Surface reconstruction using fractal priors

Paul Duree

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

UniS

Centre for Vision, Speech and Signal Processing
School of Electronic Engineering, Information Technology and Mathematics
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

August 2000

© Paul Duree 2000
Summary

In oil exploration, changes in soil depth or thickness of a rock are indicators for the presence of the so-called "seismic horizons" that identify the possible presence of oil. The data concerned are obtained by making measurements at randomly distributed sparse points. It is of interest to reconstruct the full model of the surface of the rock or the terrain, from the knowledge of the few sparse data points. This reconstruction cannot be achieved by using ordinary interpolation methods, as these methods assume that the reconstructed surface is smooth. Instead, a fractal prior model for the terrain has to be assumed. A constraint fractal formation then follows, with the constraints being the data points available. The dimension of the fractal used is inferred from the data points that are available, on the basis of the assumption that a fractal model applies and from the fact that a fractal exhibits the same properties at all scales.

Several tools for the creation of artificial fractals of varying degrees of roughness are used to give a wide range of data for the reconstruction experiments. A tool to measure the fractal dimension of a surface, or a set of sparse data points, is an important part of the reconstruction process. Several methods of fractal dimension measurement are developed and thoroughly tested with many different surfaces. The reliability of the dimension calculation and how this changes with different levels of sparsity is investigated.

Both tools are then modified to enable the production and measurement of anisotropic fractals — fractals with different levels of roughness in different directions. These sorts of fractal surfaces have received little or no attention in the literature and fractal reconstructions using prior knowledge of the anisotropy have not been done before.

Several different versions of the fractal reconstruction method are developed and the control of the dimension of the reconstructed surface is carefully investigated. Example reconstructions are then presented, using both artificial and real fractals. The subsampling of the data is performed both at random and in regular patterns and the reconstruction is forced to extrapolate from as well as interpolate between
the data points.

Finally the reconstruction method is modified to incorporate knowledge of any anisotropy in the fractal surface. The method is tested on both real and artificial data and shows significant advantages over the regular isotropic reconstruction.
To my father.
Acknowledgements

I would like to thank my supervisor Prof. Maria Petrou for providing the opportunity to do this work and for her continuous support, advice and encouragement. I would also like to thank the members — past and present — of the Centre for Vision, Speech and Signal Processing for their help and support. Thanks are also due to BG Technology who sponsored the work, and provided some real data for the experiments. Finally I would like to thank all my friends and family for their support and patience.
Contents

1 Introduction 1

2 Literature survey 5
  2.1 Surface interpolation .................................................. 5
  2.2 Fractals ........................................................................ 8
  2.3 Fractal reconstruction .................................................. 11

3 Tools for fractal generation 13
  3.1 Fractional Brownian motion ......................................... 14
  3.2 Random midpoint displacement .................................... 15
  3.3 Successive random additions ....................................... 17
  3.4 Fast Fourier transform filtering ................................... 18
    3.4.1 Anisotropic FFT filtering fractal ......................... 20

4 Tools for fractal dimension calculation 25
  4.1 Fractal dimension calculation using FFT ....................... 27
    4.1.1 Reliability of the fractal dimension calculation using FFT . 28
  4.2 Fractal dimension calculation using difference statistics .... 34
    4.2.1 Reliability of the fractal dimension calculation using difference statistics .... 35
4.2.2 Windowed difference statistics .................................................. 38

4.3 Anisotropic fractal dimension calculation ............................................ 41

4.3.1 Anisotropic fractal dimension calculation using FFT ...................... 41

4.3.2 Anisotropic fractal dimension calculation using difference statistics .................................................. 42

5 Constrained fractal reconstruction ...................................................... 45

5.1 Surface reconstruction ........................................................................ 45

5.2 The faithfulness to the data term of the cost function ......................... 47

5.3 The prior model term of the cost function ......................................... 48

5.4 Minimising the cost function ............................................................ 52

5.5 Fractal surface reconstruction using a fractal prior .............................. 56

5.5.1 Producing a constrained fractal .................................................. 58

5.6 Multiresolution reconstruction .......................................................... 63

5.6.1 Multigrid reconstruction .............................................................. 65

5.6.2 Relative multilevel reconstruction ............................................. 66

5.7 Controlling the fractal dimension of the reconstruction .................... 69

6 Study of the robustness of the method .................................................. 83

6.1 Reconstructing artificial fractals of different dimensions ............ 84

6.2 Reconstructing a real surface ........................................................... 90

6.3 Reconstructing a real surface with a mask ............................................. 96

7 Anisotropic fractal reconstruction ....................................................... 105

7.1 Introducing anisotropy to the fractal reconstruction technique ........ 105

7.2 Anisotropic reconstruction experiments ............................................ 108
Contents

8 Conclusions and future work 115

A The random addition to midpoint displacement 119

B The matrix form of the energy equation 121
Chapter 1

Introduction

The aim of this thesis is to use "constrained fractals" to reconstruct natural terrain from a sparse set of height data points. With sparse data, surface reconstruction involves filling in the gaps in the data with values as close as possible to the real missing values.

Most methods of surface reconstruction aim to satisfy some given smoothness constraint. For example, if the noise is assumed to be additive Gaussian and the smoothness constraint is that of minimum local gradient magnitude, we have the "membrane" reconstruction obtained by minimising an appropriate energy function. If the constraint is that of minimum local curvature, then a "thin plate" reconstruction is produced.

However, these sorts of constraint would give an unnatural appearance to a reconstructed surface. The surface would be far too smooth, displaying none of the rough, random texture characteristic of natural terrain. This is because the membrane or thin plate models are not appropriate models for land surface. Less importantly, the noise model may also be not of the right type for remotely-sensed data.

A model suitable for describing natural surfaces is based on Mandelbrot’s fractal geometry [31]. In particular, fractional Brownian motion produces surfaces ("fractals") that show the same statistical characteristics over a range of scales — this stochastic self-similarity being a property observed in natural terrain. Fractal sur-
faces are characterised by their fractal dimension, which has been shown to correspond closely to our perception of roughness [40].

Fractals have been used widely in computer graphics to generate realistic-looking images of landscape. There are several methods of producing these fractal surfaces, representing different approximations to fractional Brownian motion. Voss [60] describes the different methods, which include independent cuts, fast Fourier filtering, random midpoint displacement and successive random additions. The surfaces generated with these methods look realistic, but they are random — it is not possible to constrain the surfaces to go through, or pass near observed elevation data. Attempts have been made to constrain the surface by starting with a given set of points (which must fall on a regular grid) and running the midpoint displacement method from there [12, 65]. However this method gives fractals with visible artifacts, which are the result of the non-stationary approximation to fractional Brownian motion. These artifacts are particularly visible on surfaces and cannot be removed by smoothing.

Constrained fractals attempt to combine the shape control of the deterministic shape fitting methods and the realistic detail of the fractal generation methods. The technique involves using a regularisation method to do the surface interpolation, with the addition of some controlled Gaussian noise at each step of the iteration. Szeliski has shown [50] that this is equivalent to introducing a fractal prior to the Bayesian model of the problem. The result is a surface that is consistent with the given elevation data, but which also displays random fractal variations, giving the appearance of natural terrain.

The main goal of this thesis is to produce a set of routines that can calculate the fractal dimension of some sparse data from a natural landscape, then reconstruct the terrain in such a way that the interpolated data have fractal detail of the same dimension. In order to achieve this there are several separate tasks to perform. Quick and efficient testing requires the availability of many fractal surfaces of different sizes and known fractal dimensions. Therefore chapter 3 looks at three methods of fractal surface generation. Ultimately real data will have to be used, but artificial data have the advantages of allowing us to explore many different aspects
of the various methodologies pertaining to surfaces for which the “ground truth” is available and to surfaces which may be of chosen roughness. After this exhaustive evaluation of the various techniques proposed with the help of simulated data, real terrain data supplied by BG Technology will be used to see how the proposed methods perform with them.

With data to work on the next step — covered in chapter 4 — is to develop a method of recovering the fractal dimension of a surface. One method uses the power spectrum of the elevation model of a surface. However, a more appropriate method for calculating the fractal dimension of a surface is one that can deal with sparse data points given over an irregular grid. Such a method is also explored in chapter 4, and matches the observed differences in heights and separations of pairs of points and fits them to the fractal model.

The final part of the process is the fractal reconstruction. Chapter 5 looks first at normal surface interpolation and then gives a first implementation of a version that adds fractal detail during the reconstruction. Chapter 6 evaluates in detail the performance of the methods discussed in chapter 5. One major drawback of fractals is that they are isotropic. However, real terrain may be anisotropic and what is more, any such anisotropy is very significant geologically. It is the anisotropy that allows the geologist to draw conclusions about, for example, the possible presence of oil and gas deposits. So, any terrain interpolation method, if it is to be of any use to applications like the search for oil and gas reserves, has to be able to capture and retain any existing anisotropy in the reconstructed surface. Chapter 7 is devoted to a first attempt towards the solution of such a problem.

Finally chapter 8 presents my conclusions and discusses possible future developments of the work. Chapter 2 covers previous work that has been done in the different subject areas.

The originality of this thesis rests on many points:

The fractal methods proposed by Szeliski [50] and Arakawa and Krotkov [1] are thoroughly evaluated and their applicability to fractals of various degrees of roughness and rates of subsampling is investigated (chapter 6).
Two variations of the multiresolution method of Arakawa and Krotkov are proposed and their performance is compared with the original method (chapter 5).

The fractal interpolation method is modified to produce anisotropic fractals that can be of real use in the oil industry (chapter 7).
Chapter 2

Literature survey

2.1 Surface interpolation

Surface interpolation has been the subject of a great deal of research, thanks to its important role in a number of low-level computer vision problems. The common factor in these problems is that other low-level vision processes often generate scattered information that needs to be interpolated to give information at all points in the image. Taking this sparse information a surface interpolation technique aims to find the most likely values for the missing data points, giving a best guess at the complete full resolution image of the scene.

The problem is constrained by the need for the solution surface to fall close to the data points. However, many different surfaces could satisfy this constraint and the reconstruction problem is ill-posed. This means the existence of solutions cannot be guaranteed without the addition of more constraints. This problem is solved by introducing a smoothness constraint on the reconstructed surface, thus limiting the range of possible solutions. This is an example of regularisation, a technique that was first described by Tikhonov [56, 57]. Applying regularisation to vision problems, a single-level iterative algorithm for surface reconstruction was developed by Grimson [15, 16], using a variation function to express the quality of a given solution surface. With this functional, an optimisation algorithm can then find the best interpolated surface. The quadratic nature of the function means there is a definite local minimum. Theoretical justification for the introduction of the
smoothness constraint was given by Poggio and Torre [42], who formally described the solution of the ill-posed early vision processes using regularisation theory.

The single level algorithm converges to the solution extremely slowly. This is because the algorithm is very local in nature and as a result the constraints take many iterations to be propagated over a large image. Terzopoulos [53] extends the algorithm to work at a range of different resolutions, with the solution at each stage being passed back and forth between the different levels to give solutions at each. The resulting multilevel algorithm allows the constraints to propagate much faster thanks to the coarser levels, greatly improving the speed of convergence. Terzopoulos also expresses the problem in terms of surface splines [46]. The first and second order cases can be interpreted physically as a membrane and a thin plate respectively [6]. Terzopoulos choose the thin plate spline model as the best smoothness constraint.

In [50] Szeliski formulates the regularisation problem in terms of a Bayesian modelling framework. This statistical description of the problem (first laid out in detail by Jeffreys in [17]) combines a prior model of the world and its properties and a sensor model of the stochastic process involved in collecting the data. Bayes rule combines these two to give a posterior probability for the current estimate, which in this case is a surface. A detailed description of the use of Bayesian models with regularisation can be found in MacKay [28].

Often when using interpolated data it is necessary to have a measure of the uncertainty in the new values. A probabilistic analysis of regularisation is performed by Keren and Werman in [20], including a brief survey of other approaches. This problem is also solved by Szeliski in [50].

Another problem with the standard regularisation method is that the smoothness constraints are global. This means that any discontinuities in the depth or orientation of the surface will be lost in the uniformly smooth reconstruction. A type of spline that allows for discontinuity is the spline under tension [47]. Terzopoulos in [54] uses a blend of the membrane and thin plate splines to create a controlled-continuity spline which can solve regularisation problems where there are known discontinuities.
2.1. Surface interpolation

The location of the discontinuities is more likely to be unknown, which results in the problem becoming once again ill-posed. Stevenson et al [48] describes a technique for incorporating discontinuities into the reconstruction problem while maintaining a well-posed and well-conditioned system.

Miyajima and Mukawa [36] present a discontinuity-preserving reconstruction method that uses a smoothness map. This is represented by an array of regularisation parameters, and it enables discontinuities to be preserved according to the global shape as well as the local features.

An alternative to the multigrid method of Terzopoulos [53] for speeding the rate of convergence of the surface interpolation problem is the hierarchical basis function method of Szeliski [51]. This method uses conjugate gradient decent and a hierarchical (multiresolution) set of basis functions and gives a similar speed-up to that of the multigrid methods. It is also claimed that this method is easier to implement and works in cases where multigrid may fail. Szeliski suggests that his choice of hierarchical basis functions (polynomials) could be replaced by wavelets. This suggestion is followed by Yaou and Chang in [64]. Another implementation of wavelets for surface interpolation is given by Pentland in [41].

Mizutani [37] introduces a new smoothness measure to the standard regularisation problem, using a continuous order differential operator. This generalised operator is used to replace the membrane and thin plate energies as a measure of the smoothness of the reconstructed surface.

Another multiscale algorithm for solving the regularisation problem is described by Luetgten et al in [26]. A different formulation of the prior model smoothness term allows problems requiring smooth solutions to be solved with a non-iterative algorithm. This gives faster solutions and better scaling of performance as the resolution is increased.

Rabadi et al [43] presents an iterative algorithm that uses the Fourier transform of the image to provide a better first guess at the solution. The iteration then uses a multilevel pyramid structure. It is claimed that these allow the method to avoid possible stagnation and give a better chance of finding the global minimum, rather than a local one.
Szeliski's method is adapted by Goshtasby for 3D scene recovery in [14]. Using an alternative surface reconstruction method, rational Gaussian surfaces, results are demonstrated for smooth reconstructions as both surfaces, 3D shapes and 3D cylinders.

The regularisation methods involves the minimisation of an energy functional which is a weighted sum of a data compatibility term and a smoothness term. Keren and Werman [21] look at the relative weight assigned to these terms and present a method for finding the optimal weights. This is achieved using a Bayesian approach.

2.2 Fractals

Mandelbrot's fractal geometry [31] appears to describe a number of basic physical processes and patterns, such as the shape of a cloud or a mountain or the path taken by lightning. Being able to describe them also allows one to create "forgeries" of them. Mandelbrot models natural terrain using fractional Brownian motion, which is an extension of the classical Brownian motion found in physics. The roughness of the surfaces is controlled by the fractal dimension.

Pentland [40] looks at the use of fractional Brownian motion to describe natural scenes. He suggests two methods of calculating the fractal dimension of a fractal and of an image of that fractal; using the difference statistics of the fractal function, or through its Fourier power spectrum. He then uses the fractal dimension measurement to perform segmentation of images of real-world scenes.

Yokoya et al. [65] present a method of extracting fractal-based features from a terrain map, including the fractal dimension, based on the difference statistics method. Other techniques for calculating fractal dimensions are the box-counting method [23] and the covering blanket method [38].

Wornell and Oppenheim suggest another way of measuring the fractal dimension [63] that uses wavelets. Fractional Brownian motion is modelled using Daubechies wavelets. This same approach is also taken by Deriche and Tewfik [7] and Flandrin [11]. Kaplan [18] does the same thing, except using wavelets with the Haar basis.
to improve the performance where there are few data values. Fleguth and Willsky [9] use a multiscale tree to model the fractional Brownian motion behaviour within the wavelet framework, placing the emphasis on getting the most likely estimate rather than getting the best model of fractional Brownian motion in the model.

Peli [39] calculates the fractal dimension using a method based on the covering blanket. This is used on images of scenes to pick out objects, given that natural and man-made objects will generally have very different fractal properties.

As well as the roughness, Mandelbrot describes another property of fractals that may be varied and measured: “lacunarity”. This measures the size of the gaps in a fractal, and is described in chapter 34 of [31]. In [32] Mandelbrot describes the problem of random artificial fractal landscapes having an unnatural-looking number of large creases and the failure to include river networks. [32] presents some possible solutions to these problems.

Several methods of approximating fractional Brownian motion are described by Voss [60] and Saupe [45]. Both describe the popular and fast midpoint displacement method and also the spectral synthesis method, which is slower, but is a better approximation to fractional Brownian motion. Fournier et al. [12] investigate just the midpoint displacement method. All the methods take a value for the fractal dimension and use random numbers to create a randomly-shaped surface with the desired roughness.

Krueger et al. [24] evaluate four methods for generating one-dimensional fractional Brownian motion: displaced interpolation, which is a development of midpoint displacement by Saupe [45], spectral synthesis, Karhunen-Loeve-like wavelet expansion [62], and sampled fractional Brownian motion [27]. He then finds the fractal dimensions of these signals using three methods. Spectral linear regression [40], multiresolution energy estimation [29] and maximum likelihood estimation [27]. It is found that the measurements have “reasonable accuracy”, except when the analysis and synthesis routines make different assumptions, such as using spectral linear regression on the displaced interpolation data.

Reed et al. [44] and Flandrin [10] look at the question of defining the power spectrum of fractional Brownian motion. Much work, including the spectral synthesis
Chapter 2. Literature survey

method of generating fractal terrain, depends on the power spectrum of fractional Brownian motion, which as a non-stationary process cannot have a spectrum defined in the usual way. Reed et al. [44] define a spectral representation that extends to $n$-dimensions.

Fractional Brownian motion uses only one parameter — the fractal dimension — to describe a fractal surface. This is obviously insufficient to describe all types of terrains. Amongst the failings of these fractals are they have the same fractal behaviour over all scales, whereas real world surfaces are fractal over only a limited scale, and then not always with the same fractal dimension. Real terrain is also often anisotropic, unlike fractional Brownian motion surfaces. Lewis [25] describes some of the problems with the simple fractional Brownian motion method and produces a modified displacement method called generalised stochastic subdivision. This avoids the artifacts present in other subdivision methods and enables generalisation to different power spectrums.

Klinkenberg and Goodchild [22] use seven methods of calculating the fractal properties of a large sample of real surfaces. They find these to vary widely between surfaces. Some are not described well by the fractal model at all, while others are. Dierking [8] looks at different geological terrains from roughness profiles and synthetic aperture radar images. The roughness of these surfaces is then analysed and compared to a stationary random roughness model and to a fractal model. It is found that some surfaces — such as arid terrains — are well defined by the simple random method, while others require the more complex fractal model fractal model.

In an attempt to model the extra complexity of real terrain, Kaplan and Kuo [19] introduce the extended self-similar model, giving a scale dependent random structure. This model is then used for measuring the fractal properties and for generating random fractals.
2.3 Fractal reconstruction

The reconstruction methods described previously take no account of the special fractal nature of surfaces such as natural terrain and the fractal generation techniques described in the previous section produce purely random fractal surfaces. A fractal reconstruction aims to produce an interpolated surface that displays the same sort of rough, fractal properties as the sparse data points that we are given.

In [65] Yokoya et al. calculate the fractal properties of a surface and use them to produce a fractal reconstruction using a modified version of the midpoint displacement method. The data points are held fixed and the gaps are filled in the same way as a random midpoint displacement fractal is generated.

A different approach was taken by Szeliski and Terzopoulos [52]. Instead of taking a fractal generation technique and adapting it to do a reconstruction, Szeliski and Terzopoulos adapted a method of reconstruction — regularisation — to produce fractals. Szeliski places this fractal reconstruction technique in a Bayesian framework [50] and uses the Gibbs sampling technique [13] to select a random solution from the probability distribution. However, Szeliski only describes an imprecise way of controlling the roughness of the reconstructed surface by using different blends of the membrane and thin plate models at the different resolution levels.

Arakawa and Krotkov [1, 2] use the same method but add a more precise control of the fractal dimension through the manipulation of both the model blends and the “temperature” parameter. They also test the reconstructions with some real data [2].

Vemuri and Radisavljevic [59] adapt the method to work with 3D surfaces (using the “deformable superquadrics” of Terzopoulos and Metaxas [55]) and use a wavelet modelling scheme[29]. The wavelets can be used to control the surface roughness, and also improve the convergence speed of the algorithm. This technique is adapted for use with terrain data in [58]. A preconditioner with the wavelet basis and a new conjugate gradient-based Gibbs sampler is used to find the solution.

A refinement of the subdivision methods is presented by Mandal et. al. in [30].
Based on the successive random additions method of fractal generation, it starts from an arbitrary mesh and fills in more fractal detail using a "butterfly" subdivision algorithm.

An alternative to a fractional Brownian motion based interpolation is an iterated function system. Previously used mostly in one-dimensions [3, 5, 4], these provide a method of interpolating between data points without the implied smoothness constraints of a fractional Brownian motion technique. This has been extended to two-dimensional fractal interpolation by Massopust [34, 35] and Wittenbrink [61].
Chapter 3

Tools for fractal generation

In computer graphics applications, there are several popular methods for generating fractal landscapes. This section describes the implementation of three of these methods: Random midpoint displacement, successive random additions, and the fast Fourier transform method. Random midpoint displacement is the simplest method which is computationally fast at the expense of mathematical accuracy. Successive random additions is a very similar method, introducing more computations to improve the approximation to fractional Brownian motion. The fast Fourier transform method is more computationally expensive, but gives a very accurate approximation to fractional Brownian motion. This means that it produces fractal surfaces without the visible artifacts associated with the other two methods.

There are several reasons for looking at the existing methods for fractal landscape generation. An important tool for the project is a method of calculating fractal dimension of a given surface. Therefore surfaces of known fractal dimension are necessary for testing the quality of the estimates produced by various different methods. Surfaces of known dimension will also be required for testing the surface interpolation. The generated surfaces can be sub-sampled to give an input for the interpolating routine, which will then be used to recover the original. Any method will be evaluated by comparing its output with the original full surface, which will serve as the "ground truth."
3.1 Fractional Brownian motion

Natural terrain can be simulated by Mandelbrot's fractal geometry because it is statistically invariant over a wide range of scales and each part of the surface is statistically similar to all others. The model that best describes such a shape is fractional Brownian motion [33]. This is an extension of the classical Brownian motion found in physics (the path of a particle displaying Brownian motion being an example of this type of fractal). Fractional Brownian motions model non-stationary stochastic processes, which are characterised by the long-term interdependence of their increments. Such processes are often called “1/f” noises, because of their 1/f spectral behaviour.

Using the notation of Voss [60], in two dimensions a fractional Brownian motion, \( V_H(x, y) \) has stationary increments with a Gaussian distribution whose variance is given by

\[
\langle |V_H(x_2, y_2) - V_H(x_1, y_1)|^2 \rangle \propto |(x_2 - x_1)^2 + (y_2 - y_1)^2|^{2H},
\]

where the brackets < and > denote averages.

\( H \) controls the complexity of the surface and takes values between 0 and 1. If \( H < \frac{1}{2} \) then the increments of the fractional Brownian motion, \( V_H \), are negatively correlated and two consecutive displacements along a profile in any direction, and over the same horizontal distance, are likely to have opposite signs. \( H = \frac{1}{2} \) corresponds to the case of classical Brownian motion and the increments are independent of one another. For \( H > \frac{1}{2} \) the increments are positively correlated and the two height variations are likely to have the same sign.

\( H \) is related to the fractal dimension, \( D \), of the surface by

\[
D = E + 1 - H,
\]

(3.2)
where $E$ is the dimension of the Euclidean space. Therefore terrain surfaces ($E = 2$) have a fractal dimension given by

$$D = 3 - H.$$  \hfill (3.3)

A surface with larger $D$, smaller $H$, looks more irregular, with big variations in heights. A surface with smaller $D$, larger $H$, looks smoother. The resulting surface is continuous, non-differentiable, self-affine, and isotropic.

### 3.2 Random midpoint displacement

Midpoint displacement is a simple and quick way of producing an approximation to a fractal surface. It is therefore commonly seen in computer graphics, where its use was popularised by Fournier, Fussell, and Carpenter [12]. The idea is to start with a triangle or a square — with random heights at its corners — which is then subdivided by finding its midpoint. This midpoint is then given the average height of its three or four neighbours plus a random displacement. As the scale is reduced these displacements are made to scale as expected for a fractal, i.e., at each stage the scaling factor for the random number is reduced by $1/2^H$.

The square mesh method is implemented here. This method takes two steps to go from one grid to another, with half the grid size. The first step finds the elevations of the midpoints of the squares. This leaves many points on the new smaller grid without heights. The second step finds the elevations of these remaining points required to make up the grid. This second step can also be viewed as finding the midpoints of the smaller squares that form a lattice at an angle of 45 degrees to the original. The two stages are illustrated in figure 3.1.

If the constant of proportionality in (3.1) is chosen to be $\sigma^2$, then we must always have

$$\langle |V_H(x_2, y_2) - V_H(x_1, y_1)|^2 \rangle = \sigma^2 |(x_2 - x_1)^2 + (y_2 - y_1)^2|^{2H}. \hfill (3.4)$$
Chapter 3. Tools for fractal generation

Figure 3.1: The Two Stages of Midpoint Displacement.

From this expression it is possible to derive the variance, $\Delta_n^2$, of the random variable added to the averaged midpoint height at any stage $n$. The workings are given in Appendix A.

$$\Delta_n^2 = \frac{\sigma^2}{2n^H} \left| 1 - 2^{2H-4} - 2^{H-3} \right|. \quad (3.5)$$

The resulting program generates a fractal of $N$ points in only $O(N)$ operations. However, once the elevation at a point has been determined, its value is not changed in any of the later stages — i.e. at each stage only half the points are determined more accurately. This produces a result that is not stationary [31] when $H \neq \frac{1}{2}$. This means that the method is not a perfect approximation to fractional Brownian motion and the resultant surfaces show visible artifacts in the form of creases that do not disappear with more iterations. These artifacts become more visible as $H$ approaches 1 and cannot be removed by smoothing.

Figure 3.2 shows two examples of surfaces generated with the random midpoint displacement method. They both use the same random seed, making it easier to see the effect of increasing the fractal dimension as the surfaces have the same basic shape. It is just possible to see the artifacts present in this method, where peaks or troughs are visible at regularly-spaced intervals. This is more visible in the smoother surface, and the peaks/troughs are points calculated early in the interpolation.
3.3 Successive random additions

Successive random additions is the same as the method of random midpoint displacement with the difference that extra increments are added to the height at both the new nodes, as before, and also at the existing grid points. In other words a stage of the process involves the calculation of the average height of the four corners of each square, which is assigned to the midpoint, as before. However, random numbers, of variance $\Delta^2$ as given by equation (3.5), are then added to both the midpoint and each of the four corners rather than just the midpoint.

The result of the additional increments is to improve the approximation to fractional Brownian motion. The visible artifacts present in the random midpoint displacement examples are much reduced. The amount of computation is also still low, increasing as $O(N)$. However, as $H$ approaches 1 the points calculated early in the procedure are still visible as artifacts.

Figure 3.3 shows two examples of surfaces generated with the successive random additions method. As before, they both use the same random seed, making it easier to see the effect of increasing the fractal dimension as the surfaces have the same basic shape. The artifacts are less visible in this method, and are not visible at all in these small examples.
3.4 Fast Fourier transform filtering

This method generates a fractal terrain by specifying the spectral density of the surface. Natural terrain is one of a number of physical phenomena that have spectra of the form $1/f^\beta$ over a range of frequencies. Other examples [33] include economic time series, fluctuations in solids, and water flows in hydrology. The non-stationary nature of fractional Brownian motion means that its spectral density is "difficult to interpret" [33]. This is a result of the Fourier transform of the autocorrelation function and the moments not being well-defined. However, the increments are assumed to form a stationary random process with $1/f^\beta$ spectral behaviour, and the properties of these increments can be used to give a well-defined spectral representation [44]. The spectral density, $S_V(f)$, of an $n$-dimensional fractional Brownian motion is then found to be controlled by parameter $H$ [44]:

$$S_V(f) \propto \frac{1}{f^{n+2H}},$$  \hspace{1cm} (3.6)

where $f$ is an $n$-dimensional frequency, $f \equiv |f|$, and $0 < H < 1$.

Since all directions in the $xy$-plane are statistically equivalent the spectral density in two dimensions will only depend on $\sqrt{f_i^2 + f_j^2}$, where $f_i$ and $f_j$ are the frequency components of $f$ in the $x$ and $y$ directions respectively:

$$S_V(f) \propto \frac{1}{(f_i^2 + f_j^2)^{H+1}} \propto \frac{1}{(f_i^2 + f_j^2)^{4-D}}.$$  \hspace{1cm} (3.7)
3.4. Fast Fourier transform filtering

Fast Fourier Transform filtering starts with a white noise signal, $W$, obtained from a random number generator. This noise is then filtered with a transfer function $T(f)$, producing an output with the desired spectral density. The spectral density of this output is

$$S_V(f) \propto |T(f)|^2 S_W(f). \quad (3.8)$$

Since the spectrum of the white noise, $S_W(f)$, is just a constant,

$$S_V(f) \propto |T(f)|^2. \quad (3.9)$$

Therefore the required transfer function is of the form

$$T(f) \propto \frac{1}{(f_i^2 + f_j^2)^{(H+1)/2}}. \quad (3.10)$$

The filtering involves taking the 2 dimensional Fourier transform of the white noise. From the Fourier transform of the white noise we calculate the magnitude and phase at each pair of frequencies $(f_i, f_j)$. These magnitudes are first scaled according to equation (3.10) and subsequently, using the same phases, we Fourier transform back to obtain the fractal surface.

Voss [60] gives two alternative methods for producing the surface from the same theory. One involves choosing complex random variables with mean square amplitude proportional to $1/f^\beta$ and random phases. It is also possible to use a straight line (noise-less) $1/f^\beta$ spectrum and just have random phases. The white noise method will be used here.

The resultant surface is produced in $O(N \log N)$ operations and, given the assumption that fractional Brownian motion does have a power spectral density, this is the purest method of synthesising it. The surfaces are stationary and the artifacts shown by the displacement methods are not present. Adding more Fourier coefficients improves the representation of the spectrum and adds more detail to the surface. In theory the assumed spectral density shows an infinite power at the origin but this problem is side-stepped by setting this value to zero (the origin is the zero frequency or DC component and hence controls the average height of the surface).
Figure 3.4: FFT filtering surfaces.

Figure 3.4 shows two examples of surfaces generated with the Fourier synthesis method. As before, they both use the same random seed, making it easier to see the effect of increasing the fractal dimension as the surfaces have the same basic shape. These surfaces have no visible artifacts as the generating process is stationary.

3.4.1 Anisotropic FFT filtering fractal

Real world fractals often have different fractal dimensions in different directions [49]. It would therefore be useful to be able to artificially create such anisotropic fractals. Using FFT filtering this can be achieved by manipulating the frequency-space Fourier spectrum of the fractal. Since the spectrum is itself a 2D surface, it contains directional information. We can therefore give the resultant surface different properties in different directions.

To give different fractal dimensions in different directions, we calculate the angle made with the axis by the frequency-space point under consideration. The spectrum can then be scaled with a transfer function 3.10 with the desired dimension for that direction.

The simplest such fractal has given fractal dimension along one axis and another along the other axis, with the other directions a linear combination of the two, depending on the angle. The resulting skewed spectral density surface is then
3.4. Fast Fourier transform filtering

Figure 3.5: Anisotropic FFT filtering surfaces.

For a point \((f_i, f_j)\) in frequency space, its angle with the axis is

\[
\theta = \arcsin \left( \frac{f_i}{f_j} \right). 
\]  

(3.11)

To give a fractal surface with dimension \(D_x = 3 - H_x\) along the x-axis and \(D_y = 3 - H_y\) along the y-axis the transfer function \(T_{xy}\) is used:

\[
T_{xy}(f) \propto \left[ \frac{2\theta}{\pi} \right] \frac{1}{(f_i^2 + f_j^2)^{(H_x+1)/2}} + \left( 1 - \frac{2\theta}{\pi} \right) \frac{1}{(f_i^2 + f_j^2)^{(H_y+1)/2}}.
\]  

(3.12)

An example of a fractal produced with this method is shown in figure 3.5(a). The differing fractal dimensions in different directions are not easy to see in these surfaces. However, the depth variations are perceptibly less rough in the left-right direction of this example surface. The gradual transition from the one dimension to the other will result in the directions in between the two axes displaying in-between fractal properties, and this seems to cause an averaging out of the isotropic effect.

In order to create a surface with a more obvious difference in roughness in the two directions it is possible to change the transition between the two transfer functions from a linear combination to a binary one. If the angle the point in question makes with one axis is less than 45° then the transfer function for that direction is used.
Otherwise the other transfer function is used. This produces a step between the two spectral density functions.

\[ T_{xy}(f) \propto \begin{cases} \frac{1}{(f^2 + f_f^2)^{(n-1)/2}} & \text{if } \theta < 45^\circ \\ \frac{1}{(f^2 + f_f^2)^{(n-1)/2}} & \text{if } \theta \geq 45^\circ \end{cases} \]  

(3.13)

An example of a fractal surface produced with this method is shown in figure 3.5(b). With this surface it is easier to see the difference in roughness between the two directions. The left-right direction displays less fractal variation than before and this results in linear features in the fractal stretching out in this direction.

To make the distinction between the two differing roughnesses even more stark, the angles between which the two transfer functions are applied can be reduced. This means that the fractal behaviour is limited to a small range of angles around each axis. If \( \psi \) is this limiting angle (\( \psi \leq 45^\circ \)) the transfer function is now:

\[ T_{xy}(f) \propto \begin{cases} \frac{1}{(f^2 + f_f^2)^{(n+1)/2}} & \text{if } \theta < \psi \\ \frac{1}{(f^2 + f_f^2)^{(n+1)/2}} & \text{if } \theta \geq \psi \end{cases} \]  

(3.14)

A value of \( \psi = 45^\circ \) puts no limit on the angles used and is the method becomes the same as the previous version (figure 3.5(b)). Surfaces produced with \( \psi = 35^\circ \), 22.5° and 15° are shown in figures 3.6(a), 3.6(b), and 3.6(c) respectively. Reducing \( \psi \) has the effect of reducing the fractal detail in the directions other than those of the axes, and this can be seen as the surfaces show more obvious linear features in the direction of the smaller fractal dimension as the rougher profile is stretched out of the length of the surface.

In a real world surface the best model to use would depend on which one most closely matched the physical processes that created the anisotropic fractal. Two distinct processes acting in different directions, and perhaps acting at separate times, would probably require a model such as the latter two. An isotropic fractal created by one process that acted in differing degrees in different directions might be best modelled using the first method.
3.4. Fast Fourier transform filtering

Figure 3.6: Anisotropic FFT filtering surfaces with fractal behaviour over a limited range of Angles.
Chapter 3. Tools for fractal generation
Chapter 4

Tools for fractal dimension calculation

Mandelbrot's fractal geometry models landscapes as statistically self-similar surfaces. These surfaces are characterised by their fractal dimension, $D$.

Self-similarity, or scaling, is central to fractal geometry and gives rise to the similarity or fractal dimension, $D$. A $D$-dimensional self-similar object can be divided into $N$ smaller copies of itself, each scaled down by a factor $r = 1/\sqrt{N}$. Therefore if we have a self-similar object of $N$ parts, scaled by a ratio $r$ from the whole, then its fractal dimension is given by

$$D = \frac{\log N}{\log \frac{1}{r}}.$$  \hspace{1cm} (4.1)

If the scaled objects are not exactly alike, but are instead identical in all statistical respects, then the object is said to show statistical self-similarity. This simple relation leads to the popular box-counting method of fractal dimension calculation [23], which counts the number of boxes of different sizes which are required to cover a pattern.

The scaling property of fractional Brownian motion however is different from this in that the shapes repeat statistically only when the function, $V_H$, and the coordinates $x$ and $y$ are magnified by different amounts. If the coordinates are magnified by a factor $r$, then $V_H$ must be magnified by $r^H$. This is called statistical self-affinity, where shapes are statistically invariant under transformations that scale different
coordinates by different amounts. In two dimensions this non-uniform scaling is given by

\[ \Delta V \propto \Delta t^H \propto (\Delta x^2 + \Delta y^2)^H. \]  

(4.2)

The concept of fractal dimension can still be applied, and now expresses the relative amount of detail or random irregularities present at the different scales. However, measuring the fractal dimension of fractional Brownian motion using self-similarity properties can lead to ambiguous results, because the association of a self-affine fractal with a similarity dimension fixes the scaling between the otherwise independent coordinates.

There are several other methods of calculating the fractal dimension. Spectral analysis [40] is based on the assumption that fractional Brownian motion has a power spectral density. The spectrum is obtained by a Fourier transform and \( H \) is found from the slope of the log-log plot of the spectral density versus the frequency. Other methods include multiresolution analysis [29], using wavelets, a maximum likelihood estimator [27] and Pentland's fractional Brownian function approach [40, 65].

A fractal surface has a fractal dimension between 2 and 3. The higher the fractal dimension the rougher the overall surface looks [40], with values around \( D = 2.2 \) giving surfaces that look like natural terrain.

Having a method for accurately calculating the fractal dimension of a given surface is a vital part of the surface reconstruction project. The fractal dimension is an important characteristic of terrain, which should be conserved by the method of reconstruction. It is therefore essential to be able to measure the fractal dimension of a given surface. This surface may take the form of an image on a regular grid, such as the reconstructed surface, or may sometimes be in the form of irregularly spaced data points, for example sub-sampled test images or the actual range data. The method of dimension calculation must be able to cope with both situations.
4.1 Fractal dimension calculation using FFT

The first attempt at fractal dimension calculation is the spectral analysis method. This is simple to do since it is like the Fourier filtering synthesis method used for creating artificial fractals, except applied in reverse. Instead of inputting a value of $D$ and producing a surface as the output, a surface is used as the input and a value for the dimension is now the output.

The frequency space representation of the surface is obtained using the fast Fourier transform. This gives a grid of complex numbers — a real array and an imaginary array — representing the Fourier transform of the terrain. The magnitude of these complex numbers is taken, discarding the phase information and leaving just one frequency space array.

The spectral density has a peak in the centre, at the origin, and falls off as the frequency increases in the $x$ and $y$ directions. In order to perform a fit with the $1/f^\beta$ spectral density law this must be reduced to a 1D spectral density against frequency plot. This is done by simply using the $x$-$y$ coordinates to calculate the radial distance from the origin for each point on the frequency surface, thus discarding any information about directional variations. It is then a simple job of performing a least squares fit on these data to obtain a value for the slope.

The spectral density leads to Fourier coefficients of the form given by (3.10). If $f$ is the radial frequency with $x$ and $y$ components $f_i$ and $f_j$ then this is a plot of $1/(f_i^2 + f_j^2)^{(H+1)/2}$ or $1/f^{(H+1)}$ versus $f$. A log-log plot is therefore a straight line with a slope, $m = -(H + 1)$. The fractal dimension $D$ can then be calculated as

$$D = m + 4. \quad (4.3)$$

The first point of this straight line will be heavily influenced by the value of the surface at the origin. The spectral density law for a fractal predicts this to be infinity, but in practice it is simply the DC component of the signal and depends on the average height of the surface. This means that it contains no useful information about the fractal nature of the surface, and must be ignored. The first point of the
one-dimensional spectral density line will therefore be left out of the least squares fit to calculate the slope, and hence $D$.

### 4.1.1 Reliability of the fractal dimension calculation using FFT

The first test performed concerned the calculation of the fractal dimension of a fractal that was produced starting from a perfect $1/f^{(H+1)}$ spectrum. The calculation routine was then expected to produce an exact value for $H$ (and hence $D$), with any errors being the result of bugs in the code, the finite grid size, or errors in the method. Indeed, it was found that even with very small surface patches ($8 \times 8$) the dimension was recovered exactly.

The second test involved using noisy fractals, i.e., more realistic data. These are fractals that have been produced using the FFT filtering method described in section 3.4 — by creating first uncorrelated random noise with the help of a random number generator and converting it into a fractal with the desired fractal dimension by manipulating its spectrum. Table 4.1 contains results produced using fractal surfaces created by this method, of 8 different sizes. For each size 100 different surfaces were produced, starting with different uncorrelated noise fields. In this test the whole fractal is used, so the spectrum from which the fractal dimension is measured is the same size as the one used to create the fractal.

All these fractals have the same fractal dimension, $D = 2.2$. If a single run gives the result $D_i$, its error is given by $e_i = D_i - D$. The mean error in each result, $\bar{e}$, is then given by averaging over all available fractals of the same size:

$$\bar{e} = \frac{1}{100} \sum_i e_i.$$  \hspace{1cm} (4.4)

The standard deviation of the distribution of errors, $\sigma_e$, is given by

$$\sigma_e = \frac{1}{100} \sum_i (e_i - \bar{e})^2.$$  \hspace{1cm} (4.5)

The maximum and minimum values for $D_i$ are also given ($D_{\text{min}} = \min_i D_i$ and $D_{\text{max}} = \max_i D_i$).

In this case the method of generating the fractal and the method of calculating its dimension are very similar. They are both based on the assumption that a fractal
4.1. Fractal dimension calculation using FFT

Table 4.1: Fractal Dimension Calculation for fractal surfaces of different sizes produced with the FFT method. Statistics refer to ensembles of 100 different surfaces of the same size. All surfaces have $D = 2.2$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\bar{D}$</th>
<th>$\bar{\varepsilon}$</th>
<th>$\sigma_\varepsilon$</th>
<th>$D_{\min}$</th>
<th>$D_{\max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2.0592</td>
<td>0.4475</td>
<td>0.3747</td>
<td>0.2240</td>
<td>3.3527</td>
</tr>
<tr>
<td>16</td>
<td>2.1637</td>
<td>0.1357</td>
<td>0.1034</td>
<td>1.6903</td>
<td>2.5935</td>
</tr>
<tr>
<td>32</td>
<td>2.2049</td>
<td>0.0496</td>
<td>0.0383</td>
<td>2.0327</td>
<td>2.3803</td>
</tr>
<tr>
<td>64</td>
<td>2.2003</td>
<td>0.0244</td>
<td>0.0156</td>
<td>2.1357</td>
<td>2.2528</td>
</tr>
<tr>
<td>128</td>
<td>2.2005</td>
<td>0.0120</td>
<td>0.0094</td>
<td>2.1623</td>
<td>2.2398</td>
</tr>
<tr>
<td>256</td>
<td>2.2001</td>
<td>0.0054</td>
<td>0.0041</td>
<td>2.1810</td>
<td>2.2140</td>
</tr>
<tr>
<td>512</td>
<td>2.2005</td>
<td>0.0028</td>
<td>0.0021</td>
<td>2.1913</td>
<td>2.2090</td>
</tr>
<tr>
<td>1024</td>
<td>2.1996</td>
<td>0.0014</td>
<td>0.0011</td>
<td>2.1945</td>
<td>2.2030</td>
</tr>
</tbody>
</table>

surface has a spectral density, the shape of which controls the fractal dimension. As a result, it is to be expected that the calculated dimension should be close to the real value. Table 4.1 shows this to be the case, with image sizes $32 \times 32$ and above producing less than 5% errors and $256 \times 256$ giving 0.5% error on average. The results for sizes up to $512 \times 512$ also seem to show that the method produces some residual systematic error, leading to a slight over-estimate. However the biggest image size shows an under-estimate, suggesting the errors are rather random.

The next test involves checking that the method works just as well for all fractal dimensions between 2 and 3. Table 4.2 shows the results of using the same method, with an image of size $64 \times 64$, for 100 fractals at each of five different dimensions. The results show that the accuracy is constant over the range of dimensions. The slight over-estimate is also constant over the range.

Instead of using the whole fractal, like in the previous tests, the next test involved extracting a window of height points from the fractal. The fractal dimension was then calculated from this smaller amount of data so that the influence of the size of the available data in the calculation could be assessed.

Different size windows were used from each of 100 different fractals created with the same spectral synthesis method. The fractals were all of size $256 \times 256$ with dimension, $D = 2.2$. 
Table 4.2: Fractal Dimension Calculation for surfaces of different fractal dimensions produced with the FFT method. Statistics refer to ensembles of 100 different surfaces of the same $D$. All surfaces have size $N = 64$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\tilde{D}$</th>
<th>$\bar{e}$</th>
<th>$\sigma_e$</th>
<th>$D_{min}$</th>
<th>$D_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>2.1003</td>
<td>0.0244</td>
<td>0.0156</td>
<td>2.0357</td>
<td>2.1528</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3003</td>
<td>0.0244</td>
<td>0.0156</td>
<td>2.2357</td>
<td>2.3528</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5003</td>
<td>0.0244</td>
<td>0.0156</td>
<td>2.4357</td>
<td>2.5528</td>
</tr>
<tr>
<td>2.7</td>
<td>2.7003</td>
<td>0.0244</td>
<td>0.0156</td>
<td>2.6357</td>
<td>2.7528</td>
</tr>
<tr>
<td>2.9</td>
<td>2.9003</td>
<td>0.0244</td>
<td>0.0156</td>
<td>2.8357</td>
<td>2.9528</td>
</tr>
</tbody>
</table>

Table 4.3: Fractal Dimension Calculation using windows of various sizes. Statistics refer to ensembles of 100 different windows, one from each surface created. All surfaces, produced with the FFT method, are $256 \times 256$ in size and have $D = 2.2$.

<table>
<thead>
<tr>
<th>Window Size</th>
<th>$\tilde{D}$</th>
<th>$\bar{e}$</th>
<th>$\sigma_e$</th>
<th>$D_{min}$</th>
<th>$D_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2.6705</td>
<td>0.5640</td>
<td>0.4474</td>
<td>1.4881</td>
<td>4.6445</td>
</tr>
<tr>
<td>16</td>
<td>2.2717</td>
<td>0.1557</td>
<td>0.1147</td>
<td>1.7812</td>
<td>2.6428</td>
</tr>
<tr>
<td>32</td>
<td>2.2165</td>
<td>0.0778</td>
<td>0.0557</td>
<td>2.0060</td>
<td>2.4341</td>
</tr>
<tr>
<td>64</td>
<td>2.1880</td>
<td>0.0428</td>
<td>0.0290</td>
<td>2.0827</td>
<td>2.2969</td>
</tr>
<tr>
<td>128</td>
<td>2.1903</td>
<td>0.0269</td>
<td>0.0196</td>
<td>2.1247</td>
<td>2.2851</td>
</tr>
</tbody>
</table>
The results show that patches of size $8 \times 8$ or $16 \times 16$ produce very unreliable results. We can see that a minimum size of a $32 \times 32$ patch is needed for a reliable estimate, and even then there is a possibility of estimating the fractal dimension to be as low as 2.01 or as high as 2.43, instead of 2.20.

A more difficult test for the fractal dimension calculation method is to use it on other types of fractals, where the generation and calculation techniques are not as closely related. The successive random additions method 3.3 is used to generate random fractals, whose dimensions are then calculated using the same method as before, again ignoring the direct component. The results are found to be far less accurate than with the Fourier synthesised fractals. A possible reason for this is revealed by looking at the spectral density plot, which should be a straight line for a fractal.

Inspection of the spectral density in figure 4.1 shows that the otherwise straight line expected of a fractal shows a significant curve towards the higher frequencies. This is probably due to the lack of random additions given to the points introduced later in the algorithm, which as a result don't scale in quite the right fashion. This leads to this visible curve at high frequencies, or at the fine detail level of the surface. Given that the fault is with the generation routine, rather than improving the routine to enable testing, it is decided to perform the test on the straight line section.
of the spectrum, discarding the high frequency points. It is found that the more high frequency points that are thrown away the more accurate the results become. Using half the points for the calculation is found to be the best compromise, as this covers most of the straight line portion of the curve on most examples, without losing too much of the data.

The results of the fractal dimension calculation method using the first half of the frequency points on each spectrum of different-sized surfaces with \( D = 2.2 \) are given in table 4.4. These results all show significant over-estimates. The smaller images give errors of around 20%, falling to 5% for large images. This 5% error seems to remain present from \( N = 128 \) upwards, suggesting that it is an intrinsic error in the method.

The final test of the fractal dimension calculation method examines the results for successive random additions surfaces of different dimensions. Images of size 128 \( \times \) 128 are used. Table 4.5 shows that the errors increase as the dimension decreases. The method is therefore most reliable on rough surfaces and less so as the dimension approaches 2 and the surfaces become smooth. This is not a desirable behaviour since it is these surfaces that are the most common in nature and so good performance at low values of \( D \) is important.

Apart from the high frequency deviation from a straight line the successive random additions spectrum shows another unusual feature. Unlike the spectral synthesis
4.1. Fractal dimension calculation using FFT

Table 4.5: Fractal dimension calculation for fractal surfaces of various dimensions produced with the successive random additions method. Statistics refer to ensembles of 100 different surfaces of the same $D$. All surfaces have $N = 128$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\tilde{D}$</th>
<th>$\bar{e}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>2.1688</td>
<td>0.0726</td>
<td>0.0472</td>
<td>2.0389</td>
<td>2.3064</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3345</td>
<td>0.0604</td>
<td>0.0406</td>
<td>2.1666</td>
<td>2.4619</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5424</td>
<td>0.0651</td>
<td>0.0399</td>
<td>2.3449</td>
<td>2.6770</td>
</tr>
<tr>
<td>2.7</td>
<td>2.7518</td>
<td>0.0630</td>
<td>0.0365</td>
<td>2.5749</td>
<td>2.8684</td>
</tr>
<tr>
<td>2.9</td>
<td>2.9430</td>
<td>0.0512</td>
<td>0.0319</td>
<td>2.7932</td>
<td>3.0465</td>
</tr>
</tbody>
</table>

Figure 4.2: Spectral density of a successive random additions surface.

method, where the spectrum is defined to have the right shape, the spectral surface is not isotropic. As can be seen in the example given in figure 4.2 the surface has prominent ridges along the $x$ and $y$ axes.

Along any particular angle the scaling behaviour is not that expected of a fractal, with the directions along the axes showing significantly different behaviour compared to all other directions. However the results show that the process of ignoring the radial dependence, to give an "average" scaling behaviour, gives values close to those expected. Nevertheless, this averaging is based on the assumption that the spectrum is isotropic, which this spectrum is obviously not.

The ridges along the axes suggest that the spectrum is dominated by $x$-direction frequency components ($f_x$) — and especially higher frequency components — where the other component, $f_y = 0$. And similarly $f_y$ values with $f_x = 0$ also dominate.
Large high frequency components are found in the Fourier transform when there is a sharp feature such as a discontinuity in the real domain. The fact that these features are found on the axes in the frequency domain tells us that the discontinuities must have orientations parallel to the axes. This means they must be located along the edges of the fractal surface. There are discontinuities here because the Fourier method requires a surface that stretches to infinity in all directions. In order to calculate the Fourier transform of a finite surface it is assumed that the given surface wraps around repeatedly. Therefore these artifacts are seen in figure 4.2 because this wrapping around of the surface introduces discontinuities along the edges of the surface, which are then detected by the Fourier transform.

These results all involve finding the fractal dimension of a surface sampled with a regular grid. However, in practice the data will often be sparse and irregular and it is necessary to be able to find the dimension of these surfaces. This would cause a problem for this method since the FFT requires a regular grid. Therefore the method needs to be modified to cope with irregular grids, or a different method must be used.

### 4.2 Fractal dimension calculation using difference statistics

An alternative method for finding the fractal dimension is to use the difference statistics of the surface. Unlike the Fourier transform method, this can be used on irregular, sparse data of the sort that is likely to be encountered during a typical reconstruction.

This method is based on the equation that defines the relationship between the difference in height of a pair of points on a fractal surface and the distance between the two points (see equation (3.4)). If $\Delta V$ is the difference in heights and $\Delta x$ is the distance between two points, then

$$
\langle \Delta V^2 \rangle = \sigma^2 |\Delta x|^{2H}.
$$

To find the fractal dimension, all possible pairs of points on the surface are considered. For a given $\Delta x$ several values of $\Delta V^2$ are found. The average of these
4.2. Fractal dimension calculation using difference statistics

$\Delta V^2$ values is computed, to create $<\Delta V^2>$ corresponding to the given $\Delta x$. If $<\log|\Delta V^2|>$ is then plotted against $\log \Delta x$, equation (4.6) says that these values must fall on a straight line with gradient $2H$. The fractal dimension can then be found using equation (3.3).

The intercept along the vertical axis corresponds to the value of $<\Delta V^2>$ for $|\Delta x| = 1$, i.e. the variance of the distribution of the difference in height between points which are 1 unit in distance apart. If a fractal is stationary, this variance is $\sigma^2$, independent of the fractal dimension that is defined by the slope of the line.

4.2.1 Reliability of the fractal dimension calculation using difference statistics

To test the reliability of calculating the fractal dimension using difference statistics, fractal surfaces were generated using the Fourier synthesis method. The difference statistics method builds up a histogram for the difference in heights versus the difference in distance between two points, and it does this by considering each possible pair of points in the image. An example plot of $<\log|\Delta V|>$ versus $\log \Delta x$ is shown in figure 4.3.

Due to boundary effects and the fact that the larger distances occur far less frequently than the smaller ones — thus having less reliable statistics — the resulting
plot of $< \log |\Delta V| >$ versus $\log \Delta x$ does not always appear as the expected straight line for the whole range of distances. For this reason it is decided to calculate the plot only up to a distance which is a fraction of the size of the surface. Table 4.6 shows the results when the fraction of the distances used is varied. Building the histogram with distances up to 10% of the size of the surface is found to give the best results. This result is likely to be different for different fractals, depending on over how many distance scales they display fractal behaviour.

<table>
<thead>
<tr>
<th>Distance Fraction</th>
<th>$\bar{D}$</th>
<th>$\bar{\varepsilon}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2.6163</td>
<td>0.4163</td>
<td>0.0536</td>
<td>2.4670</td>
<td>2.7410</td>
</tr>
<tr>
<td>0.7</td>
<td>2.5111</td>
<td>0.3111</td>
<td>0.0625</td>
<td>2.3331</td>
<td>2.6580</td>
</tr>
<tr>
<td>0.5</td>
<td>2.4521</td>
<td>0.2521</td>
<td>0.0720</td>
<td>2.2337</td>
<td>2.6149</td>
</tr>
<tr>
<td>0.3</td>
<td>2.3868</td>
<td>0.1870</td>
<td>0.0673</td>
<td>2.1885</td>
<td>2.5530</td>
</tr>
<tr>
<td>0.1</td>
<td>2.2859</td>
<td>0.0865</td>
<td>0.0340</td>
<td>2.1785</td>
<td>2.3714</td>
</tr>
</tbody>
</table>

Table 4.7 contains the results for 100 fractal surfaces, of 4 different sizes. All the fractals have a dimension $D = 2.2$ and in each case the histogram is formed using pairs a maximum distance apart equal to 10% of the image size. The results show a consistent and significant overestimate, that does not reduce as the surfaces get bigger. The average error also remains around 8%. The only change for the larger surface sizes is a slight reduction in the error in the maximum and minimum dimension values obtained from the hundred fractals.

Table 4.8 shows the results for 100 fractals at each of 5 different fractal dimensions, with all the fractal surfaces having size $64 \times 64$. These results show the overestimates observed at the lower fractal dimensions give way to significant underestimates at the higher fractal dimensions. The crossover occurs somewhere between $D = 2.3$ and 2.5, with these dimensions being most accurately recovered, with average errors around 4%.

The reason for using this method of fractal dimension calculation is to be able to
4.2. Fractal dimension calculation using difference statistics

Table 4.7: Fractal dimension calculation for fractal surfaces of different sizes produced with the FFT method. Statistics refer to ensembles of 100 different surfaces of the same size. All surfaces have $D = 2.2$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\bar{D}$</th>
<th>$\bar{e}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>2.2742</td>
<td>0.0754</td>
<td>0.0437</td>
<td>2.1739</td>
<td>2.3901</td>
</tr>
<tr>
<td>64</td>
<td>2.2859</td>
<td>0.0865</td>
<td>0.0340</td>
<td>2.1785</td>
<td>2.3714</td>
</tr>
<tr>
<td>128</td>
<td>2.2819</td>
<td>0.0819</td>
<td>0.0314</td>
<td>2.2023</td>
<td>2.3522</td>
</tr>
<tr>
<td>256</td>
<td>2.2760</td>
<td>0.0760</td>
<td>0.0297</td>
<td>2.2029</td>
<td>2.3486</td>
</tr>
</tbody>
</table>

Table 4.8: Fractal dimension calculation for surfaces of different fractal dimensions produced with the FFT method. Statistics refer to ensembles of 100 different surfaces of the same $D$. All surfaces have size $N = 64$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\bar{D}$</th>
<th>$\bar{e}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>2.2343</td>
<td>0.1343</td>
<td>0.0351</td>
<td>2.1290</td>
<td>2.3230</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3433</td>
<td>0.0475</td>
<td>0.0283</td>
<td>2.2396</td>
<td>2.4221</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4692</td>
<td>0.0349</td>
<td>0.0250</td>
<td>2.3889</td>
<td>2.5354</td>
</tr>
<tr>
<td>2.7</td>
<td>2.5975</td>
<td>0.1025</td>
<td>0.0238</td>
<td>2.5398</td>
<td>2.6580</td>
</tr>
<tr>
<td>2.9</td>
<td>2.7145</td>
<td>0.1855</td>
<td>0.0182</td>
<td>2.6778</td>
<td>2.7653</td>
</tr>
</tbody>
</table>

work with irregular, sparse data. In order to test the effect of sparsity, table 4.9 shows the results for 100 different fractals of dimension $D = 2.2$, with the calculation performed using only a random subsample of the available points. The fraction of points taken in this subsample is varied. The results show that the average error ($\bar{e}$) remains constant down to 30%. At 10% of the points the average error increases slightly and the range in which the answers fall (between $D_{\text{min}}$ and $D_{\text{max}}$) widens. Both these increase much more when only 1% of the points are taken and, although the altered $\bar{D}$ is coincidentally actually better, the individual results are less reliable.

The important factor in this calculation is having sufficient points to produce good statistics. Therefore it is the absolute number of points that is important. A $64 \times 64$ fractal contains 4096 points, so these results suggest that less than around 400 points leads to unreliable fractal dimension results. For a larger fractal surface the
percentage value needed would be smaller.

Table 4.9: Fractal dimension calculation for fractal surfaces produced with the FFT method, varying the fraction of points used in the difference statistics calculation. Statistics refer to ensembles of 100 different surfaces of size $N = 64$ and $D = 2.2$.

<table>
<thead>
<tr>
<th>Fraction of Points</th>
<th>$\bar{D}$</th>
<th>$\tilde{\epsilon}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>2.2858</td>
<td>0.0864</td>
<td>0.0337</td>
<td>2.1770</td>
<td>2.3735</td>
</tr>
<tr>
<td>0.7</td>
<td>2.2851</td>
<td>0.0857</td>
<td>0.0334</td>
<td>2.1816</td>
<td>2.3684</td>
</tr>
<tr>
<td>0.5</td>
<td>2.2850</td>
<td>0.0858</td>
<td>0.0342</td>
<td>2.1795</td>
<td>2.3744</td>
</tr>
<tr>
<td>0.3</td>
<td>2.2848</td>
<td>0.0852</td>
<td>0.0361</td>
<td>2.1858</td>
<td>2.3675</td>
</tr>
<tr>
<td>0.1</td>
<td>2.2874</td>
<td>0.0911</td>
<td>0.0439</td>
<td>2.1485</td>
<td>2.4447</td>
</tr>
<tr>
<td>0.01</td>
<td>2.2061</td>
<td>0.5315</td>
<td>0.3827</td>
<td>0.7739</td>
<td>3.7346</td>
</tr>
</tbody>
</table>

4.2.2 Windowed difference statistics

The non-linear part of the graph in figure 4.3 is thought to be produced by the fact that the largest pair separations only occur a small number of times in each surface. And these small numbers of points lying at the large separation distances from the point under consideration lie in a restricted set of directions, rather than the full 360° that we get for the smaller separations. In order to remove this effect, experiments are done with only a small window out of a larger fractal. However the algorithm is allowed to look for pairs outside the window, up to a maximum separation of the size of the window. This means that there are now far more examples of the larger separations, and each point has an equal number of pairing points from all directions at each given any distance. Figure 4.4 shows an example of a plot of $<\log |\Delta V|>$ versus $\log \Delta x$ produced with this method. Comparing this with figure 4.3, it can be seen that the plot now remains linear for most of the graph, as expected.

Tables 4.10 and 4.11 contain the results of taking a single surface of a certain fractal dimension, and calculating the dimension of the surface inside 100 different, non-overlapping windows over the fractal. This is repeated for five surfaces of different
4.2. Fractal dimension calculation using difference statistics

dimensions, with table 4.10 giving the results for 20 x 20 windows inside a 256 x 256 fractal, and table 4.11 giving the results 40 x 40 windows inside a 512 x 512 fractal.

The results show similar average errors to those using difference statistics on the whole of a 64 x 64 surface, showing that the same accuracy can now be obtained from smaller patches of surface. The results show the same trend of an overestimate at low dimensions, becoming an underestimate at higher dimensions, as seen in the previous difference statistics results. However the better straight lines obtained here mean that the calculation no longer has to be restricted to maximum separations of 10% of the image size.

Table 4.12 shows the results from the same method, but this time with the maximum distance between points restricted to a fraction of the window size. 100 20 x 20 windows are again used from a single 256 x 256 fractal. The results show a further improvement in the calculated dimensions as smaller fractions are used for the maximum separation. This suggests that the line is still not perfectly straight, since we would expect these results would stay the same with a straight line.

Using smaller windows and restricting the maximum separation used both amount to throwing away information. In a situation where it is critical that the amount of data is sufficient to produce reliable statistics, it is important not to throw away too
Table 4.10: Windowed fractal dimension calculation for a surface of each of five different fractal dimensions, produced with the FFT method. Statistics refer to ensembles of 100 different windows of size $20 \times 20$ on one surface with the dimension $D$. All five surfaces have size $N = 256$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\bar{D}$</th>
<th>$\bar{\varepsilon}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>2.2059</td>
<td>0.1090</td>
<td>0.0719</td>
<td>2.0597</td>
<td>2.4009</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3230</td>
<td>0.0693</td>
<td>0.0519</td>
<td>2.1414</td>
<td>2.5306</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4647</td>
<td>0.0715</td>
<td>0.0487</td>
<td>2.2689</td>
<td>2.6542</td>
</tr>
<tr>
<td>2.7</td>
<td>2.6102</td>
<td>0.0944</td>
<td>0.0591</td>
<td>2.4559</td>
<td>2.7615</td>
</tr>
<tr>
<td>2.9</td>
<td>2.7394</td>
<td>0.1606</td>
<td>0.0492</td>
<td>2.6176</td>
<td>2.8507</td>
</tr>
</tbody>
</table>

Table 4.11: Windowed fractal dimension calculation for a surface of each of five different fractal dimensions, produced with the FFT method. Statistics refer to ensembles of 100 different windows of size $40 \times 40$ on one surface with the dimension $D$. All five surfaces have size $N = 512$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\bar{D}$</th>
<th>$\bar{\varepsilon}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>2.2098</td>
<td>0.1098</td>
<td>0.0648</td>
<td>2.0999</td>
<td>2.4031</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3375</td>
<td>0.0647</td>
<td>0.0421</td>
<td>2.1905</td>
<td>2.5012</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4872</td>
<td>0.0493</td>
<td>0.0392</td>
<td>2.3283</td>
<td>2.6075</td>
</tr>
<tr>
<td>2.7</td>
<td>2.6368</td>
<td>0.0652</td>
<td>0.0475</td>
<td>2.5053</td>
<td>2.7256</td>
</tr>
<tr>
<td>2.9</td>
<td>2.7669</td>
<td>0.1331</td>
<td>0.0357</td>
<td>2.6674</td>
<td>2.8317</td>
</tr>
</tbody>
</table>

much information. For this reason it is decided that the best compromise solution for fractal dimension calculation is the windowed calculation method used with the maximum possible amount of information. This is when the window is a third of the size of the whole surface, located in the centre of the image, with a maximum separation of also a third of the size of the whole surface. This means that most of the surface is used, except for a small area in the four corners. This setting will be used for all subsequent fractal dimension calculations.
4.3 Anisotropic fractal dimension calculation

Both the FFT method (section 4.1) and the difference statistics method (section 4.2) can produce information on the fractal dimension in different directions. However, the directional information has previously been discarded as an average is taken over all directions to give a single value for the fractal dimension $D$. By retaining the directional information it is possible to calculate the fractal dimension in any direction required.

4.3.1 Anisotropic fractal dimension calculation using FFT

The FFT method takes the fractal surface and produces a spectral density surface in frequency space. This two-dimensional surface contains directional information that could be used for calculating $D$ in different directions. However, the experiments in section 4.1.1 show that for a finite-sized surface the spectral density surface can have large artifacts along the axes, due to the discontinuities introduced when the Fourier transform wraps the surface around to form the infinite surface it requires. These ridges along the axes, as seen in figure 4.2, overpower the directional fractal information and as a result this FFT fractal dimension calculation method is unsuitable for use with anisotropic fractals unless these artifacts

Table 4.12: Windowed fractal dimension calculation for a surface of size $N = 256$ and fractal dimension $D = 2.2$, produced with the FFT method. Statistics refer to ensembles of 100 different windows of size $20 \times 20$ on the surface, varying the maximum distance between the points used in the difference statistics calculation, as a fraction of the window size.

<table>
<thead>
<tr>
<th>Distance Fraction</th>
<th>$\bar{D}$</th>
<th>$\bar{e}$</th>
<th>$\sigma_e$</th>
<th>$D_{\text{min}}$</th>
<th>$D_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2.2604</td>
<td>0.0797</td>
<td>0.0624</td>
<td>2.1028</td>
<td>2.4659</td>
</tr>
<tr>
<td>0.75</td>
<td>2.2556</td>
<td>0.0741</td>
<td>0.0602</td>
<td>2.0961</td>
<td>2.4647</td>
</tr>
<tr>
<td>0.5</td>
<td>2.2468</td>
<td>0.0667</td>
<td>0.0545</td>
<td>2.0909</td>
<td>2.4592</td>
</tr>
<tr>
<td>0.25</td>
<td>2.2259</td>
<td>0.0525</td>
<td>0.0394</td>
<td>2.0909</td>
<td>2.4179</td>
</tr>
<tr>
<td>0.1</td>
<td>2.1699</td>
<td>0.0427</td>
<td>0.0312</td>
<td>2.0678</td>
<td>2.3032</td>
</tr>
</tbody>
</table>
are somehow removed.

4.3.2 Anisotropic fractal dimension calculation using difference statistics

The difference statistics method works by considering each possible pair of points on a surface and compiling the statistics of the relationship between their difference in height ($\Delta V$) and their horizontal distance apart ($\Delta x$). Whereas previously only the length of the line joining the two points was considered, if the angle that this line makes with the $x$-axis is also measured then we also have directional information. By creating bins for different angles, several $\Delta V$ versus $\Delta x$ plots can now be produced, in place of the previous single plot, and a separate value for the fractal dimension can be calculated for each different direction. The smaller the angular bins are, the better the directional information is. However making the bins smaller will also reduce the amount of data in each, reducing the accuracy of each individual plot and hence each value of $D$, as less points go into its forming and the statistics become poorer. It is also possible to define single angles and produce plots for only those pairs that lie exactly in those directions, for example using just the horizontally and vertically separated pairs.

Figure 4.5 takes each of the three methods for creating anisotropic fractals and generates a surface with $(D_x = 2.1, D_y = 2.5)$ and one with $(D_x = 2.5, D_y = 2.1)$ with the same seed with each method. The fractal dimension is then measured with 10 angular bins and the results for each generation method are shown on the same graph.

These graphs show that the surfaces do display different fractal properties in different directions and the difference statistics method is capable of measuring the anisotropy. Repeating the measurements for many different seeds to the generation process, leading to different random fractal surfaces, some examples show larger degrees of anisotropy than others, some examples displaying very little at all. The general trend is as evident in these examples — the first and second methods usually have similar levels of anisotropy, with the stepped method showing slightly more difference between the dimension in the two directions. The third method
4.3. Anisotropic fractal dimension calculation

Figure 4.5: Fractal dimension measurement of anisotropic FFT filtering surfaces.
— leaving out directions greater than $\psi$ away from the axes — displays the most isotropy, as expected, with the $0^\circ$ and $90^\circ$ peaks getting closest to the input values. In contrast the first method — linearly varying the transfer function with the angle — creates the least anisotropy as the fractal is more of an in-between dimension all over rather than distinctly one dimension in one direction and the other dimension at right angles to it.

The same arguments about number of points required for a good estimate apply here as in the previous examples. Except here the number of bins used further reduces the number of points available for each plot. However on the other hand, large bins lose a lot of the directional resolution.
Chapter 5

Constrained fractal reconstruction

Having developed the tools we need, we turn our attention now to the main problem we have to solve, namely that of fractal surface reconstruction. There are two major approaches we can take. We either view the problem as one that requires a globally acceptable solution, or we view the problem as one where things must be locally as correct as possible, even at the expense of some global inconsistencies. We shall concentrate here on the first approach.

5.1 Surface reconstruction

Surface reconstruction is an important part of computer vision and has been the subject of much research. A surface interpolation algorithm fills in the gaps in a dataset with appropriate height values. However, given a set of data points there are many possible surfaces that pass through them, in other words the problem is under-constrained. A popular solution is to define surface interpolation as an optimisation problem and to maximise the smoothness of the surface, while minimising the error in the fit to the data points. This is a regularisation method, and different choices of the smoothness constraint lead to different reconstructed surfaces.

Each possible solution, \(u\) (a 2D grid of height values stacked into a column vector), to the surface interpolation has a probability, \(p(u|d)\). This expresses the likelihood
of this solution, given the data \( d \). This probability is given by Bayes rule

\[
p(u|d) = \frac{p(d|u)p(u)}{p(d)}.
\] (5.1)

The conditional probability, \( p(d|u) \), gives the probability of the data to exist given that the solution, \( u \), is correct. The prior probability, \( p(u) \), is a probability given to the set of height assignments \( u \) by the prior model. This prior model expresses what kind of solution is desired, and in regularisation the prior model is a smoothness constraint. The denominator, \( p(d) \), is the prior probability of the data, which is a constant.

Given that \( p(d) \) is a constant the posterior probability only depends on the two other probabilities:

\[
p(u|d) \propto p(d|u)p(u)
\] (5.2)

Making the assumption that each of these probabilities is given by an exponential to the power of some function, we get:

\[
p(u|d) \propto e^{-E_d(u,d)}e^{-E_p(u)}
\] (5.3)

The goal is to maximise the posterior probability, \( p(u|d) \), in other words, to choose the most probable height assignment, given the data. Therefore, we must minimise the expression \( E_d(u, d) + E_p(u) \). This is the “cost function” or the “energy function”. \( E_d(u, d) \) is the data compatibility term, and expresses how close the solution \( u \) is to the data \( d \). \( E_p(u) \) is the prior constraint term and contains the prior model. In regularisation this term expresses how smooth the solution is.

The various normalisation constants and \( p(d) \) are collected together and form the “partition function” \( Z \). In addition two new parameters, \( \lambda \) and \( T \), both called “temperature” by some people, are also introduced, to allow control of the structure of the configuration space.

Thus, \( p(u|d) \) can be written as:

\[
p(u|d) = \frac{1}{Z}e^{\frac{\lambda}{T}E(u,d)}
\] (5.4)
5.2 The faithfulness to the data term of the cost function

where we define:

\[ E(u, d) = (1 - \lambda)E_d(u, d) + \lambda E_p(u). \tag{5.5} \]

We can understand the role of the first temperature parameter \( T \) as follows: Consider two different configurations, \( u_1 \) and \( u_2 \), with energies \( E_1 > E_2 \). The probabilities of the two configurations are:

\[ p(u_1|d) = \frac{1}{Z} e^{-\frac{E_1}{T}} \quad \text{and} \quad p(u_2|d) = \frac{1}{Z} e^{-\frac{E_2}{T}}. \tag{5.6} \]

This gives

\[ \frac{p(u_1|d)}{p(u_2|d)} = e^{-\frac{\Delta E}{T}} \tag{5.7} \]

where \( \Delta E \equiv E_1 - E_2 > 0 \).

This means that if the temperature parameter, \( T \), has a high value, this ratio will be close to 1 i.e. \( p(u_1|d) \) and \( p(u_2|d) \) will be roughly equal. In other words two very different configurations will appear to be almost equally probable. On the other hand, a low value of the temperature parameter will make this ratio close to 0, which corresponds to \( p(u_2|d) \) having a much higher value than \( p(u_1|d) \). Therefore the configuration with the lower energy has a much greater probability of existence. In this way the temperature parameter gives us control of the configuration space, and allows us to sharpen it or make it more blunt accordingly.

This temperature parameter is particularly useful when the minimum of the energy function \( E(u, d) \) is sought with a stochastic optimisation method like simulated annealing.

The second "temperature" parameter, \( \lambda \) is also sometimes called the Markov parameter or superparameter. This allows us to control the relative importance given to the term that controls faithfulness to the data and the model term.

5.2 The faithfulness to the data term of the cost function

Let us assume that the measured height at a certain point \((i, j)\) is given as \( d_{ij} \), while the true height, which we are trying to recover, is \( u_{ij} \). Let us also assume that the
noise in the measurement process is additive, white and Gaussian, and resulted in a value $n_{ij}$ being added to $u_{ij}$. We have, therefore:

$$d_{ij} = u_{ij} + n_{ij} \implies n_{ij} = d_{ij} - u_{ij}.$$

Since the noise is white, i.e. uncorrelated, the joint probability of having a particular combination of noise values in the various grid points for which data is available, must be given by:

$$p(n) = \frac{1}{Z_1} e^{-\sum_i \sum_j \frac{(d_{ij} - u_{ij})^2}{2\sigma_{ij}^2}}$$

(5.8)

where $\sigma_{ij}$ is the standard deviation of the noise at position $(i, j)$ and $Z_1$ is a normalising constant. For homogeneous noise, $\sigma_{ij}$ is independent of $(i, j)$ and we may simplify the above expression by introducing parameter $c \equiv \frac{1}{\sigma_{ij}^2}$. Further, we recognise in (5.8) the posterior probability of the particular combination of data values to arise, given the true height values of the points. In other words, $p(n)$ is nothing other than $p(d|u)$. It is easy to identify then the $E_d(u, d)$ term of the cost function defined by equation (5.5) as:

$$E_d(u, d) = \frac{1}{2} c \sum_i \sum_j (d_{ij} - u_{ij})^2.$$  

(5.9)

Note that the summation here is only over all the grid points for which measurements are available, not over the whole lattice.

5.3 The prior model term of the cost function

There are a number of possible smoothness constraints [53], implying higher energies for rougher surfaces. The membrane model is a two-dimensional version of the simple elastic string (i.e. a rubber sheet), whose energy is a function of the surface area and thus increases with stretching. This model gives higher probability of existence to those configurations that have smaller gradient magnitudes. In a
discrete lattice the value of the magnitude of the local gradient can be computed as the sum of the squares of the local first differences along the two axes.

The term of the cost function that corresponds to such a prior model can be written as:

\[ E_p(u) = \frac{1}{2} \sum_{i,j} [(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2] \]  \hspace{1cm} (5.10)

The thin plate model is a two-dimensional version of the elastic beam, whose energy is a function of the surface curvature and therefore increases with bending. In one dimension this reduces to the common cubic spline. This model favours configurations that have minimum possible values of the second derivative. In a discrete lattice the second derivative of the function at a certain point can be approximated by the sum of the squares of the local second differences. The term of the cost function that corresponds to such a prior model, has the form:

\[ E_p(u) = \frac{1}{2} \sum_{i,j} \{ (u_{i+1,j} - 2u_{i,j} + u_{i-1,j})^2 + 2(u_{i+1,j+1} - u_{i,j+1} - u_{i+1,j} + u_{i,j})^2 + (u_{i,j+1} - 2u_{i,j} + u_{i,j-1})^2 \} \]  \hspace{1cm} (5.11)

Note that the summation in both equations (5.10) and (5.11) is now over all the lattice points, not only over those for which we have data.

The membrane model requires continuity in the surface only, not in its first (and higher) order partial derivatives. This leads to surfaces which display spikes at the data points and are usually considered not smooth enough. The thin plate model requires continuous first partial derivatives and gives much smoother surfaces, sometimes considered too smooth. A combination of these two models gives a controlled-continuity spline [54], or a spline under tension.

This model also allows the introduction of two types of discontinuity to the surface. A depth discontinuity is where the heights are no longer continuous and the surface is allowed to fracture, and an orientation discontinuity which removes the first order derivative continuity and allows the surface to crease. We shall discuss next how this can be achieved.
The forms of the prior model expressed by equations (5.10) and (5.11) do not allow the preservation of discontinuities in the data. For this reason, in addition to the grid of surface depth \((u_{i,j})\), line variables are added \((l_{i,j} \text{ and } m_{i,j})\) to mark depth discontinuities and crease variables \((n_{i,j})\) to mark orientation discontinuities. The crease variables are located at the nodes and the line variables are located on a dual grid, as shown in figure 5.3. In all three cases a value of 1 marks a discontinuity at that point and 0 means no discontinuity. For each node \((i, j)\), \(l_{i,j}\) marks a discontinuity in the first derivative of the energy function in the direction parallel to the x-axis, located half a grid spacing to the right of the node. Similarly \(m_{i,j}\) marks a discontinuity in the first derivative in the direction parallel to the y-axis, located half a grid spacing below the node. A crease discontinuity \((n_{i,j})\) is located at the node and represents a discontinuity in the second derivative. Continuity strengths \((\beta^x_{i,j}, \beta^y_{i,j}, \beta^{xx}_{i,j}, \beta^{xy}_{i,j}, \text{ and } \beta^{yy}_{i,j})\) are defined in terms of \(l_{i,j}, m_{i,j}, \text{ and } n_{i,j}\) in order to determine which of the surrounding points are included in the energy equation for each point, and with what weight. We define the finite differences and continuity
5.3. The prior model term of the cost function

strengths as follows

\[
\begin{align*}
    u_{i,j}^x &= u_{i+1,j} - u_{i,j} \\
    u_{i,j}^y &= u_{i,j+1} - u_{i,j} \\
    u_{i,j}^{xy} &= u_{i,j}^x - u_{i-1,j}^x = u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \\
    u_{i,j}^{xy} &= u_{i,j+1} - u_{i,j} = u_{i+1,j} - u_{i,j} \\
    u_{i,j}^{yy} &= u_{i,j}^y - u_{i,j-1} = u_{i,j+1} - 2u_{i,j} + u_{i,j-1}
\end{align*}
\]

(5.12)

\[
\begin{align*}
    \beta_{i,j}^x &= (1 - l_{i,j}) \\
    \beta_{i,j}^y &= (1 - m_{i,j}) \\
    \beta_{i,j}^{xy} &= (1 - l_{i,j})(1 - l_{i-1,j})(1 - n_{i,j}) \\
    \beta_{i,j}^{yy} &= (1 - l_{i,j})(1 - l_{i,j+1})(1 - m_{i,j})(1 - m_{i+1,j})(1 - n_{i,j}n_{i+1,j+1})(1 - n_{i+1,j}n_{i,j+1}) \\
    \beta_{i,j}^{yy} &= (1 - m_{i,j})(1 - m_{i,j-1})(1 - n_{i,j})
\end{align*}
\]

(5.13)

These are then used to define the most general prior model term of the energy function, that combines both the thin plate and membrane models, as well as the line and node processes:

\[
E_p(u) = \sum_{i,j} \left\{ w_1[\beta_{i,j}^x(u_{i,j}^x)^2 + \beta_{i,j}^y(u_{i,j}^y)^2] + w_2[\beta_{i,j}^{xy}(u_{i,j}^{xy})^2 + 2\beta_{i,j}^{yy}(u_{i,j}^{yy})^2 + \beta_{i,j}^{yy}(u_{i,j}^{yy})^2] \right\}
\]

(5.14)

where \{w_1, w_2\} define the weighting of the different levels of site interaction.
5.4 Minimising the cost function

The overall energy function can be written as:

$$E(u) = \frac{1}{2} \sum_{i,j} w_i [\beta_{ij}(u_{ij})^2 + \beta_{ij}(u_{ij})^2]$$

$$+ w_2 [\beta_{ij}(u_{ij})^2 + 2\beta_{ij}(u_{ij})^2 + \beta_{ij}(u_{ij})^2]$$

$$+ c(1 - \lambda) \sum_{i,j} (u_{ij} - d_{ij})^2. \quad (5.15)$$

We separate the two sums over indices \((i, j)\) in order to remember that the first sum is over all points of the grid we wish to reconstruct and the second sum is only over the grid points for which the data is available.

If the surface grids are stacked into column vectors \((u_{ij} = u_{iN+j}, \text{ where } N \times N \text{ is the size of the lattice, with } (i, j) \text{ starting from } (0,0))\) this equation can be written in matrix form. The data energy, \(E_d\) becomes:

$$E_d(u, d) = \frac{1}{2}(u - d)^T A_d (u - d) \quad (5.16)$$

where \(A_d\) is a diagonal matrix. Vector \(u - d\) is of size \(N^2 \times 1\). Matrix \(A_d\) is \(N^2 \times N^2\) in size. Along its diagonal it has elements equal to \(c\), at the positions which correspond to grid points for which data are available. At all other positions, which correspond to grid points for which no data are available, we set the elements to 0. This is in order to avoid having terms in the cost function equal to the square of the unknown variables \(u_i\). The inclusion of such terms would mean that we were trying to minimise the sum of the squares of our variables, favouring values close to 0 and thus imposing an extra smoothing constraint on the solution.

The prior energy \(E_p\) in matrix form is:

$$E_p(u) = \frac{1}{2} u^T A_p u. \quad (5.17)$$

Matrix \(A_p\) is \(N^2 \times N^2\) in size, with a rather complicated structure. Its elements depend on the various \(\beta\) parameters.
5.4. Minimising the cost function

The overall energy \( E \) is a combination of \( E_d \) and \( E_p \), with the Markov parameter or super parameter controlling the proportions of each:

\[
E(u, d) = (1 - \lambda)E_d(u, d) + \lambda E_p(u). \tag{5.18}
\]

Substituting 5.16 and 5.17 into 5.18 gives:

\[
\begin{align*}
E(u, d) &= \frac{1}{2} (1 - \lambda)(u^T - d^T)(A_d u - A_d d) + \frac{\lambda}{2} u^T A_p u \\
&= \frac{1}{2} (1 - \lambda)u^T A_d u - \frac{1}{2} (1 - \lambda)u^T A_d d + \frac{1}{2} (1 - \lambda)d^T A_d u \\
&\quad - \frac{1}{2} (1 - \lambda)u^T A_p u \\
&= \frac{1}{2} u^T (1 - \lambda)A_d + \lambda A_p)u - \frac{1}{2} (1 - \lambda) [u^T A_d d + d^T A_d u] \\
&\quad + \frac{1}{2} (1 - \lambda)A^T A_d u. \tag{5.19}
\end{align*}
\]

If we examine the term in the square brackets:

\[
\begin{align*}
u^T A_d d &= \sum_i u_i \sum_j (A_d)_{ij} d_j \\
&= \sum_i \sum_j (A_d)_{ij} u_i d_j \\
&= \sum_i \sum_j (A_d)_{ij} u_i d_j \\
&= d^T A_d u
\end{align*}
\]

The term in the square brackets in equation (5.19) is therefore \( 2u^T A_d d \), giving an overall matrix equation:

\[
\begin{align*}
E(u, d) &= \frac{1}{2} u^T Au - u^T (1 - \lambda)A_d d + s \\
&= \frac{1}{2} u^T Au - u^T b + s \tag{5.20}
\end{align*}
\]

Where we have defined matrix \( A \), constant \( s \) and vector \( b \) as:

\[
\begin{align*}
A &\equiv (1 - \lambda)A_d + \lambda A_p \\
s &\equiv \frac{1}{2} (1 - \lambda)A^T A_d d \\
b &\equiv (1 - \lambda)A_d d. \tag{5.21}
\end{align*}
\]
Matrix $A_p$ can be calculated by finding the correspondence of the terms of the matrix form of the energy equation (i.e. equation (5.17)) and equation (5.14). The details can be found in Appendix B.

This leads to the calculation of the elements of matrix $A_p$. These are then combined with the data terms in $A_d$ to give matrix $A$, which is then defined as given below. Any point $(i, j)$ on the surface corresponds to the row $k = iN + j$ in $A$ and $b$.

\begin{equation}
    a_{k,k} = \lambda w_1 \left( \beta_{i,j}^x + \beta_{i,j-1}^x + \beta_{i,j+1}^x + \beta_{i,j-1}^y \right) \\
    + \lambda w_2 \left( 4\beta_{i,j}^x + \beta_{i+1,j}^x + \beta_{i+1,j+1}^x + 4\beta_{i+1,j}^y + \beta_{i+1,j-1}^y \right) \\
    + 2\beta_{i,j}^{xy} + 2\beta_{i-1,j}^{xy} + 2\beta_{i+1,j-1}^{xy} + 2\beta_{i,j-1}^{xy} + (1 - \lambda)c \\
    \end{equation}

\begin{align}
    a_{k,k+N} &= -\lambda w_1 \beta_{i,j}^x - 2\lambda w_2 \left( \beta_{i+1,j}^x + \beta_{i,j}^x + \beta_{i,j+1}^x + \beta_{i,j-1}^y \right) \\
    a_{k,k-N} &= -\lambda w_1 \beta_{i,j}^x - 2\lambda w_2 \left( \beta_{i,j}^x + 4\beta_{i,j+1}^y + \beta_{i,j-1}^y + \beta_{i,j-1}^y \right) \\
    a_{k,k+1} &= -\lambda w_1 \beta_{i,j}^y - 2\lambda w_2 \left( \beta_{i,j}^y + \beta_{i,j+1}^{xy} + \beta_{i,j}^{xy} + \beta_{i,j}^{xy} \right) \\
    a_{k,k-1} &= -\lambda w_1 \beta_{i,j}^y - 2\lambda w_2 \left( \beta_{i,j}^y + \beta_{i,j-1}^{xy} + \beta_{i,j-1}^{xy} + \beta_{i,j-2}^{xy} \right) \\
    a_{k,k-2N} &= -\lambda w_1 \beta_{i,j}^y \\
    a_{k,k+2N} &= -\lambda w_1 \beta_{i,j}^y \\
    a_{k,k-2} &= -\lambda w_1 \beta_{i,j}^y \\
    a_{k,k+2} &= -\lambda w_1 \beta_{i,j}^y \\
    a_{k,k+1N+1} &= 2\lambda w_2 \beta_{i,j}^{xy} \\
    a_{k,k-1N-1} &= 2\lambda w_2 \beta_{i-1,j}^{xy} \\
    a_{k,k+1N}-1 &= 2\lambda w_2 \beta_{i,j}^{xy} \\
    a_{k,k-1N+1} &= 2\lambda w_2 \beta_{i-1,j}^{xy} \\
    \end{align}

\begin{equation}
    b_k = (1 - \lambda)c d_{i,j} 
\end{equation}

$A$ is therefore a sparse, banded matrix and has at most five elements per row for the membrane model (when $w_2 = 0$) and thirteen elements per row for the thin plate model (when $w_1 = 0$) or for a combination of them both. The number of elements in a row decide how many of the points surrounding the point of interest (the point...
5.4. Minimising the cost function

in the corresponding row of vector \( u \) are included in the calculation of its energy. Therefore, at the boundary of the region and at known discontinuities inside the region fewer points are included.

As the \( \beta \) parameters do not depend on the values of the unknown vector \( u \), function (5.20) is a quadratic with a minimum energy solution occurring when

\[
\frac{\partial E}{\partial u_{i,j}} = Au - b = 0,
\]

i.e. when

\[
Au = b.
\]

The problem therefore is to invert matrix \( A \), which is \( N^2 \times N^2 \) for a lattice of size \( N \times N \), in order to solve the equation:

\[
u = A^{-1}b.
\]

Direct methods (such as Gaussian elimination) could be used to solve this, but these would change the matrix \( A \) by "fill-in", losing its sparseness and order, and produce prohibitively large storage requirements for the large systems involved. Since direct methods are impractical, an iterative scheme must be used. Possible methods include relaxation methods (Jacobi, Gauss-Seidel and successive over-relaxation) and gradient methods (gradient descent, conjugate gradient), with different speeds of convergence and degrees of complexity. The speed can be improved by solving at multiple levels of differing resolution and by implementing the solver on a parallel architecture.

For this implementation Gauss-Seidel is used. This method takes each node in turn and minimises the energy equation for that node, assuming that all other variables are constant. The value of the solution for that node is then updated with the new value and the process is repeated on the next node.

Matrix \( A \) is a symmetric matrix and so for node \((i, j)\) the energy function can be written as:

\[
E(u_{i,j}) = \frac{1}{2} a_{iN+j, iN+j} u_{i,j}^2 + \left( \sum_{(q, r) \in (i,j)} a_{iN+j, iN+r} u_{q,r} - b_{iN+j} \right) u_{i,j} + \text{constant}.
\]

(5.27)
This has a minimum at
\[ u_{i,j}^* = b_{iN+j} - \sum_{(q,r)\neq (i,j)} a_{iN+j,qN+r} u_{q,r} \]  
(5.28)

This method will produce a surface that slowly converges towards the minimum energy solution.

## 5.5 Fractal surface reconstruction using a fractal prior

Normal reconstruction methods do a good job of reconstructing regular, smooth surfaces. However, they are unsuitable for reconstructing natural terrain because the membrane ("rubber sheet") or thin plate prior models do not describe correctly the nature of real landscapes. For a set of sparse data points drawn from a natural terrain, the interpolating surface should not be smooth, but instead it should be rough and random-looking. Fractals are useful for modelling this roughness and Szeliski [50] modified the surface interpolation methods described above so that the reconstructed surface has a fractal nature. This is done by replacing the prior model with a "fractal prior" to reflect different prior expectations of how the surface will appear in the case of natural terrain.

By taking the Fourier transform of the cost function and applying Rayleigh's theorem, Szeliski showed that a membrane prior has a spectral density of the form
\[ S_{\text{mem}}(f) \propto \frac{1}{|f|^2}. \]  
(5.29)

This is in the same form as the spectral density of a fractional Brownian motion (3.6), giving \( H = 0 \) and a fractal dimension \( D = 3 \). For the thin plate prior case the spectral density is
\[ S_{\text{tp}}(f) \propto \frac{1}{|f|^4}. \]  
(5.30)

This corresponds to \( H = 1 \) and a fractal dimension \( D = 2 \). These \( D \)-values fit in with the appearance of the resulting surfaces. The thin plate solutions look very smooth, whereas the membrane surfaces are much more space-filling.

The "fractal prior" proposed by Szeliski uses a combination of these two priors to form a controlled continuity spline, which allows the reconstruction to produce
surfaces with fractal dimensions between 2 and 3. In order to gain better control of the power spectrum over a larger range of frequencies, Szeliski performs the optimisation at multiple levels, with the relative weights given to each of the membrane and thin plate models varying between the different levels. The coarse levels provide control of the low frequency shape of the spectrum, while the fine levels shape the high frequency end of the spectrum. Although proposing it as a method of controlling the dimension, Szeliski does not give a direct relationship between the combination weights at the different levels and the fractal dimension.

Arakawa and Krotkov [1, 2] take Szeliski’s multiresolution fractal reconstruction and attempt to add precise control of the fractal dimension. Szeliski varies the weights — governing the blend of thin plate and membrane models, that are used on each level of the multilevel reconstruction, to give a surface whose fractal dimension is shown to be 2.5. Arakawa and Krotkov redefine these weights so that surfaces with any fractal dimension can be produced. However they find that the rough, fractal texture is only present at the lower frequencies, resulting in a smoother than expected surface. The reason for this is identified as the temperature parameter $T$, which they hold constant at zero at all levels. The temperature controls the variance of the Gaussian noise added at each stage of the optimisation. An attempt is made to empirically calculate the best temperatures to use for each fractal dimension by generating fractal surfaces, sub-sampling them, and then reconstructing them with different temperatures. This produces reasonable results, but the temperatures have no meaning in terms of the fractal dimension.

In order to find a relation between the temperature parameter and fractal dimension Arakawa and Krotkov propose varying the temperatures between the levels in a way analogous with the “successive random additions” method of generating fractals [60]. This method involves adding random values to the elevations, with the variance scaling as the algorithm moves to finer resolutions. Varying the temperatures in a similar way between levels of the multiresolution optimisation algorithm allows the fractalness to be preserved at higher frequencies and gives the temperature parameter a more physical meaning. However this method still requires the precomputation of three parameters before the temperatures for the
different levels can be calculated, although one is claimed to be constant over surfaces created by the same generating process, and another is calculated during their fractal dimension estimation, which is necessary for a dimension-preserving reconstruction.

Section 5.1 described the surface interpolation, which uses a single-level Gauss-Seidel relaxation algorithm. Section 5.5.1 describes the constrained fractal interpolation using this single-level Gauss-Seidel scheme. This allows the reconstruction of surfaces with fractal detail, with limited control over the amount of roughness, but no precise control over the fractal dimension of the reconstructed surface. Section 5.7 presents in detail the method of Arakawa and Krotkov.

5.5.1 Producing a constrained fractal

The posterior distribution, $p(u|d)$, of equation (5.1) defines a set of shapes that are consistent with the given elevation data $(d)$, but which display random, fractal variations, giving them the appearance of natural terrain. The most probable of these shapes is the Maximum a Posteriori (MAP) estimate of the surface we wish to reconstruct. This is the minimum energy solution, corresponding to the smooth surface reconstruction produced previously. Obtaining a fractal surface corresponds to sampling one of these random samples from the posterior distribution. However, calculating the Boltzmann distribution for each node would require a summation over all possible states to obtain the partition function, $Z$. Instead, the Gibbs Sampler [13] algorithm is used. This is an iterative stochastic algorithm where the updating rule is replaced by drawing a sample from the "local" Boltzmann (or Gibbs) distribution. By "local" here we mean the Gibbs distribution where all variables are considered fixed except the variable currently being updated. With Gauss-Seidel the new value of $u$ that locally minimizes the energy is given by (5.28).

If we solve equation (5.28) for

$$b_{iN+j} = \sum_{(q,r) \neq (i,j)} a_{iN+j,qN+r} u_{q,r} + a_{iN+j,iN+j} u_{i,j}^+$$

(5.31)
and substitute it into equation (5.27) and add and subtract \( u_{i,j}^+ \) we may write

\[
E(u_{i,j}) = \frac{1}{2} a_{iN+j,iN+j}(u_{i,j} - u_{i,j}^+)^2 + \text{constant.} \tag{5.32}
\]

This is a quadratic with minimum value \( u_{i,j}^+ \) and a second derivative of \( a_{iN+j,iN+j} \). The local Gibbs distribution is therefore

\[
p(u_{i,j} | u) \propto \exp \left( -\frac{a_{iN+j,iN+j}(u_{i,j} - u_{i,j}^+)^2}{2T} \right). \tag{5.33}
\]

This is a Gaussian with mean \( u_{i,j}^+ \) and variance \( T/a_{iN+j,iN+j} \). This means that a constrained fractal may be produced by adding some Gaussian noise with this mean and variance at each updating step. The temperature \( T \) controls how rough the reconstructed surface will appear.

Figure 5.2 shows the data used to test the surface interpolation and fractal generation functions. There are nine elevation points, with heights between 20 and 100, and two discontinuities, shown here with negative values to distinguish them from the elevation points. Between (8,0) and (8,16) there is a line discontinuity \((l=1)\) and between (15,23) and (31,23) it is a crease discontinuity \((n=1)\). Figure 5.3 shows the results of running the program with \( \{w_1, w_2\} = (0,1) \), i.e. a thin plate solution.
Figures 5.3(a) and 5.3(b) have the temperature parameter $T$ set to zero and therefore show just the surface interpolation with no random components added to the nodes. Figure 5.3(c) shows an example of a fractal surface produced after 1000 iterations with a temperature $T = 30$. It is just an example because every run will produce a slightly different surface, with the amount of variation proportional to the temperature parameter $T$.

Figure 5.4 shows the corresponding output using the membrane cost function: \( \{w_1, w_2\} = \{1, 0\} \). The membrane converges to a minimum energy solution slightly quicker than the thin plate and is more sensitive to the temperature parameter. A value of 5 was found to give a reasonable roughness.

It can be seen that the membrane produces a flatter, less smoothly varying surface, with the elevation points clearly visible as spikes. The thin plate produces a more smooth surface, which is perhaps too smooth.

Figure 5.5(a) shows a Fourier synthesised fractal, of dimension $D = 2.2$ and size $64 \times 64$, which has been sub-sampled so that only 1% of the points are used. These points (figure 5.5(b)) are then taken as the data constraints for a fractal surface reconstruction. The result after 10000 iterations of the reconstruction are shown in figure 5.5(c). This surface was reconstructed using \( (w_1, w_2, T) = (1, 1, 1) \). Figure 5.6 shows another reconstruction, this time for a fractal of dimension $D = 2.35$. In this example we used \( (w_1, w_2, T) = (1, 1, 2) \).

In both cases the reconstructions look very similar to the originals. The reconstruction process has filled the gaps in the data with terrain consistent with the original fractal dimension, and the difference statistics method gives close results for the dimension of both surfaces. However the drawback to these results is that the method of determining the parameters for the reconstruction \( (w_1, w_2, T) \) has more free parameters than just the fractal dimension. These free parameters require “tuning” to achieve the best result — whereas ideally good results should be obtainable by entering just the fractal dimension $D$. In addition, the very small number of points sub-sampled from the original are insufficient to accurately calculate the fractal dimension of the original. For a surfaces as small as $64 \times 64$ a higher percentage of the points would be needed, but real world surfaces are likely
5.5. Fractal surface reconstruction using a fractal prior

Figure 5.3: Thin plate solutions
Chapter 5. Constrained fractal reconstruction

Figure 5.4: Membrane solutions
Figure 5.5: Fractal surface reconstruction, with $D = 2.2$ and $(w_1, w_2, T) = (1, 1, 1)$.

to be much larger than this, giving better statistics for finding the dimension.

5.6 Multiresolution reconstruction

Single level reconstruction methods are slow to converge and do an imperfect job of reproducing the spectrum of the fractal at all scales. Using a multiresolution reconstruction method, in place of the simpler single-level, has the potential to provide solutions to both these problems.

Szeliski [50] outlines three different multi-level methods: the multigrid method,
Figure 5.6: Fractal surface reconstruction, with $D = 2.35$ $(w_1, w_2, T) = (1, 1, 2)$. 
5.6. Multiresolution reconstruction

relative multilevel reconstruction, and hierarchical basis functions. The first two of these are described in the following sections. Both these methods can produce a fractal reconstruction through the use of different blending parameters or weights \( w_1 \) and \( w_2 \) at each of the different resolution levels. At the maximum resolution level \( (l = 0) \) the same weights as in the single level method are used. New weights are then calculated for the coarser levels, according to the desired fractal dimension, \( D \). The coarse levels control the low frequency properties of the surface, while the fine resolution level control the high frequency properties of the surface and its fractal spectrum.

As with the single level reconstructions, it is found that varying the blend of the models used between membrane and thin-plate like this produces a surface with fractal properties at lower frequencies, but with insufficient fine-scale detail. In order to add high frequency fractal detail to the reconstruction a controlled amount of noise is added to each new height estimate during the Gauss-Seidel relaxation using the temperature parameter, \( T \). As with the blending parameters \( (w_1 \) and \( w_2) \) the amount of noise added is varied between the different levels. Larger random fluctuations will be added at the coarser levels since these levels control the large-scale details, while only small fluctuations are added at the fine resolution levels that control the fine-scale detail of the reconstruction.

5.6.1 Multigrid reconstruction

The simplest is the multigrid method, where the reconstruction is performed first on a coarser grid than the desired final result. The solution at this level is passed as a starting point for a reconstruction at a finer level. This process is continued until the full-resolution level is reached. Interpolation is required for mapping the input data constraints from the and the solution data between levels and the energy equations must be redefined for the new levels.

There are also more complex multigrid methods, where the communication between levels is not just a one-way process and the information is passed back-and-forth to ensure equal accuracy at all levels. This approach also allows for the solution of the problem at each level to be executed in parallel, whereas in the simpler
approach the relaxation at each level has to be executed strictly in series. However in cases where the final, fine-level solution is the only one of interest, and the course level data is to be discarded, it is obviously less important to include this extra complexity to ensure the accuracy of the coarse solutions.

A fractal of in-between dimension is produced by varying the blend of thin plate and membrane models used in the energy equation in each level, as described in the previous section.

The spectrum of the resulting fractal is also dependent on the number and size of the different levels and the number of iterations performed at each.

An example multigrid reconstruction is shown in figure 5.7. This surface uses the data points from figure 5.2 and 5000 iterations at the course level, 2500 at the medium and 100 at the fine level.

5.6.2 Relative multilevel reconstruction

An alternative multiresolution method uses a relative representation. In this scheme each level only contains data relevant to its own scale, and the solution is the sum of the different levels. This means that, as before, the coarsest level contains the low frequency, large-scale detail, however the other levels no longer also contain this same information, only holding data that is of too high a resolution to be represented at the coarser levels.

The data compatibility energy is defined on the overall, summed result and the prior energy is defined at each of the different levels. The whole system is then expressed as one matrix equation which is solved by iteration, producing results from each level which, when summed together, give the final surface.

The solution vector $\mathbf{u}$ is related to the individual level vectors $\mathbf{u}_i$ by an interpolation matrix $I_i$:

$$
\mathbf{u} = \sum_{i=1}^{L} I_i \mathbf{u}_i.
$$

(5.34)
5.6. Multiresolution reconstruction

Figure 5.7: Multigrid reconstruction
Each product $J_i u_i$ produces a vector of the same size as the solution vector interpolated from the smaller-sized level vectors using the chosen method of interpolation (e.g., bilinear).

The data compatibility energy is the same as the single-level case (5.16):

$$E_d(u, d) = \frac{1}{2} (u - d)^T A_d (u - d)$$

(5.35)

but each level has its own prior energy:

$$E_p(u_l) = \frac{1}{2} u_l^T A_p^l u_l.$$  

(5.36)

To combine these individual level energy equations with the overall energy equation form of the data constraints the $u_l$ vectors are concatenated to form $\bar{u} = [u_1^T \ldots u_L^T]$ and the interpolation matrices are also similarly concatenated to form $\bar{I} = [I_1 \ldots I_L]$. A composite prior energy matrix is formed from the individual matrices $A_p^l$ defined by (5.36):

\[
A_p = \begin{bmatrix}
A_1^p & 0 & \cdots & 0 \\
0 & A_2^p & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & A_L^p
\end{bmatrix}
\]

(5.37)

Using these the energy function can be written as:

\[
E(\bar{u}) = E_d(\bar{u}, d) + \sum_{l=1}^{L} E_p^l(u_l)
\]

(5.38)

\[
= \frac{1}{2} (\bar{I} \bar{u} - d)^T A_d (\bar{I} \bar{u} - d) + \frac{1}{2} \bar{u}^T \tilde{A} \bar{u} - \bar{u}^T \tilde{b} + c
\]

(5.39)

where $\tilde{A} = \bar{I}^T A_d \bar{I} + \bar{I}^T A_p \bar{I}$, $\tilde{b} = \bar{I}^T A_d d$ and $c$ is a constant. This is a quadratic; the same form as equation (5.20) in the single level solution.

The $A_p$ matrix from the single level reconstruction is used as each $A_p^l$ submatrix in (5.37). Szeliski [50] reports that this produces a singular matrix $\tilde{A}$. To solve this problem a small energy term, $\epsilon$ is added to each level.
The magnitudes of the submatrices $A_p$ are also varied across the levels so that the coarser levels represent the larger features. This is achieved by multiplying the submatrices by a factor $s^{-l}$, where $s > 1$.

$$A_p^I = s^{-l}(A_p + \epsilon)$$

5.7 Controlling the fractal dimension of the reconstruction

The fractal reconstruction method controls the roughness of the surface by varying the blend of thin plate and membrane models used, and varying the temperature parameter $T$ used when adding noise during the optimisation.

Arakawa and Krotkov [1] attempt to link these parameters to the fractal dimension. The power spectral density of the blend of thin plate and membrane prior models is derived by Szeliski [50] by taking the Fourier transform of the prior energy, $E_p(u)$. If $U(f) = \mathcal{F}\{u\}$ then we can write a new energy function, $E_p'(U)$ in terms of $U(f)$.

$$E_p'(U) = \frac{1}{2} \int |H_p(f)|^2 |U(f)|^2 df,$$

where

$$|H_p(f)|^2 = \sum_m w_m |2\pi f|^{2m} = w_1 |2\pi f|^2 + w_2 |2\pi f|^4. \quad (5.42)$$

Since the Fourier transform is a linear operation, if $u$ is a random variable with a Gibbs distribution of energy $E_p(u)$ then $U(f)$ is also a random variable with a Gibbs distribution of energy $E_p'(U)$. Therefore, at a given frequency $f$ the probability distribution is

$$p(U) \propto \exp \left(-\frac{1}{2} |H_p(f)|^2 |U(f)|^2 \right). \quad (5.43)$$

In other words $U(f)$ is a random Gaussian variable with a variance of $|H_p(f)|^{-2}$ and $u$ is a correlated Gaussian noise, with a spectrum

$$S(f) = |H_p(f)|^{-2}. \quad (5.44)$$
Substituting for $H_p(f)$ from (5.42) gives

$$S(f) = \frac{1}{w_1|2\pi f|^2 + w_2|2\pi f|^4}. \quad (5.45)$$

The general spectral density for a fractal is:

$$S(f) \propto |f|^{2D-8}. \quad (5.46)$$

Therefore the membrane term $S \propto |2\pi f|^{-2}$ corresponds to $D = 3.0$ and the thin plate term $S \propto |2\pi f|^{-4}$ has a fractal dimension of $D = 2.0$.

Szeliski shows experimentally that $S(f)$ behaves as $S(f) \propto |f|^{-3}$ in the vicinity of the frequency $f_0$ when

$$w_1 = |2\pi f_0|^2 w_2, \quad (5.47)$$

producing a fractal of dimension $D = 2.5$. Below this frequency the spectral behaviour is that of a membrane ($D = 3.0$) and above it that of a thin plate ($D = 2.0$). Arakawa and Krotkov [1] extend this to produce a general dimension $D$ by setting $S(f_0) = k_1|f_0|^{2D-8}$ (where $k_1$ is a constant) and substituting this into equation (5.45). This produces a new equation for the blend of the two models:

$$w_1 = \frac{f_0^{5-2D}}{(2\pi)^2 k_1} - (2\pi f_0)^2 w_2. \quad (5.48)$$

These are the weights that are used in a single level reconstruction or at the highest resolution level of a multilevel scheme. In a multilevel scheme the final reconstruction is a combination of more than one surface, and its spectrum is therefore the combination of the spectra of the individual levels. The desired result is to have an overall spectrum of the form $S(f) \propto f^{-\beta}$. If we set the relationship between the $H_p$ values at two adjacent levels $l + 1$ and $l$ to be

$$|H_{p,l+1}(f)|^2 = 2^{-\beta}|H_{p,l}(2f)|^2 \quad (5.49)$$

then the spectrum for the two levels is

$$S(f) = |H_{p,l+1}(f)|^{-2} + |H_{p,l}(f)|^{-2} = 2^\beta|H_{p,l}(2f)|^{-2} + |H_{p,l}(f)|^{-2} \propto 2^\beta (2f)^{-\beta} + f^{-\beta} \propto f^{-\beta}.$$
as required. Combining equations (5.42) and (5.49) gives
\[
\begin{align*}
  w_{m+1} |2\pi f|^{2m} &= 2^{-\beta} (w_m |2\pi f|^{2m})
  \\
  w_{m+1} f^{2m} &= 2^{2m-\beta} f^{2m} w_m
\end{align*}
\]

Therefore the relation between the weights used at adjacent levels is

\[
  w_{m+1} = 2^{2m-\beta} f^{2m} w_m.
\]  (5.50)

As well as varying the weights controlling the blend of thin plate and membrane models to use at each level, the roughness of the fractal is also controlled by varying the temperature parameter $T$. Szeliski sets this by trial and error, but Arakawa and Krotkov use an analogy with the successive random additions method of generating fractals to give a relation between the temperature and the dimension. Again this relationship varies over the different levels of the multilevel optimisation.

In the successive random additions method of generating random fractal surfaces, the variance of the noise at level $l$ is given by

\[
  \sigma^2 = \frac{\sigma_0^2 (1 - 2^{4-2D})}{2^{2D-6l}}.
\]  (5.51)

Therefore the temperature at the finest level is set to

\[
  T(D) = k_2 \sigma_0^2 (1 - 2^{4-2D})
\]  (5.52)

where $\sigma_0$ is the standard deviation of the elevation distribution and $k_2$ is a constant, which is claimed to be dependent on the generation process of the fractal.

The temperatures to be used at the coarser levels are then obtained from the temperature at the previous level using the relation

\[
  T_l(D) = T_{l-1}(D) 2^{5-2D}.
\]  (5.53)

Equations (5.48), (5.50), (5.52), and (5.53) now contain five parameters that must be input to the reconstruction, in addition to the value of $D$: $f_0$, the value of $w_2$ at the first level, $k_1$, $\sigma_0^2$, and $k_2$. Some may be set to a fixed value for all surfaces, whilst others may vary with the properties of the surface.
\( \sigma_0^2 \) is defined as the variance of the heights of the fractal surface. This value may therefore be simply calculated from the data at the same time when the fractal dimension of the is estimated.

A value for \( k_1 \) can be fixed by insisting that the reconstruction is a pure thin plate when \( D = 2.0 \). The thin plate produces the smoothest reconstructions, which is what we want at \( D = 2.0 \). Putting \( (w_1, w_2) = (0,1) \) and \( D = 2.0 \) into equation (5.48) produces a value for the constant \( k_1 = \frac{1}{16\pi^2} \). Equation (5.48) then becomes

\[
\begin{align*}
w_1 &= (2\pi)^2 f_0^2 (f_0^{4-2D} - w_2).
\end{align*}
\]

A value of \( w_2 \) must be chosen for the fine-detail level of the reconstruction. \( w_1 \) can then be found from (5.54) and the weights at the other levels can be found from (5.50). The chosen value of the weights must be such that they never become negative at any time, or else the reconstruction does not work. Also, it can be seen from the example plot of \( w_2 \) versus \( w_1 \) for different values of \( f_0 \) in figure (5.8), that as the frequency \( f_0 \) is increased, at values of \( w_2 \) around 1 the value of \( w_1 \) changes direction, first rising and then falling back. Since we want to investigate the behaviour of the reconstructions at different values of \( f_0 \), we need the weights to behave across the range of frequencies. A value of \( w_2 = 0.1 \) is therefore chosen since the relationship is well-behaved in this region.

The frequency constant \( f_0 \), in equation (5.48) effects the weights ratio over the range of \( D \). Setting \( f_0 = 1 \) produces constant weights at all dimensions. Setting \( f_0 > 1 \) produces a negative value for \( w_1 \), which must be a positive value. Therefore, given that negative frequencies have no meaning, the valid range for \( f_0 \) is \( 0 < f_0 < 1 \).

In figures 5.9 and 5.10 \( f_0 \) is set to a different values and the same \( D = 2.1 \) surface is reconstructed using a 3 level multilevel method. \( f_0 \) is set to fractions of the size of the surface, \( N \), ranging from \( \frac{1}{N} \) to \( \frac{6}{N} \). 10% of the original surface data is used in each reconstruction. Table 5.1 shows the fractal dimensions calculated from each of the reconstructions, along with the results from three other values of \( f_0 \), and the results for two rougher surfaces.

The fractal dimension plots (figure 5.9) show that the \( f_0 = \frac{1}{64} \) reconstruction gets the closest match to the original, with the larger values producing lines increas-
5.7. Controlling the fractal dimension of the reconstruction

![Figure 5.8: Plot of $w^2$ versus $w_1$ for $f_0 = 0.1-1.0$ with $D = 2.4$.](image)

ingly far away from that of the original data. Looking at the actual reconstructions in figure 5.10 $f_0 = \frac{1}{64}$ also gives the best-looking reconstruction, with the higher frequency reconstructions becoming increasingly flat, stiff and smooth. The calculated dimensions in table 5.1 also tell a similar story, with $f_0 = \frac{1}{64}$ leading to the $D$ value closest to the original value of 2.1. The higher frequencies lead to large underestimates of $D$. The higher values of $f_0$ can be seen to produce the same smooth reconstructions even for higher dimension surfaces; only $f_0 = \frac{1}{64}$ allows the reconstruction to become suitably rough as the dimension increases. It is clear from these results that $f_0 = \frac{1}{64}$, or more generally $f_0 = \frac{1}{k}$, is the frequency that gives the best reconstructions.

Now that $f_0$, the value of $w_2$ at the first level, $k_1$, and $\sigma_0^2$ have been set, the only free parameter in the reconstruction is $k_2$. Arakawa and Krotkov [1] say that this is a constant dependent on the generation process of the fractal. If this is true, it would be expected that for one generation process, once a suitable value of $k_2$ is found, it should produce good reconstructions for all surfaces sharing the same generation process, whatever their fractal dimension.

To test this, a large number of $512 \times 512$ surfaces are generated using the FFT method, with dimensions between $D = 2.0$ and $D = 2.7$. Starting with 20 different
Figure 5.9: Fractal dimension plots for a surface reconstruction with $D = 2.1$, using various values of $f_0$. 
5.7. Controlling the fractal dimension of the reconstruction

Figure 5.10: Fractal surface reconstructions with $D = 2.1$, using various values of $f_0$. 
Table 5.1: Fractal dimensions of reconstructions of three different fractal surfaces using different values of $f_0$. The original fractal surfaces have $D = 2.1$, $D = 2.2$ and $D = 2.3$, $N = 64$ and they were produced using the Fourier method.

<table>
<thead>
<tr>
<th>Value of $64f_0$</th>
<th>$D_{calc}(orig) = 2.141$</th>
<th>$D_{calc}(orig) = 2.203$</th>
<th>$D_{calc}(orig) = 2.266$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.136</td>
<td>2.172</td>
<td>2.303</td>
</tr>
<tr>
<td>2</td>
<td>2.133</td>
<td>2.162</td>
<td>2.257</td>
</tr>
<tr>
<td>4</td>
<td>2.129</td>
<td>2.150</td>
<td>2.202</td>
</tr>
<tr>
<td>8</td>
<td>2.123</td>
<td>2.136</td>
<td>2.159</td>
</tr>
<tr>
<td>16</td>
<td>2.108</td>
<td>2.116</td>
<td>2.127</td>
</tr>
<tr>
<td>32</td>
<td>2.084</td>
<td>2.089</td>
<td>2.101</td>
</tr>
<tr>
<td>64</td>
<td>2.059</td>
<td>2.063</td>
<td>2.080</td>
</tr>
</tbody>
</table>

surfaces at each dimension, each surface is subsampled by 10% and its dimension ($D_{orig}$) and $\sigma_0^2$ are found from the retained 10% of the data, using the windowed difference statistics method. A reconstruction is then made with these values and some constant value of $k_2$. Finally, the dimension of the resulting surface ($D_{recon}$) is calculated and this is plotted against $D_{orig}$. Figure 5.11 shows the resulting plots using values of $k_2$ from 1 to 750.

The results using $k_2 = 1$ in figure 5.11(a) show the reconstructions have the correct dimension for the very smoothest surfaces; around $D = 2.1$. For higher dimension surfaces the plot displays a tight straight line relation between the input dimension $D_{orig}$ and the reconstructed dimension $D_{recon}$. However this straight line reveals that the reconstructions are slightly smoother than the originals for surfaces above $D = 2.1$ and they become increasingly so as the dimension gets larger. The relation eventually becomes a curve at the very highest dimensions, suggesting an inability to reconstruct the highly random, uncorrelated surfaces with dimensions above $D = 2.6$, at least when using $k_2 = 1$.

The results of increasing $k_2$ to a value of 100 can be seen in figure 5.11(b). The plot is very similar, with a very slight spreading of the points to give a less tight straight line. The point at which the line crosses the $D_{recon} = D_{orig}$ line can also been seen to have increased slightly, to approximately 2.15.
5.7. Controlling the fractal dimension of the reconstruction

Figure 5.11: Plots of $D_{\text{recon}}$ against $D_{\text{orig}}$ using 10% of the original data for $N = 512$ and for various values of the constant $k_2$. 

(a) $k_2 = 1$
(b) $k_2 = 100$
(c) $k_2 = 200$
(d) $k_2 = 300$
(e) $k_2 = 500$
(f) $k_2 = 750$
The subsequent higher values of $k_2$ can be seen to continue these two trends. The points become more scattered — especially at the lower dimensions — and the point at which the curve crosses $D_{\text{recon}} = D_{\text{orig}}$ is raised to a higher dimension. The first effect means that the higher values of $k_2$ cause the reconstruction method to produce surfaces of widely-varying dimensions for input data with the same dimension. The reconstructions also often display much more roughness than the original data from which they were reconstructed. The second effect means that the method produces a reconstructed surface with the same dimension as the input data — a good reconstruction — for a higher value of $k_2$ for input data with higher fractal dimensions.

These results suggest that a constant, low value of $k_2$ is suitable for reconstructing relatively smooth fractal surfaces with a dimension around $D = 2.1$. However, in order for the method to be able to reconstruct a wide range of surfaces of differing dimensions, a varying value of $k_2$ is required.

To this end the plots in figure 5.11 may be used to find an empirical relationship between $k_2$ and $D_{\text{orig}}$ by reading the dimension at which each curve crosses the $D_{\text{recon}} = D_{\text{orig}}$ line. In order to be able to give a value of $k_2$ for any value of $D$, a curve is fit to the points taken from figure 5.11. A fourth order relationship is found to give a good curve — figure 5.12 shows the curve found by fitting five of the points in figure 5.11. For $D = 2.1$ and below a constant value of $k_2 = 1$ is used.

If the reconstruction experiments with the 512 $\times$ 512 surfaces, generated using the FFT method, is repeated with values of $k_2$ taken from the curve shown in 5.12, we get the results shown in 5.13.

For higher dimensions this curve displays significantly more scatter than in figure 5.11(a) — the $k = 1$ plot. However the line is now centred on the $D_{\text{recon}} = D_{\text{orig}}$ line, so the method is producing reconstructions of a roughness closer to that of the original data for surfaces up to about $D = 2.5$ or 2.6, which covers all the normal range of naturally occurring fractal terrain. Surfaces above $D = 2.5$ are still reconstructed as too smooth by this method, but such highly random surfaces are so noisy and uncorrelated as to be unsuitable for terrain modelling anyway.

Taking the same original surface data we can now see the effect that the percentage
5.7. Controlling the fractal dimension of the reconstruction

Figure 5.12: Empirical relation between $k_2$ and $D_{\text{orig}}$.

Figure 5.13: Plot of $D_{\text{recon}}$ against $D_{\text{orig}}$ using 10% of the original data, $N = 512$ and an empirical relation for $k_2(D)$. 
of data kept has on the spread of the points around the central line. Figure 5.14 shows the results when the proportion of the data used as the starting point for the reconstruction is set to 1%, 5%, 15%, and 20%.

Keeping only 1% of the data results in a fairly large spread of the points about the central line. The dashed lines in figure 5.14(a) and figure 5.14(b) show where points with a 5% error would lie. From this it can be seen that the maximum error in the fractal dimension is around 5% when 1% of the data is used for the reconstruction. Using 5% of the data results in errors of around half the size for the lower dimension surfaces, rising to a maximum of 5% for the roughest surfaces.

Increasing the amount of data kept to 15% and 20% produces the plots shown in figures 5.14(c) and 5.14(d). The scatter of the points is much reduced as more of the original data is kept, and the errors in the dimension of the reconstruction are very small for all of these surfaces below $D = 2.5$. This is especially so using 20%, where the curve makes a very good, tight straight line. Above $D = 2.5$ the reconstructions are still consistently too smooth, and the method probably requires a larger value of $k_2$ in this region, should we be interested in reconstructing these sort of fractals and not the smoother terrains that we are interested in here.
5.7. **Controlling the fractal dimension of the reconstruction**

![Plots of $D_{recon}$ against $D_{orig}$ using different percentages of the original data, $N = 512$ and an empirical relation for $k_2(D)$.](image)

Figure 5.14: Plots of $D_{recon}$ against $D_{orig}$ using different percentages of the original data, $N = 512$ and an empirical relation for $k_2(D)$. 

(a) Keeping 1% of the data (dashed lines show 5% error)

(b) Keeping 5% of the data (dashed lines show 5% error)

(c) Keeping 15% of the data

(d) Keeping 20% of the data
Chapter 5. Constrained fractal reconstruction
Chapter 6

Study of the robustness of the method

The tools for generating fractals developed in chapter 3 and for measuring fractal dimensions developed in chapter 4 can now be used to test the quality of the reconstructions produced by the different methods. The simplest test is to generate a random fractal surface of a given dimension, then throw away most of the points and attempt to reconstruct the original from what remains. The advantage of using artificial data is that we have total control over what surfaces we attempt to reconstruct. In particular it is easy to produce surfaces with a wide range of different fractal dimensions, whereas it would be hard to find examples for all dimensions using real data. However, the fractal reconstruction technique is supposed to be used for reconstructing natural terrain, and hence the most important test is how well it reconstructs real, natural surfaces. Here we will use real data from two different sources, sub-sample in different ways, then attempt to reconstruct the original surface. A common real world situation would be to have good, dense data in one region and very sparse data in an adjoining region. The quality of the reconstructions in this sort of situation can be tested by keeping all the data in one area and keeping a random subsample over the rest of the surface. This would involve the method doing a certain amount of extrapolation of the data, a more difficult task than interpolation.

All the experiments will involve calculating the fractal dimension of the surface from the remaining sub-sampled sparse data using the windowed difference statis-
Chapter 6. Study of the robustness of the method

tics method of section 4.2.2. The values given by this method will then provide the input parameters for the reconstructions. The success of the reconstruction can then be determined by comparing it point-by-point with the original surface, and by calculating the fractal properties of the reconstruction and comparing with those of the original surface.

The statistics used for comparing the closeness of the reconstruction to the original are the Chebychev norm, the mean error, the root mean square (RMS) error, and the median of the absolute errors at each point. The Chebychev norm is the maximum absolute error anywhere on the surface, i.e. the maximum difference found between any reconstructed point and its corresponding original value.

6.1 Reconstructing artificial fractals of different dimensions

Six fractals with fractal dimensions between 2.0 and 2.5, and all of size 128 x 128, are produced using the FFT filtering method described in section 3.4. In all cases 5% of the data points are selected at random for use as the input to the reconstruction methods.

The windowed difference statistics method is used to calculate the fractal dimension, $D_{calc}$, of the surface from the 5% of points that are kept. For a 128 x 128 surface 5% gives 819 points, which leads to a reasonably reliable fractal dimension calculation. Much less than this and the data points are too few to give enough statistical information over enough scales and the estimate of the fractal dimension can be very inaccurate.

Table 6.1 shows the results of comparing the original fractals with the reconstructed fractals created using multigrid and the relative multilevel methods. A third set of reconstructions is performed without the Gauss-Seidel random additions step of the multigrid reconstruction. This gives a perfectly smooth reconstruction that doesn’t look at all fractal. This reconstruction will be used as a non-fractal comparison, although the method does still use the fractal data to decide the thin plate and membrane model blending parameters.
6.1. Reconstructing artificial fractals of different dimensions

Table 6.1: Fractal surface reconstructions of surfaces of different fractal dimensions produced with the FFT method. 5% of the original points are used in each reconstruction.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.1798</td>
<td>2.1536</td>
<td>1.6159</td>
<td>-0.0282</td>
<td>0.2670</td>
<td>0.1658</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2234</td>
<td>2.1979</td>
<td>2.1743</td>
<td>-0.0393</td>
<td>0.3531</td>
<td>0.2182</td>
</tr>
<tr>
<td>2.2</td>
<td>2.2763</td>
<td>2.2496</td>
<td>2.6605</td>
<td>-0.0505</td>
<td>0.4395</td>
<td>0.2731</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3379</td>
<td>2.2818</td>
<td>3.0986</td>
<td>-0.0626</td>
<td>0.5356</td>
<td>0.3360</td>
</tr>
<tr>
<td>2.4</td>
<td>2.4062</td>
<td>2.5192</td>
<td>3.6461</td>
<td>-0.0787</td>
<td>0.6727</td>
<td>0.4251</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4780</td>
<td>2.3709</td>
<td>4.4299</td>
<td>-0.1029</td>
<td>0.8844</td>
<td>0.5660</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.1798</td>
<td>2.1150</td>
<td>1.6510</td>
<td>-0.0126</td>
<td>0.1865</td>
<td>0.1099</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2234</td>
<td>2.1612</td>
<td>2.0663</td>
<td>-0.0169</td>
<td>0.2511</td>
<td>0.1524</td>
</tr>
<tr>
<td>2.2</td>
<td>2.2763</td>
<td>2.2084</td>
<td>2.4435</td>
<td>-0.0238</td>
<td>0.3271</td>
<td>0.2042</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3379</td>
<td>2.2523</td>
<td>2.8395</td>
<td>-0.0342</td>
<td>0.4250</td>
<td>0.2709</td>
</tr>
<tr>
<td>2.4</td>
<td>2.4062</td>
<td>2.3016</td>
<td>3.3377</td>
<td>-0.0480</td>
<td>0.5669</td>
<td>0.3656</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4780</td>
<td>2.3646</td>
<td>4.0964</td>
<td>-0.0650</td>
<td>0.7745</td>
<td>0.5026</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.1798</td>
<td>2.0803</td>
<td>1.0378</td>
<td>-0.0158</td>
<td>0.1598</td>
<td>0.0927</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2234</td>
<td>2.0961</td>
<td>1.3055</td>
<td>-0.0210</td>
<td>0.2098</td>
<td>0.1236</td>
</tr>
<tr>
<td>2.2</td>
<td>2.2763</td>
<td>2.1142</td>
<td>1.6596</td>
<td>-0.0283</td>
<td>0.2788</td>
<td>0.1670</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3379</td>
<td>2.1320</td>
<td>2.1203</td>
<td>-0.0385</td>
<td>0.3763</td>
<td>0.2286</td>
</tr>
<tr>
<td>2.4</td>
<td>2.4062</td>
<td>2.1462</td>
<td>2.7023</td>
<td>-0.0522</td>
<td>0.5159</td>
<td>0.3227</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4780</td>
<td>2.1532</td>
<td>3.4190</td>
<td>-0.0700</td>
<td>0.7139</td>
<td>0.4358</td>
</tr>
</tbody>
</table>
Figure 6.1 shows the original $D = 2.2$ fractal, along with the three reconstructions. Figure 6.1 shows the difference statistics plots using 5% of the original points and from the three reconstructions. The gradients of these lines give the fractal dimensions of the surfaces.

The results in section 4 show that, at values close to $D = 2.0$, the FFT fractals had consistently higher calculated fractal dimensions, $D_{calc}$, when measured with the difference statistics method than the input value of $D$. This can again be seen in these results, with all the $D_{calc}$ values below $D = 2.4$ being significant over-
6.1. Reconstructing artificial fractals of different dimensions

Estimates. It is important therefore to note that the reconstruction tries to reconstruct a surface of dimension $D_{\text{calc}}$ rather than $D$, since the method relies on calculating the dimension from the sparse data it is given.

Both the fractal reconstruction methods do a good job of producing a surface with similar measured fractal dimension. The error $D_{\text{calc}} - D_{\text{recon}}$ for the relative multiresolution method, rises from a value of 0.03 at $D = 2.0$ to a large value of 0.11 at $D = 2.5$. And we can see that this method produces over-smooth reconstructions at all dimensions here, and especially at $D = 2.5$. The multigrid method shows errors of between 0.06 and 0.11 across the $D = 2.0 - 2.5$ range, a larger error at $D = 2.0$, but the same at $D = 2.5$.

Far greater errors in the dimension of the reconstruction are seen in the smooth examples, with an underestimate of 0.10 at $D = 2.0$ rising to 0.32 at $D = 2.5$. In other words the smooth reconstructions display very little variation in fractal dimension, whether the initial data is rough or smooth.

Figure 6.1 shows the fractal dimension plots used to generate these results. From
this plot we can see that the original data display a good straight line at the higher scales, but the line becomes more erratic at the smaller scales. This is due to there being a relatively small number of points close together, giving more unreliable statistics at the small-scale level. However the fractal reconstructions do a good job of matching the overall gradient of the line. The smooth reconstruction also matches the gradient at the large-scale end of the plot, but not at the fine-detail end, where the gradient diverges from the others. The figures given in the table are calculated from the average gradients, but we can see here that the smooth reconstruction has the correct dimension at the large scale level — the scale of most of the data used — but has a smaller dimension at the fine-scale level. In other words it fills in the gaps in the fractal data with a smooth interpolation and the remaining fractal characteristic of the smooth reconstructions is just that of the overall shape of the sparse data.

On a point-by-point basis, all the error measures show that the smoother surfaces are reconstructed with significantly smaller errors than the rougher ones. The relative multilevel method gives the worst results by all these measures. The smooth reconstructions give the best results in terms of the RMS error, and the median of the errors for all the surfaces, while the multigrid reconstruction has a slightly better mean error.

It can be seen therefore that the smooth multigrid reconstruction performs slightly better than the fractal multigrid reconstruction in three out of four of these error measures. However figure 6.1 shows that the smooth reconstruction looks very different from the original, as seen by its calculated fractal dimension. Comparing the surfaces visually, the multigrid method seems to give the best reconstruction, by far. If looks are important the multigrid is obviously the winner, and perhaps the small but consistent advantage in the mean error is in fact the most significant error measure, indicative of the random fractal noise improving the average closeness of the interpolated points.

Table 6.2 shows the results of repeating the experiment, but this time keeping 10% of the data points. This improves the accuracy of the initial fractal dimension calculation, and leaves smaller gaps for the reconstruction to fill in. Overall the same
Table 6.2: Fractal surface reconstructions of surfaces of different fractal dimensions produced with the FFT method. 10% of the original points are used in each reconstruction.

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.2098</td>
<td>2.1771</td>
<td>1.3906</td>
<td>-0.0058</td>
<td>0.2335</td>
<td>0.1448</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2559</td>
<td>2.2178</td>
<td>1.6997</td>
<td>-0.0086</td>
<td>0.2930</td>
<td>0.1813</td>
</tr>
<tr>
<td>2.2</td>
<td>2.3107</td>
<td>2.2498</td>
<td>2.0503</td>
<td>-0.0124</td>
<td>0.3559</td>
<td>0.2223</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3733</td>
<td>2.2811</td>
<td>2.5098</td>
<td>-0.0178</td>
<td>0.4390</td>
<td>0.2764</td>
</tr>
<tr>
<td>2.4</td>
<td>2.4412</td>
<td>2.3226</td>
<td>3.2060</td>
<td>-0.0262</td>
<td>0.5693</td>
<td>0.3641</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5113</td>
<td>2.3740</td>
<td>4.2413</td>
<td>-0.0396</td>
<td>0.7674</td>
<td>0.4998</td>
</tr>
</tbody>
</table>

Multigrid reconstructions

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.2098</td>
<td>2.1445</td>
<td>0.9728</td>
<td>-0.0025</td>
<td>0.1514</td>
<td>0.0945</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2559</td>
<td>2.1896</td>
<td>1.2959</td>
<td>-0.0041</td>
<td>0.2036</td>
<td>0.1287</td>
</tr>
<tr>
<td>2.2</td>
<td>2.3107</td>
<td>2.2325</td>
<td>1.6663</td>
<td>-0.0070</td>
<td>0.2680</td>
<td>0.1712</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3733</td>
<td>2.2752</td>
<td>2.1386</td>
<td>-0.0112</td>
<td>0.3572</td>
<td>0.2307</td>
</tr>
<tr>
<td>2.4</td>
<td>2.4412</td>
<td>2.3258</td>
<td>2.8275</td>
<td>-0.0165</td>
<td>0.4911</td>
<td>0.3202</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5113</td>
<td>2.3841</td>
<td>3.7308</td>
<td>-0.0227</td>
<td>0.6868</td>
<td>0.4538</td>
</tr>
</tbody>
</table>

Smooth multigrid reconstructions

<table>
<thead>
<tr>
<th>$D$</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>2.2098</td>
<td>2.0923</td>
<td>0.6586</td>
<td>-0.0059</td>
<td>0.1146</td>
<td>0.0691</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2559</td>
<td>2.1117</td>
<td>0.8660</td>
<td>-0.0083</td>
<td>0.1567</td>
<td>0.0961</td>
</tr>
<tr>
<td>2.2</td>
<td>2.3107</td>
<td>2.1332</td>
<td>1.1496</td>
<td>-0.0116</td>
<td>0.2170</td>
<td>0.1344</td>
</tr>
<tr>
<td>2.3</td>
<td>2.3733</td>
<td>2.1541</td>
<td>1.5396</td>
<td>-0.0162</td>
<td>0.3048</td>
<td>0.1908</td>
</tr>
<tr>
<td>2.4</td>
<td>2.4412</td>
<td>2.1702</td>
<td>2.1180</td>
<td>-0.0223</td>
<td>0.4331</td>
<td>0.2759</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5113</td>
<td>2.1775</td>
<td>2.9185</td>
<td>-0.0302</td>
<td>0.6190</td>
<td>0.4002</td>
</tr>
</tbody>
</table>
basic patterns can be seen across the results; the relative multilevel performs the worst all round; the smooth multigrid the best in all but fractal dimension and mean error; and the fractal multigrid again produces the best overall reconstructions: closest in fractal dimension and with the lowest mean errors.

6.2 Reconstructing a real surface

The next experiments will test how well the reconstructions work with real world data. The first set of real data is a dense, regular grid of depth data for a rock surface under the sea bed. It is a region under the North sea and was provided by BG Technology. The fact that the data points are dense and regular make it a simple task to sub-sample them and to compare how close the reconstruction gets to the original data, as in the previous section with the artificial data. A test surface of size 128 X 128 is cut from the dense data area (see Figure 6.3(a)) to be used for the reconstructions.

Unlike when using artificial data, we can no longer vary the fractal dimension of the surface, this is obviously fixed with each piece of real data. The dimension will likely also vary across the surface more with real data than with the surfaces generated to follow perfectly the statistical definition of a fractal. Real surfaces could also have different dimensions at different scales, or perhaps not be fractal at all at certain scales.

The first experiment with this set of data again takes a random subsample of the data, evenly distributed over the whole surface. The sampling rate is varied from 0.5% to 40%, and the same error statistics are calculated as before.

Table 6.3 gives the numerical results and figure 6.3 shows the original data and the reconstructions. The first thing we notice about the original data is that the surface does not appear very fractal, especially at small scales. Large areas are either completely flat, or else extremely smooth. The larger scale detail is however fractal in nature and the first effect of this is that when sampling only a small number of well-spaced points, the fractal dimension estimate is too high. Thus the 0.5% and
### 6.2. Reconstructing a real surface

Table 6.3: Fractal surface reconstructions of real sub-surface data, using different percentages of the original points to perform the reconstruction.

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{\text{calc}}$</th>
<th>$D_{\text{recon}}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.3528</td>
<td>2.1703</td>
<td>2.7268</td>
<td>-0.1044</td>
<td>0.6684</td>
<td>0.3431</td>
</tr>
<tr>
<td>1</td>
<td>2.3710</td>
<td>2.1836</td>
<td>2.0561</td>
<td>0.0162</td>
<td>0.4476</td>
<td>0.2610</td>
</tr>
<tr>
<td>5</td>
<td>2.1622</td>
<td>2.0919</td>
<td>1.5754</td>
<td>0.0027</td>
<td>0.1675</td>
<td>0.0700</td>
</tr>
<tr>
<td>10</td>
<td>2.2446</td>
<td>2.1422</td>
<td>1.4937</td>
<td>0.0050</td>
<td>0.1604</td>
<td>0.0782</td>
</tr>
<tr>
<td>20</td>
<td>2.2275</td>
<td>2.1444</td>
<td>1.0683</td>
<td>0.0035</td>
<td>0.1100</td>
<td>0.0574</td>
</tr>
<tr>
<td>40</td>
<td>2.2318</td>
<td>2.1521</td>
<td>0.8544</td>
<td>-0.0004</td>
<td>0.0905</td>
<td>0.0476</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{\text{calc}}$</th>
<th>$D_{\text{recon}}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.3528</td>
<td>2.0509</td>
<td>3.1897</td>
<td>-0.1192</td>
<td>0.6851</td>
<td>0.2717</td>
</tr>
<tr>
<td>1</td>
<td>2.3710</td>
<td>2.0504</td>
<td>1.7386</td>
<td>0.0196</td>
<td>0.3606</td>
<td>0.1811</td>
</tr>
<tr>
<td>5</td>
<td>2.1622</td>
<td>2.0811</td>
<td>1.5763</td>
<td>0.0032</td>
<td>0.1564</td>
<td>0.0471</td>
</tr>
<tr>
<td>10</td>
<td>2.2446</td>
<td>2.0996</td>
<td>1.5218</td>
<td>0.0036</td>
<td>0.1193</td>
<td>0.0285</td>
</tr>
<tr>
<td>20</td>
<td>2.2275</td>
<td>2.1193</td>
<td>1.0925</td>
<td>0.0023</td>
<td>0.0819</td>
<td>0.0170</td>
</tr>
<tr>
<td>40</td>
<td>2.2318</td>
<td>2.1329</td>
<td>0.8978</td>
<td>-0.0003</td>
<td>0.0656</td>
<td>0.0117</td>
</tr>
</tbody>
</table>
Chapter 6. Study of the robustness of the method

(a) Real subsurface data
(b) Multigrid reconstruction using 0.5% of the data
(c) Multigrid Reconstruction using 5% of the data
(d) Multigrid Reconstruction using 10% of the data
(e) Smooth multigrid Reconstruction using 0.5% of the data
(f) Smooth multigrid Reconstruction using 5% of the data

Figure 6.3: Fractal surface reconstructions of real sub-surface data.
1% reconstructions add far too much small-scale roughness to the surface, and the resultant errors are large.

When more data points are kept the dimension measured is smaller and the reconstructions are better. However the reconstruction method assumes the same fractal dimension holds over the whole surface and at all scales, and clearly in this case the original data points are not fractal at fine scales. The surface has large, perfectly flat regions, with sharp creases. Only a small part in the centre looks truly fractal all the way down to the finest scales. Since the fractal reconstruction assumes uniform fractal characteristics, it keeps on adding small detail variations where the original shows none and as a result, the smooth reconstructions actually look more like the original here.

The creases in the surface are totally lost from the reconstructions when very few data points are kept. As the amount of data kept is increased, the creases appear in the reconstructions, but they are never as sharp and straight as in the original since the smoothness term of the reconstruction smoothes them out.

The type of fractal assumed by the reconstruction is obviously different in character to what we have in this example of real data. Using our fractal reconstruction here does not produce a result closer to the original than a non-fractal reconstruction technique. To achieve that we would need to be able to instruct the reconstructed surface to be a fractal over only a defined range of scales, rather than all scales as we have now, or possibly also to be able to vary the dimension over the surface. Another possibility to improve these reconstructions would be to add in crease discontinuities. This however would raise the problem of having to locate the creases in the sparse data and to somehow extend them into the regions with no data.

Examination of the error surfaces for these results shows that the errors are only significantly better for the smooth reconstruction around the flat sections of the original data, and it is these areas that give rise to the better overall results for the smooth reconstruction with these data points. The fractal dimension calculation also gives an average value over the whole surface, and so the fractal reconstruction is therefore too smooth in the rough, valley areas and too rough on the perfectly flat regions.
Chapter 6. Study of the robustness of the method

Table 6.4: Fractal surface reconstructions of real USGS terrain data, using different percentages of the original points to perform the reconstruction.

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.1148</td>
<td>2.0687</td>
<td>523.2063</td>
<td>-2.7175</td>
<td>155.1344</td>
<td>84.6165</td>
</tr>
<tr>
<td>1</td>
<td>1.9877</td>
<td>2.1042</td>
<td>493.0034</td>
<td>-0.8459</td>
<td>93.5179</td>
<td>46.1423</td>
</tr>
<tr>
<td>5</td>
<td>2.2807</td>
<td>2.1939</td>
<td>336.9905</td>
<td>1.4040</td>
<td>53.4641</td>
<td>29.9700</td>
</tr>
<tr>
<td>10</td>
<td>2.1795</td>
<td>2.1661</td>
<td>275.3535</td>
<td>0.5136</td>
<td>30.1430</td>
<td>14.9202</td>
</tr>
<tr>
<td>20</td>
<td>2.2134</td>
<td>2.1796</td>
<td>258.8085</td>
<td>-0.1438</td>
<td>23.6699</td>
<td>13.7959</td>
</tr>
<tr>
<td>40</td>
<td>2.1931</td>
<td>2.1768</td>
<td>163.9338</td>
<td>-0.1395</td>
<td>17.0423</td>
<td>9.7620</td>
</tr>
</tbody>
</table>

Multigrid reconstructions

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.1148</td>
<td>2.0671</td>
<td>532.0375</td>
<td>-9.2454</td>
<td>153.1331</td>
<td>85.0642</td>
</tr>
<tr>
<td>1</td>
<td>1.9877</td>
<td>2.0842</td>
<td>493.0034</td>
<td>-0.8459</td>
<td>93.5179</td>
<td>46.1423</td>
</tr>
<tr>
<td>5</td>
<td>2.2807</td>
<td>2.0982</td>
<td>408.6667</td>
<td>1.4253</td>
<td>40.0471</td>
<td>18.4631</td>
</tr>
<tr>
<td>10</td>
<td>2.1795</td>
<td>2.1026</td>
<td>277.4659</td>
<td>0.2937</td>
<td>25.4512</td>
<td>10.4000</td>
</tr>
<tr>
<td>20</td>
<td>2.2134</td>
<td>2.1112</td>
<td>263.4738</td>
<td>-0.0205</td>
<td>16.9942</td>
<td>7.0869</td>
</tr>
<tr>
<td>40</td>
<td>2.1931</td>
<td>2.1070</td>
<td>172.3474</td>
<td>0.0769</td>
<td>12.9628</td>
<td>5.0741</td>
</tr>
</tbody>
</table>

Smooth multigrid reconstructions

The second example of real data we will use is some terrain data taken from the US Geological Survey (USGS). A 128 x 128 section of mountainous terrain (from near Billings, Montana) is selected at random and is sub-sampled and reconstructed, as before. The original data is shown in figure 6.4(a).

Table 6.4 shows the results of comparing the original surface with the reconstructions at the different sampling rates. Figure 6.4 shows the original data and three examples of the reconstructions.

As seen before the extremely low sampling rates produce unreliable fractal dimension estimates, and while the 0.5% value is quite good here, the value calculated at 1% is an impossible value, lower than 2.0, the Euclidean dimension of a 2D surface. A value of $D = 2.0$ is used for this reconstruction.
6.2. Reconstructing a real surface

(a) Real USGS data

(b) Smooth multigrid reconstruction using 5% of the data

(c) Multigrid Reconstruction using 5% of the data

(d) Multigrid Reconstruction using 10% of the data

Figure 6.4: Fractal surface reconstructions of real USGS data.
Chapter 6. Study of the robustness of the method

These data points have a more uniform roughness than the BG data, but still have quite a few flat areas, on top of the ridges. The error statistics again show the smooth reconstruction giving the smaller errors in most cases, though they are all close and the fractal reconstruction does better for some. The fractal reconstruction again matches the fractal dimension of the original more closely in most cases.

From figure 6.4(b) we see that the smooth reconstruction is indeed too smooth in appearance, but is otherwise a good, close reconstruction. The 5% fractal reconstruction suffers from a too high dimension estimate, and is too uniformly noisy. The 10% reconstruction is much better however, and looks far more true to the original than the other reconstructions. It does however apply the roughness all over the fractal, including the small regions where the original has flat areas. This will adversely affect the error measures, even though it is a better reconstruction overall.

6.3 Reconstructing a real surface with a mask

The next experiments take the real data and keep all the points inside a mask, while keeping none or only a random sample outside the mask. Mask 1 (figure 6.5(e)) attempts to simulate the situation where dense data points are available for one region, but only a small number of points are known in an adjoining region. When 0% of the data points are kept outside the mask, the only points at which heights are known lie on four straight lines extending away from the dense data region. This could occur if one region is easy and cheap to completely survey, while in the other region it is only possible to do a few passes across it. The reconstruction will assume that the surface in the half where it does the reconstructing has the fractal properties as those it calculates from the dense region.

Table 6.5 and figure 6.5 show the results using the BG sub-surface data (figure 6.3(a)). On the left of figure 6.5 are two of the reconstructions, while on the right are the corresponding error surfaces. These are calculated by subtracting the height of each point on the reconstructed surface from the height of the same point on the original.
6.3. Reconstructing a real surface with a mask

Figure 6.5: Fractal surface reconstructions of real sub-surface data, using a mask.
Chapter 6. Study of the robustness of the method

Table 6.5: Fractal surface reconstructions of real sub-surface data, keeping all points within a masked region (mask 1) and different percentages of points outside the mask.

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.1940</td>
<td>2.1389</td>
<td>5.1944</td>
<td>-0.0989</td>
<td>0.9477</td>
<td>0.3518</td>
</tr>
<tr>
<td>0.1</td>
<td>2.1941</td>
<td>2.1388</td>
<td>4.8350</td>
<td>-0.0529</td>
<td>0.8416</td>
<td>0.3355</td>
</tr>
<tr>
<td>0.5</td>
<td>2.1940</td>
<td>2.1377</td>
<td>1.9684</td>
<td>-0.1088</td>
<td>0.4437</td>
<td>0.2140</td>
</tr>
<tr>
<td>1</td>
<td>2.1939</td>
<td>2.1376</td>
<td>1.7801</td>
<td>-0.0310</td>
<td>0.3465</td>
<td>0.1513</td>
</tr>
<tr>
<td>5</td>
<td>2.1948</td>
<td>2.1485</td>
<td>1.3616</td>
<td>-0.0019</td>
<td>0.1745</td>
<td>0.0750</td>
</tr>
<tr>
<td>10</td>
<td>2.1950</td>
<td>2.1507</td>
<td>1.3848</td>
<td>0.0053</td>
<td>0.1425</td>
<td>0.0639</td>
</tr>
<tr>
<td>20</td>
<td>2.1971</td>
<td>2.1624</td>
<td>1.0916</td>
<td>0.0012</td>
<td>0.1100</td>
<td>0.0614</td>
</tr>
<tr>
<td>40</td>
<td>2.1974</td>
<td>2.1711</td>
<td>0.8949</td>
<td>0.0010</td>
<td>0.1047</td>
<td>0.0610</td>
</tr>
</tbody>
</table>

Comparing the figures in table 6.5 with those for the same data in table 6.3, we can see that when we keep 0% of the data outside the mask, the errors are similar in size to the case when we have only a very small percentage of the data, evenly spread over the whole region. This is despite having half the data points in this case.

Looking at the reconstructions even the 0% case appears to be a reasonable reconstruction. The data points along the four straight lines of the mask appear sufficient to define the overall shape in this region. However the reconstruction again adds data with a uniform roughness, whereas the original shows more localised areas of higher fractal roughness and creasing, interspaced with large flat areas. The reconstructions can't recreate this.

Keeping more data outside the mask leads to a better reconstruction, with the creases included. However these creases are still not as sharp as in the original surface.

Looking at the error surfaces we see the biggest errors occur the furthest away from any data points, as we would expect. Where the area with no data is bounded by two of the mask lines, the errors are kept quite small. However where there are no
6.3. Reconstructing a real surface with a mask

Table 6.6: Fractal surface reconstructions of real US terrain data, keeping all points within a masked region (mask 1) and using different percentages of points outside the mask.

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.1854</td>
<td>2.1768</td>
<td>763.1922</td>
<td>20.7243</td>
<td>194.9586</td>
<td>94.9316</td>
</tr>
<tr>
<td>0.1</td>
<td>2.1855</td>
<td>2.1700</td>
<td>659.0296</td>
<td>26.1147</td>
<td>131.6531</td>
<td>81.7670</td>
</tr>
<tr>
<td>0.5</td>
<td>2.1859</td>
<td>2.1669</td>
<td>659.0296</td>
<td>26.1147</td>
<td>131.6531</td>
<td>81.7670</td>
</tr>
<tr>
<td>1</td>
<td>2.1857</td>
<td>2.1653</td>
<td>517.3035</td>
<td>4.7142</td>
<td>92.5023</td>
<td>42.2523</td>
</tr>
<tr>
<td>5</td>
<td>2.1871</td>
<td>2.1714</td>
<td>381.5060</td>
<td>3.5075</td>
<td>43.1043</td>
<td>20.7411</td>
</tr>
<tr>
<td>10</td>
<td>2.1889</td>
<td>2.1742</td>
<td>271.1337</td>
<td>2.0363</td>
<td>30.4604</td>
<td>15.3533</td>
</tr>
<tr>
<td>20</td>
<td>2.1922</td>
<td>2.1768</td>
<td>256.1135</td>
<td>0.4195</td>
<td>22.1884</td>
<td>12.2236</td>
</tr>
<tr>
<td>40</td>
<td>2.1936</td>
<td>2.1783</td>
<td>164.1366</td>
<td>-0.2982</td>
<td>17.1632</td>
<td>9.8828</td>
</tr>
</tbody>
</table>

When more data points are kept we see that the errors are more evenly distributed, now with peaks around the rougher, creased areas, as before. There are also still significant areas of high error near the edges where small extrapolations are required.

This experiment is repeated with the USGS data (figure 6.4(a)). Table 6.6 shows the error data, while figure 6.6 shows two of the reconstructions and their corresponding error surfaces.

The more uniformly fractal nature of this terrain again leads to a better-looking reconstructions. However, the original terrain has large valleys running parallel to the lines of data sampling on the mask. These are large changes in the heights in the
Chapter 6. Study of the robustness of the method

Figure 6.6: Fractal surface reconstructions of real USGS data, using mask 1.
Table 6.7: Fractal surface reconstructions of real sub-surface data, keeping all points within a masked region (mask 2) and different percentages of points outside the mask.

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.1833</td>
<td>2.1427</td>
<td>3.3806</td>
<td>0.0689</td>
<td>0.8763</td>
<td>0.3301</td>
</tr>
<tr>
<td>0.1</td>
<td>2.1834</td>
<td>2.1463</td>
<td>2.7393</td>
<td>0.0305</td>
<td>0.6301</td>
<td>0.2625</td>
</tr>
<tr>
<td>0.5</td>
<td>2.1840</td>
<td>2.1514</td>
<td>2.1301</td>
<td>-0.0474</td>
<td>0.3430</td>
<td>0.1374</td>
</tr>
<tr>
<td>1</td>
<td>2.1841</td>
<td>2.1510</td>
<td>2.0681</td>
<td>-0.0546</td>
<td>0.3193</td>
<td>0.1086</td>
</tr>
<tr>
<td>5</td>
<td>2.1861</td>
<td>2.1520</td>
<td>1.3386</td>
<td>-0.0036</td>
<td>0.1367</td>
<td>0.0442</td>
</tr>
<tr>
<td>10</td>
<td>2.1876</td>
<td>2.1525</td>
<td>1.3159</td>
<td>-0.0019</td>
<td>0.1101</td>
<td>0.0357</td>
</tr>
<tr>
<td>20</td>
<td>2.1897</td>
<td>2.1538</td>
<td>1.0814</td>
<td>-0.0011</td>
<td>0.0845</td>
<td>0.0321</td>
</tr>
<tr>
<td>40</td>
<td>2.1889</td>
<td>2.1554</td>
<td>0.7698</td>
<td>0.0007</td>
<td>0.0711</td>
<td>0.0289</td>
</tr>
</tbody>
</table>

direction perpendicular to the sampling lines and the reconstruction completely misses them since it has very limited information in this direction. This leads to ridges of large errors on the error surface, where the valleys are flattened out.

We can see that the surface produced when 5% of the data points outside the mask are kept has sufficient information to reconstruct the valleys well and its biggest errors occur at the edges where small extrapolations are needed.

Finally another mask is created and used in the same way with both sets of data. Mask 2 is shown in figure 6.7(e). This mask keeps all the data in a central band and keeps a few lines of data on both sides, but with the lines at the extreme edges of the surface on one side of the central band of dense data. This forces extrapolation on one side and guarantees no extrapolation on the other side.

Table 6.7 and figure 6.7 show the results for the BG sub-surface data, and table 6.8 and figure 6.8 show the results for the USGS data.

In both cases we see the biggest errors where difficult extrapolation is required. The areas where three sides are bounded by lines of data display very similar errors to those that are bounded on all four sides, the worst errors occurring where only two sides are bounded.
Figure 6.7: Fractal surface reconstructions of real sub-surface data, using a mask.
6.3. **Reconstructing a real surface with a mask**

(a) Multigrid reconstruction using 0% of the data outside the mask

(b) Error surface for the 0% reconstruction

(c) Multigrid reconstruction using 5% of the data outside the mask

(d) Error surface for the 5% reconstruction

Figure 6.8: Fractal surface reconstructions of real USGS data, using mask 2.
Chapter 6. Study of the robustness of the method

Table 6.8: Fractal surface reconstructions of real US terrain data, keeping all points within a masked region (mask 2) and using different percentages of points outside the mask.

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{calc}$</th>
<th>$D_{recon}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.1811</td>
<td>2.1809</td>
<td>938.7159</td>
<td>13.8037</td>
<td>198.8082</td>
<td>96.1670</td>
</tr>
<tr>
<td>0.1</td>
<td>2.1812</td>
<td>2.1792</td>
<td>561.9097</td>
<td>-10.4536</td>
<td>154.4242</td>
<td>86.5510</td>
</tr>
<tr>
<td>0.5</td>
<td>2.1813</td>
<td>2.1790</td>
<td>519.6597</td>
<td>-11.0152</td>
<td>108.4368</td>
<td>52.5530</td>
</tr>
<tr>
<td>1</td>
<td>2.1815</td>
<td>2.1792</td>
<td>384.7714</td>
<td>2.0868</td>
<td>72.5298</td>
<td>37.5896</td>
</tr>
<tr>
<td>5</td>
<td>2.1835</td>
<td>2.1780</td>
<td>445.0278</td>
<td>1.6115</td>
<td>38.1156</td>
<td>16.7670</td>
</tr>
<tr>
<td>10</td>
<td>2.1816</td>
<td>2.1779</td>
<td>291.0460</td>
<td>0.3543</td>
<td>25.2960</td>
<td>12.7693</td>
</tr>
<tr>
<td>20</td>
<td>2.1832</td>
<td>2.1786</td>
<td>274.9730</td>
<td>0.1046</td>
<td>19.3802</td>
<td>10.2739</td>
</tr>
<tr>
<td>40</td>
<td>2.1865</td>
<td>2.1791</td>
<td>162.2347</td>
<td>-0.1276</td>
<td>15.7378</td>
<td>8.6471</td>
</tr>
</tbody>
</table>

The sub-surface data reconstructions again show large errors along the central ridge where the original surface is more rough and creased than the rest of the terrain. The USGS reconstructions also repeat the filling-in of the valleys missed by the data points.
Chapter 7

Anisotropic fractal reconstruction

The fractal techniques used so far have all assumed that fractal properties of a surface are the same in all directions. However a real-world fractal surface is unlikely to be perfectly isotropic, and will often have significantly different fractal properties in different directions. Chapter 4 describes how we can adapt the fractal dimension calculation methods in order to reveal this anisotropic nature of a fractal, producing directional information. In this chapter we will attempt to take this information and incorporate it into the fractal reconstruction.

7.1 Introducing anisotropy to the fractal reconstruction technique

The isotropic fractal reconstruction method makes the reconstructed surface into a fractal through two techniques. These are varying the blending parameters, which control the amount of thin-plate and membrane models to use at each level, and varying the magnitude of the random noise element added to each update of the optimisation. Ideally both would be modified to give different dimensions in different directions. It is not obvious how the adding of noise can be done in an anisotropic fashion, however the blending parameters can easily be separated out into $x$ and $y$ components to give an anisotropic fractal reconstruction.

We do this by going back to the equations for minimising the cost function in chapter 5.4. Equation (5.14) gives the prior model term of the energy equation, $E_p(u)$. 

105
The parameters $w_1$ and $w_2$ define the combination of membrane and thin plate models. In order to achieve different combinations in the $x$ and $y$ directions $w_1$ is replaced by $w_{1x}$ and $w_{1y}$ and $w_2$ by $w_{2x}$ and $w_{2y}$.

The $x$ direction membrane parameter, $w_{1x}$, acts on the first order $x$ direction finite difference $u_{i,j}^x$ found in the membrane term of equation (5.14), while the $y$ direction membrane parameter, $w_{1y}$, acts on the corresponding first order $y$ direction finite difference $u_{i,j}^y$.

Similarly for the second thin plate part of the equation, the $x$ direction thin plate parameter, $w_{2x}$, acts on the second order $x$ direction finite difference $u_{i,j}^{xx}$ found in the thin plate term and the $y$ direction thin plate parameter, $w_{2y}$, acts on the second order $y$ direction finite difference $u_{i,j}^{yy}$. The cross term $v_{i,j}^{xy}$ includes differences in both directions and is multiplied by both $w_{2x}$ and $w_{2y}$. However a cross term of $w_{2x}w_{2y}$ means the equation does not reduce to the isotropic equation in the case where $w_{2x} = w_{2y} = w_2$ and $w_{1x} = w_{1y} = w_1$. Therefore a cross term of

$$w_{2x}w_{2y} = \frac{w_{2x}w_{2y}}{\sqrt{(w_{2x}^2 + w_{2y}^2)}}$$

is used. Applying these new anisotropic blending parameters to equation (5.14) gives a new version of the general prior model term of the energy function:

$$E_p(u) = \sum_{i,j} \left\{ w_{1x}\beta_{i,j}^x(u_{i,j}^x)^2 + w_{1y}\beta_{i,j}^y(u_{i,j}^y)^2 + \frac{w_{2x}\beta_{i,j}^{xx}(u_{i,j}^{xx})^2}{2} + \frac{w_{2y}\beta_{i,j}^{yy}(u_{i,j}^{yy})^2}{2} + \frac{w_{2x}w_{2y}\beta_{i,j}^{xy}(u_{i,j}^{xy})^2}{2} \right\}$$

This energy is then used to form the prior matrix $A_p$ using the same method as described in chapter 5.4. This leads to a new set of thirteen equations for the non-zero elements of each row of matrix $A$, given below.
7.1. Introducing anisotropy to the fractal reconstruction technique

\[ a_{k,k} = \lambda w_{1x} (\beta_{i,j}^x + \beta_{i-1,j}^x) + \lambda w_{1y} (\beta_{i,j}^y + \beta_{i,j-1}^y) \]  
\[ + \lambda w_{2x} \left( 4\beta_{i,j}^{xx} + \beta_{i-1,j}^{xx} + \beta_{i+1,j}^{xx} \right) + \lambda w_{2y} \left( 4\beta_{i,j}^{yy} + \beta_{i,j-1}^{yy} + \beta_{i,j+1}^{yy} \right) \]  
\[ + \lambda w_{2x} w_{2y} \left( 2\beta_{i,j}^{xy} + 2\beta_{i-1,j-1}^{xy} + 2\beta_{i,j-1}^{xy} + 2\beta_{i-1,j}^{xy} \right) \]  
\[ + (1 - \lambda)c \]  
\[ a_{k,k+N} = -\lambda w_{1x} \beta_{i,j}^x - 2\lambda w_{2x} (\beta_{i,j}^{xx} + \beta_{i+1,j}^{xx}) - 2\lambda w_{2x} w_{2y} (\beta_{i,j}^{xy} + \beta_{i,j+1}^{xy}) \]  
\[ a_{k,k-N} = -\lambda w_{1x} \beta_{i,j}^x - 2\lambda w_{2x} (\beta_{i,j}^{xx} + \beta_{i-1,j}^{xx}) - 2\lambda w_{2x} w_{2y} (\beta_{i,j}^{xy} + \beta_{i,j-1}^{xy}) \]  
\[ a_{k,k+1} = -\lambda w_{1y} \beta_{i,j}^y - 2\lambda w_{2y} (\beta_{i,j}^{yy} + \beta_{i,j+1}^{yy}) - 2\lambda w_{2x} w_{2y} (\beta_{i,j}^{xy} + \beta_{i,j+1}^{xy}) \]  
\[ a_{k,k-1} = -\lambda w_{1y} \beta_{i,j}^y - 2\lambda w_{2y} (\beta_{i,j}^{yy} + \beta_{i,j-1}^{yy}) - 2\lambda w_{2x} w_{2y} (\beta_{i,j}^{xy} + \beta_{i,j-1}^{xy}) \]  
\[ a_{k,k-2N} = w_{2x} \lambda \beta_{i,j}^{xx} \]  
\[ a_{k,k+2N} = w_{2x} \lambda \beta_{i,j}^{xx} \]  
\[ a_{k,k-2} = w_{2y} \lambda \beta_{i,j}^{yy} \]  
\[ a_{k,k+2} = w_{2y} \lambda \beta_{i,j}^{yy} \]  
\[ a_{k,k+N+1} = 2w_{2x} w_{2y} \lambda \beta_{i,j}^{xy} \]  
\[ a_{k,k-N-1} = 2w_{2x} w_{2y} \lambda \beta_{i,j}^{xy} \]  
\[ a_{k,k+N-1} = 2w_{2x} w_{2y} \lambda \beta_{i,j}^{xy} \]  
\[ a_{k,k-N+1} = 2w_{2x} w_{2y} \lambda \beta_{i,j}^{xy} \]  
\[ (7.4) \]

For a single level reconstruction the optimisation can now be performed in the same way as before. For a multigrid reconstruction the above is used to form the matrix at each successive level in turn, and for a relative multiresolution reconstruction it is used to form the larger matrix including all of the levels.

The blending parameters at each level of the two multilevel methods are also related in the same way. We replace \( w_1 \) and \( w_2 \) with \( w_{1x} \) and \( w_{1y} \) and \( w_{2x} \), \( w_{2y} \) and \( w_{2x} w_{2y} \) in equations (5.48) and (5.50). At the maximum resolution level the parame-
ters are now given by

\[ w_{1x} = \frac{f_0^{5-2D_x} - (2\pi f_0)^2 w_{2x}}{(2\pi)^2 k_1} \]

\[ w_{1y} = \frac{f_0^{5-2D_y} - (2\pi f_0)^2 w_{2y}}{(2\pi)^2 k_1} \]  

(7.5)

The parameters at the coarser levels are now found with

\[ u^{l+1}_{m_x} = 2^{2m_x + 2D_x - 8} u^l_{m_x} \]

\[ u^{l+1}_{m_y} = 2^{2m_y + 2D_y - 8} u^l_{m_y} \]  

(7.6)

The rest of the method is the same as before and gives a fractal reconstruction with anisotropic properties. Since the variation of the blending parameters mostly controls the large-scale fractal properties and the addition of the controlled noise controls the fine-scale fractal detail, we would expect the anisotropy to be more evident in the overall shape than in the fine details.

### 7.2 Anisotropic reconstruction experiments

The first candidate for a test reconstruction is the artificial anisotropic fractal generation method, described in section 3.4.1. Figure 7.1 shows an example fractal with a dimension in the \( x \) direction of \( D_x = 2.2 \) and in the \( y \) direction of \( D_y = 2.5 \). A random subsample of the points is taken and reconstructions are performed with both the regular multigrid reconstruction method and the anisotropic multigrid reconstruction. The resultant surfaces are compared point by point with the original, and the error statistics are calculated. Experiments are run with percentages of data between 0.5% and 20% and the results are shown in table 7.1. \( D_{rx} \) and \( D_{ry} \) are the fractal dimensions calculated in the \( x \) and \( y \) directions from the reconstructed surfaces.

Figure 7.2 shows both reconstructions using 0.5%, 1% and 5% of the original data points. From both the statistics and from looking at the pictures we can see that the anisotropic reconstruction does a better job than the normal isotropic reconstruction method. This is especially true at the lower percentages of data kept. When
7.2. Anisotropic reconstruction experiments

Figure 7.1: Artificially-generated anisotropic fractal surface, with $D_x = 2.2$ and $D_y = 2.5$

Table 7.1: Isotropic and Anisotropic fractal surface reconstructions of an artificially-generated anisotropic fractal with $D_x = 2.2$ and $D_y = 2.5$

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{rx}$</th>
<th>$D_{ry}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.2037</td>
<td>2.3277</td>
<td>150.5478</td>
<td>1.7328</td>
<td>18.2431</td>
<td>7.2290</td>
</tr>
<tr>
<td>1.0</td>
<td>2.1786</td>
<td>2.3302</td>
<td>80.5813</td>
<td>0.8859</td>
<td>10.4072</td>
<td>4.7632</td>
</tr>
<tr>
<td>5</td>
<td>2.2231</td>
<td>2.2886</td>
<td>31.0738</td>
<td>-0.1184</td>
<td>4.4981</td>
<td>2.4954</td>
</tr>
<tr>
<td>10</td>
<td>2.2397</td>
<td>2.3972</td>
<td>15.8613</td>
<td>0.0004</td>
<td>3.0770</td>
<td>1.8960</td>
</tr>
<tr>
<td>20</td>
<td>2.2645</td>
<td>2.3925</td>
<td>11.9837</td>
<td>-0.0095</td>
<td>2.3386</td>
<td>1.4705</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{rx}$</th>
<th>$D_{ry}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2.2201</td>
<td>2.4378</td>
<td>35.7137</td>
<td>0.2407</td>
<td>7.8080</td>
<td>4.7405</td>
</tr>
<tr>
<td>1.0</td>
<td>2.2032</td>
<td>2.3925</td>
<td>38.4097</td>
<td>0.3269</td>
<td>6.7447</td>
<td>3.7839</td>
</tr>
<tr>
<td>5</td>
<td>2.2349</td>
<td>2.4128</td>
<td>17.0588</td>
<td>-0.1263</td>
<td>3.7958</td>
<td>2.2208</td>
</tr>
<tr>
<td>10</td>
<td>2.2573</td>
<td>2.4334</td>
<td>14.2244</td>
<td>-0.0499</td>
<td>2.8438</td>
<td>1.7410</td>
</tr>
<tr>
<td>20</td>
<td>2.2697</td>
<td>2.4484</td>
<td>11.2007</td>
<td>-0.0526</td>
<td>2.2145</td>
<td>1.3840</td>
</tr>
</tbody>
</table>
Chapter 7. Anisotropic fractal reconstruction

Figure 7.2: Fractal surface reconstructions of an artificial anisotropic fractal
using the lowest number of data points, the anisotropic reconstruction has a maximum error of nearly five times smaller, the mean error is seven times smaller and the root mean squared error is half that of the normal reconstruction.

As the amount of data kept is increased, we see that the results get more similar, both statistically and visually. Once 10% of the data points are kept, the results are basically the same, except for the mean error which is now smaller for the isotropic case. This is to be expected, because once a certain amount of the data points are kept, any reconstruction has enough information to force the overall shape to be anisotropic. Since we are unable to introduce anisotropy at the finest scales, the reconstructions are bound to become similar when the data is no longer sparse and the reconstruction has less freedom in choosing its shape.

The fractal dimensions of the reconstructed surfaces illustrate how the normal reconstruction method is trying to make an isotropic fractal out of the data. As more of the original points are kept, the more anisotropic its result becomes, as the data points are a bigger proportion of the surface.

The second experiment is a repeat of one in section 6.3. It is the one where we take the real USGS terrain data, and use the mask designed to force the reconstruction to perform an extrapolation (mask 1). Qualitatively this terrain looks like a good candidate for the anisotropic reconstruction because of its prominent ridges. These cross the surface in one direction and are indicative of an anisotropic fractal, as we have seen from creating them artificially. Measuring the dimension of the surface in the $x$ and $y$ directions we get figures of $D_x = 2.40$ and $D_y = 2.15$. The anisotropic reconstruction will use these values, while the normal method will use the averaged, overall dimension, as usual. Table 7.2 gives the statistical results for both methods, keeping all data inside the mask and using different percentages of data outside the mask from 0% to 5%.

Figure 7.3 shows the original data and the mask, as well as both the isotropic and anisotropic reconstructions for the 0% and 0.1% cases.

Looking at the error statistics we again see a significant advantage for the anisotropic reconstructions, and again the advantage is most striking when the data points are
Table 7.2: Isotropic and Anisotropic fractal surface reconstructions of a section of real USGS data, using mask1

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{rx}$</th>
<th>$D_{ry}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.2116</td>
<td>2.1549</td>
<td>922.4751</td>
<td>-21.9694</td>
<td>240.0140</td>
<td>131.3146</td>
</tr>
<tr>
<td>0.1</td>
<td>2.2026</td>
<td>2.1586</td>
<td>860.7922</td>
<td>-19.6507</td>
<td>201.3529</td>
<td>112.9799</td>
</tr>
<tr>
<td>0.5</td>
<td>2.2053</td>
<td>2.1687</td>
<td>410.8350</td>
<td>-12.1780</td>
<td>110.6185</td>
<td>57.1075</td>
</tr>
<tr>
<td>1</td>
<td>2.2047</td>
<td>2.1702</td>
<td>416.0945</td>
<td>-9.5630</td>
<td>84.7408</td>
<td>35.8019</td>
</tr>
<tr>
<td>5</td>
<td>2.2147</td>
<td>2.1761</td>
<td>229.0540</td>
<td>-1.8681</td>
<td>34.1927</td>
<td>17.2031</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sampling Rate</th>
<th>$D_{rx}$</th>
<th>$D_{ry}$</th>
<th>Chebychev Norm</th>
<th>Mean Error</th>
<th>RMS Error</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.4117</td>
<td>2.1331</td>
<td>588.1182</td>
<td>-28.3218</td>
<td>192.7098</td>
<td>114.6713</td>
</tr>
<tr>
<td>0.1</td>
<td>2.4047</td>
<td>2.1379</td>
<td>538.7935</td>
<td>-17.6828</td>
<td>151.5585</td>
<td>95.3513</td>
</tr>
<tr>
<td>0.5</td>
<td>2.4069</td>
<td>2.1469</td>
<td>370.4000</td>
<td>-0.3786</td>
<td>93.9495</td>
<td>47.9153</td>
</tr>
<tr>
<td>1</td>
<td>2.4065</td>
<td>2.1508</td>
<td>354.4399</td>
<td>-5.2028</td>
<td>72.2930</td>
<td>33.7131</td>
</tr>
<tr>
<td>5</td>
<td>2.4168</td>
<td>2.1572</td>
<td>237.0906</td>
<td>-2.5870</td>
<td>33.3532</td>
<td>16.6760</td>
</tr>
</tbody>
</table>
the most sparse. This trend is present in all the error measures, except for the mean error, which is better for one method in some cases and for the other method in others. The calculated dimensions show that the isotropic reconstruction no longer makes the result totally isotropic. This is because half of the surface is fixed and so the average dimension will remain a little anisotropic overall even if the reconstructed section is done in an entirely isotropic way. We can see this in the examples in the figure 7.3. The areas reconstructed by the isotropic method look entirely isotropic, while the anisotropic reconstruction can be seen trying to extend the characteristic ridge features. This works fine where the ridges extend indefinitely, but falls down where the ridge suddenly stop, as the reconstruction continues them on, unless it has data points to tell it otherwise.

Comparing the results with the real data with those using the artificial fractal, the anisotropic method has less of an advantage with these real data points. This is not surprising since the artificial fractal is created to be isotropic over its whole surface and at all scales — the anisotropy is homogeneous — which is the assumption used in this reconstruction method. Real data is unlikely to have this property, especially when looking at large scales such as these, where many different natural processes have gone to form the shape of the terrain. If we could find real data where only one natural process has gone into forming its shape, then the anisotropy might be homogeneous. However, this example has anisotropy that is sufficiently constant for this method to produce better results than assuming no anisotropy.
Chapter 7. Anisotropic fractal reconstruction

Figure 7.3: Fractal surface reconstructions of an artificial anisotropic fractal
Chapter 8

Conclusions and future work

The first task was to produce tools for the creation of fractals of all kinds quickly and easily. Three different methods were implemented: random midpoint displacement, successive random additions and FFT filtering. FFT filtering was then adapted in a novel way to produce anisotropic fractals, fractals possessing different fractal properties in different directions. The random midpoint displacement surfaces were found to display prominent artifacts — peaks visible at regular intervals where the original starting points were. These made it an undesirable method to use. The successive random additions method was also found to occasionally suffer from this problem, but to a lesser degree and often not at all. It is therefore a better choice for fractal surface generation. However FFT filtering was found to be the best method to use, creating the purest fractals the closest to the theoretical definition of a fractal. Although such pure fractals are probably not very common in the real world, where fractal behaviour is usually limited to a finite range of scales.

The second piece of work required was a reliable way of calculating the fractal dimension of a given surface. Two methods were used here: a reverse Fourier transform, and difference statistics. Each method was extensively tested using the surfaces made with the different creation tools. The FFT method was found to work well, especially with fractals also created with a Fourier transform. However this method can only work for a regular, dense grid of data, not irregularly-sampled sparse data points. The difference statistics method does not have this problem, and was found to give the best results when used on a central window, so that
all directions and distances were equally represented in the statistics. Finally, this
difference statistics fractal dimension calculation method was adapted so that the
directional information is kept and any anisotropy in the fractal can be measured.

Using these tools the method of constrained fractal reconstruction could be tested.
Simple fractal reconstructions were performed with single-level reconstructions.
However these were found to give insufficient control over the resultant fractal
dimension of the reconstructed surface, and are also slow to converge to a so­
lution. Two different multilevel methods — multigrid and relative multilevel —
were found to speed up the convergence times and to give more control over the
fractal dimension of the reconstruction. The relative multilevel method, however,
takes a prohibitively long time to set up, and has very high storage requirements.
It was found that the sparse matrix storage scheme could be tuned to be quick to
set up, to take up less memory, or to be quick to solve, but the nature of the relative
multilevel equations that no scheme could be found to do all three well at once.
Therefore it was found that the multigrid is the best compromise in terms of speed,
storage requirements and versatility.

The multilevel reconstruction methods have a large number of free parameters that
need to be set — we can’t just specify the dimension and get a reconstruction. How
to set these parameters was investigated and a method was found that holds for
all surfaces created by the same generation process. It was found that once the
method is set up for a particular family of surfaces — done with a single constant
— then it correctly reconstructs surfaces of the correct dimension, for any surface
within that same family.

Testing the reconstruction methods on both real and artificially-generated frac­
tal data, it was found that the fractal reconstruction technique produced results
that looked far more like the original data than non-fractal reconstructions. Our
multigrid method was found to produce better results than the relative multilevel
method, both visually and statistically. The multigrid results were also found to be
at least as good statistically as those produced with a smooth multigrid reconstruc­
tion, with some giving better results, in terms of the mean error over the surface.

Using real data it was found that some sub-surface terrain data from BG Technol-
ogy could be well reconstructed in areas where the data showed a consistent fractal nature. However the reconstructions applied the same fractal properties over the whole surface, and thus the reconstruction was less successful in areas where the data showed no fractal behaviour at all — flat plateaus. A possible future adaptation might be able to apply the random noise only in certain areas of the reconstruction and so vary the fractal dimension over the surface. A second example of real data — the USGS data — was more successfully reconstructed because this surface showed a more consistent roughness over the whole surface.

Both sets of real data were also reconstructed using masks, in order to see if the reconstruction performed better in an area of sparse data if this was near to an area of dense data. The dense data provided a reliable estimate of the fractal properties, assuming again they held constant over the whole surface. The reconstructions however didn’t show there was any other obvious advantage in having a dense area of data nearby, compared to a similarly sparse set of data points without the adjacent dense data. These masks also forced the reconstructions to extrapolate out beyond the data points, rather than just to interpolate in between them. This was found to be successful only over small distances and where the direction of the surface didn’t change dramatically beyond the data points. Possible improvements to the extrapolation might include geological information into the prior model to decide in what directions the surface is likely to go.

These experiments show how important it is to match the prior assumptions that go into the reconstruction method to the physical characteristics of the surface being reconstructed. In the cases where these assumptions don’t match completely, the result is a bad reconstruction, or one that only works in a limited section of the surface. One such assumption used is that the surfaces are equally rough in all directions — that the fractality is isotropic. The USGS data is imperfectly reconstructed in part because its fractal properties appear quite anisotropic.

Adapting the prior assumptions of the method so that we assume anisotropy results in a novel technique for reconstructing these sort of fractals. The method is imperfect in that all the ways that go into making our reconstructions fractal can not be applied differently in different directions. However enough anisotropy can
be introduced into the reconstructed surfaces to produce good results when reconstructing fractals that display significant anisotropy. In such cases — where the assumptions match — the reconstructions are significantly better than those produced with the standard isotropic method. The extrapolation involved in the USGS data experiment is also performed better using the anisotropic fractal reconstruction.

Possible improvements to this method would be to be able to define the dimension in all directions, rather than just along the perpendicular axes. Future improvements to the overall reconstruction technique could include a way of detecting and predicting discontinuities and creases in the surfaces, and thus including them in the reconstruction since the method is already capable of this. Parallelisation of the optimisation code would lead to the speeding up of the present method, or else a better optimisation technique might be used such as those involving wavelets. Wavelets offer a possible other way of representing data across differing scales, and therefore offer some interesting possibilities for working with fractals.
Appendix A

The random addition to midpoint displacement

At the nth stage of the random midpoint displacement process the centre of any given square is given a height that is the average of the four surrounding points plus a random addition, $a_n$. This random number has a Gaussian distribution with a zero mean and a standard deviation which can be derived from (3.4).

If the whole image has a side of length 1, the centre of any square at stage n is at $\frac{1}{2 \sqrt{2^{n+1}}} - \frac{1}{2^{n+1}}$ away along each coordinate from the corner of the square. For simplicity take the corner of the square to be at (0,0). Then we have

$$V_H\left(\frac{1}{2^{n+1}}, \frac{1}{2^{n+1}}\right) = \frac{1}{4} \left[ V_H(0,0) + V_H(0,1) + V_H(1,0) + V_H(1,1) \right] + a_n. \tag{A.1}$$

Rearranging,

$$V_H\left(\frac{1}{2^{n+1}}, \frac{1}{2^{n+1}}\right) - V_H(0,0) = \frac{1}{4} \left[ V_H\left(\frac{1}{2^{n+1}}, 0\right) - V_H(0,0) \right] + \frac{1}{4} \left[ V_H\left(0, \frac{1}{2^{n+1}}\right) - V_H(0,0) \right] + \frac{1}{4} \left[ V_H\left(\frac{1}{2^{n+1}}, \frac{1}{2^{n+1}}\right) - V_H(0,0) \right] + a_n. \tag{A.2}$$
Appendix A. The random addition to midpoint displacement

Squaring both sides

\[
\left[ V_H\left(\frac{1}{2^{n+1}},\frac{1}{2^{n+1}}\right) - V_H(0,0) \right]^2 = \\
\frac{1}{16} \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right]^2 + \frac{1}{16} \left[ V_H(0,\frac{1}{2^{n+1}}) - V_H(0,0) \right]^2 + \frac{1}{8} \left[ V_H(0,\frac{1}{2^{n+1}}) - V_H(0,0) \right] \left[ V_H(0,\frac{1}{2^{n+1}}) - V_H(0,0) \right] + \sigma_n^2 \\
\frac{1}{8} \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right] \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right] + \frac{1}{8} \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right] \left[ V_H(0,\frac{1}{2^{n+1}}) - V_H(0,0) \right] + \frac{1}{8} \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right] \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right] + \frac{1}{8} \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right] \left[ V_H\left(\frac{1}{2^{n+1}},0\right) - V_H(0,0) \right]
\]

We take the expectation values of both sides and assume that the data is uncorrelated, i.e.

\[
\langle [V_H(x_1,y_1) - V_H(x_2,y_2)][V_H(x_3,y_3) - V_H(x_4,y_4)] \rangle = \\
\langle [V_H(x_1,y_1) - V_H(x_2,y_2)] \rangle \langle [V_H(x_3,y_3) - V_H(x_4,y_4)] \rangle = 0.
\]

Using equation (3.4) and the above observation we obtain:

\[
\sigma^2 \left[ \frac{1}{2^{n+1}} + \frac{1}{2^{n+1}} \right]^H = \frac{\sigma^2}{16} \left[ \frac{1}{2^{n-1}} \right]^H + \frac{\sigma^2}{16} \left[ \frac{1}{2^{n-1}} \right]^H + \frac{\sigma^2}{16} \left[ \frac{1}{2^{n-1}} + \frac{1}{2^{n-1}} \right]^H + \Delta_n^2,
\]

where \( \Delta_n^2 \) is the expectation value of \( \sigma_n^2 \).

An expression for the variance of the random increment can then be found:

\[
\Delta_n^2 = \sigma^2 \left[ \frac{1}{2^{n+1}} \right]^H - \frac{\sigma^2}{8} \left[ \frac{1}{2^{n-1}} \right]^H - \frac{\sigma^2}{16} \left[ \frac{1}{2^{n-1}} \right]^H \\
= \sigma^2 \left[ \frac{1}{2^{nH}} \right]^H - \frac{1}{2^{(n-1)H+3}} - \frac{1}{2^{H+4}} \\
= \frac{\sigma^2}{2^{nH}} \left[ 1 - 2^{H-nH+H-3} - 2^{nH-nH+2H-4} \right].
\]

Hence,

\[
\Delta_n^2 = \frac{\sigma^2}{2^{nH}} \left[ 1 - 2^{-3} - 2^{H-4} \right].
\]
Appendix B

The matrix form of the energy equation.

\[ k = iN + j \]

\[ i = \left\lfloor \frac{k}{N} \right\rfloor \text{ integer part of } \frac{k}{N} \]

\[ j = k - \left\lfloor \frac{k}{N} \right\rfloor N \]

\[ E_p = \frac{1}{2} \sum_{k=0}^{N^2-1} \sum_{l=0}^{N^2-1} u_k a_{kl} u_l \quad (B.1) \]

\[ u_{ij} = \begin{bmatrix}
  u_{0,0} & u_{1,0} & \cdots & u_{N-1,0} \\
  u_{0,1} & u_{1,1} & \cdots & u_{N-1,1} \\
  \vdots & \vdots & \ddots & \vdots \\
  u_{0,N-1} & u_{1,N-1} & \cdots & u_{N-1,N-1}
\end{bmatrix} \quad (B.2) \]

Each column of this matrix \( u_{ij} \) is stacked into a vector to form \( u_k \).
Appendix B. The matrix form of the energy equation.

\[ u_{i,j} = \begin{bmatrix} u_{0,0} \\ u_{0,1} \\ \vdots \\ u_{0,N-1} \\ u_{1,0} \\ u_{1,1} \\ \vdots \\ u_{1,N-1} \\ \vdots \\ u_{N-1,0} \\ u_{N-1,1} \\ \vdots \\ u_{N-1,N-1} \end{bmatrix} \quad \text{and} \quad u_k = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \\ u_{N+1} \\ \vdots \\ u_{2N-1} \\ \vdots \\ u_{N^2-N} \\ u_{N^2-N+1} \\ \vdots \\ u_{N^2-1} \end{bmatrix} \]  

(B.3)

Substituting the finite differences (5.12) and the continuity strengths (5.13) into the energy equation (5.15) we have:

\[
E(u) = \frac{1}{2} \lambda \sum_{ij} \beta_{ij}^x w_1 (u_{i+1,j}^2 + u_{i,j}^2 - 2u_{i,j+1}u_{i+1,j}) \\
+ \beta_{ij}^y w_1 (u_{i,j+1}^2 + u_{i,j}^2 - 2u_{i+1,j}u_{i,j+1}) \\
+ w_2 \beta_{ij}^{xx} (u_{i+1,j}^2 + 4u_{i,j}^2 + u_{i-1,j}^2 - 4u_{i,j}u_{i+1,j} + 2u_{i+1,j}u_{i-1,j} - 4u_{i,j}u_{i+1,j}) \\
+ 2w_2 \beta_{ij}^{yy} (u_{i+1,j+1}^2 + u_{i,j+1}^2 + u_{i,j}^2 - 2u_{i,j+1}u_{i+1,j} - 2u_{i+1,j+1}u_{i,j}) \\
+ 2u_{i,j+1}u_{i+1,j} + 2u_{i,j+1}u_{i+1,j} - 2u_{i,j+1}u_{i+1,j} - 2u_{i+1,j}u_{i,j}) \\
+ w_2 \beta_{ij}^{yy} (u_{i,j+1}^2 + 4u_{i,j}^2 + u_{i,j-1}^2 - 4u_{i,j}u_{i+1,j+1} + 2u_{i,j+1}u_{i,j-1} - 4u_{i,j}u_{i,j-1})
\]
\[ E(u) = \frac{1}{2} \sum_{i,j} \left[ w_1 \beta_{i,j}^x + w_1 \beta_{i,j}^y + 4w_2 \beta_{i,j}^{xy} + 2w_2 \beta_{i,j}^{yy} + 4w_2 \beta_{i,j}^{yy} \right] u_{i,j}^2 + \frac{1}{2} \sum_{i,j} \left[ 2w_2 \beta_{i,j}^{xy} \right] u_{i+1,j+1}^2 + \frac{1}{2} \sum_{i,j} \left[ w_1 \beta_{i,j}^x + w_2 \beta_{i,j}^{xy} + 2w_2 \beta_{i,j}^{yy} \right] u_{i+1,j}^2 + \frac{1}{2} \sum_{i,j} \left[ w_1 \beta_{i,j}^y + 2w_2 \beta_{i,j}^{xy} + 2w_2 \beta_{i,j}^{yy} \right] u_{i,j+1}^2 + \frac{1}{2} \sum_{i,j} \left[ w_1 \beta_{i,j}^y + 2w_2 \beta_{i,j}^{xy} + 2w_2 \beta_{i,j}^{yy} \right] u_{i,j+1}^2 + \frac{1}{2} \sum_{i,j} \left[ 2w_2 \beta_{i,j}^{xy} \right] u_{i-1,j}^2 + \frac{1}{2} \sum_{i,j} \left[ 2w_2 \beta_{i,j}^{xy} \right] u_{i-1,j}^2 + \frac{1}{2} \sum_{i,j} \left[ -2w_1 \beta_{i,j}^x - 4w_2 \beta_{i,j}^{xy} - 4w_2 \beta_{i,j}^{yy} \right] u_{i,j+1}u_{i+1,j} + \frac{1}{2} \sum_{i,j} \left[ -2w_1 \beta_{i,j}^y - 4w_2 \beta_{i,j}^{xy} - 4w_2 \beta_{i,j}^{yy} \right] u_{i,j}u_{i,j+1} + \frac{1}{2} \sum_{i,j} \left[ 2w_2 \beta_{i,j}^{xy} \right] u_{i+1,j}u_{i-1,j} + \frac{1}{2} \sum_{i,j} \left[ -4w_2 \beta_{i,j}^{xy} \right] u_{i,j}u_{i-1,j} + \frac{1}{2} \sum_{i,j} \left[ -2w_2 \beta_{i,j}^{xy} \right] u_{i+1,j}u_{i,j+1} + \frac{1}{2} \sum_{i,j} \left[ 4w_2 \beta_{i,j}^{xy} \right] u_{i+1,j}u_{i,j} + \frac{1}{2} \sum_{i,j} \left[ 4w_2 \beta_{i,j}^{xy} \right] u_{i,j+1}u_{i+1,j} + \frac{1}{2} \sum_{i,j} \left[ 2w_2 \beta_{i,j}^{xy} \right] u_{i,j+1}u_{i,j-1} + \frac{1}{2} \sum_{i,j} \left[ -4w_2 \beta_{i,j}^{xy} \right] u_{i,j}u_{i,j-1} \]

As we sum over all pixels, we can collect the terms that represent identical interactions.

The interactions represented by \[ \square, \square, \square, \square, \square, \square, \square, \square \] and \[ \square \] are all the same and
can be collected together if the indices are changed so that they all refer to the $u_{i,j}$ or $u_{i,j}u_{i+1,j}$ interaction. Similarly $M$ and $N$ may be expressed in terms of the $u_{i,j}u_{i,j}$ or $u_{i,j}u_{i,j+1}$ interaction. And $E_{ni}$ and $E_{n+1}$ may be expressed in terms of the $u_{i,j}u_{i,j}$ or $u_{i,j}u_{i,j+1}$ interaction.

The remaining four terms are $\Delta u_{i,j}u_{i,j}$, $\Delta u_{i,j}u_{i+1,j}$, and $\Delta u_{i,j}u_{i,j+1}$. The indices first three are shifted so that all terms now contain a central $u_{i,j}$ term. This gives the new $\Delta u_{i,j}u_{i,j}$, $\Delta u_{i,j}u_{i+1,j}$, and $\Delta u_{i,j}u_{i,j+1}$ terms.

$$E(u) = \frac{1}{2} \sum_{i,j} \left[ w_{u_{i,j}} + w_{u_{i+1,j}} + 4w_{u_{i,j+1}} + 2w_{u_{i+1,j+1}} + 2w_{u_{i,j+1}} + 2w_{u_{i+1,j}} + w_{u_{i,j}} + w_{u_{i+1,j}} + w_{u_{i,j+1}} \right] u_{i,j}$$

$$+ \frac{1}{2} \sum_{i,j} \left[ -2w_{u_{i,j}} - 4w_{u_{i,j-1}} - 4w_{u_{i,j+1}} - 4w_{u_{i,j+1-1}} \right] u_{i,j}$$

$$+ \frac{1}{2} \sum_{i,j} \left[ 2w_{u_{i+1,j}} u_{i+1,j} \right] u_{i,j}$$

$$+ \frac{1}{2} \sum_{i,j} \left[ 2w_{u_{i,j+1}} u_{i,j+1} \right] u_{i,j}$$

$$+ \frac{1}{2} \sum_{i,j} \left[ 2w_{u_{i,j+1}} u_{i,j+1} \right] u_{i,j}$$

$$+ \frac{1}{2} \sum_{i,j} \left[ 2w_{u_{i+1,j}} u_{i+1,j} \right] u_{i,j}$$

The coefficient of $u_{i,j}^2$ is $a_{i,j}$. The factor $\frac{1}{2}$ is the same in both expressions, so

$$a_{i,j} = \lambda w_{v_{i,j}} \left[ \beta_{i,j}^x + \beta_{i,j}^y + \beta_{i,j-1}^x + \beta_{i,j-1}^y \right]$$

$$+ \lambda w_{v_{i,j}} \left[ 4\beta_{i,j}^x + 2\beta_{i,j}^y + 4\beta_{i,j-1}^x + 2\beta_{i,j-1}^y + 2\beta_{i,j+1}^x + 2\beta_{i,j+1}^y \right]$$

If $k$ corresponds to $(i,j)$ then $k = iN + j$. For $(i+1,j)$ we then have $(i+1)N + j = iN + j + N = k + N$. So the term that expresses the interactions between $(i,j)$ and

...
$(i + 1,j)$ is the $a_{k,k+N}$ term. This gives

$$a_{k,k+N} = -2\lambda_1 \beta_{i,j}^p - 4\lambda_2 \left[ \beta_{i,j}^{xx} + \beta_{i+1,j}^{xx} + \beta_{i,j}^{yy} + \beta_{i,j-1}^{yy} \right]$$

For symmetry, we may wish to express this as the interaction between $(i-1,j)$ and $(i,j)$, and $(i,j)$ and $(i+1,j)$. The first is expressed by $a_{k,k-N}$ and the second by $a_{k,k+N}$

$$a_{k,k+N} = -\lambda_1 \beta_{i,j}^p - 2\lambda_2 \left[ \beta_{i,j}^{xx} + \beta_{i-1,j}^{xx} + \beta_{i,j}^{yy} + \beta_{i,j+1}^{yy} \right]$$

$$a_{k,k-N} = -\lambda_1 \beta_{i-1,j}^p - 2\lambda_2 \left[ \beta_{i-1,j}^{xx} + \beta_{i,j}^{xx} + \beta_{i-1,j}^{yy} + \beta_{i,j+1}^{yy} \right]$$

The interaction between vertical neighbours $(i,j)$ and $(i,j+1)$ is given by

$$a_{k,k+1} = -2\lambda_1 \beta_{i,j}^p - 4\lambda_2 \left[ \beta_{i,j}^{yy} + \beta_{i,j+1}^{yy} + \beta_{i,j}^{x} + \beta_{i,j+1}^{y} \right]$$

$(i,j) \rightarrow iN + j = k$

$(i,j+1) \rightarrow iN + j + 1 = k + 1$

For symmetry

$$a_{k,k+1} = -\lambda_1 \beta_{i,j}^p - 2\lambda_2 \left[ \beta_{i,j}^{yy} + \beta_{i,j+1}^{yy} + \beta_{i,j}^{x} + \beta_{i,j+1}^{y} \right]$$

$$a_{k,k-1} = -\lambda_1 \beta_{i-1,j}^p - 2\lambda_2 \left[ \beta_{i-1,j}^{yy} + \beta_{i,j}^{yy} + \beta_{i-1,j}^{x} + \beta_{i,j+1}^{y} \right]$$

For the interaction between $(i,j)$ and $(i+2,j)$

$(i,j) \rightarrow iN + j = k$

$(i+2,j) \rightarrow (i+2)N + j = k + 2N$

$$a_{k,k+2N} = \lambda_2 \beta_{i+1,j}^{xx}$$

$$a_{k,k-2N} = \lambda_2 \beta_{i-1,j}^{xx}$$

For the interactions between $(i,j)$ and $(i,j+2)$

$(i,j) \rightarrow iN + j = k$

$(i,j+2) \rightarrow iN + j + 2 = k + 2$

$$a_{k,k+2} = \lambda_2 \beta_{i,j+1}^{yy}$$

$$a_{k,k-2} = \lambda_2 \beta_{i,j-1}^{yy}$$
Appendix B. The matrix form of the energy equation.

For the interactions between \((i, j)\) and \((i + 1, j + 1)\)

\[(i, j) \rightarrow iN + j = k\]

\[(i + 1, j + 1) \rightarrow (i + 1)N + j + 1 = k + N + 1\]

\[a_{k,k+N+1} = 2\lambda w_2 \beta_{i,j}^{xy}\]

\[a_{k,k-N-1} = 2\lambda w_2 \beta_{i-1,j-1}^{xy}\]

For the interactions between \((i, j)\) and \((i - 1, j + 1)\)

\[(i, j) \rightarrow iN + j = k\]

\[(i - 1, j + 1) \rightarrow (i - 1)N + j + 1 = k - N + 1\]

\[a_{k,k-N+1} = 2\lambda w_2 \beta_{i-1,j}^{xy}\]

\[a_{k,k+N-1} = 2\lambda w_2 \beta_{i,j-1}^{xy}\]
Bibliography


