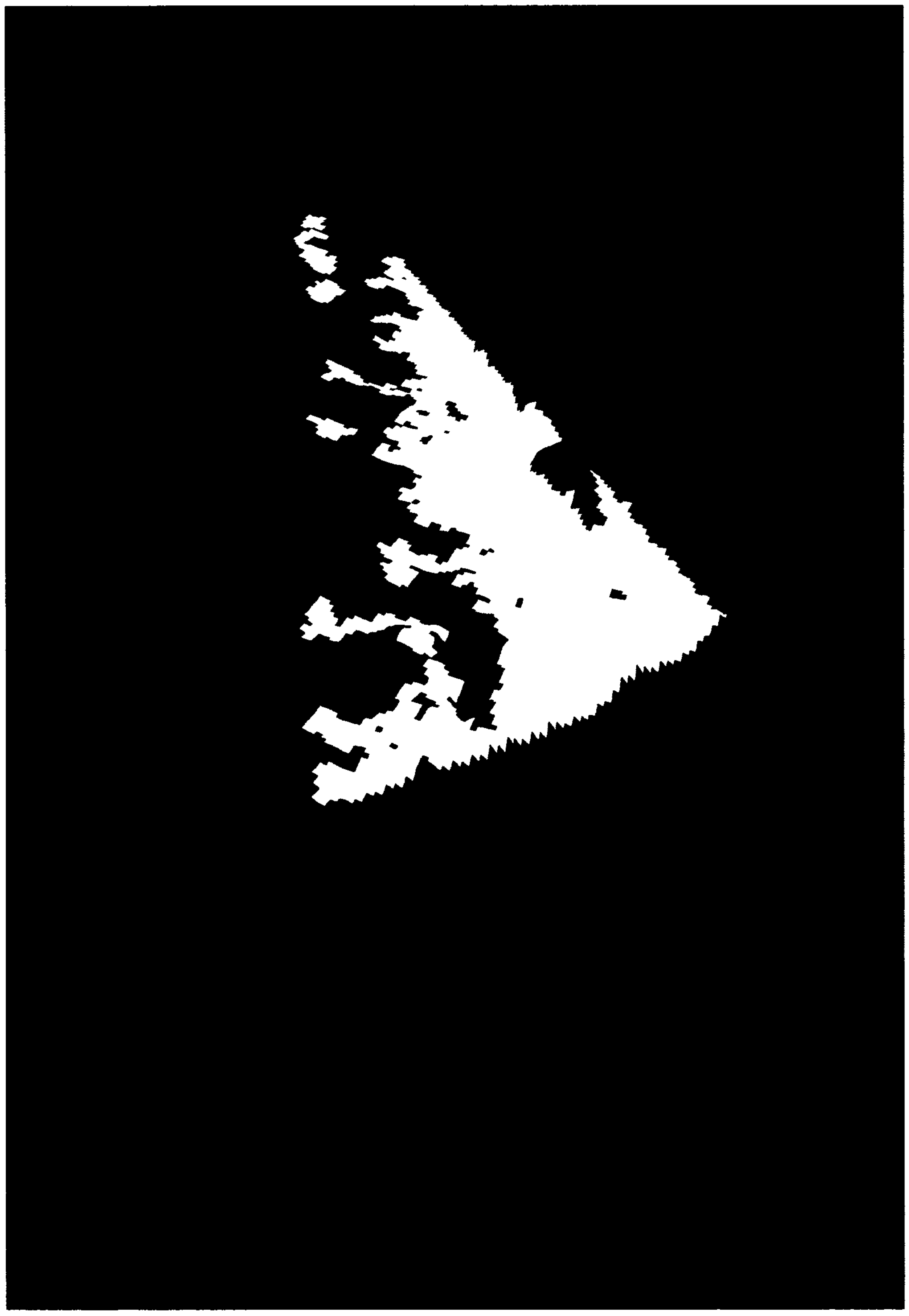


Figure 6.7: A void in some unusual terrain.

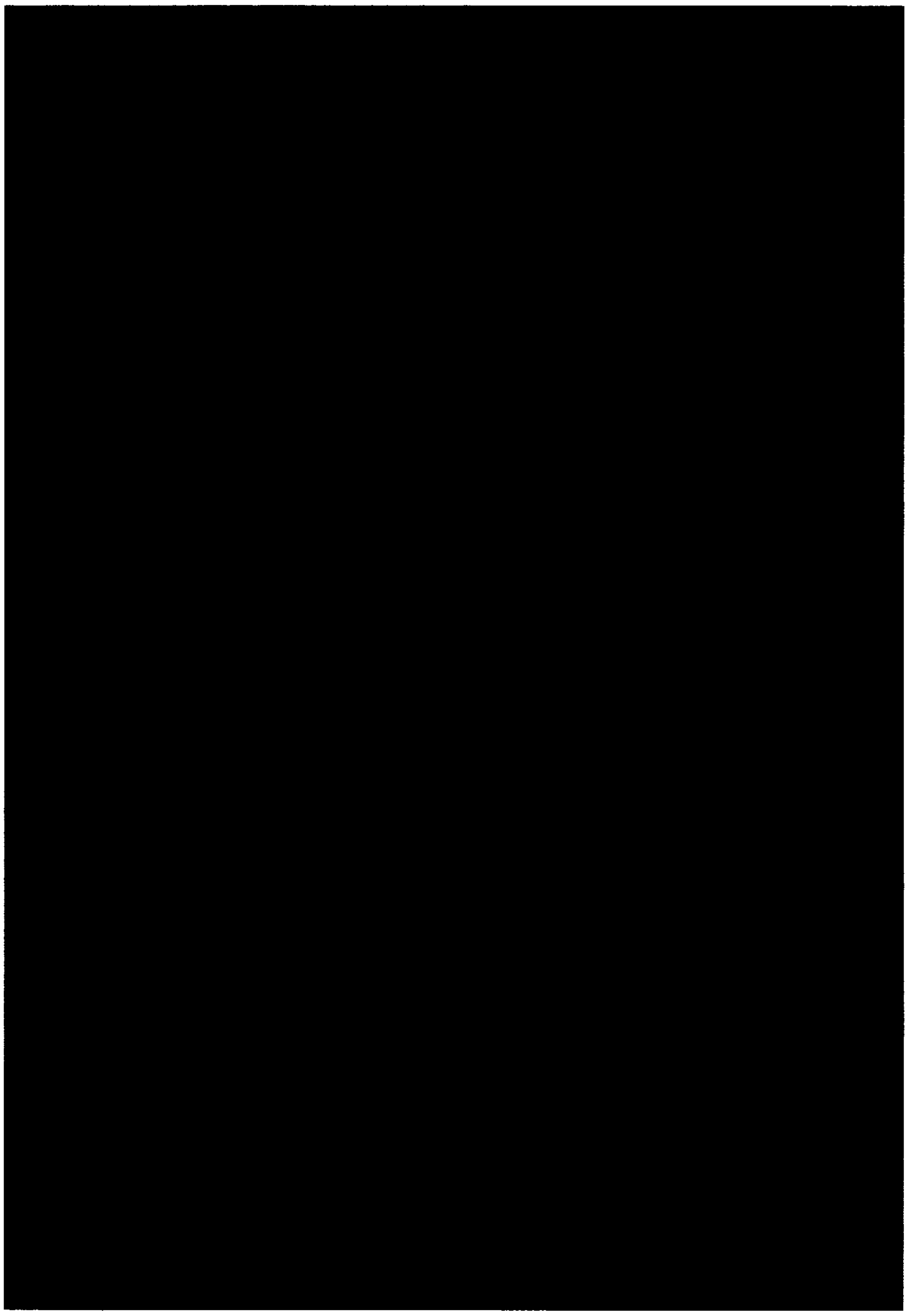


Figure 6.8: A patched for the void from Figure 6.1.3.

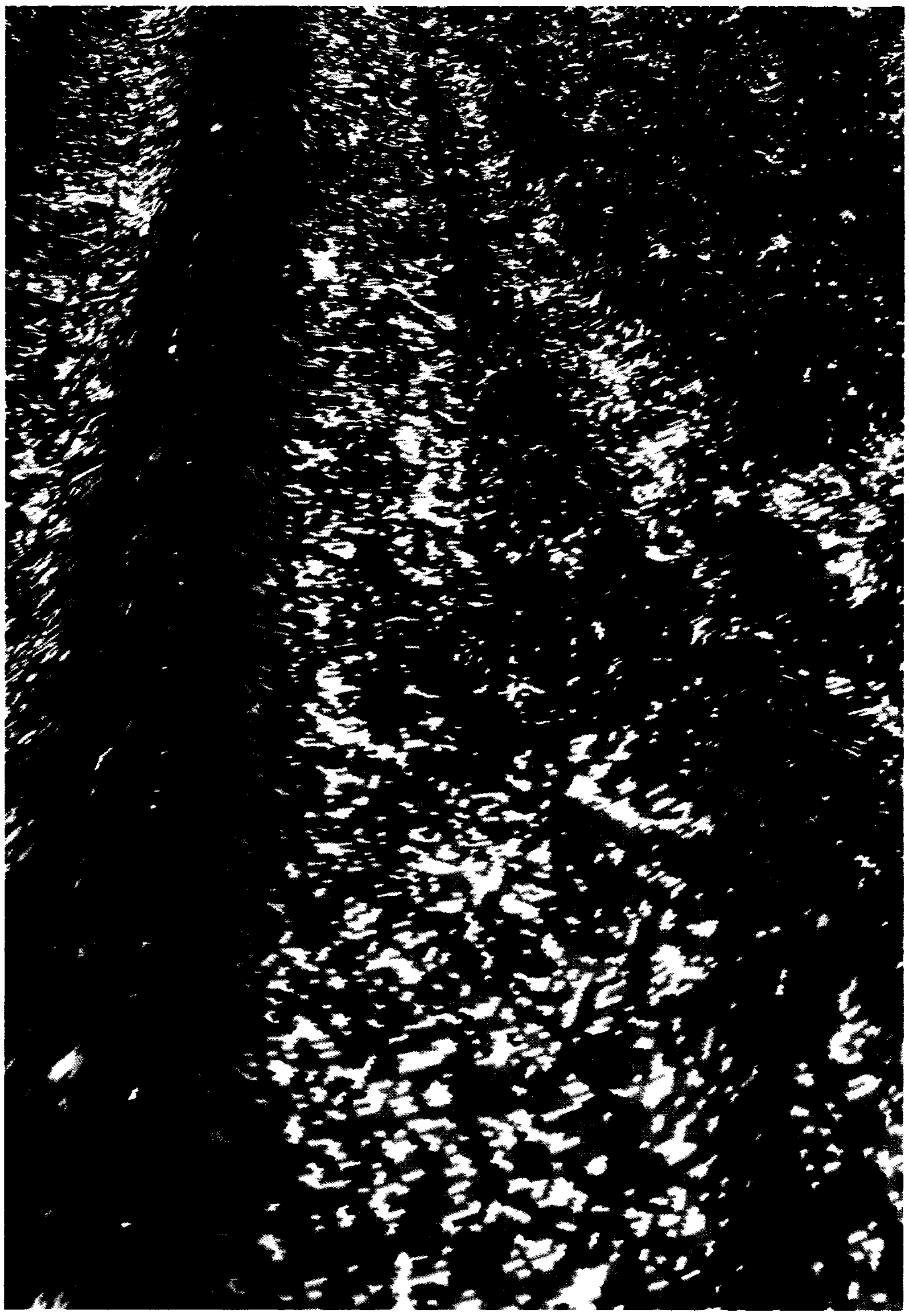


Figure 6.9: Terrain characteristic rendering for patched void from Figure 6.8.

Figure 6.10: An example steep void that has less characteristics. The first row illustrates the voids in the terrain. The second row illustrates the fixed voids with a phong shaded rendering. The third row illustrates the fixed voids with our characteristics rendering.

individually in terms of quality. We also needed to test the method's ability to significantly increase the number of images in a particular database. However, as the number of output images increases, our ability to individually consider each output image significantly decreases. Therefore, we decided to not generate the exhaustive set of possibilities. Rather, we generate a smaller subset that represents a good distribution of all possible combinations. To achieve this goal we exclude the component used as the base color from the selection process, reducing the possibilities to N^4, where N is the number of irises in the database. The component to be used for the base color information is then selected uniformly from the original set of irises that provides an even distribution of the color schemes amongst the generated irises. Next, we use only the unique combinations of the remaining four components, rather than the ordered permutations. This further reduces the number of images to the unique combinations of four iris images chosen from N. In our case, each database will contain 128 iris images. Even with the reduced number of possibilities, using our method on a database of this size will generate approximately 10 million iris images. To reduce this to a reasonable number, we separate each database into groups of eight images, and consider each group as a separate database. With each of these databases we obtain 70 new iris images from unique combinations of four irises from eight. Therefore, the total number of iris images synthesized is reduced to:

$$\frac{N}{8} \times 70, \tag{6.1}$$

or in our case: 1120 synthesized irises.

Next, the synthesized iris images were examined and rated according to the clarity and realism of the characteristics. We also used the uniqueness of the synthetic iris when compared with those used to create it. This visual process required that we learn the patterns and characteristics found in the real irises from the database in order to distinguish the sometimes subtle differences between two similar irises. Selecting which irises to use when creating a new iris is an important step in the process as it influences the quality of the resulting images. Results show that the classification process improves the overall quality of the synthetic irises, as seen in Table 6.1.

Figure 6.12 contains an example synthesized iris image with the five iris images used to create it. The synthesized image displays similar characteristics and color patterns to the real irises and thusly is a very realistic iris image. In addition, it is significantly different from each of the irises used to create it and therefore is a unique iris image that can be used in addition to the existing irises in the database.

In Figure 6.13, 6.14, and 6.15, we display a set of eight iris images taken from the UPOL database, alongside 16 iris images that were created from various combinations of the eight. Not only does the algorithm produce proper individual results but also it can significantly increase the number of unique iris images contained in the database, as the chart in Figure 6.16 demonstrates. Our implementation did not generate an exhaustive set of all possible irises for logistical reasons, yet still made a significant increase to the UPOL database by providing 1098 new unique and realistic iris images.

6.2.2 Usefulness for Iris Recognition Testing

The final test that we performed on our iris data set, was to submit them to an iris recognition algorithm. The goal of this test is to show that our synthetic iris

Database	Irises Created	High Quality Irises	Ratio
Complete UPOL	1120	1025	91.5%
Grouped UPOL	1120	1098	98.0%

Table 6.1: The quality results of the method with the Complete UPOL database as input, and with the Grouped UPOL database as input.

images exhibit the same properties as the existing iris images. We use the iris image recognition code provided by Masek and Kovesi [52]. Their system is based on the Daugman iris recognition algorithm [25]. The Daugman algorithm reduces each iris to a binary iris code that uniquely identifies each iris image. The Hamming distance between two iris codes is the number of positions for which the corresponding values are different. The system takes two input irises images and produces the Hamming distance that represents the differences between the irises. As described in Chapter 5, each of our irises represents a unique iris in the database. The comparisons between our irises and the others in both the existing and synthetic databases is considered to be an inter-class comparison. First, we ran the iris recognition algorithms on all of the inter-class comparisons in the existing UPOL database, and recorded the resulting hamming distances (Figure 6.17a) and (Table 6.2). Then, we ran the algorithm on all of the inter-class comparisons for the augmented UPOL database, and recorded the resulting hamming distances (Figure 6.17b) and (Table 6.2). As can be seen from the charts, the distribution of hamming distances for the existing database and our augmented database is nearly identical. Table 6.2 shows that there are some differences in the percentage of comparisons that fall into each range. The largest difference is in the 0.45 – 0.50 range, a 10% shift, although the average difference is only a few percent. This indicates that our synthetic images do not drastically

Range	Original UPOL	Augmented UPOL
0.00 - 0.05	4%	8%
0.05 - 0.10	2%	2%
0.10 - 0.15	2%	3%
0.15 - 0.20	4%	5%
0.20 - 0.25	5%	7%
0.25 - 0.30	7%	7%
0.30 - 0.35	9%	11%
0.35 - 0.40	15%	19%
0.40 - 0.45	29%	26%
0.45 - 0.50	22%	12%
0.50 - 0.55	1%	0%
0.55 - 0.60	0%	0%
0.60 - 0.65	0%	0%
0.65 - 0.70	0%	0%
0.70 - 0.75	0%	0%
0.75 - 0.80	0%	0%
0.80 - 0.85	0%	0%
0.85 - 0.90	0%	0%
0.90 - 0.95	0%	0%
0.95 - 1.00	0%	0%

Table 6.2: A table illustrating the percentage of inter-class comparisons in our testing of the synthesis method.

change the properties of the original database. Thus, our method provides good results for the purposes of testing iris recognition algorithms.

Figure 6.11: An example of an extremely large void that spans many hundreds of kilometers.

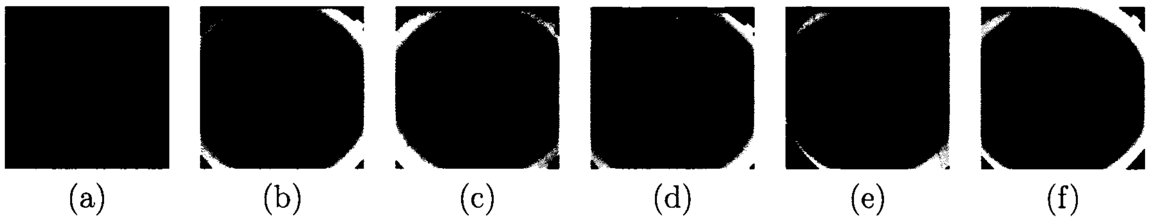


Figure 6.12: Synthesized iris beside real iris samples used to create it. (a) The synthesized image. (b) The base iris image, I_{base}. (c)(d)(e)(f) The irises used for details, D_1^{k-1}, D_2^{k-2}, D_3^{k-3}, and D_4^{k-4} respectively.

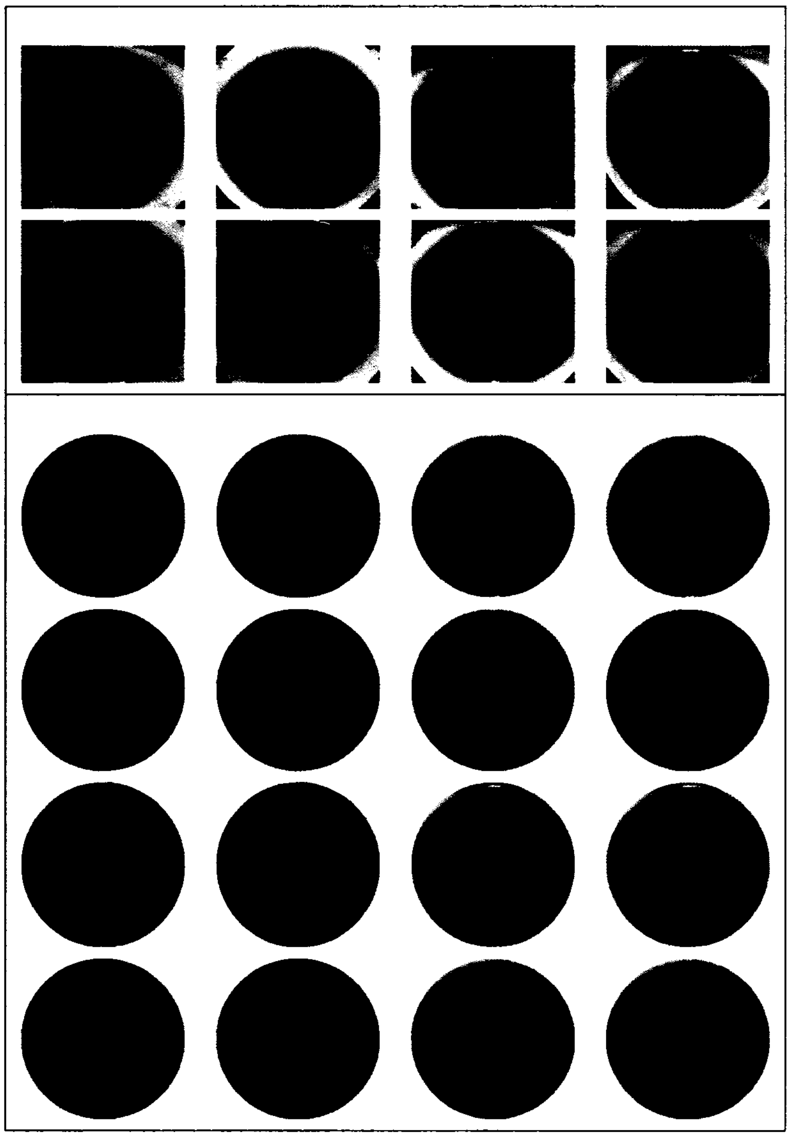


Figure 6.13: Eight original irises from the UPOL [27] database. Synthetic iris images generated by our algorithm in the bottom box by using combinations of the original eight.

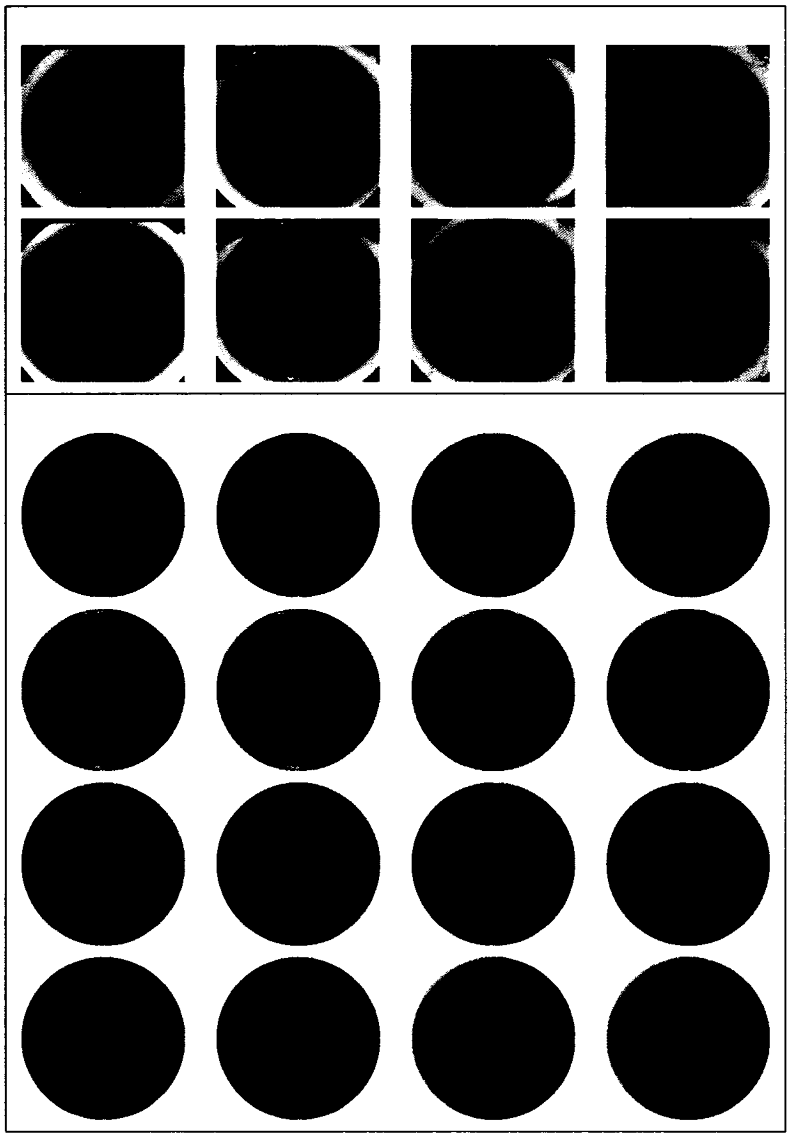


Figure 6.14: Eight original irises from the UPOL [27] database. Synthetic iris images generated by our algorithm in the bottom box by using combinations of the original eight.

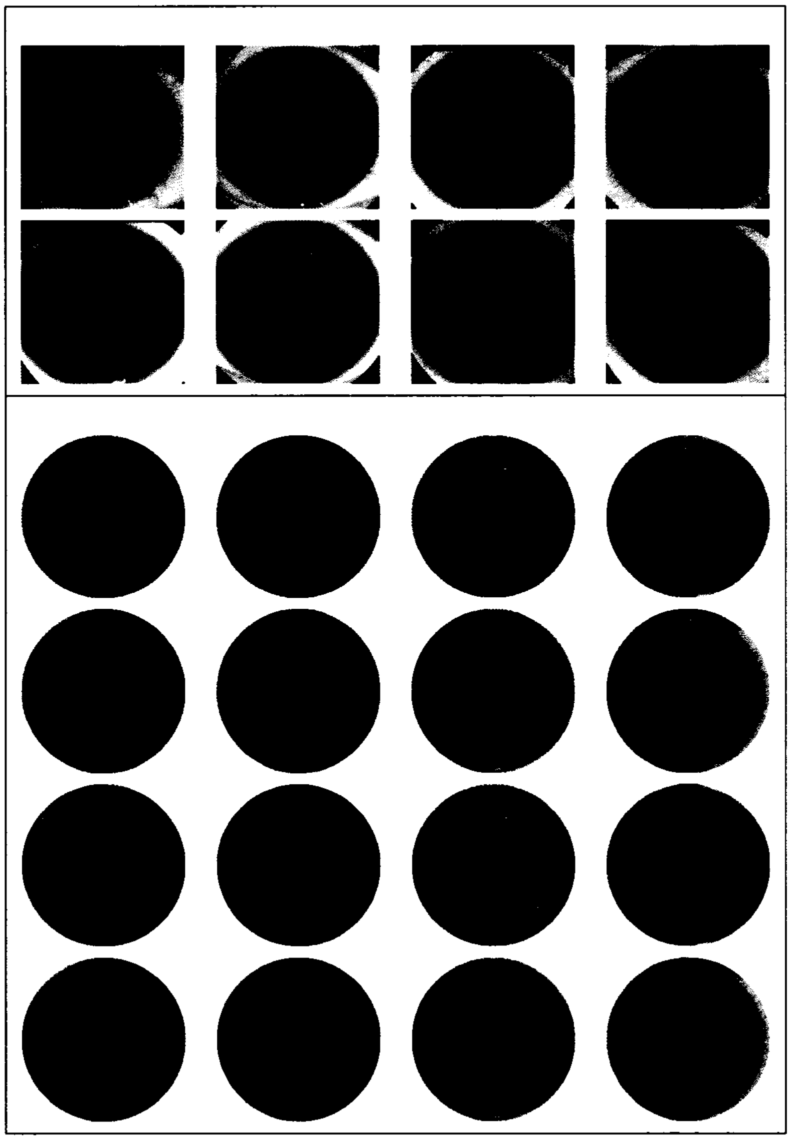


Figure 6.15: Eight original irises from the UPOL [27] database. Synthetic iris images generated by our algorithm in the bottom box by using combinations of the original eight.

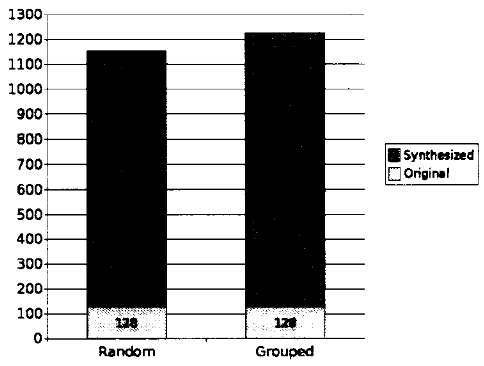


Figure 6.16: Original database size and the increase provided by our augmentation method.

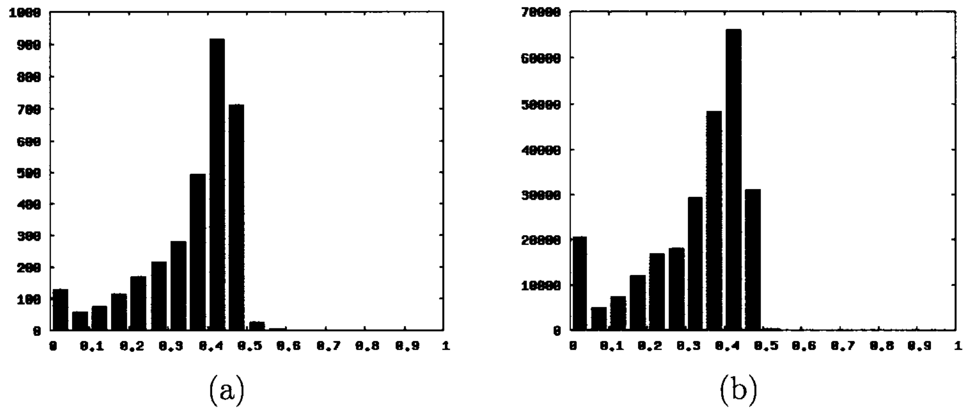


Figure 6.17: Hamming distances for inter-class comparisons of (a) the original UPOL database and (b) the UPOL database augmented with our irises.

Chapter 7

Conclusions and Future Work

In this thesis, we presented a multiresolution framework for the synthesis of digital models. We showed how the multiresolution details can be viewed as the characteristics of existing models. These details were used to synthesize new digital models with realistic properties. Reverse Chaikin subdivision filters were used to provide the multiresolution representation of our digital models. It has proven to be very effective at extracting characteristics from the digital models. Additionally the coarse representation that is given by reverse Chaikin subdivision can be used as the overall structure of the digital model. To demonstrate the usefulness of this method we developed two applications of data synthesis: terrain void repair and iris image synthesis.

7.1 Thesis Contributions

In the first application, we have presented a method for repairing the voids that are commonly found in DEMs. We presented a new technique to building smooth patches for the voids. The method uses a curve-based approach and has been effective at providing patches that are consistent with the surrounding terrain's structure. In addition, we have shown that reverse Chaikin subdivision filters are very effective at capturing the characteristics of terrain. These characteristics can be used to greatly improve the realism of the smooth patches, making them fit with the surrounding terrain. Finally, we have presented a new rendering method that can emphasize the characteristics of terrain.

In the second application, we have presented a novel method to augment existing iris image databases through iris image synthesis. We showed how the complete characteristics of an iris image can be extracted using multiple levels of reverse subdivision multiresolution. We developed a new method for iris image synthesis based on combining various levels of iris characteristics that were extracted from multiple real iris images. Next, we proposed criteria for classifying iris images in order to improve the results of the method. To validate the approach, we provided experimental results that demonstrate its effectiveness at synthesizing individual and realistic iris images and increasing the size of the UPOL iris image database. Finally, we showed that our synthetic irises are suitable for testing iris recognition methods.

The overall framework has been shown to be effective for the synthesis of digital models. The resulting models are very realistic, and exhibit the same characteristics as the original models. In addition, we have demonstrated that the framework can be easily adapted to fit the needs of particular applications.

7.2 Future Work

Investigation into deriving multiresolution filters that consider the smoothness of the coarse representation will be interesting as a future work. It is expected that these filters will provide a more efficient method of extracting the characteristics from existing objects. We also expect that the method will be applicable to any news areas that need to synthesize digital models with characteristics similar to existing objects.

For the terrain repair application, investigation into automated methods of validating the results should be performed. Additionally, the quality of the smooth patches might be improved if they are considered as a patch based problem. The SRTM data has a few areas where there are extremely large areas of missing data, in some cases spanning many hundreds of kilometers (Figure 6.11). In our implementation we can only consider terrain areas for which there is enough system memory to hold all the information at once. It would be interesting to investigate methods for fixing data sets that contain voids larger than the available system memory.

Our iris image method will benefit from development of algorithmic approaches to classifying sample irises, by allowing any user who is unfamiliar with iris images to easily extend their iris image databases. Although, the method's ability to significantly increase the size of a database reduces this requirement significantly. In order to better test iris recognition algorithms, a number of image transformation steps can be added to our post processing stages. These steps, when combined with iris scaling algorithm, can be used to produce multiple samples of the same iris with varying rotations, pupil sizes, and noise levels. This will allow our method to produce image databases similar to the UBIRIS database where multiple samples of varying quality exist for each iris. As a final step, we should further investigate automated iris-recognition algorithm testing of our synthetic results. Not only should the general statistical properties be taken into account, but it would be interesting to study the effect that each synthesis component has on the comparison between the synthetic iris and those used to create it.

Bibliography

[1] A. Aldroubi, M. Eden, and M. Unser. Discrete spline filters for multiresolutions and wavelets of l_2. *SIAM Journal on Mathematical Analysis*, 25(5):1412–1432, 1994.

[2] M. Ashikhmin. Synthesizing natural textures. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226, New York, NY, USA, 2001. ACM Press.

[3] G. Baraquet and M. Sharir. Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design*, 12(2):207–229, March 1995.

[4] R. H. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kauffman, 1987.

[5] R. H. Bartels, G. H. Golub, and F. F. Samavati. Some observations on local least squares. *BIT Numerical Mathematics*, 46:455–477, 2006.

[6] R. H. Bartels and F. F. Samavati. Reversing subdivision rules: Local linear conditions and observations on inner products. *Journal of Computational and Applied Mathematics*, 119(1-2):29–67, 2000.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[8] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2:II–707–12 vol.2, 18-20 June 2003.

[9] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Computer Graphics*, pages 361–368. ACM SIGGRAPH, 1997.

[10] J. Brosz. Terrain modeling by example. Master's thesis, University of Calgary, Calgary, Alberta, August 2005.

[11] J. Brosz, F. F. Samavati, and M. Costa Sousa. Terrain synthesis by-example. In *International Conference on Computer Graphics Theory and Applications.* In Association with Eurographics, 2006.

[12] M. Brunn, M. Costa Sousa, and F. F. Samavati. Capturing and re-using artistic styles with reverse subdivision-base multiresolution methods. Accepted 2004.

[13] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 31(4):532– 540, Apr 1983.

[14] R. Cappelli, D. Maio, and D. Maltoni. Synthetic fingerprint-database generation. In *Proc. 16th International conference on Pattern Recognition*, volume 3, pages 744–747, 2002.

[15] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C.

McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM Press, 2001.

[16] CASIA. Casia iris image database. In *http://www.sinobiometrics.com*, 2004.

[17] G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics Image Processing*, pages 346–349, 1974.

[18] C. K. Chui and E. Quack. Wavelets on a bounded interval. *Numerical methods in approximation theory*, 9(105):53 - 75, 1994.

[19] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu. A finite element method for surface restoration with smooth boundary conditions. *Computer Aided Geometric Design*, 21(5):427–445, 2004.

[20] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *Proceedsing of SIGGRAPH 2003*, 2003.

[21] M. Costa Sousa, K. Foster, and B. Wyvill. Precise ink drawing of 3d models. *Special issue of Computer Graphic Forum*, 22(3), 2003.

[22] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2:II–721–II–728 vol.2, 18-20 June 2003.

[23] J. L. Crowley. A representation for visual information. Technical report, November 1981.

[24] J. Cui, Y. Wang, J. Huang, T. Tan, and Z. Sun. An iris image synthesis method based on pca and super-resolution. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, pages 471 – 474, 2004.

[25] J. Daugman. How iris recognition works. *IEEE Trans, CSVT*, 14:21–30, 2004.

[26] J. Davis, S.R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *First International Symposium on 3D Data Processing Visualization and Transmission*, pages 428–861, 2002.

[27] M. Dobeš and L. Machala. Iris database. In *http://www.inf.upol.cz/iris/*, 2005.

[28] S. Dowding, T. Kuuskivi, and X. Li. Void fill of srtm elevation data - principles, processes and performance. In *ASPRS Images to Decision: Remote Sensing Foundation for GIS Applications*, 2004.

[29] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM Trans. Graph.*, 22(3):303–312, 2003.

[30] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, August 2001.

[31] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.

[32] A. Finkelstein and D. H. Salesin. Multiresolution curves. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 261–268, New York, NY, USA, 1994. ACM Press.

[33] K. Foster, M. Costa Sousa, F. F. Samavati, and B. Wyvill. Reverse subdivision multiresolution for polygonal silhouette error correction. In *Lecture Notes in Computer Science 3045*, pages 247–259, 2004.

[34] M. L. Gavrilova. Algorithms in 3d real-time rendering and facial expression modeling. In *3IA'2006 Plenary Lecture, Eurographics*, pages 5–8, May 2006.

[35] M. L. Gavrilova. Computational geometry and image processing techniques in biometrics: on the path to convergence. In *Image Pattern Recognition: Synthesis and Analysis in Biometrics.* World Scientific Publishers, December 2006.

[36] A. Grossman and J. Morlet. Decomposition of functions into wavelets of constant shape, and related transforms. *Mathmatics and Physics: Lectures on Recent Results*, 1985.

[37] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, New York, NY, USA, 2001. ACM Press.

[38] Y. Huang, S. Luo, and E. Chen. An efficient iris recognition system. In *Proc. of the IEEE*, pages 1348–1363, 1997.

[39] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5):363–370, 1984.

[40] J. J. Koenderink and A. J. Doorn. Visual detection of spatial contrast; influence of location in the visual field, target extent and illuminance level. *Biological*

Cybernetics, 30(3):157–167, 1978.

[41] Jan J. Koenderink and A. J. Doorn. The structure of two-dimensional scalar fields with applications to vision. *Biological Cybernetics*, 33(3):151–158, 1979.

[42] J. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 34–46. IEEE, 1975.

[43] A. Lefohn, R. Caruso, E. Reinhard, B. Budge, and P. Shirley. An ocularist's approach to human iris synthesis. *IEEE Computer Graphics*, pages 70–75, 2003.

[44] B. Li, Y. Qi, and X. Shen. An image inpainting method. *Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on*, pages 6 pp.–, 7-10 Dec. 2005.

[45] L. Liang, C. Liu, Y. Xu, B. Guo, and H. Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.

[46] P. Liepa. Filling holes in meshes. In *SGP '03: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 200–205. Eurographics Association, 2003.

[47] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.*, 16(1):34–73, 1997.

[48] Y. Luo and M. Gavrilova. 3d facial model synthesis using voronoi approach. In *IEEE-CS proceedings, ISVD*, pages 132–137, July 2006.

[49] T. Lyche and K. Mørken. Spline-wavelets of minimal support. *Numerical methods in approximation theory*, 9(105):177–194, 1994.

[50] S. G. Mallat. Multiresolution representation and wavelets. Master's thesis, University of Pennsylvania, 1988.

[51] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11:674 – 693, 1989.

[52] L. Masek and P. Kovesi. Matlab source code for a biometric identification system based on iris patterns. *The School of Computer Science and Software Engineering, The University of Western Australia*, 2003.

[53] Y. Meyer. Wavelets - Algorithms and applications. *Applied Mathematics*, 1993.

[54] A. N. Netravali and G. G. Haskell. *Digital pictures: Representation, Compression, and Standards.* Plenum Press, New York, 2nd edition, 1995.

[55] D. Nie, L. Ma, and S. Xiao. Similarity based image inpainting method. *Multi-Media Modelling Conference Proceedings, 2006 12th International*, pages 4 pp.–, 4-6 Jan. 2006.

[56] H. Proença and L. A. Alexandre. Ubiris: A noisy iris image database. In *Proc. 13th International conference on image Analysis and Processing*, 2005.

[57] E. Quack and N. Weyrich. Decompision and reconstruction algorithms for spline wavelets on a bounded interval. *Applied and Computational Harmonic Analysis*, 1(3):217 – 213, 1994.

[58] R. Sabourin, S. E. N. Correia, and J. M. de Carvalho. On the performance of wavelets for handwritten numerals recognition. In *Proc. 16th International conference on Pattern Recognition*, volume 3, pages 127–130, 2002.

[59] F. F. Samavati and R. H. Bartels. Multiresolution curve and surface representation by reversing subdivision rules. *Computer Graphics Forum*, 18(2):97–120, 1999.

[60] F. F. Samavati, R. H. Bartels, and L. Olsen. Local b-spline multiresolution with examples in iris synthesis and volumetric rendering. In *Synthesis and Analysis in Biometrics*. World Scientific Publishing, December 2006.

[61] F. F. Samavati, N. Mahdavi-Amiri, and R. H. Bartels. Multiresolution representation of surface with arbitrary topology by reversing doo subdivision. *Computer Graphic Forum*, 21(2):121–136, 2002.

[62] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Trans. Graph.*, 23(3):878–887, 2004.

[63] E. Stollinitz, T. Derose, and D. Salesin. *Wavelets for Computer Graphics.* Morgan Kaufmann, San Francisco, 1996.

[64] J. Sun, L. Yuan, J. Jia, and H. Shum. Image completion with structure propagation. In *SIGGRAPH '05: Proceedings of the 31st annual conference on Computer graphics and interactive techniques.* ACM Press, 2005.

[65] T. Taerum, M. Costa Sousa, F. F. Samavati, S. Chan, and R. Mitchel. Real-time super resolution contextual close-up of clinical volumetric data. In *Proc.*

Of Eurographics/IEEE-VGTC Symposium on Visualization. IEEE, 2006.

[66] M. A. Unser. Ten good reasons for using spline wavelets. In *Proc. SPIE, Wavelet Applications in Signal and Image Processing V*, volume 3169, pages 422–431, Oct 1997.

[67] USGS. Shuttle radar topography mission (srtm) faq. *http://seamless.usgs.gov/website/seamless/faq/srtm_faq.asp*, Last visited January 27th, 2005:Last updated October 28th, 2004.

[68] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro. Inpainting surface holes. *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 2:II–903–6 vol.3, 14-17 Sept. 2003.

[69] J. M. Van Verth and L. M. Bishop. *Essential Mathematics for Games and Interactive Applications.* Morgan Kaufmann, San Francisco, 2004.

[70] C. Wang, Y. Luo, M.L. Gavrilova, and J. Rokne. Fingerprint image matching using a hierarchical approach. In *Computational Intelligence in Information Assurance and Security.* Springer SCI Series, To Appear, 2007.

[71] A. Watt. *3D Computer Graphics.* Addison-Wesley Publishing Ltd., 3 edition, 2000.

[72] J. Wayman, A. Jain, D. Maltoni, and D. Maio. *Biometric Systems: Technology, Design and Performance Evaluation.* Springer, 2005.

[73] L. Wecker, F. F. Samavati, and M. Gavrilova. Iris synthesis: A multiresolution approach. In *Proc. Graphite 2005*, December 2005.

[74] L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[75] R. Wildes and J. Asmuth. A system for automated iris recognition. In *Proc. of the Second IEEE Workshop on Application of Computer Vision*, pages 121–128, 1994.

[76] R.P. Wildes. Iris recognition: an emerging biometric technology. In *Proceedings of the IEEE*, volume 85, pages 1348 - 1363, 1997.

[77] L. H. Yang, L. Feng, and Y. Y. Tang. A wavelet approach to extracting contours of document images. In *Proc. of the 5th Int'l Conference on Document Analysis and Recognition*, pages 71–74, 1999.

[78] S. Yanushkevich, P. Wang, S. Srihari, and M. Gavrilova. *Image Pattern Recognition: Synthesis and Analysis in Biometrics.* World Scientific Publishers, December 2006.

[79] M. Yasuda, J. Ohkubo, and K. Tanaka. Digital image inpainting based on markov random field. *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, 2:747–752, 28-30 Nov. 2005.

[80] S. Zelinka and M. Garland. Jump map-based interactive texture synthesis. *ACM Trans. Graph.*, 23(4):930–962, 2004.

[81] Y. Zhu, T. Tan, and Y. Wang. Biometric personal identification based on iris patterns. In *Proc. of 15th Int'l conference on Pattern Recognition*, volume 2, pages 801–804. IEEE, 2000.

Appendix A

Software Systems

In order to test our synthesis method, we built software for both application areas. For the first application area, we built a program which can demo the repair process for the DEMs from the SRTM mission (Section A.1). For the second application area, we built a number of tools for use in iris synthesis (Section A.2). We used C++ as the primary development language, and Python for scripting [1]. All of the user interface are done with Qt4 [2], the Trolltech UI framework. For rendering purposes, we used the high-performance object-oriented rendering framework called Ogre3D [3]. Finally, for image loading and saving we used the cimg library [4].

A.1 Terrain Repair Interface

The main interface for the Terrain Repair Software shows the program options on the left and a high level view of the terrain for the continental united states on the right (Figure A.1). The user may move the terrain by clicking and dragging with the left mouse button. In addition, the user may zoom into the terrain to see the terrain from a zoomed in view which displays each height sample as one pixel on the screen. There is a blue square in the center of the view which represents the area which will be shown if the user switches to the 3D view. The user can switch to the

[1]http://www.python.org/
[2]http://www.trolltech.com/products/qt
[3]http://www.ogre3d.org/
[4]http://cimg.sourceforge.net/

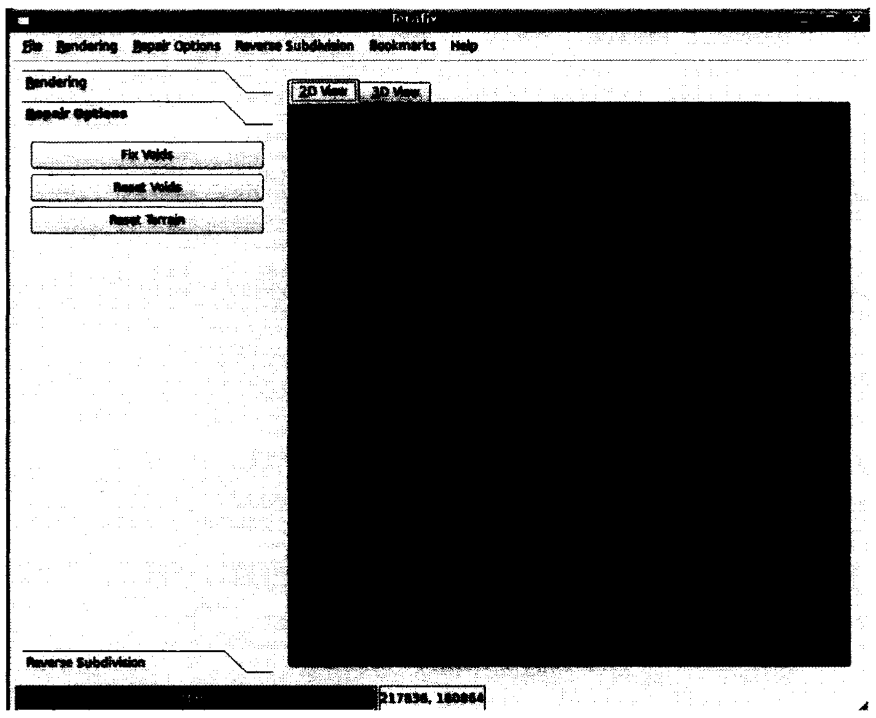


Figure A.1: Terrain repair main interface.

3D view by selected the 3D view tab above the main view panel (Figure A.2 (b)).

The area of terrain which is included in the program is large. To facilitate navigation of this terrain and returning to previously visited areas, the user can bookmark certain areas of the terrain (Figure A.3).

Finally, on the left of the program are three option panels which allow the user to interact with the program. First, there is a rendering option panel where the user can select the desired rendering style for the terrain. Second, there is a repair option panel where the user can repair the terrain, and reset it as need be. Third, there is a reverse subdivision panel where the user can use RS to decompose the terrain or view its details with either Haar or Chaikin reverse subdivision filters.

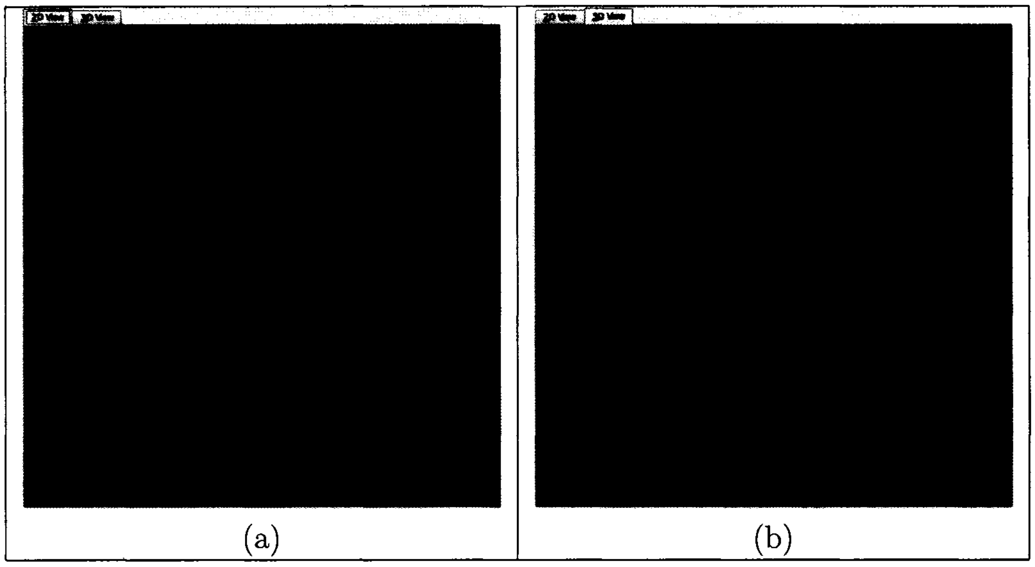


Figure A.2: (a) Terrain repair 2D interface zoomed in. The user can pan by dragging with the left mouse button. The user can zoom out with the scroll button. (b) The user can move the terrain by dragging with the right mouse button. The user can rotate the terrain by dragging with the left mouse button. The user can zoom with the scroll button.

A.2 Iris Image Interface

In the iris image software, there was no need for a GUI application as a most operating systems provide image viewing capabilities. This software uses cimg for all of its loading and saving routines, which means that it supports many image formats, most notably JPG and PNG. For this system we built a command line program to do the main synthesis method. This program takes 6 parameters:

- The output image name.
- The input image for the base color I_5.
- The input image for the 4th level of details I_4.
- The input image for the 3th level of details I_3.

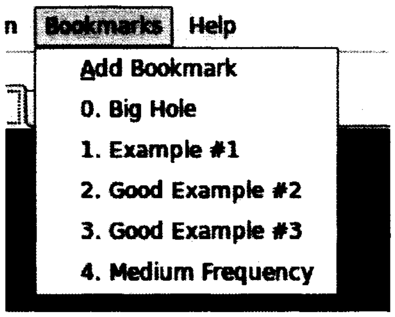


Figure A.3: The user can bookmark certain positions and return to them at a later date.

- The input image for the 2th level of details I_2.

- The input image for the 1th level of details I_1.

The interface to this program is somewhat tedious to use when generating large amounts of iris images. Therefore, we also built a python script to make the decisions of which iris images to combine. The script is given a list of iris images, and it will perform the iris image selection algorithm described in Section 6.2.1. Then the python script will invoke the command line program to perform the synthesis using the selected iris images.

A.3 Implementation Details

A.3.1 Class Structures

The application was designed in an object-oriented paradigm (OOP). Where core features are described with an interface and the concrete implementation of features is provided in the child classes. The class hierarchy is shown in Figure A.6 as a UML diagram. There are six main areas of functionality where Iris Synthesis, Ter-

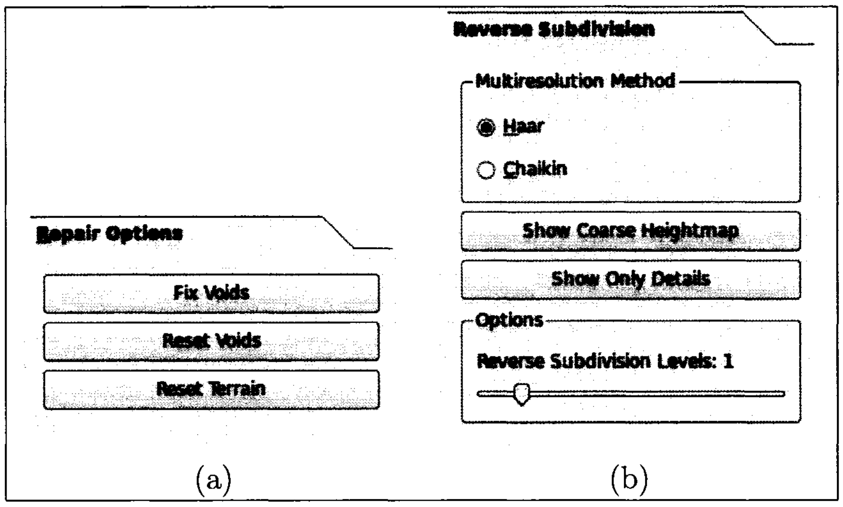


Figure A.4: (a) The repair option panel allows the user to perform the repair, reset the voids and reset the entire terrain. (b) The reverse subdivision panel allows the user to decompose the terrain with Chaikin and Haar RS filters.

rain Repair and Rendering provide the application specific features and Iteration, Multiresolution, and File operations provide the common features.

All of the classes are templated in order to provide generic implementation regardless of the base data type. This allows us to use the same implementation of features for both applications.

Iteration

The main data structure in our system is **TPSurface** which encapsulates a tensor product surface. It is composed of a two-dimensional array of values which represent the color of the iris or the height of the terrain. In the case of a DEM single floating point values are stored at each point. In contrast, an iris image contains three floating point values as at each point, one for each of red, green and blue respectively. However, for the purposes of most of our algorithms the necessary operations are the same: *getXSize*, *getYSize*, *getValue* and *setValue* and center around iteration

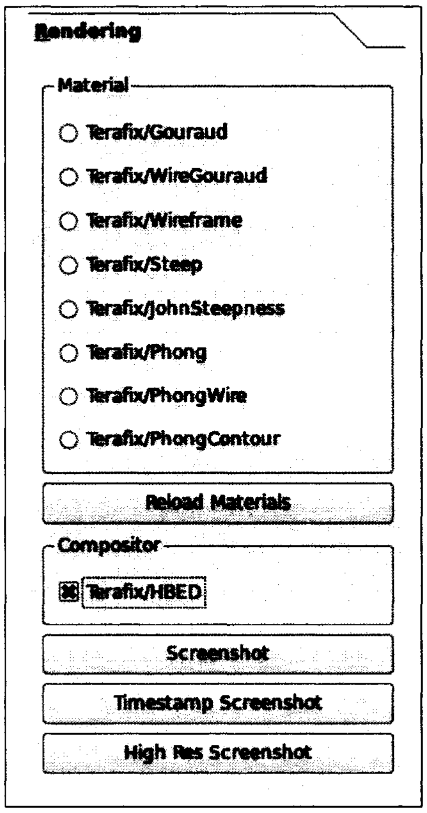


Figure A.5: The rendering option panel allows the user to choose the rendering styles, and take screen shots.

concepts.

As discussed in Chapters 2 and 3 the Chaikin multiresolution scheme that we use is based on curves. Therefore we have **Curve** class which exposes the usual operations: *size*, *getValue* and *setValue*. The **UCurve**, **VCurve**, **WCurve**, and **TCurve** provide the necessary implementations of these operations for iterating over the u, v, w, and t curves of a **TPSurface** respectively. In some situations, it is desirable to iterate over only a subsection of a curve and these operations are implemented in the **USubCurve**, **VSubCurve**, **WSubCurve**, and **TSubCurve** respectively. It is important to note that for all of the **Curve** specializations are associated with a **TPSurface** and the *getValue*, *setValue* operations retrieve and set the value within that surface.

Finally, the **CurveSet** class encapsulates the necessary operations to describe the number of curves in a curve set for a **TPSurface** and to obtain a reference to those curves. **UCurveSet**, **VCurveSet**, **WCurveSet**, and **TCurveSet** provide the actual implementations for each curve direction.

Multiresolution

The main multiresolution operations are based on curves, as described in [60] and are encapsulated in **CurveMR**. They include *decomposition* and *reconstruction* of a curve. **ChaikinMR** and **HaarMR** provide the implementations for the Chaikin and Haar subdivision schemes respectively. As described in [60] multiresolution operations for tensor product surface can be described in terms of curves. Therefore the **TPSurfaceMR** class can implement the multiresolution operations for a **TPSurface** generically and will use the provided **CurveMR** to actually perform the calculations.

File Input and Output

File input and output operations are defined by the **TPSurfaceIO** and implemented by the **PngIO**, **JpgIO**, and **HgtIO** for the PNG, JPG and HGT file formats respectively.

Iris Synthesis

The iris synthesis applications contains a number of operations that are specific to it. First and foremost the pre-processing steps are encapsulated in the **IrisStretch** and **PolarTransform** respectively. The **ChooseIrises** class encapsulates the iris component selection process for each synthetic iris. Finally, the **BuildIris** class encapsulates the operations of extracting components from existing irises and combining them to form the synthetic irises.

Terrain Repair

The terrain repair applications also contains operations that are specific to it. From a high level, the **Repair** class encapsulates the entire repair process and delegates specific portions to the appropriate classes. The first step in the terrain process involves the **SmoothPatch** class which uses the **Hermite** class to provide an initial patch for the void. In the second step the **FindDetails** is first used to find details from terrain surrounding the void and the **CopyDetails** class is used to extract and apply these details to the smooth patch. In the final step the **CombineSolutions** class combines each of the curve set's intermediate solutions into the final solution for each void.

Rendering

Rendering duties are handled by an excellent open source engine: **Ogre3D**. Using it abstracts many of the oddities of cross-platform graphics rendering. Specifically, it allows easy cross-platform usage of the OpenGL shading language, which we used to implement our rendering algorithms. Although the **Contour**, **Wireframe**, **Details**, and **Phong** are represented as classes in Figure A.6 they are actually OpenGL Shading scripts which are uploaded to the graphics card at runtime. The **2D** class is an implementation of a 2D rendering style that converts heightmaps to Qt4 compatible images which could then be easily displayed.

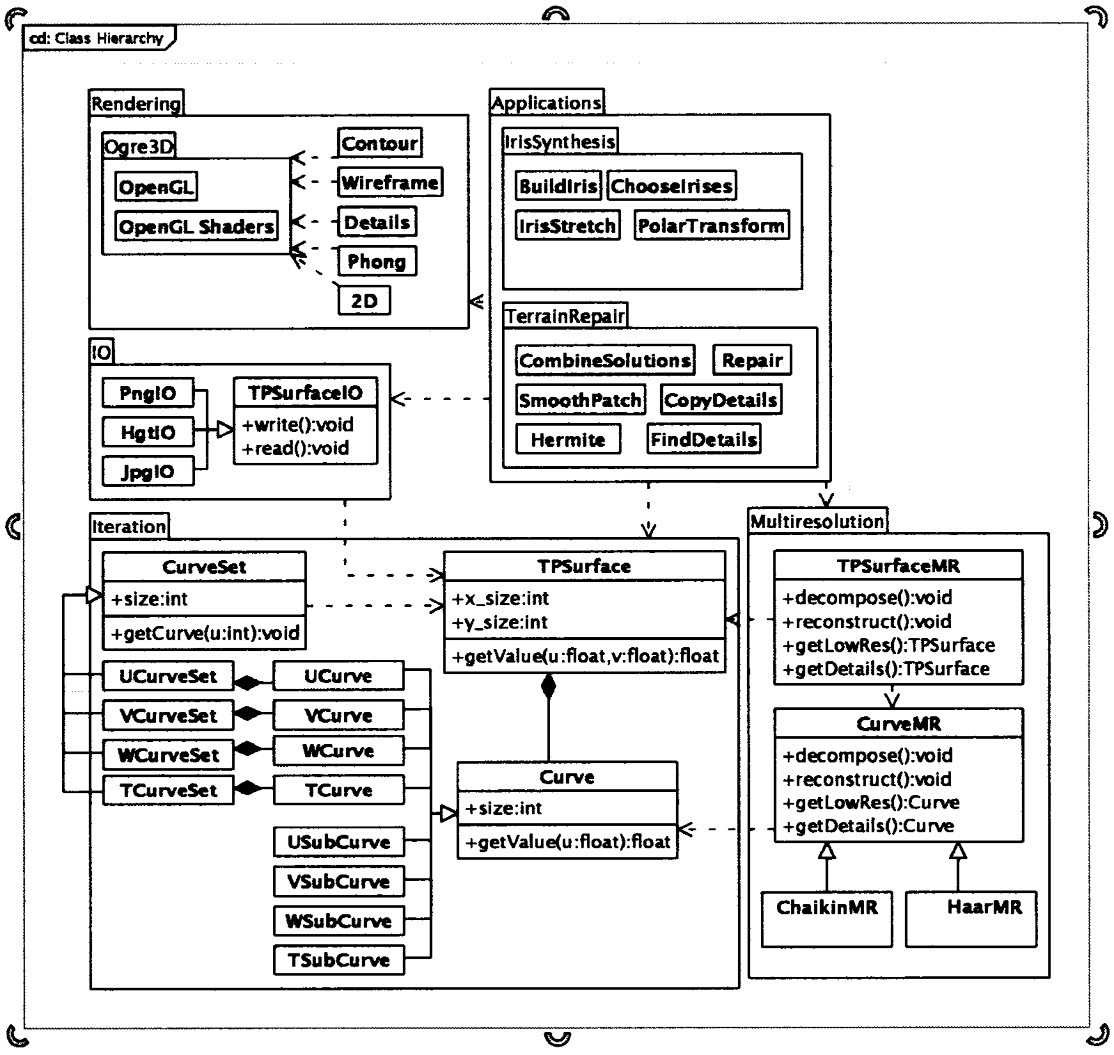


Figure A.6: A UML diagram which illustrates the class hierarchy of our implementation.