Smoothing 3D Meshes using Markov Random Fields

Master's thesis

Vedrana Andersen
vedrana@itu.dk
130274-xxxx

Supervisors:
Mads Nielsen, Professor, DIKU
Henrik Aanæs, Associate Professor, DTU

ITU, September 2006 - April 2007

Abstract

This thesis investigates the use of Markov Random Fields (MRF) for formulating priors on 3D surfaces represented as triangle meshes. The problem is addressed by focusing on mesh smoothing, which is of great interest in many applications of geometry processing, e.g., computer vision and reverse engineering.

Firstly, a mesh-smoothing vertex process is developed. It is a process that combines a smoothness prior described through MRF with the simple observation model into MAP-MRF framework. An edge process for detecting features (ridges) is developed next, where the feature-detecting function allows specifying the sharpness of the ridge. Lastly, vertex process and edge process are coupled in a feature-preserving mesh-smoothing method. Smoothing is done by iterative vertex replacement, using Metropolis sampling and simulated annealing scheme.

Preliminary but promising experimental results are presented, proving the feasibility and demonstrating the use of MRF on triangular meshes. Developed priors and the optimization methods are discussed, and some possible improvements suggested.

Preface

For most students the work on the Master's thesis is a continuation of previous project or course. I have not followed that pattern. In choosing my project I left the safety of the regular image grid and the familiar Matlab environment and took a small step from Image Analysis towards Computer Graphics.

Well, looking back at the past half year, I must confess that the great steepness of the learning curve was discouraging during the first months, as I had to become familiar with C++, OpenGL and the GEL framework, and get into a whole new field of surface modeling. In the mean time the literature has become comprehensible, tools familiar and endless possibilities folded out. Now, a week before the hand-in date, it feels as if I have just started.

Acknowledgements

My thanks to Jakob Andreas Bærentzen, Associate Professor, DTU, whose GEL framework provided the basis for my work, and whose advice helped a lot along the way.

Many thanks also to Mads Nielsen, Professor, DIKU, for his ideas and suggestions for this thesis; someday I hope to follow up on all of them.

Heartfelt thanks to my project advisor Henrik Aanæs, Associate Professor, DTU, for his advice, support, and encouragement, and for never showing even the slightest sign of annoyance when I regretfully did not follow his advice.

Thanks to Renée for being there in the moment of crisis.

Thanks to Ante for helping with one of my thesis, again.

And finally, my thanks to Per, Mia and Freja for being so patient.

Contents

1 Introduction **13**

2 Related Work **15**

3 Markov Random Field Theory **17**
3.1 Sites and Labels 17
3.2 Neighborhood System and Cliques 18
3.3 Random Fields 19
3.4 Markov Random Fields 19
3.5 Gibbs Random Fields 19
3.6 Markov-Gibbs Equivalence 20
3.7 MAP-MRF Labeling 20

4 Optimization **23**
4.1 Metropolis Sampler 23
4.2 Simulated Annealing 24

5 Markov Random Fields on Surface Meshes **27**
5.1 Triangular Meshes 27
5.2 Markov Random Fields on Triangular Meshes 28
5.3 Assumptions 29

6 Mesh Smoothing **31**
6.1 What is a Smooth Mesh? 31
6.2 MRF on Mesh Vertices 32
6.3 Smoothness Potential Based on Mean Curvature 32
6.3.1 MRF Framework, Neighborhoods and Clique Potentials . 33
6.4 Alternative Formulations of the Smoothness Potential 35
6.4.1 Quadratic and Square-Root Smoothness Potentials 36
6.4.2 Thresholded Smoothness Potential 38
6.4.3 Comparison of the Four Potentials 38
6.4.4 Normals Under Noise 40
6.5 Likelihood Function 40
6.6 Probability Distributions 41

6.7 Optimization for Mesh Smoothing 42
6.7.1 Metropolis Sampler 42
6.7.2 Simulated Annealing 44
6.8 Summary 45

7 Mesh Smoothing, Results and Discussion 47
7.1 Typical Smoothing 47
7.2 Parameter Choice 48
7.2.1 Cooling Scheme 50
7.2.2 Sampling Size 50
7.2.3 Data Weight 55
7.3 Local Versus Global Smoothing 55
7.4 Efficiency 58
7.5 Comparison of the Different Smoothness Potentials 59
7.6 Conclusion 60

8 Feature Detection 63
8.1 What is a Feature Edge? 64
8.2 Edge Labeling 64
8.3 Ridge Sharpness 66
8.4 Neighborhood Support 67
8.4.1 Formulations of the Neighborhood Support 69
8.5 Stochastic Edge Labeling 71
8.5.1 Optimization 72
8.6 Deterministic Edge Labeling 72
8.7 Summary 73

9 Feature Detection, Results and Discussion 75
9.1 Stochastic Edge Labeling 75
9.1.1 Thresholds 77
9.1.2 Neighborhood Support Question 78
9.2 Deterministic Edge Labeling 80
9.3 Conclusion 80

10 Feature-Preserving Mesh Smoothing 81
10.1 Piecewise Smoothness Prior 81
10.2 Optimization Scheme 82
10.3 Two Questions 83
10.4 Summary 83

11 Final Results and Discussion 85
11.1 Smoothing Evolution 85
11.2 Final Result 87
11.3 Why Not Just Thresholded? 90

11.4 Conclusion 90

12 Future Work **95**

12.1 Improvements of The Existing Algorithm 95

12.2 A Functional Mesh Smoother 95

12.3 Extensions of the Model 96

13 Conclusion **97**

A Box-Müller Transformation **99**

B Hausdorff Distance **101**

Bibliography **103**

Chapter 1

Introduction

The primary aim of this thesis is to investigate the use of Markov Random Field (MRF) theory in formulating priors on surface meshes.

Markov Random Fields have been used extensively for solving Image Analysis problems at all levels. The local property of MRF makes it convenient for modeling dependencies of image pixels, and the MFRs-Gibbs equivalence theorem provides a joint probability in a simple form, making MRF theory useful for statistical Image Analysis. Typical applications include image restoration and segmentation, edge detection, texture analysis and many more. The majority of these MRF applications are built on a regular pixel grid, even though MRF theory, in its general form, does not require regularity of the sites.

The biggest challenge in formulating MRF on triangular meshes is dealing with the irregularity of the surface mesh – vertices can have a different number of adjacent vertices, and the distance between the two adjacent vertices can vary. MRF has not yet found its application in the field of 3D mesh modeling, and some formulations described in this thesis are modeling experiments.

Two processes are developed in this thesis. First, the vertex process utilizes a smoothness prior based on absolute mean curvature and a likelihood function to smooth the mesh in MAP-MRF framework. Second, the edge process is a feature-detecting MRF, which is based on edge sharpness and the support from neighboring edges. Those two processes are then coupled together, so that the vertex process smooths, but not across the feature edges of the edge process — the labels returned from the edge process are used as the weights for the vertex process.

The primary aim of the thesis is addressed by focusing on mesh smoothing because it is easy to see the effects of the priors in this setting. Besides, as the use of the geometry scanning devices increases, efficient mesh-smoothing algorithms that can be used for denoising the acquired data are highly desirable. Models acquired by geometry scanners are often noisy and require smoothing before further processing. To remove the noise while preserving the features is not a trivial problem.

In the next chapter, Related Work, the brief overview of the current state in the field of mesh smoothing is presented. Chapters 3 and 4, Markov Random Field Theory and Optimization, introduce the theory and terminology of MRF, together with the optimization scheme which is used throughout the work. Chapter 5, Markov Random Fields on Surface Meshes, sets the basis for the work described in following chapters. In Chapter 6, Mesh Smoothing, different smoothness priors are is developed and compared. The results of using the developed priors are presented in Chapter 7. Chapter 8, Feature Detection, outlines the formulation of the feature-detecting function, with the result and discussion following in Chapter 9. The smoothness prior and feature detecting function are in Chapter 10, Feature-Preserving Mesh Smoothing, combined into feature-preserving mesh smoothing scheme. Chapter 11 contains more of the experimental results. Chapter 12 suggests some possible improvements and extensions of the developed model and the final chapter offers some concluding remarks.

The implementation of the smoothing algorithm was done in C++ using the GEL framework, which contains a half-edge data structure for representing triangle meshes. No special care has been taken to make the implementation as efficient as possible, the focus being put on flexibility and transparency. The analysis of the results was done in Matlab, and Matlab was also used for analysis of some smaller issues that occurred along the way.

By the end of the project, quite a large collection of images and movies was produced. Some of them, together with the C++ and Matlab code, may be found online at *http://www.itu.dk/people/vedrana/smooth*. The electronic version (can be practical for a more detailed inspection of the images) of this report is also available there.

Chapter 2

Related Work

Mesh-smoothing algorithms have a long history in the field of geometry processing. In [Taubin, 1995] a linear and isotropic technique is proposed, as a generalization of frequency-domain image filtering. The diffusion process that uses a geometry flow analogy [Desbrun *et al.*, 1999] is introduced next. Both techniques are efficient, but fail to distinguish between noise and features of the underlying object.

To address this problem, anisotropic diffusion [Desbrun *et al.*, 2000] and diffusion smoothing of the normal field [Tasdizen *et al.*, 2002] are proposed. The results are impressive, but the computation complexity puts a limit on the size of the model. More efficient methods are also proposed, such as non-iterative feature-preserving smoothing [Jones *et al.*, 2003] based on robust statistics, and an adaptation of bilinear filtering to surface meshes [Fleishman *et al.*, 2003]. More recently, in [Diebel *et al.*, 2006] Bayesian approach is presented using the smoothness prior and the conjugate gradient for optimization. Methods here are feature-preserving, but without an explicit feature detection scheme.

The method for recovering feature edges proposed in [Attene *et al.*, 2005] is based on the dual process of sharpening and straightening feature edges. Vertex-based feature detection using an extension of the fundamental quadric is utilized in a smoothing method described by [Jiao and Alexander, 2005].

Comprehensive study on the use of Markov Random Field theory for solving Image Analysis problems can be found in books by [Li, 2001] and [Winkler, 2003]. Markov Random Field theory is convenient for addressing the problem of piecewise smooth structures. In [Geman and Geman, 1984] a foundation for the use of MRF in Image Analysis problems is presented in an algorithm for restoration of piecewise smooth images, where gray-level process and line processes are used. Some of the other applications of Markov Random Fields for problems involving reconstruction of piecewise smooth structures include [Diebel and Thrun, 2005], where high-resolution range-sensing images are reconstructed using weights obtained from a regular image. In [Hartelius and Carstensen, 2003] a coupled MRF is used for locating grids with possible cracks in the structure. In [Sun *et al.*, 2003] a stereo matching problem is addressed by three coupled MRFs modeling piecewise smoothness and occlusion.

The only example of applying MRF on 3D meshes is presented in [Willis *et al.*, 2004], where MRF are used as deformable 3D models for surface sculpting. The potentials used for sculpting control the deformation of the surface by modeling elasticity and plasticity and are as such quite different then the smoothness priors developed here.

This work investigates the possibility of formulating surface priors in terms of MRF, and using those priors for reconstructing the surface from the noisy date. In particular, a prior for piecewise smooth surface has been developed and successfully used for feature-preserving mesh smoothing. Unlike most other mesh smoothing algorithms, this approach does not only preserve sharp ridges, but also explicitly detects the ridges. This makes it possible to expand the model by defining the prior on the surface ridges.

The method described here is not automatic, and calls for an estimation of a considerable set of parameters. However, this allows a great control over the performance of the priors.

Chapter 3

Markov Random Field Theory

The content of this chapter is almost entirely adopted from [Li, 2001] and is presented here in condensed form as an introduction to the theory and the terminology of MRF.

3.1 Sites and Labels

Many problems can be posed in terms of sites and labels, where the solution to a problem is a set of labels assigned to sites. This involves a discrete set of sites

$$S = \{1, \dots, m\}$$

and a set of labels L. A label is an event that may happen to a site. Depending on the problem at hand, a label set may be continuous or discrete.

The labeling problem is to assign a label from the label set L to each of the sites in S. The set

$$f = \{f_1, \dots, f_m\}$$

is called a labeling of sites in S. This can be regarded as a mapping from S to L

$$f : S \rightarrow L, \; f(i) = f_i \,.$$

Typical use of MRF in Image Analysis problems are image restoration and smoothing, where the image pixels take the role of the sights, and labels are continuous or discrete pixel values. A labeling would then be any assignment of pixel values to pixels. Similarly, an edge detection in an image could be posed as assigning a label from a set $\{$*edge, non-edge*$\}$ to image pixels.

Sites on a lattice, as for example image pixels, are considered spatially regular. The sets of sites that do not present spatial regularity are considered irregular. In the remainder of the thesis we deal with irregular sets of sites, which contain vertices or edges of 3D meshes.

In the terminology of random fields, a labeling is also called a configuration. In Image Analysis, a configuration or labeling can correspond to an image or an edge map. The set of all possible configurations is called the configuration space Ω. The cardinality of configurations space is

$$|\Omega| = |L|^m ,$$

where m is the size of S. For the discrete labeling problem with m sites and M labels there exist a total number of M^m possible configurations. Among all labelings, there are only a small number of them which are good solutions and maybe just a few that are optimal in terms of a certain criterion. Defining the optimal solution and finding it are two important topics in the optimization approach to labeling.

3.2 Neighborhood System and Cliques

To develop MRF theory we need to define spatial dependencies between sites, and we do that by choosing a neighborhood system.

A neighborhood system is defined as

$$\mathscr{N} = \{\mathscr{N}_i | i \in S\} ,$$

where $\mathscr{N}_s$ is the set of sites neighboring s. The neighboring system has the following properties:

1. the site is not neighboring itself: $i \notin \mathscr{N}_i,\ \forall i \in S$,

2. neighboring is mutual: $i \in \mathscr{N}_j \iff j \in \mathscr{N}_i,\ \forall i, j \in S$.

The neighborhood system induces a symmetric binary relation over S, so for two sites $i, j \in S$ we say that they are neighbors if $i \in \mathscr{N}_j$ and $j \in \mathscr{N}_i$.

The set $\mathscr{N}_i$ is typically defined as a collection of sites within a certain radius from i. For regular sites (e.g., image pixels) this results in all internal sites having neighborhoods of the same size and shape. For irregular sites this results in neighborhoods of irregular shapes and sizes. Neighborhood system does not need to be defined in terms of spatial proximity, alternative definitions of neighborhood can be used, for example, in terms of Delaunay triangulation of the irregular sites. In the remainder of the thesis we deal with 3D meshes, where the neighborhood system is defined in terms of mesh connectivity.

A pair $(S, \mathscr{N})$ defines a graph in the usual sense, where S contains the nodes and $\mathscr{N}$ determines the links between the nodes according to the neighboring relationship. A clique c for $(S, \mathscr{N})$ is defined as a a complete subgraph of $(S, \mathscr{N})$. In other words, clique c is a subset of sites in S that are all neighbors to one another.

The collection of all cliques is denoted $\mathscr{C}$. Every site defines a single-site clique; a pair of neighboring sites defines a pair-site clique; a triple of neighboring sites defines a triple-site clique; and so on. The collections of single-site, pair-site and triple-site cliques are denoted by $\mathscr{C}_1$, $\mathscr{C}_2$ and $\mathscr{C}_3$ respectively.

3.3 Random Fields

Let $F = \{F_1, \dots, F_m\}$ be a family of random variables defined on the set of sites S, where each random variable F_i takes a value from the set of labels L. The family F is called a random field.

The notation $F_i = f_i$ denotes the event that F_i takes the value f_i and the notation $(F_1 = f_1, \dots, F_m = f_m)$ or $F = f$ denotes a joint event, where $f = f_1, \dots, f_m$ is a configuration of F. A labeling f of the sites is a configuration of F, corresponding to a realization of the field.

For a discrete label set, the probability that a random variable F_i takes the value f_i is denoted $P(F_i = f_i)$, abbreviated $P(f_i)$. The joint probability is denoted $P(F = f)$, abbreviated $P(f)$. For a continuous label set we have corresponding probability density functions.

3.4 Markov Random Fields

A random field F is said to be a Markov Random Field on S with respect to $\mathcal{N}$ if and only if the following conditions are satisfied:

1. positivity: $P(f) > 0, \forall f \in \Omega$,
2. Markovianity: $P(f_i | f_{S-\{i\}}) = P(f_i | f_{\mathcal{N}_i})$.

where $f_{S-\{i\}}$ denotes the set of labels of all sites but i, and $f_{\mathcal{N}_i}$ denotes the set of labels at the sites neighboring i.

The Markovianity describes the local characteristics of the random field — only neighboring labels have direct interaction with each other.

MRF can be specified in terms of the conditional probabilities $P(f_i | f_{\mathcal{N}_i})$, but MRF provide no obvious methods for deducing the joint probability from the associated conditional probabilities.

3.5 Gibbs Random Fields

A random field F is said to be a Gibbs Random Field (GRF) on S with respect to $\mathcal{N}$ if and only if its configurations obey a Gibbs distribution. A Gibbs distribution takes the form

$$P(f) = \frac{1}{Z} e^{-\frac{1}{T}U(f)} ,$$

where

$$Z = \sum_{f \in \Omega} e^{-\frac{1}{T}U(f)}$$

is a normalization constant, T is a constant called the temperature, and $U(f)$ is the energy function.

The energy

$$U(f) = \sum_{c \in \mathscr{C}} U_c(f)$$

is a sum of clique potentials $U_c(f)$ over all possible cliques c. The value of $U_c(f)$ depends on the local configuration on the clique c.

$P(f)$ measures the probability of the occurrence of a particular configuration. The configurations with lower energy are more probable. The temperature controls the sharpness of the distribution — when it is high all configurations tend to be equally probable, while near zero temperature the probability distribution concentrates around the global energy minima, see also Figure 4.1.

3.6 Markov-Gibbs Equivalence

An MRF is characterized by its local property (Markovianity), whereas a GRF is characterized by its global property (Gibbs distribution). The Hammersley-Clifford theorem [Hammersley and Clifford, 1971] establishes the equivalence of these two types of properties. The theorem states that F is an MRF on S with respect to $\mathscr{N}$ if and only if F is a GRF on S with respect to $\mathscr{N}$.

The theorem provides a simple way of obtaining the joint probability $P(f)$ by specifying the clique potential functions $U_c(f)$. Clique potential functions should be chosen depending on the desired system behavior, encoding a-priori knowledge about interactions between labels. Choosing the forms and parameters of the potential functions is a major topic in MRF modeling.

3.7 MAP-MRF Labeling

Bayes statistics is a theory widely used for estimation and decision making.

The probability of the labeling f given the observation d is called the posterior probability $P(f|d)$ and can be evaluated using the Bayes rule

$$P(f|d) = \frac{p(d|f)P(f)}{p(d)} \ .$$

The term $P(f)$ is called the prior probability of labeling f, $p(d|f)$ is the conditional probability density function of the observations d, also called the likelihood function of f for d fixed, and $p(d)$ is the density of d, also called the evidence.

When d is given (i.e., the evidence is constant) the Bayes rule can be written as

$$P(f|d) \propto p(d|f)P(f) \ .$$

In the maximum a-posteriori (MAP) solutions, as a special case in the Bayes framework, only the most probable estimate is of interest. The optimal MAP solution is obtained by maximizing the posterior probability

$$f^* = \arg\max_{f \in \Omega} P(f|d) = \arg\max_{f \in \Omega} p(d|f)P(f) \ .$$

In MAP-MRF labeling, $P(f|d)$ is the posterior distribution of an MRF. An important modeling step in MRF-MAP labeling is to derive the posterior distribution. The prior is obtained from the prior energy $U(f)$ for the labeling, which in turn is a sum of clique potentials. Likelihood distribution can often also be expressed in terms of likelihood energy

$$p(d|f) \propto e^{-U(d|f)}$$

leading to the posterior probability

$$P(f|d) \propto e^{-U(f|d)} ,$$

where

$$U(f|d) = U(d|f) + U(f)$$

is the posterior energy.

The MAP estimate is then equivalently found by minimizing the posterior energy function

$$f^* = \arg\min_{f \in \Omega} U(f|d) .$$

The procedure of the MAP-MRF approach summarized in [Li, 2001] is also followed in the presentation of the mesh-smoothing MRF process presented later. The summary is:

1. Pose a labeling problem as a discrete or continuous, on a regular or irregular set of sites, and chose an appropriate MRF representation of f.

2. Derive the posterior energy to define the MAP solution to a problem in the following four steps:

   - Define a neighborhood system $\mathcal{N}$ on S and the set of cliques $\mathcal{C}$ for $\mathcal{N}$.
   - Define a prior clique potentials $U_c(f)$ to give $U(f)$.
   - Define the likelihood energy $U(d|f)$.
   - Add $U(f)$ and $U(d|f)$ to yield the posterior energy $U(f|d)$.

3. Find the MAP solution.

In the above summary, the prior model depends on the output that we expect and the likelihood model depends on physical considerations such as the sensor noise. The parameters on both models have to be specified, manually or automatically. An optimization algorithm for finding the MAP solution needs to be chosen, where the main issues are the quality and the efficiency.

Chapter 4

Optimization

After the modeling is done and energy function described, the remaining problem is to find the optimal solution. A variety of local and global optimization methods exist. This chapter briefly presents only the optimization method used here and only in it's most general form. In Section 6.7, after the mesh-smoothing prior is defined, optimization is re-visited to discuss specific mesh-smoothing issues.

4.1 Metropolis Sampler

The Metropolis sampler is a random sampling algorithm, which generates a sequence of configurations from a probability distribution using a Monte Carlo procedure. The sampling scheme is:

1. randomly initialize f,
2. for $i \in S$ do
   (a) let $f'_{i'} = f_{i'}$ for all $i' \neq i$, choose $f'_i \in L$ at random,
   (b) replace f by f' with probability $p = \min\{1, P(f')/P(f)\}$, where P is the given Gibbs distribution,
3. repeat 2. N times.

At each step, a new configuration f' is chosen by randomly changing only one label. The new configuration is accepted according to the Metropolis criterion. Metropolis criterion assures that the new configuration will be accepted as soon as it has a higher probability. Still, even with a smaller probability the new configuration has a chance of being accepted, depending on the ratio $P(f')/P(f)$. This allows the algorithm to leave the local energy minima.

Looking at the Metropolis criterion for accepting a less probable configuration, and using Gibbs distribution we obtain the following probability of acceptance

$$p = \frac{P(f')}{P(f)} = e^{-\frac{1}{T}(U(f')-U(f))} ,$$

where U is the potential of the configuration. This is function of the energy *difference* and the temperature T.

To investigate the role of the temperature T, we look at the case when

$$\Delta U = U(f') - U(f) > 0$$

which means that the new configuration results in a higher energy.

For a limit of $T \rightarrow \infty$ the probability of accepting the new configuration

$$p = e^{-\frac{1}{T}\Delta U}$$

approaches 1, and the Metropolis sampler reduces to the random sampler, accepting any new configuration regardless of its energy. But as T falls, p also falls, and for $T \rightarrow 0$ it approaches 0, so no increases in the energy are accepted.

To summarize, if T is large, many 'bad' moves are accepted, and a large part of configuration space is accessed, while for small T mostly 'good´ moves are accepted and the sequence of samples will converge towards the local energy minimum. It is this property that allows the use of the Metropolis sampling as the optimization method.

4.2 Simulated Annealing

Simulated annealing is a stochastic optimization algorithm that simulates the physical annealing process of melting and then slowly cooling to achieve the optimal energy configuration. The annealing algorithm is:

1. randomly initialize T and f,
2. do
   (a) randomly sample f under T,
   (b) decrease T,
3. repeat 2. until $T \rightarrow 0$.

A random search method, such as Metropolis sampler, is used to locate next configuration. The random search is controlled by the temperature T. In the simulated annealing scheme the T is initially high and then *gradually* decreased, to minimize the risk of the algorithm being trapped in local energy minima.

High initial temperature allows a large part of the configuration space to be examined. Gradually decreasing the temperature limits the amount of allowed 'bad´ moved, slowly reducing the part of the configuration space that gets examined.

For a particular logarithmic cooling scheme it is proven that the system converges to minimal energy [Geman and Geman, 1984]. This scheme is unfortunately too slow for practical applications, but other cooling schemes also produce good results.

Figure 4.1 illustrates the effect of decreasing the temperature for a constructed energy function. For a high temperature the probability distribution is almost uniform, while for low temperature it concentrates around the minimum of the energy function.

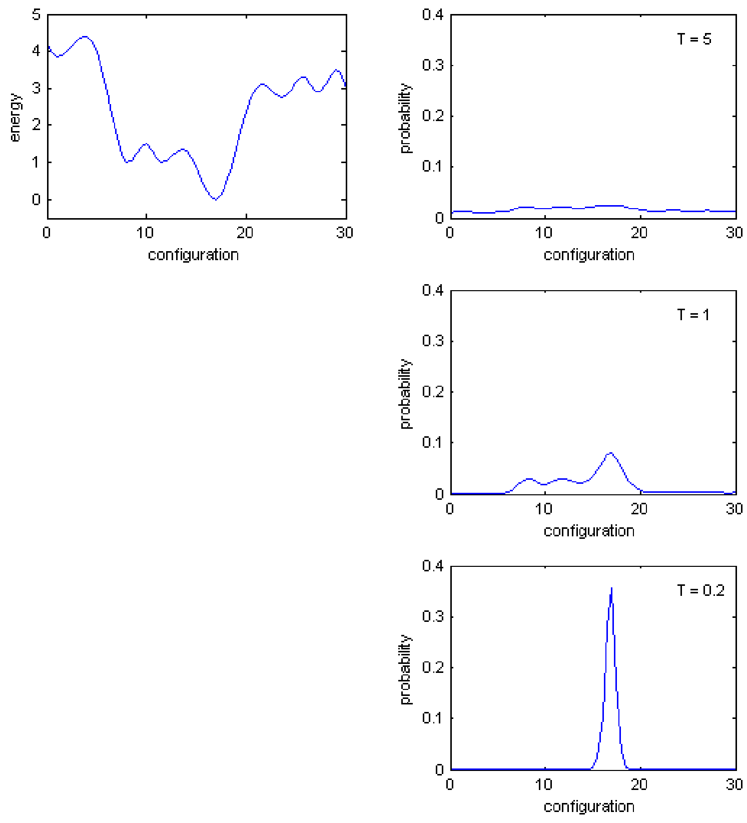


Figure 4.1: The effect of decreasing the temperature on the probability distribution. *Right:* A constructed energy function. *Left:* Corresponding Gibbs probability distribution for temperature $T = 5$, $T = 1$ and $T = 0.2$ (from top to bottom).

Chapter 5

Markov Random Fields on Surface Meshes

This chapter sets the basis for the work that follows. An idea for defining MRF in terms of mesh connectivity is discussed first, followed by a section covering the few assumptions about the meshes dealt with here.

5.1 Triangular Meshes

The surface meshes considered in this thesis are triangular meshes. A comprehensive coverage of the geometric modeling based on triangle meshes is presented in [Botsch *et al.*, 2006].

A triangle mesh $\mathscr{M}$ consists of geometric and a topological component. The topology of the mesh can be represented by a set of m vertices

$$V = v_1, \ldots v_m$$

and a set of triangular faces connecting them

$$F = f_1, \ldots, f_k \ , \qquad f_i \in V \times V \times V \ ,$$

where each triangle specifies its three vertices from V. However, it is often more efficient to represent the connectivity of a triangle mesh in terms of the edges of the respective graph

$$E = e_1, \ldots, e_l \ , \qquad e_i \in V \times V \ .$$

The geometric embedding of a triangle mesh into $\mathbb{R}^3$ is specified by associating a 3D position $\mathbf{v}$ to each vertex $v \in V$, so each face $f \in F$ actually represents a triangle in 3D space, specified by its three vertex positions.

An important topological characterization of a surface is whether or not it is two-manifold, which is the case if for each point the surface is locally homeomorphic to a disk (or a half-disk at boundaries). Two-manifold mesh does not contain the edges that belong to more than two triangles or the vertices that belong to two surface sheets. A two-manifold mesh also does not self-intersections, so a property of being two-manifold also depends on the geometric embedding of the mesh.

5.2 Markov Random Fields on Triangular Meshes

Defining MRF on surface meshes is straightforward if one uses mesh connectivity to define the MRF neighborhood. Different possibilities are discussed here.

On the bottom line, methods based on MRF deal with labeling the set of sites to minimize the potential of site cliques. When defining MRF on the surface meshes, the MRF site can be assigned to the any basic entities of a triangular mesh — vertices, edges or faces. The vertex and edge process are indeed explored in this thesis; it is also easy to imagine a useful face process.

Further more, because of the mesh connectivity, equivalent energies can be obtained by building MRF on different mesh entities. For example, the potential of 3-clique of vertices can equivalently be expressed as the potential of mesh face, and the potential of 2-clique of vertices as the potential of an edge. This allows some flexible notation, as it will be used later in the text. The MRF sites will therefore be assigned to those entities, which we will change or label.

When wanting to smooth the mesh, an immediate approach is to do it by moving the vertices, so a MRF should be built on the mesh vertices. Alternatively, it could be done by smoothing the normals of the faces and using those to reconstruct a smooth surface. This approach has not been followed here.

The next modeling decision involves defining the neighborhood system for the sites, and the logical choice is to employ the connectivity information of the surface mesh (and therefore ensuring to stays on the surface). The main implementational advantage of this approach is obvious: with an efficient mesh data structure, connectivity provides direct access to neighbors.[1] However, one has to bear in mind that such model is generally dependent on the mesh quality, element density and mesh regularity, so these issues should be addressed when modeling.

The biggest challenge in formulating MRF on triangle meshes is dealing with its irregularity. A vertex can have a different number of adjacent vertices, and an edge can have a different number of adjacent edges (it is only faces that show some regularity there). As a result, for example, one can not directly compare the potentials corresponding to two vertices.

It is possible to imagine alternative neighborhood systems. For example, the neighborhood based on spatial proximity, which would reduce the surface mesh back to a point cloud.

Despite the simplicity and the flexibility of the formulation, MRF has not yet found its application in the field of 3D mesh modeling. Still, many concepts used

[1]Sometimes one needs to distinguish between two meanings of the word *neighbor*: neighbor in a MRF sense or neighbor in the terms of mesh connectivity. I used the expressions *adjacent* and *incident* when describing the mesh connectivity, so the word *neighbor* was used mostly in MRF sense.

in Image Analysis have over time been generalized and applied to mesh surfaces; MRF will surely follow.

5.3 Assumptions

Throughout this thesis only the two-manifold meshes are considered. All of the noise-free meshes used as underlying true surface in the testing of the developed algorithms meet this assumption. However, after adding a noise it sometimes happened that the noisy mesh contained self-intersections and was therefore not two-manifold any more. Nothing special was done with this, even though it is, for example, clear that such a mesh could not be a result of a point cloud triangulation.

Furthermore, it is assumed that the mesh has a correct topology and triangulation, so that the edges of the triangular mesh follow the ridges of the sampled surface. Based on this assumption we do not consider the topology changes of the mesh (e.g., flipping the edge that is traverse to the ridge) although this is likely to be advantageous in practical applications. Similarity, nothing is done to handle the ‘badly shaped´ triangles. The reason for this is an ambition to isolate the different effects, and focus only on the effects of developed priors. Of course, the limited time frame also called for prioritizing. Combining mesh smoothing with techniques for improving mesh triangulation and mesh quality is a part of a future work.

Chapter 6

Mesh Smoothing

The goal of this thesis is to formulate a prior for piecewise smooth surfaces and to use it for feature-preserving mesh smoothing. Two processes need to be coupled: a smoothing process defined on the mesh vertices, and an edge process for detecting discontinuities between the smooth parts. This chapter describes the first process, mesh-smoothing, where the objective is simply to smooth the mesh, without worrying about preserving features.

The presented model is an MAP-MRF, which combines an MRF smoothness prior based on absolute mean curvature and a likelihood function based on the displacements from the input mesh. A few alternative formulations of the smoothness prior are also presented. The mesh is smoothed in an optimization scheme whereby the positions of the vertices are iteratively changed according to the posterior.

6.1 What is a Smooth Mesh?

There are two general goals of the mesh soothing methods [Botsch *et al.*, 2006]. The first goal is denoising measured data. Many meshes acquired by 3D scanners contain high frequency noise, i.e., small perturbations in the vertex positions. Here, the goal is to smooth out these artifacts in such a way that the global shape, or the low frequency component, is preserved. An additional requirement is often the preservation of certain surface features like sharp edges and corners, which should not be smoothed.

The other goal of mesh smoothing is the design of high-quality, fair surfaces. This process is called mesh fairing and the resulting surfaces mush meet certain aesthetic requirements, put essentially as principle of the simplest shape. An aesthetic surface is free of unnecessary details such as noise or oscillations. Mathematical formulations of this principle often lead to the minimization of certain energy functionals, often inspired by physical processes such as spanning a membrane or bending a thin plate.

What do we expect of a mesh smoothing algorithm? Is a perfect cube less or more smooth than the perfect sphere? Should a smoothing algorithm round the sharp edges of a cube?

What we want depends of what we know about the underlying object. If it has sharp edges, we want them preserved; otherwise, we want the edges to be smoothed. In the following sections we see that by just changing a energy function we can obtain different results, in accordance with what we expect of the smoothing algorithm.

6.2 MRF on Mesh Vertices

The smoothing algorithm should iteratively re-position mesh vertices, in search of the optimal configuration. The MRF for the smoothing process is therefore built on the mesh vertices, and the role of the MRF sites is given to vertices. Each mesh vertex is associated with a MRF site. The random variable (or label) of a vertex is it's spatial position. The mesh configuration is any assignment of positions to mesh vertices.

A short note on the notation: In cases where a distinction is needed between the vertices and their labels (i.e., spatial positions), the italic font is used for the vertices and bold face font for the vectors representing the spatial positions of the vertices. For example, $\mathbf{v}_i$ is a spatial position of vertex v_i, but indices are used *only* where there is more than one vertex involved in a formula. Assigning a position to each vertex v from the set of vertices V gives a mesh configuration $\mathbf{V}$. In a similar fashion, distinction is made between the edge $e_{12} = (v_1, v_2)$ and the vector $\mathbf{e}_{12} = \mathbf{v}_2 - \mathbf{v}_1$, but the vector $\mathbf{e}_{12}$ is not a part of the random fields framework.

A smoothness prior will enable the calculation of the smoothness potential for any mesh configuration. This potential should be expressed as a sum of clique potentials.

6.3 Smoothness Potential Based on Mean Curvature

Mathematical formulation of mesh fairing leads often to minimization of certain energy functions, such as curvatures. The initial smoothing algorithm developed during the work on this thesis is based on minimizing the absolute mean curvature.

The triangle meshes are piecewise linear surfaces and curvature, as well as other differential properties, is not defined along the edges or at the vertices. However, if one assumes a triangular mesh to be a piecewise linear approximation of an underlying smooth surface, the approximation of the differential properties can be computed directly from the mesh data. A brief overview of different approaches with this as a goal can also be found in [Botsch *et al.*, 2006].

The integral absolute mean curvature at the vertex v, with respect to area attributed to the vertex can then be expressed [Dyn *et al.*, 2001] as

$$|\bar{H}_v| = \frac{1}{4} \sum_{e \sim v} |\phi_e| \, \|\mathbf{e}\| \ ,$$

where $e \sim v$ signals that the edge e is incident on the vertex v, $\|\mathbf{e}\|$ is the length of edge e and ϕ_e is the dihedral angle attributed to the edge e. The dihedral angle is the angle between the normals of the two faces sharing an edge. The absolute dihedral curvature ignores the sign of the dihedral angle, treating the convex and concave angles equally. Illustration of dihedral angles in 2D is shown in Figure 6.1

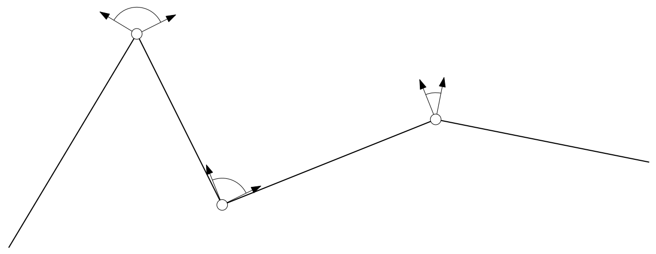


Figure 6.1: Dihedral angles (angles between the normals) in 2D. Large dihedral angle signals a sharp turn. Small dihedral angle signals smoothness.

Motivated by the above expression, we want to define the potential of the smoothness prior as

$$U_V(\mathbf{V}) = \sum_{e \in E} |\phi_e| \, \|\mathbf{e}\| \ , \tag{6.1}$$

ignoring the constant factor and summing over all non-boundary edges in the mesh (or using $\phi_e = 0$ for the border edges).

This expression is a summation over all the edges, but to fit it into the formal MRF framework a summation over the cliques of vertices is needed. This is easily achieved, as it is the positions of four vertices that define a dihedral angle of each non-boundary edge.

6.3.1 MRF Framework, Neighborhoods and Clique Potentials

The suitable neighborhood for the MRF smoothness prior is defined as follows: two different vertices v_1 and v_2 are neighbors if they belong to the faces that share an edge. This is obviously a well defined neighborhood system. Given a vertex v of valency n, the neighborhood $\mathcal{N}_v$ of the vertex v is the set containing its n adjacent vertices and n vertices that share an opposite edge with v, as illustrated in Figure 6.2.

For this neighborhood model there are 4 clique types: *i)* single vertex, *ii)* an edge, *iii)* a triangle, and *iv)* two triangles sharing an edge. For the smoothness prior we do not assign any potential to the first three types of cliques, as they do not provide any information about curvature.

The potential of the 4-clique having the connectivity setting as in Figure 6.3 is then

$$U_{V4}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4) = |\phi_{e_{23}}| \, \|\mathbf{e}_{23}\| \ ,$$

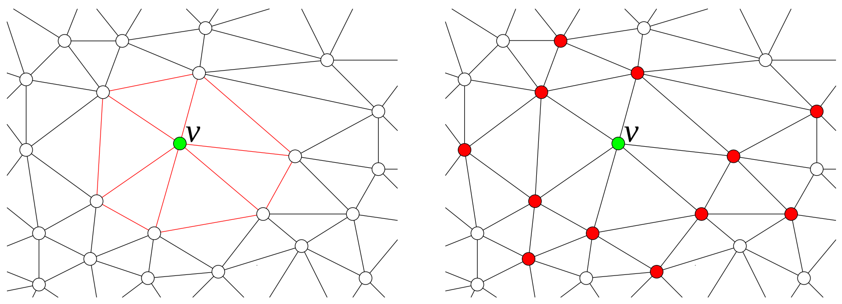


Figure 6.2: *Left:* Changing the position of a single vertex v affects the dihedral angles of the edges incident to the vertex and the edges opposite to the vertex (all marked red). The lengths of the edges incident to v are also affected. *Right:* After changing the position of the vertex v we need to access its adjacent and opposite vertices to (re-)calculate the dihedral angles of affected edges; all those vertices (marked red) form the neighborhood of the vertex v.

where $\phi_{e_{23}}$ is a dihedral angle of the edge $e_{23} = (v_2, v_3)$, and $\|\mathbf{e}_{23}\| = \|\mathbf{v}_3 - \mathbf{v}_2\|$ is the length of the edge e_{23}. Even though it is not explicitly stated in this formulation, the dihedral angle $\phi_{e_{23}}$ is also a function of the positions of vertices v_1 and v_2 (and all the positions are given by the labeling $\mathbf{V}$).[1]

For the setting shown in Figure 6.3, the dihedral angle $\phi_{e_{23}}$ is computed as

$$\phi_{e_{23}} = \arccos \frac{\mathbf{n}_{123} \cdot \mathbf{n}_{243}}{\|\mathbf{n}_{123}\| \|\mathbf{n}_{243}\|} ,$$

where $\mathbf{n}_{123}$ and $\mathbf{n}_{243}$ are normals of the two faces and $\cdot$ stands for the scalar (dot) product of vectors. The normal n_{123} is given by

$$\mathbf{n}_{123} = \frac{(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)}{\|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)\|} ,$$

where $\times$ is the vector (cross) product of two vectors. The other normal is obtained in the same manner.

We can now express the smoothness potential as the sum of potentials of the 4-cliques

$$U_V(\mathbf{V}) = \sum_{(v_1,v_2,v_3,v_4) \in \mathscr{C}_4} U_{V4}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$$

and it is indeed equivalent to the formulation of absolute mean curvature.

After this excursion in a formal MRF framework, we will close the full circle and, because of simplicity of notation, continue using summation over edges, not the 4-cliques of vertices. In a bit sloppy notation, I will say that the edge e is in the neighborhood $\mathscr{N}_v$ of the vertex v if v is in the 4-clique corresponding to that edge.

[1] I did attempt to devise the notation with explicit dependencies. It was a kingdom of brackets, obscure indices and hard-to-read formula. Only the most important dependencies are explicitly stated here, for example, the clique potential being primarily the function of labeling (i.e., vertex positions).

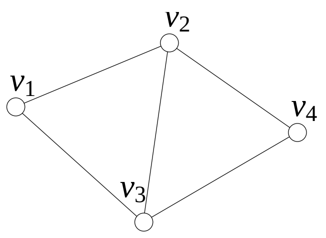


Figure 6.3: A 4-clique of vertices. The potential of a 4-clique depends on the dihedral angle between the normals of the two faces, and the length of the edge (v_2, v_3). To get hold of the dihedral angle we need to know the positions of all the vertices in the clique.

The edge-neighborhood of the vertex is the set of affected edges, as illustrated and explained in Figure 6.2. Basically the same could be obtained by defining MRF on mesh faces, and the potential defined here would be a potential of the 2-clique of faces.

This smoothness potential, based on the absolute mean curvature, depends on the dihedral angles and the edge lengths. The dependence on the edge lengths makes it prefer smaller objects — a pure scaling of a mesh would lessen the energy. This means that the use of this prior could lead to the undesirable shrinking of the mesh.

When computing the integral mean curvature it is required that the longer edges contribute more than the shorter ones, but is the dependence on the edge length also required when smoothing? This question opens the possibility of the alternative formulation, with smoothness energy being the sum of dihedral angles. A comparison of the results obtained by using the two different formulations is in the results section.

The smoothness prior as defined here served as the basis of most of the work in this thesis. However, some of the alternative formulations, which are presented next, have proved to have other desirable properties. The results of using all the priors are presented and compared later.

6.4 Alternative Formulations of the Smoothness Potential

In all the alternative formulations presented here, as in the original, smoothness energy is obtained as a sum of potentials over all non-boundary edges (or, equivalently, all vertex 4-cliques, or, also equivalently, two neighboring faces)

$$U_V(\mathbf{V}) = \sum_{e \in E} f(e) \; .$$

The function $f(e)$ is the evaluation of the smoothness based on the two neighboring faces, and can be defined in different ways. As everything else remains the same, we will look only at this function, which I call for smoothness potential contributed to an edge, even though it is a function of the position of four vertices.

The first alternative formulation is already mentioned, it is an angle-based potential that does not depend on the edge length

$$f_1(e) = |\phi_e| \, .$$

6.4.1 Quadratic and Square-Root Smoothness Potentials

Potentials here are not directly proportional to an angle between the faces that share the edge *e*, but are instead functions of the difference between the normals of the two faces.

Toward the end of the project period the need for a stronger smoothness prior has emerged – the edge process was taking care of the edges, so a stronger smoothing prior could be applied to the rest of the mesh. I therefore tested the over-smoothing quadratic potential developed by [Szeliski and Tonnesen, 1992], which takes the form

$$f_2(e) = \|\mathbf{n}_{123} - \mathbf{n}_{243}\|^2 \ ,$$

where $\mathbf{n}_{123}$ and $\mathbf{n}_{243}$ are the normals of the faces that share edge *e*, as in Figure 6.3.

A related feature-preserving square-root prior developed by [Diebel *et al.*, 2006] was used very successfully for mesh smoothing while preserving features, so I tested it for comparison. The square-root prior takes the form

$$f_3(e) = \|\mathbf{n}_{123} - \mathbf{n}_{243}\| \ ,$$

which is a Euclidean distance in normal space.[2]

That the quadratic potential is over-smoothing, and the root potential feature-preserving has been shown by experiment. The intuition behind the performances of those potentials can also be easily obtained by looking at a simple example, which also sheds a light on the feature-preserving performance of original angle-based potential.

To compare the performances of the three priors (quadratic, root and angle based) we can try to evaluate the different positions of a vertex on the ridge top. This is illustrated in Figure 6.4 where we look at the 2D problem, corresponding to the cross section of the surface ridge. Two extreme positions of the vertex connecting two slanted lines (which would in 3D correspond to the ridge connecting two slanted planes) are evaluated. The position C_1 preserves the ridge, while the position C_2 cuts the top of the ridge. The preference between C_1 and C_2 according to the three potentials can easily be found by looking at the normals, angles between the normals and the triangle formed by the tips of the normals, all shown in Figure 6.4.

[2]The square-root potential defined here is the root of a quadratic potential, so the names of the potentials can be misleading. I adapted directly from the original nomenclature.

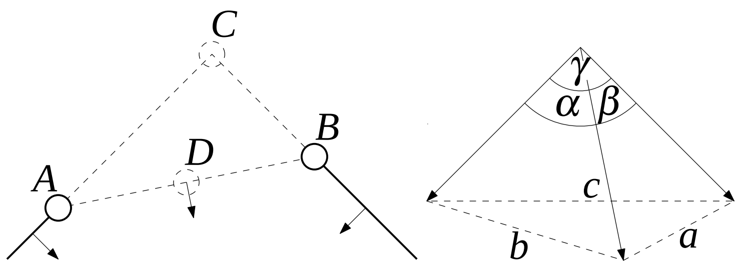


Figure 6.4: *Left:* A 2D arrangement corresponding to a cross section of the surface ridge. Vertices A and B belong to slanted planes on the opposite sides of the ridge. Vertex C connects A and B. Two extreme positions of the vertex C are evaluated: position C_1 on the top of the ridge; and the position C_2 in between vertices A and B. *Right:* The normals, the angles between normals, and the triangle defined by the tips of the normals, for the two positions of the vertex C: position C_1 results in only one non-zero angle, the ridge angle γ; position C_2 breaks γ in two smaller angles α and β.

The angle-based potentials operate on the sum of the absolute angles (dihedral angles for 3D, outer angles for 2D) in the neighborhood, which means that the choice between position C_1 and C_2 depends on the evaluation of γ and $\alpha+\beta$. But, as we have $\gamma=\alpha+\beta$, the angle-based potentials are indifferent about positions C_1 and C_2. Actually, angle-based potentials are indifferent as long as wa stay in the triangle $\triangle AC_1B$ since the sum of the outer angles around the ridge is always equal to γ. This means that, in case of the 2D shapes, the square and dodecagon (and any convex polygon) have the same potential and we can not expect the corners of the square to be rounded. A potential that rounds the corners should discriminate between a big angle and two half angles, so one should look for potentials that are not linear with angle.

The root potential operates on the Euclidean distance between the tips of the normals, so we look at the triangle defined by the tips of the normals. We have $c < a+b$ due to the triangle inequality. The root potential prefers one large angle instead of two smaller, and the position C_1 is preferred. This explains the feature-preserving properties of the root potential.

The quadratic potential operates on the squared Euclidean distances between the tips of the normals. Looking at the triangle this time we have $c^2 > a^2+b^2$, an inequality arising from the fact that the triangle in question is obtuse. This inequality holds always, since the triangle defined by the tips on the normals is obtuse as long as γ is smaller than 180°. The quadratic potential will therefore prefer breaking a large γ into two smaller angles, cutting off the top of the ridge.

Similar geometric properties, just in 3D, are causing the difference in the feature-preserving properties of the three potentials. The situation in 3D is, of course, more complex. Moving just one vertex at the time means that the vertices *along* the ridge also influence the potential. This influence will make the indifferent angle-based potential to actually preserve ridges.

A descriptive comparison of the angle-based, quadratic, square root and the thresholded potential, which is to be defined next, is found in the Figure 6.5.

6.4.2 Thresholded Smoothness Potential

A thresholded potential was tested in another attempt of achieving feature-preservation by adjusting the smoothness potential. It takes the simple form of

$$f_4(e) = \max(|\phi_e|, \phi_0)\|\mathbf{e}\| ,$$

where ϕ_0 is some threshold angle. The basic potential used here is curvature-based, but it could just as well be an angle-based or quadratic potential described above. The formulation of the potential effectively stops the smoothing of those edges that are sharper than the threshold.

If one keeps track of the occurrences of edges that reach the smoothness threshold, it will result in classifying edges into low-potential and high-potential. A thresholded smoothness potential is therefore equivalent to the implicit edge labeling.

6.4.3 Comparison of the Four Potentials

In Figure 6.5 the angle-based, quadratic, square-root and the thresholded potential are compared. The figure brings the plot of the potential functions against the dihedral angle, and a demonstration of the effect of using those potentials. A corresponding problem in 2D is again used, with two lines meeting at a different slopes, representing an intersection of two surfaces meeting in a ridge. The potentials of the connecting point are plotted in different colors.

For the small ridge sharpness, all potentials have similar effect, but as the ridge sharpness grows, four quite different patterns emerge. It is again confirmed that angle based formulation is indifferent to the position of the mid-point, as long as the curve remains concave. The quadratic formulation f_2 clearly has the tendency of over-smoothing sharp corners, while the square root formulation f_3 behaves in a drastically different way, maintaining the big turn and preserving a sharp edge. The thresholded potential f_4 looks a lot like the square root potential, and has a strong tendency of preserving, even enhancing, ridges.

To conclude, the choice of the smoothness potential will obviously have influence on the smoothing performance. If we want the smoothing algorithm to round the ridges, quadratic potential should be chosen. If we want to preserve the ridges, the choice should fall on the square-root potential, while using the thresholded potential allows specifying the angle at which the rounding of the ridge stops.

The results by applying the described potentials on mesh smoothing can be found in 7.5.

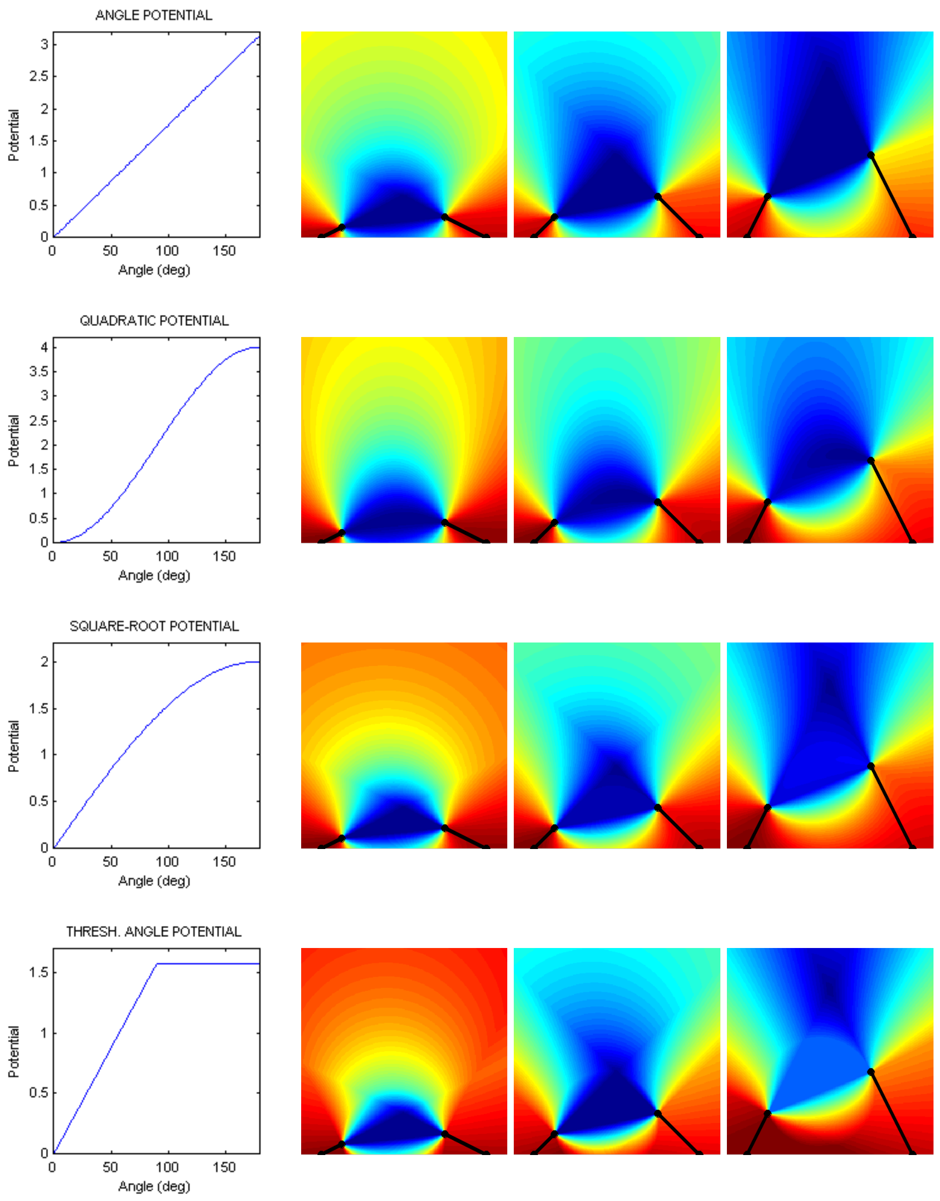


Figure 6.5: Comparison of the four smoothness potentials. *Top to bottom:* Angle potential, quadratic potential, root potential and threasholded angle potential. *Left:* Potentials versus the angle. *Columns 2–4:* Visualization of the corresponding potentials in 2D for the ridges of different sharpness. Blue color signals the small potential, red color signals the big potential.

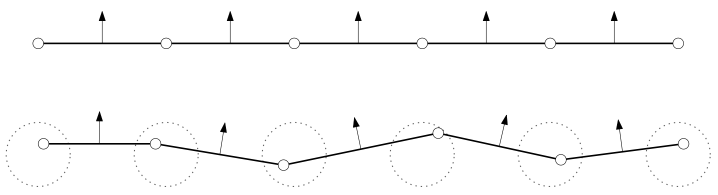


Figure 6.6: The 2D illustration of the behavior of face normals of the planar surface when positions of the vertices get perturbed by additive Gaussian noise.

6.4.4 Normals Under Noise

All described smoothness potentials are the functions of the face normals, so a question about what happens to the mesh normals under noise seemed worth examining. A distribution that can be approximated by the Gaussian would, for example, justify the use of the quadratic potential.

A plain surface represented as triangle mesh is considered, so all the face normals are the same, and all the tips of the normals coincide. We look at the distribution of the tips of the normals after adding a Gaussian noise to the vertex positions. The corresponding 2D situation is illustrated in the Figure 6.6.

The tips of the normals lie on the unit sphere, so in general the distribution of the normals can not be Gaussian. But what happens when the noise is relatively small compared to the triangle size?

The normal of the face containing vertices v_1, v_2 and v_3 can be computed by

$$\mathbf{n}_{123} = \frac{(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)}{\|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)\|} \ .$$

The denominator in this expression is twice the area of the triangular face. Assuming the noise small enough that it does not change the area of the triangular faces, we can consider the denominator to be constant.

As for the nominator, it contains a cross product of two triangle sides. Assuming the noise small compared to the length of the sides, the result of the cross product will be normally distributed. The shape of this Gaussian depends on the shape of the face triangle and can be very elongated or flat, even with the isotropic vertex noise.

To sum up, for a noise that is small compared to the size of the triangles, the distribution of the tips of the normals will indeed follow Gaussian distribution. However, the noise used in the experimental part of this thesis is usually to large for this assumption to hold.

6.5 Likelihood Function

Applying the smoothness prior should theoretically smooth the mesh until the perfectly smooth surface is found. In practical applications we want the output of smoothing to relate to the input mesh, which we denote as the initial vertex configuration $\mathbf{V}^0$.

We assume that the input mesh has an underlying true surface, which was corrupted by the noise of the data-acquisition device. It is the true surface that we want to recover, having a prior knowledge about it. In the iterative recovery method, the likelihood function describes the probability of observing the input mesh given the current candidate configuration $\mathbf{V}$.

In other words, the observation model assures that we do not move too far away from the input mesh.

Measurement noise is assumed to be isotropic and Gaussian, and therefore the quadratic function for likelihood energy is chosen

$$U_L(\mathbf{V}^0|\mathbf{V}) = \alpha \sum_{v \in V} \left\|\mathbf{v}^0 - \mathbf{v}\right\|^2 ,$$

where $\mathbf{v}^0$ denotes the initial position of vertex v in the input mesh. The constant α is used as the weight determining how much faith one has in the data.

Let me just note that the above can be seen as a contribution of single-vertex cliques, with clique potential for the label $\mathbf{v}$ being given by

$$U_{v1}(\mathbf{v}) = \alpha \left\|\mathbf{v}^0 - \mathbf{v}\right\|^2 .$$

Assuming non-isotropic Gaussian noise, which captures the characteristics of the sensor (the noise is often elongated along one axis), the likelihood function can be generalized [Diebel *et al.*, 2006] by applying Mahalanobis distance.

Another possible improvement employed by [Diebel *et al.*, 2006] involves using the distance from $\mathbf{v^0}$ to the point on the surface defined by $\mathbf{V}$, which is closest to $\mathbf{v^0}$.

6.6 Probability Distributions

After combining the smoothness priors with the observation likelihood, the posterior energy of a mesh vertex configuration takes the form

$$U_{\text{POST}}(\mathbf{V}) = \alpha \sum_{v \in V} \left\|\mathbf{v}^0 - \mathbf{v}\right\|^2 + \sum_{e \in E} |\phi_e| \, \|\mathbf{e}\| ,$$

leading directly to joint probability, which is a Gibbs distribution

$$P(\mathbf{V}) = \frac{1}{Z} \exp\left(-\frac{1}{T} U_{\text{POST}}\right) ,$$

where Z is a normalizing constant, and T is the temperature, which will play a role in the optimization scheme.

A curvature based prior is used above, but any other smoothness prior can be thresholded as well.

Similarly, the part of the energy affected by the position of the vertex v is given by

$$U_{\text{POST}}(\mathbf{v}) = \alpha \left\|\mathbf{v}^0 - \mathbf{v}\right\|^2 + \sum_{e \in \mathcal{N}_v} |\phi_e| \, \|\mathbf{e}\|$$

and it leads to conditional probability of the position of vertex v, given its neighborhood

$$P(\mathbf{v}|\mathcal{N}_v) = \frac{1}{Z} \exp\left(-\frac{1}{T} U_{\text{POST}}(\mathbf{v})\right) .$$

6.7 Optimization for Mesh Smoothing

Having defined the model one has to decide on an optimization scheme: How to find the vertex configuration that maximizes the posterior distribution?

Mesh smoothing is a problem of large dimensionality (even the labels are three dimensional), and the interaction between the vertices is complex. Instead of making any assumptions about the energy function, I applied the Metropolis sampler with simulated annealing, which was used in many vision-based MAP-MRF applications [Geman and Geman, 1984], [Hartelius and Carstensen, 2003].

The basic algorithm is already presented in Section 4. Here I describe the issues specific for the problem at hand.

6.7.1 Metropolis Sampler

The Metropolis sampler works by randomly choosing the new configuration candidate and accepting with a certain probability, according to the Metropolis criterion. As we obtain information about probability distribution by sampling it, it is important that those samples are made in the part of the configuration space where the probability density is high.

Firstly, to ensure reasonably good initial configuration the sampling is started around the input mesh $\mathbf{V^0}$, so the input mesh plays two roles: both as the initial configuration and as a data term.

Secondly, the new configuration candidate should be sampled in vicinity of the current configuration [Li, 2001]. New configuration candidate is found by changing a label of a single site, and in this case it means finding a new random position for a single mesh vertex. To stay in the vicinity of the current configuration, the new position is sampled according to a Gaussian distribution around the current position of the vertex

$$\mathbf{v}' = \mathbf{v} + \mathbf{s}, \quad \mathbf{s} \sim N(\mathbf{0}, \Sigma) .$$

Initially I used only an isotropic Gaussian, where $\Sigma = \sigma^2\mathbf{I}$, and σ is the variance, but since one is generally more interested in smoothing in the direction of the normal to the surface, the Gaussian was later on allowed to be elongated in the

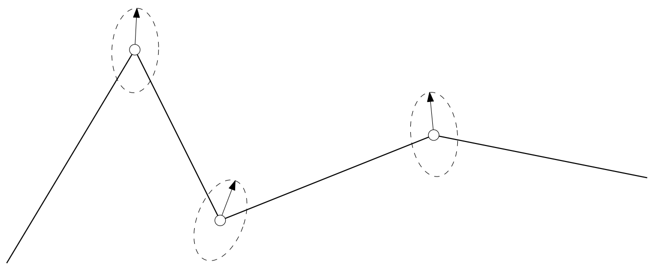


Figure 6.7: For smoothing the mesh we are mostly interested in moving the vertices along the normal direction. The Gaussian distribution used for sampling is therefore elongated in the direction of the vertex normal.

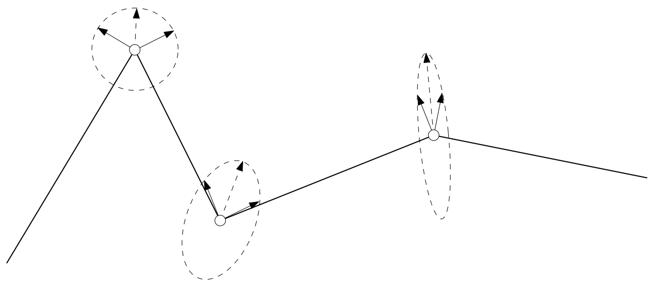


Figure 6.8: An idea for improving the sampling. In case of the vertex in relatively smooth area we want to sample mostly in normal direction. For vertices where there is less consensus between the normals of incident faces we want a less elongated Gaussian. Compare with Figure 6.7.

normal direction

$$\Sigma = \mathbf{R} \begin{pmatrix} \sigma_N^2 & 0 & 0 \\ 0 & \sigma_T^2 & 0 \\ 0 & 0 & \sigma_T^2 \end{pmatrix} ,$$

where σ_N and σ_T are the variances in the normal direction and in the tangent plane, and $\mathbf{R}$ is the rotation matrix that aligns the direction of σ_N with the direction of the vertex normal. This is illustrated by the corresponding 2D configuration in Figure 6.7.

Box-Müller transformation, which was used to generate samples from a Gaussian distribution, is outlined in Appendix A.

The normal to the surface at a mesh vertex is approximated by the angle-weighted sum of incident face normals [Bærentzen and Aanæs, 2005]

$$n_v = \frac{\sum_f \phi_f n_f}{\sum_f \phi_f} ,$$

where ϕ_f is the face angle of face f at vertex v, n_f is the normal of face f and the summation visits all the faces incident to v.

The sampling algorithm therefore calls for the estimation of the two parameters σ_N and σ_T. Those should be big enough to allow the configuration to change sufficiently fast, but small enough to allow a lot of updates. The role of σ_N and σ_T is also discussed later.

The sampling scheme proved very important for the success of optimization. Figure 6.8 illustrates an initial idea for possible improvement of the sampling scheme. Vertices in the smooth areas should move mostly in normal direction, but the vertices in noisy areas of the mesh should be allowed to move in all directions, i.e., in the direction of the normals of all faces incident to v. So, the idea is to use the following permutation scheme

$$\mathbf{v}' = \mathbf{v} + \frac{\sum_f s_f \phi_f n_f}{\sum_f \phi_f}, \quad s_f \sim N(\mathbf{0}, \sigma) \ .$$

The variance σ should probably also be adjusted differently for the smooth and for the noisy areas, and a care should be taken in the case of very sharp corners.

Another sampling issue is the ordering of the update. All vertices should be updated with the same frequency, but not in some given order to prevent the propagation of configuration in a certain direction. The scheme is used where all the mesh vertices are updated sequentially in one iteration sweep, but according to the configuration of the *previous* iteration.

Lastly, it is worth considering whether all moves should be allowed. In most practical applications the input mesh will be two-manifold, and a two-manifold output is a requirement. A question can be posed whether it is desirable to introduce self-intersections while heating the system. Simulated annealing should allow a large part of the configuration space to be examined. However, it is only a small part of the configuration space that represents the two-manifold mesh, and the optimization should maybe be limited to those configurations. This would require not allowing for the moves that introduce self-intersections. To detecting self-intersection could significantly slow down the smoothing algorithm and would require the extension of the mesh data-structure.

6.7.2 Simulated Annealing

In a simulated annealing scheme, one starts with a high temperature, which allows a higher degree of randomness, and gradually cools the system down. The choice has to be made about the initial temperature, how many iterations are performed at each temperature, and how much the temperature is decremented at each step as cooling proceeds. To assure that the system does not get trapped in a local minimum the initial temperature needs to be set high enough and the cooling should not be too rapid.

The logarithmic cooling scheme, which was theoretically proven to converge to minimal energy [Geman and Geman, 1984] is too slow for practical use, so the faster cooling scheme suggested by [Kirkpatrick *et al.*, 1983] was used. The temperature T_i of the i-th iteration is obtained by

$$T_i = kT_{(i-1)} \ ,$$

where the values of the constant k are typically chosen in the range 0.8 – 0.99.

The initial temperature should be such that the first few sweeps of the algorithm allow for almost all updates [Li, 2001]. This guideline served as basis for the (optional) estimation of the initial temperature. We know that the probability of accepting the perturbation is a function of the energy change, so an initial dummy sweep is preformed where all the vertices get perturbed according to the chosen sampling scheme. The change of energy for each dummy perturbation is monitored and the initial temperature is chosen so that even the perturbation with the biggest increase in energy has p_0 probability of being accepted. This leads to the following expression for the initial temperature

$$T_0 = -\frac{\max \Delta U}{\log p_0} \ ,$$

where

$$\max \Delta U = \max U(f') - U(f)$$

is the maximal energy change for all the perturbations of the dummy loop.

The guidelines mentioned here make it easier to choose T_0 and k high enough for a good annealing scheme, but one also has to consider the running time and the rate of convergence. Simulated annealing can be rather slow, and to investigate the possibility of applying a local optimization method for mesh smoothing, a smoothing with a zero temperature was sometimes used.

By smoothing with a zero temperature I consider a greedy optimization scheme where only nearest local minimum but it's result can help determine whether gradient descent methods could be used for the problem at hand.

The convergence of the implemented method was not decided by specific convergence criterion; instead I simply let the optimization run for a certain number of iterations and I monitored the energy and the number of re-positioned vertices. From the energy plot was is evident that the energy converges as the number of updated vertices becomes low. This makes it possible to use the number of updates as convergence criterium. The optimization was usually stopped when only a small number (a few percent) of the vertices got updated in each iteration.

6.8 Summary

In this chapter I formulated mesh smoothing as a MRF labeling problem on the set of mesh vertices. Smoothness potential was defined as a sum of the clique potentials, where a clique corresponds to a pair of neighboring faces, and a potential is a function of dihedral angle.

A number of alternative smoothness potentials was also formulated. All defined potentials are closely related but differ in the evaluation of the smoothness from the normals of two neighboring faces. Using analogy with 2D I showed that the different formulations of smoothness potentials result in different smoothing properties, e.g., quadratic potential is over-smoothing and square-root potential is feature preserving.

Smoothness prior was combined with likelihood function into MAP-MRF framework. The optimization method based on the simulated annealing and Metropolis sampler was presented. The specific issues as cooling schedule and perturbation scheme were discussed.

In the next chapter I present experimental results of using the smoothing method described here.

Chapter 7

Mesh Smoothing, Results and Discussion

In this section I first present one typical mesh-smoothing vertex process and then analyze the effect of changing input parameters. A discussion on both the modeling and optimization follows. Lastly, the different smoothness priors are compared and some possible improvements suggested.

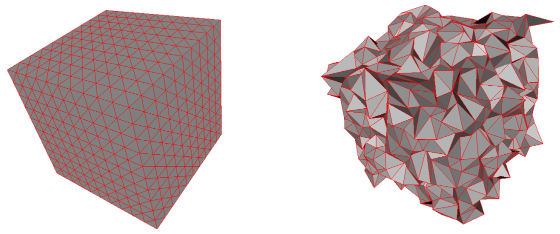
7.1 Typical Smoothing

A small $10 \times 10 \times 10$ cube is used as the testing object, since the small size allowed for many repeated testings. The cube is artificially corrupted by Gaussian noise, $\sigma_{\text{noise}} = 0.3$ of the mean edge length, see Figure 7.1. The distance between the underlying perfect cubes and the noisy cube is also shown in the figure.

To be able to quantify and compare the results, a distance measure between two meshes is needed. I used the symmetrical vertex-to-surface Hausdorff distance and symmetrical vertex-to-surface mean error. In short, Hausdorff distance is a maximum-minimum distance and measures the biggest disparity between the two surfaces. Mean error measures the mean distance between two surfaces. Those distance measurements covered in more detail in Appendix B.

The initial temperature T_0 for the typical smoothing was estimated by the dummy loop. We set the probability of accepting the worst move to $p_0 = 0.5$. Annealing constant was chosen to be $k = 0.9$, which is a reasonable balance between the quality and efficiency of optimization. The size of the sampling step was chosen to be $\sigma_N = 2\sigma_T = 0.1$ average edge length. Those are the values usually producing good result. Finally, the weight of the data term α, was chosen to be 0.1 by trying out a few different values. The size of α depends on our faith in data. It also depends on the average edge length and on the smoothness prior, which makes it hard to give a general guideline for choosing α.

A few frames from a typical smoothing of a noisy cube can be seen in Figure 7.2, together with graphs showing the energy minimization and the graph mon-



distances:
$d_H = 1.25 \quad (1.10 \text{ av.e.l.})$
$d_{ME} = 0.28 \quad (0.24 \text{ av.e.l.})$

Figure 7.1: Most used testing model, synthetically made $10 \times 10 \times 10$ cube *(left)*, and the same model corrupted with Gaussian noise, $\sigma_{noise} = 0.3$ average edge length *(center)*. *Right:* Hausdorff distance and mean error between the two models, expressed also in average edge length of the uncorrupted mesh.

itoring the number of updates. After 10 iterations the mesh is very noisy due to the high temperature. After 50 iterations the effects of cooling can be seen: the mesh surface is visibly smoother. After 100 iterations it is even smoother. The result after 100 and 200 iterations varies slightly, implying convergence.

The number of the updated vertices falls rapidly with the temperature; initially the most of the model's 602 vertices were updated, but the number fell under 100 in around 50 iterations. From the plot of the potentials we see initial increase of energy due to high temperature and a quick convergence. The likelihood potential increased as the model left the initial configuration, but likelihood contributes very little to the total potential.

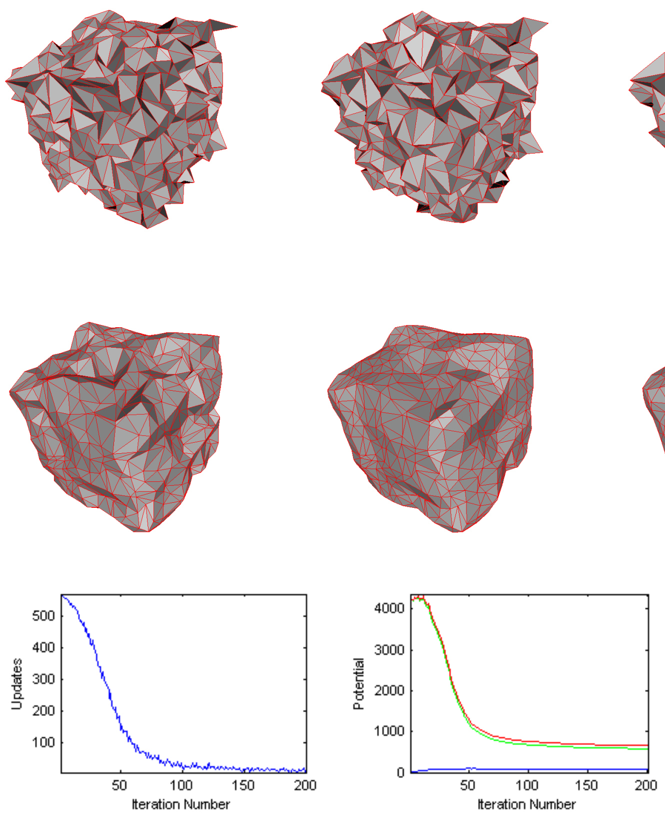
Parameters used for this smoothing are also those that typically produce good results. In the following section I justify the choice of parameters, and present a series of tests where each of the parameters is systematically changed until its influence on the smoothing scheme is visible.

7.2 Parameter Choice

The correct parameter choice is, of course, essential for good results. We can check the influence of the following parameters:

- T_0 and k, initial temperature and annealing constant,
- σ_N and σ_T, covariances of the random sampling around a vertex,
- α, weight of the data term.

Let us look at the effect of changing each of these parameters.



distances,
$N = 100$:
$d_H = 0.877$
$d_{ME} = 0.161$

distances,
$N = 200$:
$d_H = 0.875$
$d_{ME} = 0.157$

Figure 7.2: One typical smoothing of a noisy cube. Parameters used: data weight $\alpha = 0.5$, initial temperature (estimated by the dummy loop) $T_0 = 10.3$, annealing constant $k = 0.9$, sampling step $\sigma_N = 2\sigma_T = 0.1$ average edge length. *Top 2 rows:* The mesh configuration for iterations 1, 10, 20, 50, 100 and 200. *Bottom left:* The number of updated vertices over time. *Bottom center:* Potentials over time: blue — likelihood potential, green — smoothness potential, red —- total potential. *Bottom right:* Distance between the outcome mesh and the uncorrupted cube model after 100 and 200 iterations.

7.2.1 Cooling Scheme

The cooling schema is tested first. Figure 7.3 illustrates a very slow cooling process with annealing constant $k = 0.99$. To allow for the temperature to drop close to zero, 500 iterations was used. Visually this is the best smoothing result, and the final energy is also minimal. However, the distance from the uncorrupted cube is not the smallest. It seems that the high temperature allowed the strongly weighted prior to take over.

Faster cooling, with the annealing constant $k = 0.8$, is shown in the Figure 7.4. The temperature drops faster and so do the number of updated vertices and energy. Results after 100 iterations are just slightly worse than for the slower scheme.

In Figure 7.5 I tested the annealing to extremes, smoothing with a zero temperature, accepting only those updates that minimize energy, which is clearly visible in the energy plot. Results are very good — especially the distances.

To wrap up the investigation about temperature we have a smoothing with very high initial temperature in the Figure 7.6. The number of accepted updates and energy stays high longer, and the system stays longer time in the mode of randomly accepting all updates. Eventually the temperature drops and smoothing occurs, but results are not better than previously.

From the little experiment we can see that simulated annealing helps to find the low energy and visually smoothest mesh. The results obtained by faster schemes are slightly worse: high temperature introduces noise, and the system is not given enough time to recover. Cooling at zero temperature also produced relatively good result.

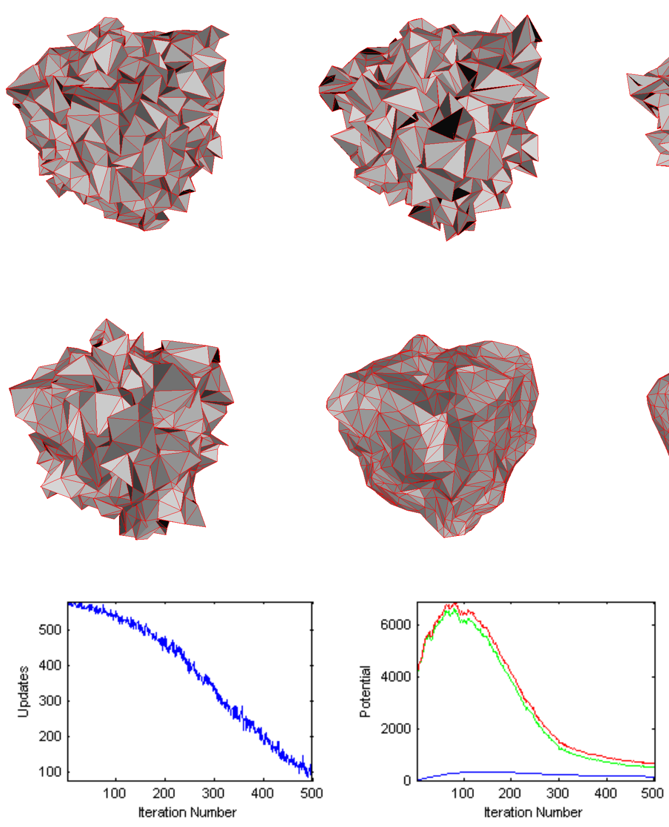
7.2.2 Sampling Size

Parameters σ_N and σ_T determine the size of the sampling step. Smaller sampling size leads to subtle changes in configuration between iterations, with a larger number of vertices updated. Bigger sampling size results in fewer and less subtle changes.

Heuristics provides the general strategy for selecting the size of the sampling step [Frost and Heineman, 1997], but I based my selection on smoothing performance. Good results were obtained with the values around $\sigma_N \approx 2\sigma_T \approx 0.1$ of the average edge length.

Choosing larger sampling size, as in Figure 7.7, would introduce a lot of noise during the initial period with large random perturbations resulting in a jumble of vertices and edges. The increase of likelihood potential is visible on the energy plot. The system does not recover from the noisy configuration, but some smoothing happens at lower temperatures. Combining big sampling size with zero temperature, in an attempt of eliminating the introduction of noise did helped a bit, see Figure 7.8.

As demonstrated in Figure 7.9, small sampling size does not allow for enough movement, vertices make small jumps around the initial positions. The likelihood

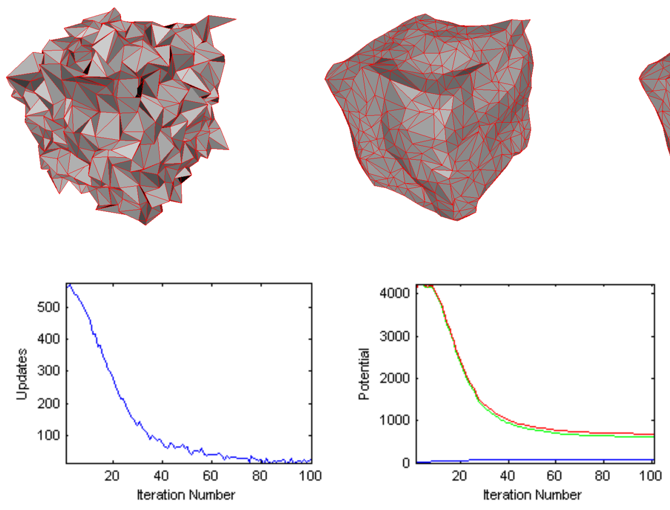


distances:

$d_H = 1.24$

$d_{ME} = 0.23$

Figure 7.3: Very slow cooling, annealing constant $k = 0.99$. *Top 2 rows:* Iterations 10, 50, 100, 200, 300 and 500. *Bottom:* Updated vertices over time, potentials over time (blue – likelihood, green – smoothness, red – total) and distance to uncorrupted cube.

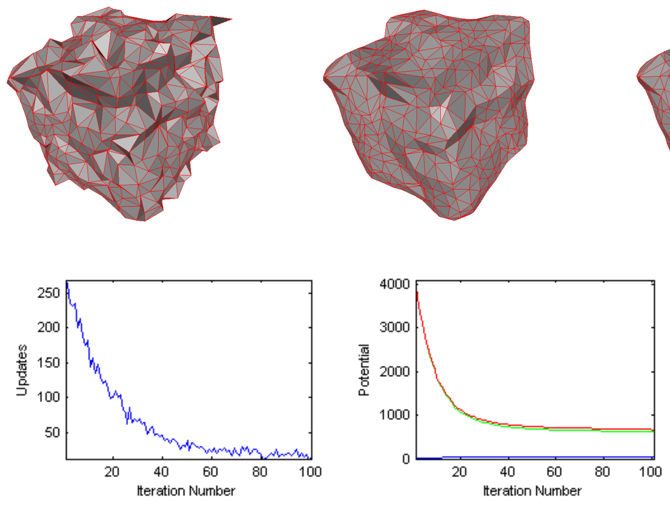


distances:

$d_H = 0.95$

$d_{ME} = 0.18$

Figure 7.4: Faster cooling scheme, annealing constant $k = 0.8$. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue – likelihood, green – smoothness, red – total) and distance to uncorrupted cube.

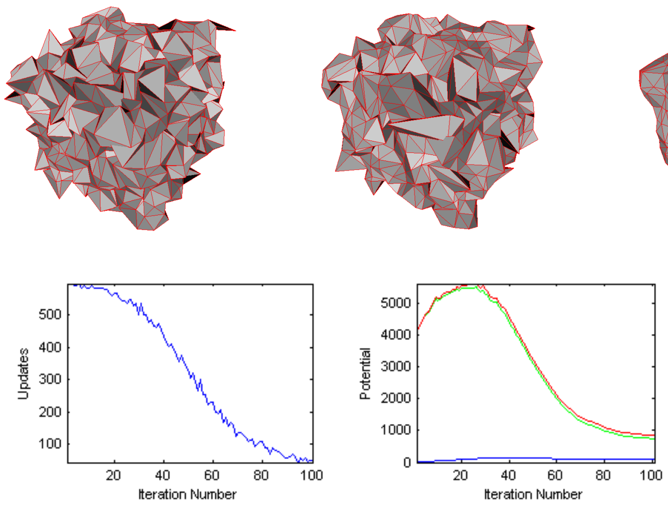


distances:

$d_H = 0.81$

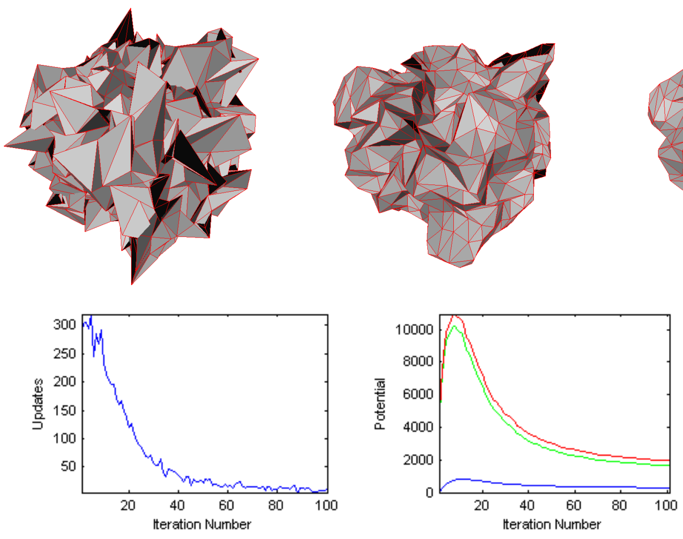
$d_{ME} = 0.14$

Figure 7.5: Smoothing with a zero temperature, $T_0 = T_k = 0$. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.



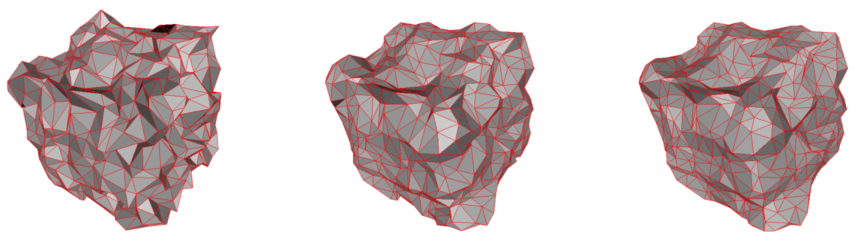
distances:
$d_H = 0.89$
$d_{ME} = 0.19$

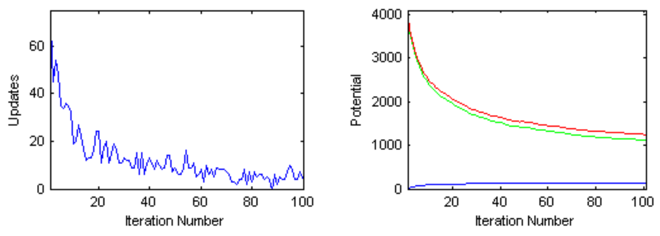
Figure 7.6: High initial temperature, $T_0 = 50$. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.



distances:
$d_H = 1.57$
$d_{ME} = 0.45$

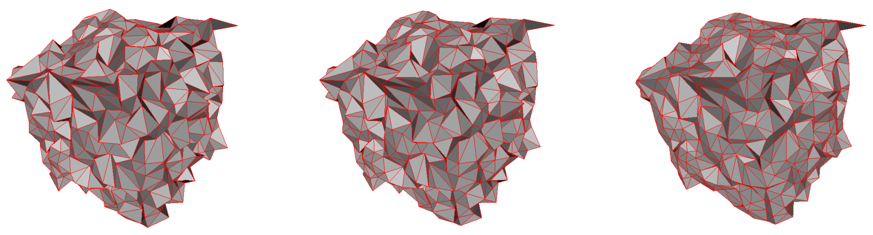
Figure 7.7: Big sampling step for random perturbation of vertices, $\sigma_N = 2\sigma_T = 1$ average edge length. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.

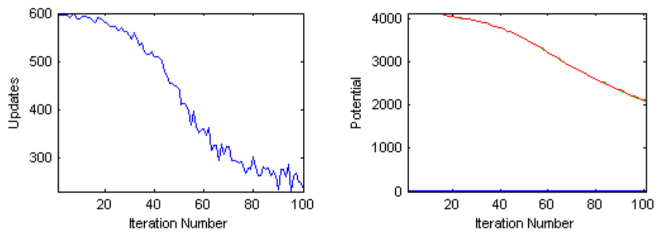




distances:
$d_H = 0.97$
$d_{ME} = 0.21$

Figure 7.8: Smoothing with a zero temperature, $T_0 = T_k = 0$, with big sampling step for random perturbation of vertices, $\sigma_N = 2\sigma_T = 1$ average edge length. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.





distances:
$d_H = 1.29$
$d_{ME} = 0.21$

Figure 7.9: Small sampling step for random perturbation of vertices, $\sigma_N = 2\sigma_T = 0.01$ average edge length. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.

potential stays very small, and the smoothness potential decreases very slowly. The number of updates falls, but stays high despite the low temperature. When it starts to get cold vertices are still close to their initial position. From there, the system will eventually find the closest local minimum, but the high update number after 100 iterations indicates a very long smoothing before convergence occurs.

With the conclusion that a choice of a good sampling size seems very important for smoothing process I wrap up the discussion of optimization parameters and look at the only modeling parameter.

7.2.3 Data Weight

The remaining parameter α weights the likelihood function, which measures the distance between the input configuration and the present configuration, so the choice of α depends upon one's faith in the data. When α is big, as in Figure 7.10, the optimal configuration did, as expected, bear a close resemblance with the input configuration. Big data term is also visible in the energy plot.

On the other hand, in the Figure 7.11 we see that even with very small parameter α, the initial configuration was not totally forgotten — a locally defined smoothness prior, together with mesh connectivity and a perturbation scheme where vertices move independently, kept the mesh in roughly the same space bounds. I return to this effect a few paragraphs down, where I discuss the performance of the smoothing prior.

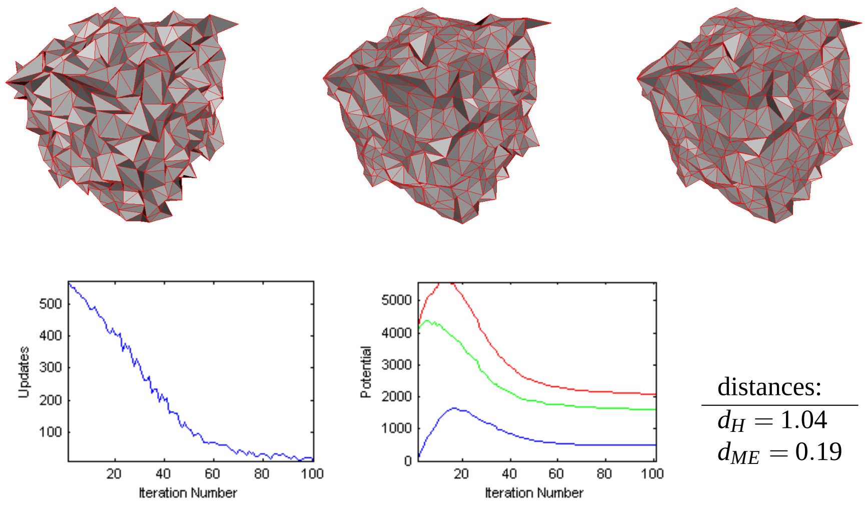
Lastly, Figure 7.12 contains a comparison of the final results from all the tests, where it is possible to verify the correlation between the posterior energy and the distance from the uncorrupted cube. We can see that the configurations resulting in small energy generally also have a small distance measures from the uncorrupted cube. The noise-free cube has the minimal total energy (likelihood is calculated in relation to the noisy cube). That verifies that the model was designed successfully.

We can also see that it is the size of the sampling step, σ_N and σ_T, having the strongest influence on the success of the optimization.

The optimization seems rather robust to the different cooling schemes — the difference between achieved minimal energies are marginal. Even though the slow cooling scheme yield visually best result, the distance measure for that mesh is rather big — the mesh is obviously over smoothed and the rounded edges contribute to high distance measure. This observation raises a question about necessity of finding the global energy minimum. Maybe it is enough to find an energy minimum which is closest to input mesh? Assuming small noise, the local energy minimum might be a good solution of the smoothing algorithm.

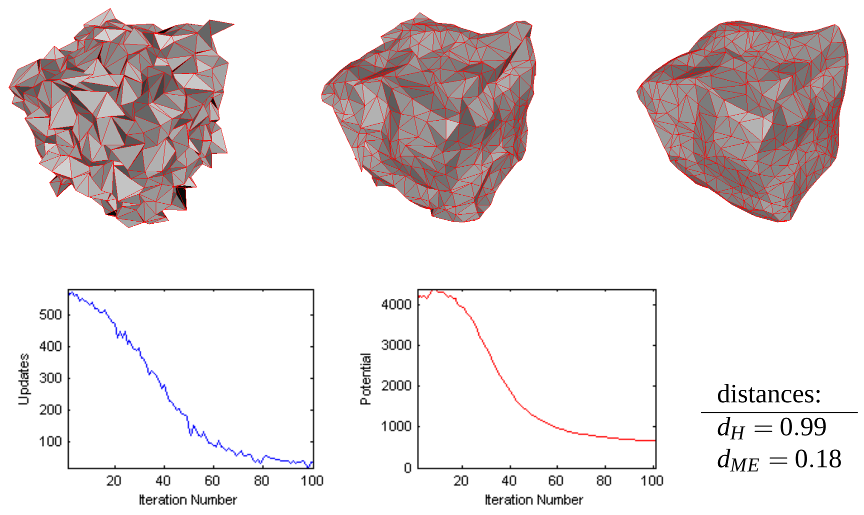
7.3 Local Versus Global Smoothing

The results presented show considerable denoising of the input mesh. But, are the results as expected? Some of the smoothed meshes seem very edgy, with almost



distances:
$d_H = 1.04$
$d_{ME} = 0.19$

Figure 7.10: Big weight for the data term, $\alpha = 20$. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.



distances:
$d_H = 0.99$
$d_{ME} = 0.18$

Figure 7.11: No data term, $\alpha = 0$. *Top:* Iterations 10, 50 and 100. *Bottom:* Updated vertices over time, potentials over time (blue — likelihood, green — smoothness, red — total) and distance to uncorrupted cube.

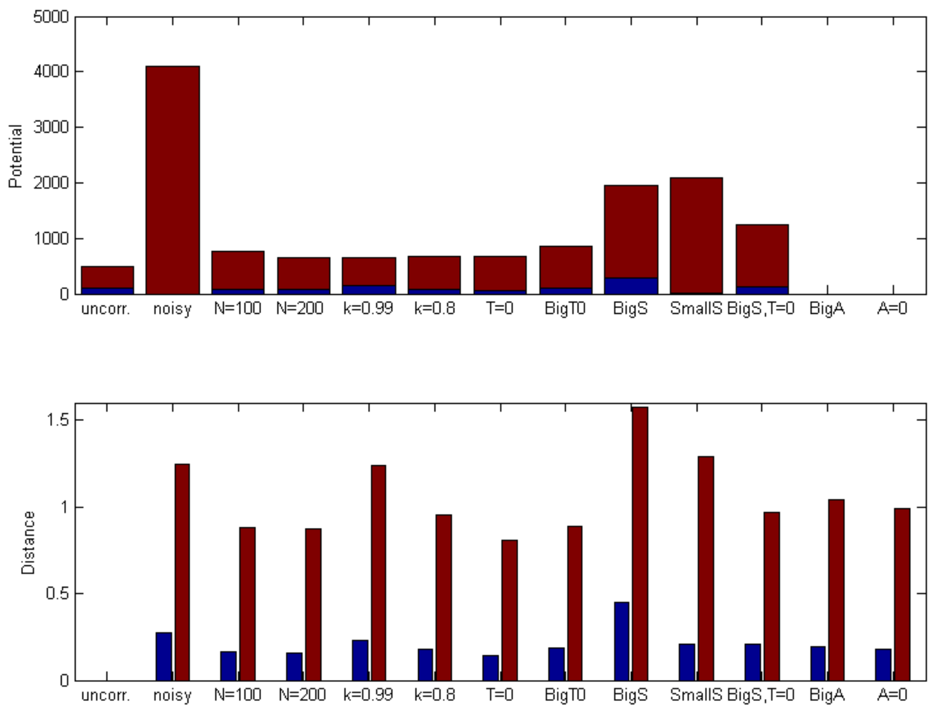


Figure 7.12: Comparison between the final potential and distance to uncorrupted cube for some mesh configurations, including all parameter testing results. *Top:* Potentials, smoothness potential (red) stacked over likelihood potential (blue). *Bottom:* Distances from the uncorrupted cube, Hausdorff distance(red), mean error (blue). The configuration are from left to right: uncorrupted cube, noisy (corrupted) cube, smoothing result after 100 iterations, smoothing result after 200 iterations, slow cooling after 500 iterations, faster cooling scheme, smoothing at zero temperature, high initial temperature, big sampling step, small sampling step, big sampling step and zero temperature, big data weight, zero data weight (potential of last two not included since posterior model changed).

well-defined ridges, even the one *without* the data term. This is not un-expected after the analysis of the curvature-based potential in Section 7.5, but should one not expect more rounding of the ridges and corners? Additionally, all of the meshes preserve their rough shape, even *without* the data term.

If we again consider smoothing a perfect cube without the data term, optimization process should round the corners and also shrink the cube. Assuming low temperature, how would the optimization perform? All the vertices on the sides of the cube are in locally optimal positions. The vertices on the edges (ridges) of the cube are also positioned optimally in regard to their neighborhoods – pulling inward just one of the ridge vertices would increase the energy. Only the 8 vertices in the corners of the cube can decrease the energy by being pulled inward. The smoothing algorithm would make a lot of sweeps just to move the 8 corner vertices. This movement should then propagate down the sides of the cube, leading again to the small number of vertices around the corner waiting to be pulled in. The change propagates slowly, and the mesh will effectively preserve the shape very close to the original shape, even though it is not in the energy minimum (or in the local minimum).

So, our algorithm generally preserves rough shape, making it applicable for mesh denoising where only high frequency noise should be removed.

7.4 Efficiency

As the mesh size grows, efficiency becomes a serious issue. For a not extremely large mesh containing 13000 faces, the smoothing would take close to ten second per iteration. Successfully smoothing large mesh with a good cooling scheme would be very time demanding.

The work of this project was focused in investigating feasibility and performance of MRF mesh-smoothing methods and little effort was put in optimizing the running time of the algorithm. While developing code it was important to leave room for many possible changes and improvements. Additionally, the performance was highly monitored all the time, and monitoring just the energy more than doubled the number of operation per iteration. It is surely possible to implement the MRF smoothing method much more efficiently than done here. However, running time will undoubtedly be an issue with large meshes.

From the performance analysis it is clear that simulated annealing does produce the visually best results, but only with a very slow cooling scheme. A lot of noise is introduced while the temperature is high, and the mesh is not given enough time to recover if it is cooled down too fast. On the other side, cooling at the zero temperature produces reasonable results in much less time. The benefits of simulated annealing are surely present in some cases, but whether it is worth the extra running time is questionable.

Another efficiency issue is the number of updates per iteration. Toward the end of the smoothing, just a very small number of vertices are updated. This surely

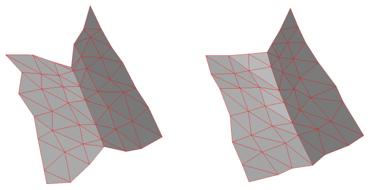


Figure 7.13: The comparison of the smoothness potential, which is proportional to the edge length *(left)* and the smoothnes potential depending only on the dihedral angle *(right)*. Considerable shrinking of the ridge can be seen. Data term weight $\alpha = 0$ for this experiment.

indicates convergence, but it also means that a lot of time is used to evaluate bad moves. As one attempt of helping this problem out, a scheme with adaptive sampling size could be devised. As smoothing progresses and becomes more subtle, the sampling size could decrease to increase the probability of a good move.

More drastic change would involve trying out deterministic optimization methods. The success of smoothing at zero temperature raises hopes that even a simple gradient-based optimization could produce reasonable results. For global optimization graduated-non-convexity or belief propagation could possibly be used.

7.5 Comparison of the Different Smoothness Potentials

A comparison of the different smoothness potentials is presented here. For a quick initial comparison a small model of a ridge has been used first. To see the effect of different priors on smoothing a more realistic model the noisy cube has been used again, this time with $\sigma_{\text{noise}} = 0.2$ average edge length.

On the Figure 7.13 the curvature based potential f is compared with the potential that does not depend on the edge length on testing model. I apply only a few quick sweeps of the smoothing algorithm for this test. As expected, the curvature potential did shrink the ridge. In most meshes, however, it is not possible to shrink just the ridge edges, and the data term does not allow significant mesh shrinking.

Figure 7.14 tests all the potentials $f, f_1, \ldots, f_4$ on the same small testing model. The only potential rounding the ridge is the over-smoothing quadratic potential. Other potentials do not round the ridge, but the original formulation and thresholded version shrink the ridge a bit due to the dependency upon the edge length.

The same five potentials are also compared in the Figure 7.15. For this experiment all the parameters were estimated so that the potentials have comparable starting positions. The initial temperature T_0 has been estimated by the dummy loop, and the date weight α has been calculated from the average initial edge potential.

Each potential was used in two optimization processes: one that used simulated annealing, and one that smoothed with zero temperature. The distance from

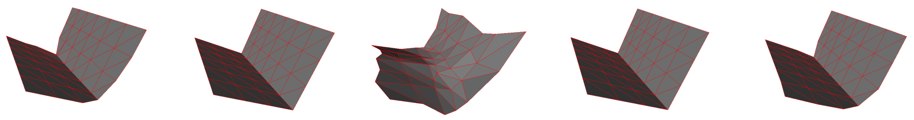


Figure 7.14: Comparison of the five smoothness potentials. *Left to right:* The initial formulation based on absolute curvature, potential without the dependence on the edge length, quadratic potential, square root potential and thresholded potential. Data term weight $\alpha = 0$ for this experiment.

uncorrupted cube was calculated for each final configuration. The likelihood energy (relative to α) is also marked to help comparison.

It is possible to draw a number of conclusions from the figure. Firstly, all potentials apparat from quadratic give similar results. The quadratic potential, unlike the others, rounds the sharp surface edges. This again confirms the conclusion about performances of developed priors.

Secondly, the likelihood energy is always higher when simulated annealing was used. Simulated annealing therefore makes it possible to for the system to 'forget' initial configuration and search for an optimal solution further from it. However, in most cases smoothing with zero temperature produced the result that was closer to the noise-free mesh. The best result was produced by using thresholded potential and it is also the configuration that has relatively small likelihood energy. We can again conclude that the local optimization method might suffice in some applications.

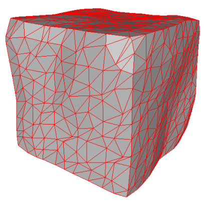
Lastly, it is hard to make a conclusion about the influence of the edge length in the formulation of the prior. When I inspected the resulting meshes it looked as if the formulation without the dependence on the edge length produces more 'badly shaped' triangles (i.e., triangles with either one very small angle or one very large angle) but more test are needed for a definite answer.

7.6 Conclusion

In this chapter I successfully used the MRF smoothness prior to smooth the noisy mesh. A typical smoothing was presented first, and general guidelines on the choice of the optimization parameters were given.

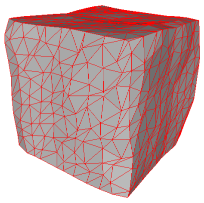
A large experiment is presented next, where I test the influence of each modeling and optimization parameter. The resulting meshes and energy plots verify that the simulated annealing optimization behaves as expected. All results show significant denoising of the input mesh, but the visually smoothest result is obtained by applying a very slow cooling scheme.

The good result of the smoothing with zero temperature raised the question



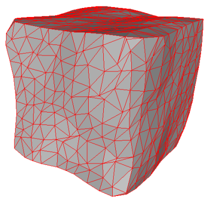
likelihood:
$U_L = 45.2$

distances:
$d_H = 0.58$
$d_{ME} = 0.092$



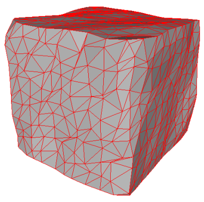
likelihood:
$U_L = 36.5$

distances:
$d_H = 0.64$
$d_{ME} = 0.077$



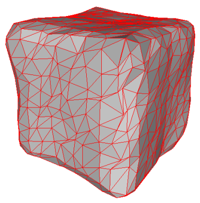
likelihood:
$U_L = 46.1$

distances:
$d_H = 0.85$
$d_{ME} = 0.099$



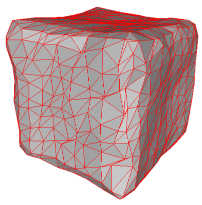
likelihood:
$U_L = 36.6$

distances:
$d_H = 0.55$
$d_{ME} = 0.079$



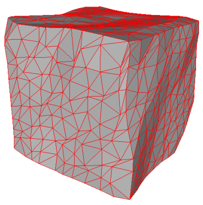
likelihood:
$U_L = 51.9$

distances:
$d_H = 0.77$
$d_{ME} = 0.148$



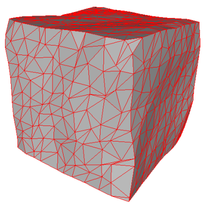
likelihood:
$U_L = 48.1$

distances:
$d_H = 0.65$
$d_{ME} = 0.136$



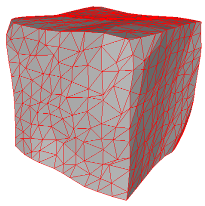
likelihood:
$U_L = 44.8$

distances:
$d_H = 0.84$
$d_{ME} = 0.092$



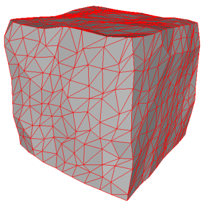
likelihood:
$U_L = 35.6$

distances:
$d_H = 0.56$
$d_{ME} = 0.075$



likelihood:
$U_L = 43.7$

distances:
$d_H = 0.47$
$d_{ME} = 0.088$



likelihood:
$U_L = 34.8$

distances:
$d_H = 0.52$
$d_{ME} = 0.071$

Figure 7.15: Another comparison of the five smoothness potentials. *Top to bottom:* The initial formulation based on absolute curvature, potential without the dependence on the edge length, quadratic potential, square root potential and thresholded potential. *Left:* Configuration after 300 loops using simulated annealing. *Right:* After 200 loops of zero-temperature smoothing. Data weight and initial temperature are estimated using the dummy loop, annealing constant $k = 0.9$, sampling step $\sigma_N = 2\sigma_T = 0.1$.

about the possibility of using a gradient-based method for optimization. It has been demonstrated that using zero temperature often yields a smooth result, which is also very close to the underlying noise-free surface. It is certainly worth examining gradient-based optimization, since the efficiency of the simulated annealing becomes a serious issue for large meshes.

As for simulated annealing, it has been demonstrated that the size of the sampling step has a strong influence on the optimization result. One of the important improvements of the algorithm would be devising the scheme where the step size becomes smaller and smaller. The criterion for determining the size of the sampling step could be a percentage of the updated vertices. This might be a solution to a low-frequency oscillations that often do not get smoothed.

Finally, another large experiment for comparing the performances of different smoothness potentials was presented. This experiment shows that quadratic potential tends to round the ridges of the surface, while the other potentials preserve the sharp ridges of the surface. An important question in the remainder of the thesis is whether an explicit edge labeling can improve feature-preserving mesh smoothing, or is enough control obtained by a good choice of the smoothness potential.

Chapter 8

Feature Detection

The initial goal of this thesis is to develop a prior for piecewise smooth surfaces. Two terms are combined in the piecewise-smoothness prior: the smoothness prior already described in previous chapters, and the edge labeling process to model the discontinuities in the smoothness of the surface.

This chapter presents a possible formulation of the edge process for detecting the ridges on the surface of the mesh. The material presented is two fold — on the one hand we have the original purpose of using the edge process in combination with the vertex process; on the other hand the edge process can be considered an independent ridge-detection process.

The aim is to label each mesh edge as either a ridge edge or a non-ridge edge (or maybe something in between) based on two terms: the sharpness (dihedral angle) of the edge and the neighborhood support of the other edges. Edge labels will later be used in a mesh-smoothing process to prevent it from over-smoothing across the feature edges.

Finding an appropriate formulation for the edge labeling was not straightforward. On the one hand, the used neighborhood was not big enough to distinguish between ridge or non-ridge edges. On the other hand, the sharpness of an edge is not independent of other edges — the edges are constrained by mesh connectivity.

In an attempt of finding the best (and still simple) edge labeling, a few different formulations have been developed. This chapter brings the overview of the developed formulations and some of the issues regarding the edge labeling. The experimental results and discussion on using edge labeling as an independent process follow in the Chapter 9. The results of using edge labeling in combination with vertex process can be found in Chapter 11.

8.1 What is a Feature Edge?

A ridge on a piecewise smooth surface is a line along which two smooth surfaces meet.[1] We can say that the ridge edge is a discontinuity in the smoothness of the surface (which is why we will later use ridge edges to introduce discontinuities in the smoothing process).

As stated above, the idea of our edge labeling is to use two terms. First term is the sharpness of the edge, which is again the dihedral angle of the incident faces used in the vertex process. The larger the dihedral angle, the sharper the mesh edge, and the more probable it is that the edge lies along the surface ridge.

The second term is neighborhood support. By neighborhood support we mean a presence of other ridge edges along the same ridge line. To utilize the neighborhood support a measure of parallelism between the edges needs to be devised, to model the concept of 'lying along' the ridge.

Still, this is generally not sufficient information to distinguish between ridge and non-ridge edges. Mesh edges can have large dihedral angles on the smooth parts of the surface, where the sampling density is relatively low compared to the curvature of the surface, and the neighborhood support can still be present, as demonstrated in Figure 8.1. The fandisk model shown there is noise-free, but it is still imposable to distinguish between the not very sharp feature ridge and the non-feature edges on the curved part of the surface, when only sharpness and neighborhood support are used. We would need the information about curvature of the surfaces on both sides of the edge to decide whether we deal with the feature edge or with the curved but smooth surface.

Even more, we have to acknowledge that the discontinuities of the underlying surface are ambiguous. The curved part of the fandisk model *might* actually be piecewise planar, composed of many elongated rectangles. So, on the bottom line, it is impossible to label edges with certainty, but the large neighborhood might provide a better starting point for finding a more probable labeling.

8.2 Edge Labeling

In edge labeling process we want to assign a label to each mesh edge. The label should signal the presence of surface feature edge along the mesh edge.

There are few modeling options worth considering:

Continuous or discrete labeling. Intuitively, a choice would probably fall on discrete labeling, as an edge either is or is not a ridge edge. On the other hand, knowing that the labels will be used as weights in the vertex process the continuous labeling also becomes interesting, as it might be desirable to leave

[1]A small terminology clash is encountered here. When working with surface meshes, word *edge* is reserved for describing a connectivity relation between vertices, but talking about surfaces, an edge would be what I am here calling *ridge*.

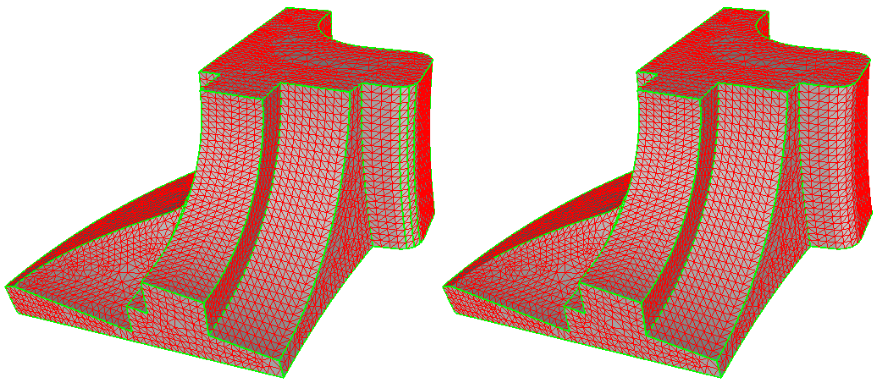


Figure 8.1: Feature and non-feature edges, sharpness and neighborhood support. Edges marked green are sharper than 160° *(left)* and 156° *(right)*. The feature edge dividing the two smooth surfaces on the left of the fandisk model is not sharper then the non-feature vertical edges on the curved part of the fandisk. Neighborhood support of adjacent sharp edges would not help distinguish between feature and non-feature in this case, since the non-feature edge has an ideal support. To distinguish between feature and non-feature in this case, one should look at the larger neighborhood.

some room for uncertainty. A label can in that context be seen as probability of the ridge-edge.

Deterministic or stochastic labeling. In deterministic case the labels are calculated from the sharpness (and neighborhood) information for the edge. In stochastic case a random field is associated with each edge with the rest of the MRF framework built upon it.

In search of the best labeling method all of the combinations were tried and tested. The overview of the approaches follows, and the the results are presented in the next chapter.

The additional question that had to be answered was whether the neighborhood support contributes to the performance of the edge detection or could the same effect be obtained by utilizing only the information about the edge sharpness. This will also be re-visited later.

A short note on notation: An edge e_i is assigned label ε_i, and indices are used *only* when there is more than one edge involved. Assigning a label to each of the edges from the set of edges E gives a label configuration $\mathbf{E}$.

In case of discrete labeling, the set of labels L is

$$L = \{0,1\} ,$$

while in the case of continuous labeling the labels can take any value in the interval between 0 and 1

$$L = [0,1] .$$

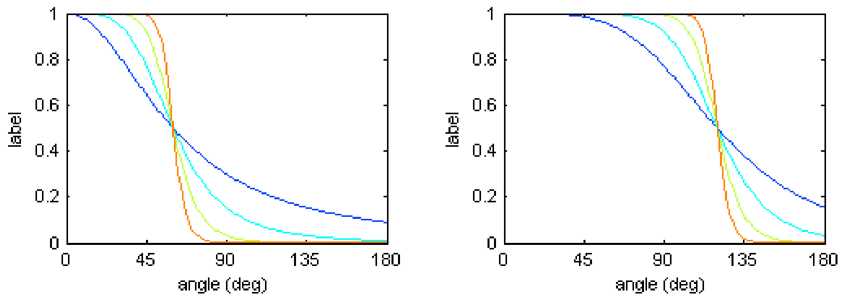


Figure 8.2: The cut-off function used for labeling the mesh-edges depending on the dihedral angle ϕ. *Left:* Cut-off angle $\phi_0 = 60°$. *Right:* Cut-off angle $\phi_0 = 120°$. The slopes at the cut-off angle in both cases are $k = 0.5$ for blue, $k = 1$ for cyan, $k = 2$ for yellow and $k = 4$ for red color.

A bit counter-intuitive, the label 0 signals the occurrence of a ridge along the edge, while label 1 signals the non-ridge edges. This choice is motivated by the desire to use labels as weights for the vertex process, and the need to have 0 weight where smoothing should stop. There are quite a few $(1-\varepsilon)$ expressions in the coming text due to this choice.

8.3 Ridge Sharpness

Looking only at ridge sharpness is the simplest way of assigning labels to dihedral edges. In discrete deterministic case this is obtained by putting a threshold to the dihedral angle. As mentioned in Section 6.4.2 putting a threshold to dihedral angle is equivalent to implicit edge labeling — each occurrence of the dihedral angle larger than the threshold is equivalent to saying that the edge is sharp. As demonstrated in the Figure 6.5 thresholded smoothness potential is in itself feature-preserving.

For a continuous and deterministic case we want a smooth transition between the labels 1 and 0. A cut-off function is needed, where both the cut-off angle ϕ_0 and the steepness of the transition can be controlled.

The desired properties were found in the function

$$h(\phi) = \frac{1}{1 + (\phi_e/\phi_0)^s}$$

illustrated on Figure 8.2. Each edge is assigned the label $\varepsilon = h(\phi_e)$.

When $\phi_e = \phi_0$ the edge will be assigned label $\varepsilon = h(\phi_0) = 0.5$. The parameter s controls the sharpness of the transition and the slope at the cut-off angle is

$$k = g'(\phi_0) = -\frac{s}{4\phi_0} \; .$$

So, θ_0 is an estimation of the sharpness of the ridges that we want to detect, while k (or s) serves to model our confidence in θ_0.

For the stochastic edge labeling, looking only at edge sharpness means that we assign non-zero potential only to single-edge cliques. In a discrete case this is again only a small step from implicit edge labeling by thresholding the smoothness potential. In the related field of MRF image reconstruction, the similar step is the introduction of line process [Geman and Geman, 1984] to signal the occurrences of edges.

To define the labeling that tends to assigns label 0 to edges sharper than a given threshold ϕ_0, and label 1 to less sharp edges, each edge is assigned the potential

$$U_{E1}(e) = (\phi_0 - \phi_e)(1-\varepsilon) \ ,$$

where ε is the label from the set $L_E = \{0,1\}$ and ϕ_e is the dihedral angle of the edge e. For non-sharp edges, which have a dihedral angle under the threshold, this expression is minimized by assigning the label 1, and for larger dihedral angles it is energy-wise better to insert the ridge-edge.

To complete the coverage of possible sharpness formulations let me mention also the continuous stochastic case. One possibility is to take the discrete case and allow the labels to be continuous, for the sake of adding a certain lever of uncertainty to the edge labeling.

In the other formulation of the continues stochastic sharpness potential we add a level of randomness to the deterministic labeling based on cut-off function. This sharpness potential penalizes the difference between the assigned label and the label obtained by the cut-off function

$$U_{E1}(e) = (\varepsilon - h(e))^2 \ .$$

8.4 Neighborhood Support

The neighborhood support stands for the contribution of the edges lying on the same ridge line. We first define the edge neighborhood system, which is initially used in MRF framework, but can also be used for deterministic edge labeling.

Mesh connectivity is again employed for neighborhood relation: two different edges e_1 and e_2 are neighbors if they share a vertex. This is obviously well defined neighborhood system. The neighborhood of an edge e consists of all edges that are incident to one of its two vertices, as shown in the Figure 8.3.

For this neighborhood system we have many types of cliques: *a)* a single edge, *b)* a different number of edges incident on the same vertex and *c)* a triangle. The contribution of the single-edge cliques is covered by the edge sharpness part. The neighorhood support, which is examined here, deals with the two-edge cliques.

A non-boundary edge in a regular mesh will have 10 neighboring edges (5 at each side), an it is clear that only parallel or close-to-parallel neighboring edges should act supportive. As the measure of how parallel the two edges e_1 and e_2

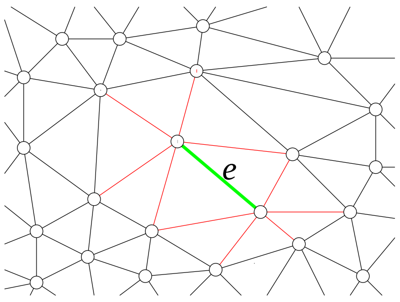


Figure 8.3: Edges marked red are the neighborhood of the edge e. A non-boundary edge in a regular mesh will have 10 neighbors.

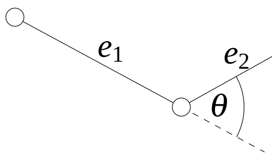


Figure 8.4: A 2-clique of edges. The potential of a 2-clique depends on labels and the angle of the bend.

are we use the angle θ_{e_1,e_2} between the directions that the neighboring edges take on a potential ridge line, see Figure 8.4. When $\theta_{e_1,e_2} = 0$, the edges e_1 and e_2 are parallel and this configuration yields the maximal support.

As already mentioned and demonstrated in Figure 8.1 considering only edge sharpness and neighborhood support is not sufficient for classification of mesh edges into ridge and non-ridge edges, even in the case of noise-free meshes. The re-appearing question during the work on this thesis relates to the significance of the neighborhood support in edge labeling.

The initial idea of employing the neighborhood support was to assist linking of mesh edges into ridge lines. If, for example, in a chain of three mesh edges the outer two edges are labeled as ridge edges, we would prefer the middle edge to be labeled as ridge too, even if it's dihedral angle maybe does not reach the sharpness threshold. The change of sharpness along the edge could be contributed to noise, and neighborhood support is supposed to help detect the ridges across the noise.

One thing is overlooked in the above reasoning — the dihedral angles of the neighboring edges are not totally unrelated. The constraint lies in a fact that vertices of the mesh form a surface. So, in the example of three linked edges where the middle one gets a wrong label due to the noise, a little bit of denoising might be sufficient for the recovery of the middle edge to? In other words, maybe the mesh connectivity automatically preforms some form of the neighborhood support, making its explicit formulation unnecessary?

Before trying to formulate the neighborhood support and solving the dilemma by experiment, we can analyze the question by focusing on a simple example presented on Figure 8.5 where we add more and more noise to the cube model and plot the histograms of dihedral angels ϕ_e and support angles θ_{e_1,e_2}.

In a case of a noise-free cube the edge sharpness is enough for a perfect edge classification. Even with the presence of a small noise the edge sharpness will suffice. At some point the nose will spread the distribution of dihedral angles and the correct classification based only on dihedral angle is not possible any more. To examine the possibility of using the neighborhood support we look at the distribution of the support angles θ. With the increasing noise the distribution of support angles has experienced the equivalent (if not larger) spreading and mixing, so using neighborhood support is likely to find features in noise.

Of course, the *combination* of edge sharpness and neighborhood support can still prove to be useful, but it clearly requires more analysis and experiments.

8.4.1 Formulations of the Neighborhood Support

Initially, we again consider only the discrete labeling. The idea is to make non-ridge edges neutral in all configurations, and to assign potentials only to interaction between two ridge edges. For the small angles θ_{e_1,e_2} the potential of the two-clique should be negative, thus supporting two neighboring and close-to-parallel ridge edges; and for the angles between 90° and 180° the potential should be positive, thus preventing the sharp corners in the ridge line. This leads to the following formulation

$$U_{E2}(e_1,e_2) = (\theta_{e_1,e_2} - \theta_0)\,(1-\varepsilon_1)\,(1-\varepsilon_2)$$

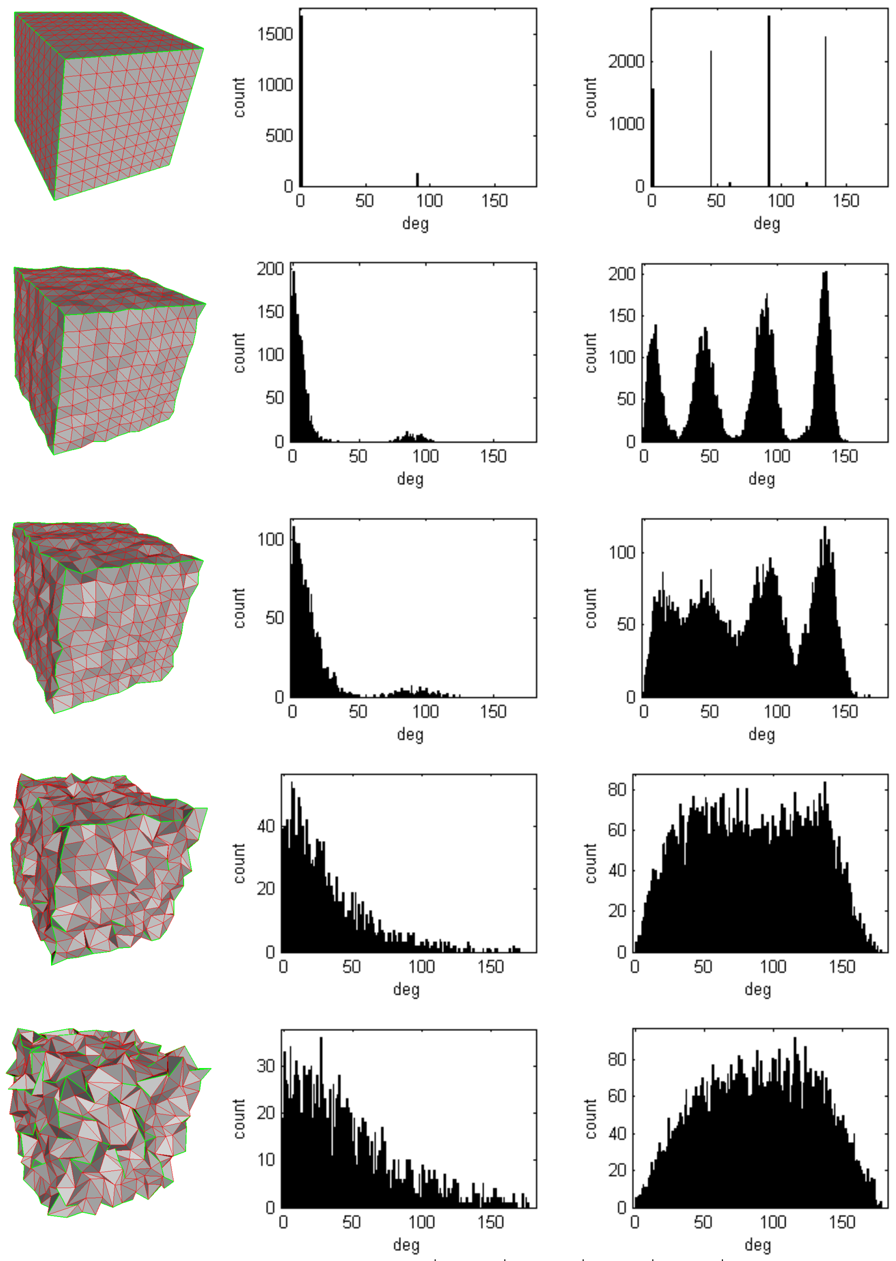
where θ_{e_1,e_2} is the angle between the edges e_1 and e_2, ε_1 and ε_2 are the labels assigned to edges e_1 and e_2, and θ_0 is the support threshold, which is typically set to 90°.

Figure 8.6 shows some of the possible two-edge clique configurations and their potentials. Just as desired, only the clique containing two ridge edges has non-zero potential, negative for support and positive for un-support.

In the same figure we can also see the potentials in case where the labels were allowed to be continuous. The desired properties are maintained — ridge edges interact stronger than non ridge edges; edges that point in the same direction support each other; orthogonal edges are relatively neutral; and sharp edges meeting at a small angle show negative support to each other. It is therefore worth trying to use this linear potential for continuous labels, too.

In the above formulation it is the presence of ridge edges that is supportive and this support sums up to influence the edge labeling. This potential does not use the intuition we have about the ridge line, as a chain of ridge edges. In the next formulation an attempt is made in modeling the feature line for what it is: a sequence of close-to-parallel edges that have the same sharpness. The two-clique potential is formulated as

$$U_{E2}(e_1,e_2) = (\varepsilon_1 - \varepsilon_2)^2\ \max(0, \theta_0 - \theta_{e_1,e_2})\ .$$



σ_{noise} (average edge length)	0	0.05	0.1	0.2	0.3
average dihedral angle ϕ_e	6°	11°	16°	32°	45°
average support angle θ_{e_1,e_2}	75°	77°	79°	84°	90°

Figure 8.5: The distributions of dihedral angles and support angles for the different levels of noise. *Top to bottom:* Noise level $\sigma_{\text{noise}} = 0, 0.05, 0.1, 0.2, 0.3$ average edge length. *Left:* Model with the edges sharper than 75° marked green. *Center:* Distribution of the dihedral angles ϕ_e. *Right:* Distribution of support angles θ_{e_1,e_2}.

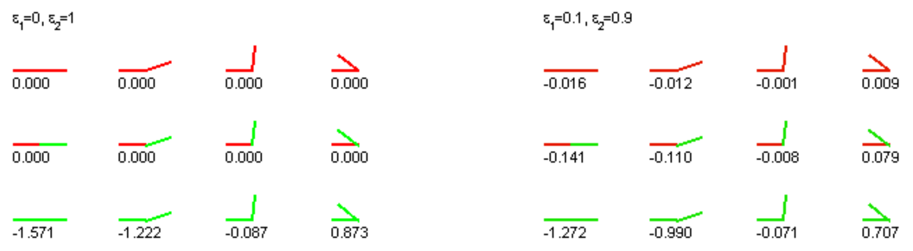


Figure 8.6: Potentials for some configuration of 2-edge cliques. Red color used for non-ridge edge, green color used for ridge edge. *Left:* Discrete edge labeling with the labels from $L_E = \{0,1\}$. *Right:* Continuous labeling with labels from $L_E = [0,1]$.

For those edges that are up to θ_0 parallel we penalize the difference in the labels. In this case we do not look at the configuration of sharp edges that un-support each other, but that can be easily included.

8.5 Stochastic Edge Labeling

Let's first look at the discrete case and the linear potentials. The total potential of an edge labeling configuration is

$$U_E(\mathbf{E}) = \sum_{e \in E} (\phi_0 - \phi_e)(1-\varepsilon) + \beta \sum_{(e_1,e_2) \in \mathscr{C}_2} (\theta_{e_1,e_2} - \theta_0)(1-\varepsilon_1)(1-\varepsilon_2) ,$$

where β is the weighting constant for the support energy. We will refer to the first part as the sharpness energy (or edginess energy) and the second part as the support energy. For the configuration of all edges being non-ridge, the total edge labeling energy is 0, and it can be improved by inserting edges at the right places.

If we operate on a stationary mesh, the angles ϕ_e and θ_{e_1,e_2} are fixed and the above reduces to an auto-logistic MRF model [Li, 2001] where the energy takes the form

$$U(f) = \sum_{\{i\} \in \mathscr{C}_1} \alpha_i f_i + \sum_{\{i,i'\} \in \mathscr{C}_2} \beta_{i,i'} f_i f_{i'} .$$

The constants $\beta_{i,i'}$ are interaction coefficients reflecting the pair-site interaction.

The energy of an edge labeling leads directly to the joint probability of the labeling configuration, which is

$$P(\mathbf{E}) = \frac{1}{Z} \exp\left(-\frac{1}{T} U_E(\mathbf{E})\right) ,$$

where T is the temperature and Z is the normalization constant.

To obtain conditional probability I looked at the part of the energy that changes when one alters a single label, which is

$$U_E(\varepsilon) = \left[(\phi_0 - \phi_e) + \beta \sum_{e' \in \mathcal{N}_e} \left(\theta_{e,e'} - \theta_0\right)(1-\varepsilon') \right](1-\varepsilon) \; . \tag{8.1}$$

The optimal labeling (given the neighborhood) depends only on the sign of the expression in the square brackets.

The conditional probability is given as

$$P(\varepsilon|\mathcal{N}_e) = \frac{1}{Z} \exp\left(-\frac{1}{T} U_E(\varepsilon)\right) \; .$$

Simply allowing the labels to be continuous does not change the optimal configuration and it has the positive effect on optimization. Despite it being oversimplistic approach, this method produces good results.

The stochastic labeling based on the alternative sharpness and support potentials are formulated in similar way.

8.5.1 Optimization

At high temperatures Metropolis sampling always prefers the new label. In the case of discrete binary labeling a new label is the *other* label, and this results in a lot of flickering on the high temperatures. As the temperature decreases flickering ceases and algorithm starts converging.

The Gibbs sampler is maybe more suited for binary labeling since it evaluates the probabilities of all the labels, in this case $P(0)$ and $P(1)$, at each step of the algorithm. A label is set to 0 with probability

$$p = \frac{P(0)}{P(0) - P(1)} \; .$$

The temperature plays similar role in Gibbs sampling, as it does in Metropolis sampling. In the limit of $T \to \infty$ the probability p is 1/2, and in the limit $T \to 0$ the probability is either 0 or 1, depending on the sign of the energy distance.

Allowing the continuous labels is another way of dealing with flickering.

8.6 Deterministic Edge Labeling

Deterministic edge labeling was mostly used only for the pure sharpness labeling according to the cut-off function, as described in Section 8.3.

It is also possible to define the deterministic labeling, which includes neighborhood support. To formulate this type of edge labeling we notice that in Equation (8.1) only a sign of the expression in the square brackets is needed to determine the label of the edge. This expression can be used to assign the continuous label to an the edge.

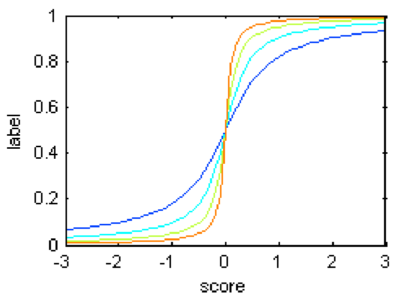


Figure 8.7: The cut-off function used for labeling the mesh-edges based on sharpness and support measure. The cut-off value is 0, the slopes at the cut-off value are $k = 0.5$ for blue, $k = 1$ for cyan, $k = 2$ for yellow and $k = 4$ for red color. Compare with Figure 8.2

The labeling function L is formulated as

$$h(e) = g\left[(\phi_0 - \phi_e) + \beta \sum_{e' \in \mathcal{N}_e} \left(\theta_{e,e'} - \theta_0\right)(1 - \varepsilon')\right]$$

where g has to be monotonically increasing function that has a codomain in the interval $[0,1]$ and where $g(0) = 0.5$. Function g serves as a cut-off function, so it is desirable that its steepness can also be controlled.

Function
$$f(x) = \frac{1}{\pi}\arctan(\gamma x) + \frac{1}{2}$$
has desired properties. The constant γ can be used to control steepness of the function and thus the level of uncertainty we want the labeling to exhibit. For the large γ labels will be concentrated close to the asymptotical values of 0 and 1. For the smaller γ labels will be more evenly distributed.

The direct method can be used with some success in combination with vertex smoothing process, but it was not useful on its own, as a ridge detecting function. Due to its deterministic nature and the neighborhood support it would often flicker between two configurations.

8.7 Summary

In this chapter I presented some methods for detecting ridges on the surface represented as triangle mesh. The developed methods can be categorized as continuous or discrete, deterministic or stochastic, with or without neighborhood support.

One formulation used extensively in the remainder of the text is continues, stochastic and with neighborhood support. In this formulation the edge process is modeled as MRF on mesh edges, with two kinds of clique potentials. The sharpness potential of a single edge is linear to the dihedral angle of the edge. The support potential of two edges is linear to the angle between the edges. This formulation is often referred to as the linear edge labeling.

Another often used formulation is based on the cut-off function of the dihedral angle. This is continues edge labeling without neighborhood support, and it has the deterministic and the stochastic version.

The important question of whether neighborhood support can contribute to edge labeling is raised in this chapter. It is in an attempt of finding the answer to this question that so many different formulations have been developed.

The success of the edge labeling as an independent ridge-detecting process will be investigated in the next chapter. After coupling the edge process with the vertex process, the success of different edge labeling methods is investigated again in Chapter 11.

Chapter 9

Feature Detection, Results and Discussion

The results presented here cover only the edge labeling with fixed vertex positions. Even though this is not the primary goal of edge labeling process it can be useful to see, especially in order to understand the effects of the neighborhood support.

In all figures, the edges shown green are those with the label smaller than 0.5, and it is those edges that are referred to as *detected* edges.

9.1 Stochastic Edge Labeling

The Figure 9.1 shows a discrete edge labeling, using the linear potentials and neighborhood support. Gibbs sampling and simulated annealing were used for optimization. The effects of the high temperature are clearly visible. Final configuration has successfully identified some of the cube ridges.

The discrete labeling would often result in a lot of flickering (i.e., alternating between two configurations). This could easily be addressed by allowing the labels to be continues. Labels would still tend to be concentrated around 0 and 1, but with much nicer optimization.

Instead of applying simulated annealing, the fastest results were found by initially labeling edges using the threshold on the dihedral angle, and in a few quick sweeps of the algorithm letting the neighborhood support connect the ridge edges. This approach is presented in Figure 9.2. The energy plot shows negative energy already for the initial configuration. The subsequent decrease in energy is contributed exclusively to the decrease in the support energy, with the sharpness potential slowly growing as the less sharp edges get labeled as ridges.

Due to the noise, initially are just some of the edges on the models true ridges labeled as ridge edges. After a few sweeps of the algorithm, ridge edges are connected in longer structures. Surface ridges are rather well detected, but some spurious structures occur when a ridge gets detected in a noise. One of the horizontal

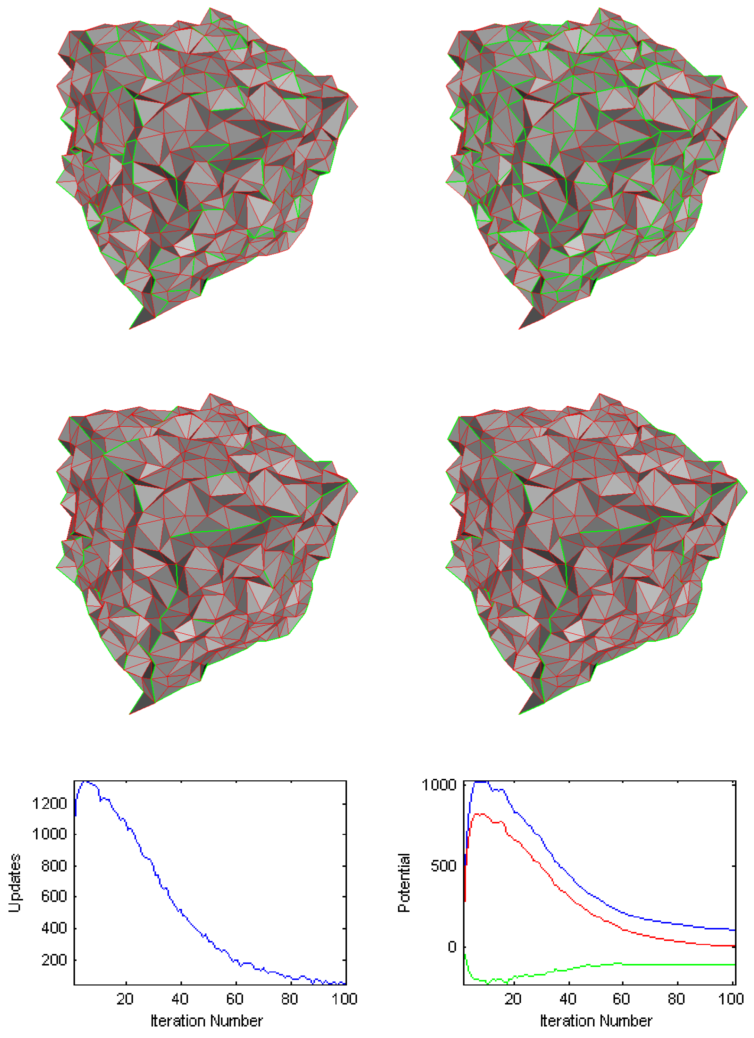


Figure 9.1: Stochastic discrete edge labeling of a noisy cube using a Gibbs sampler and simulated annealing. Parameters used: neighborhood support weight $\beta = 1$, sharpness threshold $\phi_0 = 120°$, neighborhood support threshold $\theta_0 = 90°$, initial temperature $T^0 = 1$, annealing constant $k = 9.5$. *Top and middle:* Iterations 3, 10, 50 and 100. *Bottom left:* The number of updated edges over time. *Bottom right:* Potentials over time: blue — sharpness potential, green — neighborhood support potential, red — total potential.

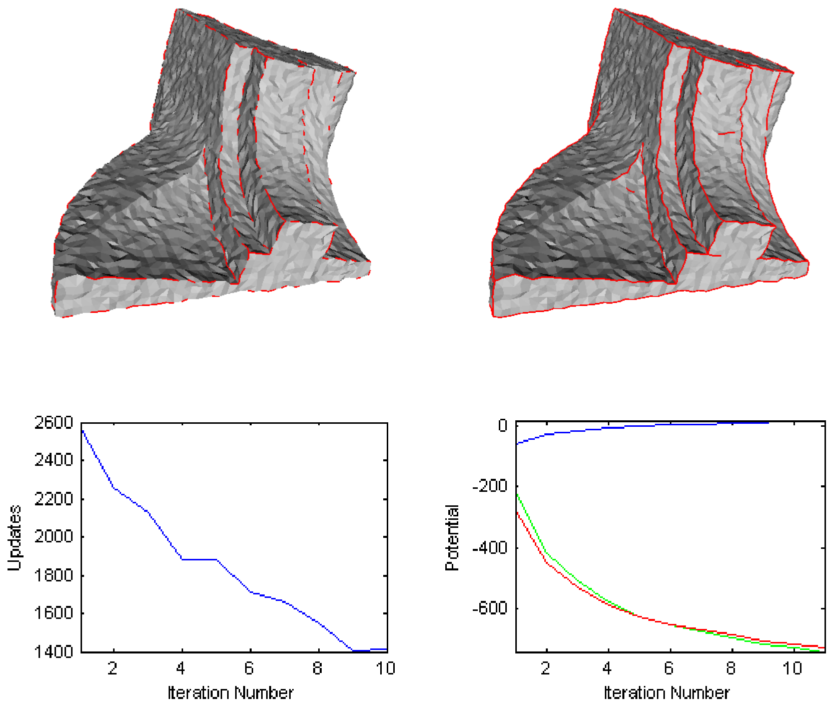


Figure 9.2: Fast stochastic edge labeling. Fandisk model corrupted with Gaussian noise, $\sigma_{NOISE} = 0.1$ average edge length. Parameters used: neighborhood weight $\beta = 1$, sharpness threshold $\phi_0 = 90^\circ$, neighborhod support threshold $\theta = 90^\circ$. *Top:* Initial labeling based only on edge sharpness and labeling after 10 iterations. *Bottom left:* The number of updated edges over time. *Bottom right:* Potentials over time: blue — sharpness potential, red — neighborhood support potential, red — total potential.

ridges framing the front face of the model gets continued across the planar surface due to the neighborhood support.

9.1.1 Thresholds

Sharpness threshold ϕ_0 was originally used to model the sharpness of the ridge edges. With support threshold θ_0 we can alter the neighborhood support and decide, for example, that only edges with the difference in directions less than 45° support each other. However, once we add the sharpness and support potential together those thresholds start to interact. Choosing a small θ_0 will cause a lot of unsupport between edges and the total number of edges will decrease. This means that even some of the edges sharper than ϕ_0 will not get labeled as ridge edges. Increasing θ_0 will have the opposite effect.

The best strategy is therefore to fix θ_0 at 90° and use ϕ_0 to regulate the number

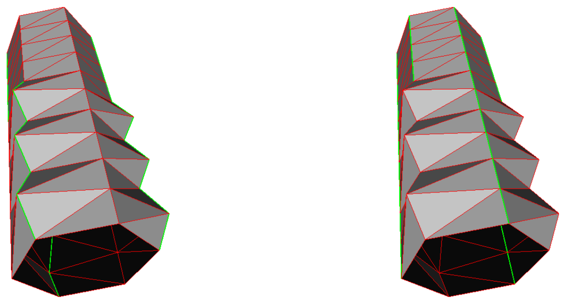


Figure 9.3: The demonstration of the effect of using the neighborhood support. *Left:* Thresholding the dihedral angle at 60°. *Right:* Result of the stochastic edge labeling using linear potentials for neighborhood support. Parameters used: sharpness threshold $\phi_0 = 50°$, neighborhood support threshold $\theta = 60°$, neighborhood support weight $\beta = 1$.

of detected edges. At this setting the orthogonal edges are neutral to each other, but because of an abundance of very supportive parallel (or near-parallel) edges and no equivalently unsupportive edges, there will generally be more detected edges than what one could desire when choosing ϕ_0.

9.1.2 Neighborhood Support Question

As already discussed, and important question is whether neighborhood support contributes to the correct labeling of edges. Again, the most interesting is in the case where edge labels are used as the weights for smoothing, but the fixed-shape situation can be examined first.

In Figure 9.3 a small testing model is used to demonstrate that the neighborhood support works as planned. The model is constructed so that the edges of the straight ridge are less sharp than the edges of the zigzag ridge. Due to neighborhood support, which is stronger for the straight ridge, it is the less sharp edges that get labeled as ridges.

Next we want to investigate whether using neighborhood support contributes to good edge labeling. In Figure 9.4 the edge labeling based exclusively on the dihedral angle is compared with the edge labeling that incorporates the neighborhood support. We can see that both support based labelings successfully found the upper cube ridge and the parts of the front ridge, without founding many spurious ridges. To find the upper edge sharpness-only labeling should put the threshold down to 60°, and at that point a lot of noise gets labeled as ridges, too.

The conclusion is that the neighborhood support *does* contribute to the correct labeling in the case of the fixed mesh. How significant that contribution is, and does it lead to better smoothing when edge labeling is coupled with vertex process

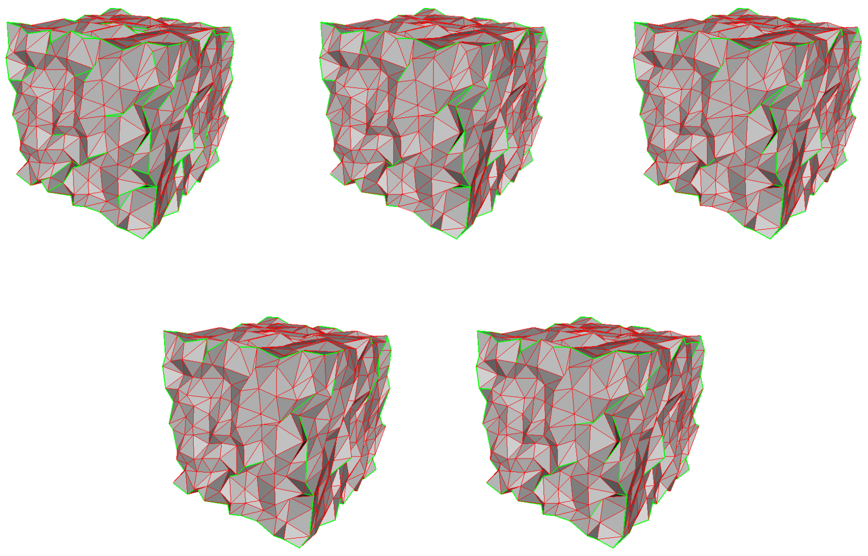


Figure 9.4: Comparison of edge labeling with and without the neighborhood support. *Top:* Edges thresholded using $\theta_0 = 60^\circ$, $\theta_0 = 70^\circ$, $\theta_0 = 80^\circ$ (left to right). *Bottom:* Two labelings that used linear neighborhood support. Parameters used: sharpness threshold $\phi_0 = 90^\circ$, neighborhood support threshold $\theta = 90^\circ$, neighborhood support weight $\beta = 0.8$ (left); sharpness threshold $\phi_0 = 80^\circ$, neighborhood support threshold $\theta = 90^\circ$, neighborhood support weight $\beta = 0.5$ (right).

is still to be answered.

9.2 Deterministic Edge Labeling

There is not so much to investigate about the deterministic edge labeling on the fixed meshes. In case of the sharpness-only function, the deterministic edge labeling is basically identical to putting threshold on the dihedral angle. Using constant labels and different cut-off functions can result in labels concentrated around 0 and 1, or in more evenly distributed labels. However, when displaying (and classifying) the edges the threshold of 0.5 is always used, resulting always in same edge classification.

The case where edge labeling incorporates neighborhood support was hard to analyze in a fixed mesh setting. Due to the neighborhood support the labeling would often flicker between two (often quite different) configurations.

9.3 Conclusion

In this section I presented a few results of using an edge labeling as an independent process for detecting the ridges on the surface representing as a mesh.

A couple of points is demonstrated here. First, it is shown that MRF framework can be used for the problem of the ridge detection. Secondly, it is demonstrated that the neighboring support does work as planned and that it contributes positively to the edge labeling — the results obtained by using neighborhood support were better than the results when using just the threshold on the edge sharpness. Still, the truly big question, which will be addressed in the upcoming chapters is: Does neighborhood support help when we combine the edge and the vertex process?

Chapter 10

Feature-Preserving Mesh Smoothing

The two processes, vertex process based on the smoothness prior and the edge process for labeling ridge edges, are in this chapter combined into a coupled MRF, which can be used for feature-preserving mesh-smoothing.

Coupling the vertex and the edge process is a straightforward step. The only adjustment needing to be done is to ensure that the vertex process does not smooth over the sharp edges. However, considering that we have few different smoothness potentials, a number of different edge labeling schemes, an open question about the significance of the neighborhood support for the edge labeling, and an ample amount of parameters for each process, it is clear that examining the coupled model took some time, patience and a significant flexibility of the implemented code.

The chapter contains the basic formulation of a finished piecewise smoothness prior and the description of the optimization method. Results and discussion follow in the next chapter.

10.1 Piecewise Smoothness Prior

To ensure that the vertex smoothing process does not smooth over the sharp edges, the smoothness potential from Equation (6.1) is modified to the potential when the edge labeling **E** is given. In the case of the curvature-based potential we have the following expression

$$U_V(\mathbf{V}|\mathbf{E}) = \sum_{e \in \mathbf{E}} \varepsilon \, |\phi_e| \, \|\mathbf{e}\| \ .$$

The presence of a ridge edge cuts (or, in continuous case, lessens) the smoothness constraint between the vertices. Any other smoothness potential can be modified in the same way.

In case of the deterministic edge labeling, U_V is the prior energy. Having the edge labeling defined as the MRF on mesh edges, the total prior energy of the piecewise smoothness prior is

$$U_{PR} = U_V(\mathbf{V}) + \delta\ U_E(\mathbf{E}) \ ,$$

where δ is the weight between the smoothness potential and the edge labeling potential.

Directly from the definition, the vertex potential depends on the edge labeling. Edge potential also depends on vertex configuration, since one needs to get hold of dihedral and support angles to calculate the edge energy and all angles change with the vertex configuration. So, we have a doubly-coupled MRF.

The prior distribution is expressed as

$$P(\mathbf{V},\mathbf{E}) \propto \exp\left(-\frac{1}{T}U_{\text{PR}}\right)$$

and the posterior distribution is obtained by applying the Bayes rule

$$P(\mathbf{V},\mathbf{E}|\mathbf{V^0}) \propto P(\mathbf{V^0}|\mathbf{V},\mathbf{E})P(\mathbf{V},\mathbf{E}) \ .$$

As the edge labeling **E** is not observed for the likelihood probability we use

$$P(\mathbf{V^0}|\mathbf{V},\mathbf{E}) = P(\mathbf{V^0}|\mathbf{V}) \ .$$

10.2 Optimization Scheme

The posterior energy consists of three terms: likelihood energy, smoothness energy and edge-labeling energy (the last again consisting of two terms: sharpness energy and neighborhood support energy). In case of a discrete edge-labeling MRF, this is a combination of the real and the combinatorial problems.

To use Metropolis sample for optimization, at each iteration of the algorithm every vertex and every edge should be perturbed, evaluated, and allowed to be updated according to the Metropolis criterion. For each perturbation a conditional probability should be evaluated by looking at the part of the total energy, which is affected by the perturbation.

Evaluating the edge perturbation is almost the same as in the pure edge process, where the edge sharpness and neighborhood support are evaluated. Only additional factor to consider is the change in the smoothness potential due to the labeling of the edge, so only one additional term needs to be evaluated.

Vertex perturbation, on the other hand, has a big effect on the edge-labeling energy. Changing a vertex position affects all edges incident to the vertex, which in turn affects all two-edge cliques. For a regular mesh where a vertex is incident to 6 edges only this would mean evaluating 105 additional angles, a considerable computations burden.

So, instead of minimizing the total posterior energy, a much simpler optimization scheme is used where I alternate between the vertex process and the edge process, optimizing one at a time. The convergence is not guaranteed, but the experiment show that this also leads to the minimization of the joint energy. This also means that instead of using weight δ to balance the smoothness potential and edge labeling potential, I can use two different temperatures T_V and T_E, for vertex process and one edge process respectively. This also makes it possible to use different cooling schemes for vertex and edge process. It proves very useful later.

10.3 Two Questions

The work on the coupled model consisted mainly in experiments to determine how do the developed priors preform and what is the best way of formulating the prior for piecewise smooth surfaces. Additionally, experiments were done to cast the light on two open questions:

1. Does the edge labeling that includes neighborhood support contribute positively to smoothing piecewise smooth surfaces, or can the same effect be achieved by using only sharpness of the edges?
2. Does the separate edge labeling process contribute positively to smoothing piecewise smooth surfaces, or can the same effect be achieved by using only well-chosen smoothness prior?

The reason for asking the first question is already covered in Section 8.4. Second question is interesting if the answer to the first question is 'No'. If the neighborhood support proves not useful, does it make sense to explicitly label the edges if we can chose a feature-preserving smoothness prior as demonstrated in Chapter 7?

10.4 Summary

In this chapter I described the way of coupling one of the smoothness priors developed in Chapter 6 and one of the edge labeling methods developed in Chapter 8. The result is a feature preserving mesh smoothing scheme. Since the edge labeling method is in fact ridge-detecting, the scheme developed here is preforming best on piecewise smooth surfaces with sharp ridges between the smooth parts.

Instead of optimizing the total vertex and edge energy, which would be computationally very demanding, the optimization scheme with alternating vertex and edge process is developed.

There is a number of smoothness priors to chose from, and a number of edge labeling methods, so there was many coupled combinations to test. The two questions, which should be resolved by the experiment, are posed here. The first is about the role of neighborhood support, and the second about the role of the edge labeling. Experimental results addressing those questions are presented in the next chapter.

Chapter 11

Final Results and Discussion

This chapter contains some of the experiments conducted to evaluate the performance of the coupled model.

First, an evolution of the developed smoothing methods is presented — the results shown there are not the best achieved results, but are instead examples of the problems one can encounter when using the methods in question.

Some of the successful smoothing is presented and discussed next. And lastly, the question about the role of the edge process is addressed and resolved in this chapter.

11.1 Smoothing Evolution

In order to cast some light on the two questions formulated above, an experiment was preformed, where we start with pure smoothing and add first sharpness-based edge process and later also neighborhood support. This experiment does not present the best performance of a chosen smoothing methods. On the contrary, some parameters were chosen to demonstrate the pitfalls connected with each of the smoothing processes.

First, Figure 11.1 shows two runs of pure smoothing. The two potentials used are the familiar quadratic and square-root potentials. As before, quadratic potential over-smooths the ridges and leaves no dihedral angle over 75°. The square-root potentials preserves the ridges, and those mesh edges that have dihedral angle larger than 75° are shown red.

Next, in Figure 11.2 the deterministic edge labeling is used. Ridge edges are detected nicely, but quite a lot of noise got labeled (and preserved) as features, too. This is an often observed problem of deterministic edge labeling — once a noise gets labeled as a ridge its smoothing is prohibited and in turn it becomes a ridge edge. The label acts as a self-fulfilling prophecy. That kind of behavior is partly what we want a prior to do, as we want the prior to find what we are looking for. But the problem with deterministic labeling is that it does not allow for an alternative hypothesis, but gets stranded in it's initial estimations.

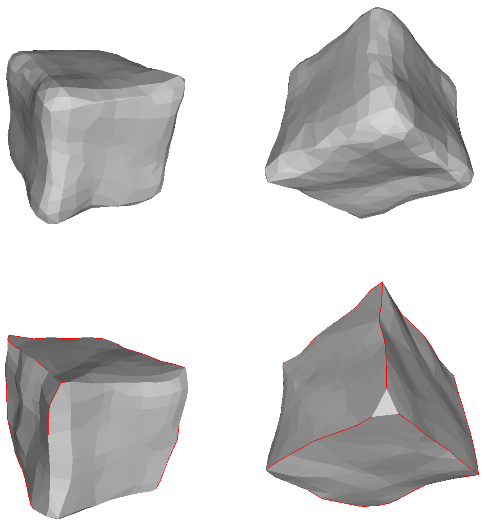


Figure 11.1: The results of the two different pure smoothing processes. *Top:* Using the over-smoothing quadratic smoothness potential. *Bottom:* Using the feature-preserving square-root smoothness potential.

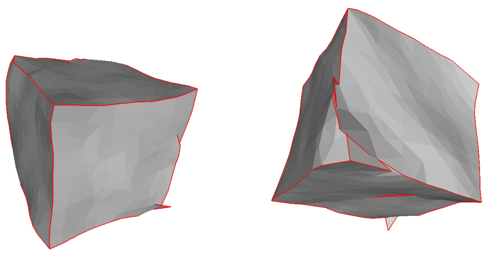


Figure 11.2: The result of the process where the quadratic smoothness prior is coupled with the deterministic edge labeling based on cut-off function.

In Figure 11.3 the stochastic process is used, introducing some randomness to exactly the same cut-off function as used in the previous figure. Improvement is obvious. Allowing the evaluation of the alternative labeling resulted in the very successful ridge detection. The only artifact is one zigzag ridge.

Finally, in the Figure 11.4 the neighborhood support is introduced and we can immediately see it's effect. Some cube ridges get detected, but so do some features in the noise. In the case of the linear prior we clearly see the effect of 'chaining' the ridge edges.

As mentioned, this experiment exposed the worst side of the used smoothing methods, but it also proved that introducing each new step in the 'evolution' of the smoothness has effect on the performance.

11.2 Final Result

The best results were obtained by using the quadratic smoothness prior and continuous stochastic edge labeling based on linear potentials. Those are the results presented here. However, all of the other combinations also produced satisfactory results, if a good choice of parameters was used.

It is understandable that over-smoothing quadratic prior is preforming best in coupled model. When the edge process takes care of the edges, the smoothing should be as strong as possible. As for the edge process, the stochastic versions proved better then the deterministic. A certain level of randomness is needed to prevent the edge process being fixed on noise, or missing to detect an over-smoothed edge.

To obtain a good smoothing result using the combined model, one has to think about a couple of issues:

Cooling schemes for vertex and edge process. The optimization method made it possible to chose the cooling schemes for the vertex process and edge process independent of each other. This proved very useful. It was observed that the best result were found when the temperature of the edge process stayed on the constant, small value. This would introduce enough randomness in the edge process, so that the alternative solutions got explored. The smoothing process could use simulated annealing scheme, but good result in considerably less time were obtained when smoothing with zero temperature.

Ordering of vertex and edge process. The smoothing process can be significantly assisted if ordering of the vertex and edge process is chosen well. Having a very noisy input mesh, it proved useful to start with a few sweeps of the vertex process to remove the worst noise, before turning the edge process on. This should, on the other hand, not be used with the meshes that are not very noisy, where it is important to detect the edges before they get rounded by the smoothing process. A possibility of turning the edge labeling off can

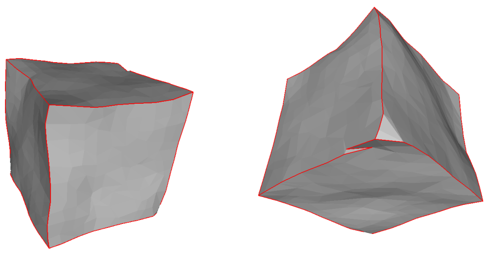


Figure 11.3: The result of the process where the quadratic smoothness prior ic coupled with the stochastic edge labeling based on applying penalty to the distance from the result of the cut-off function.

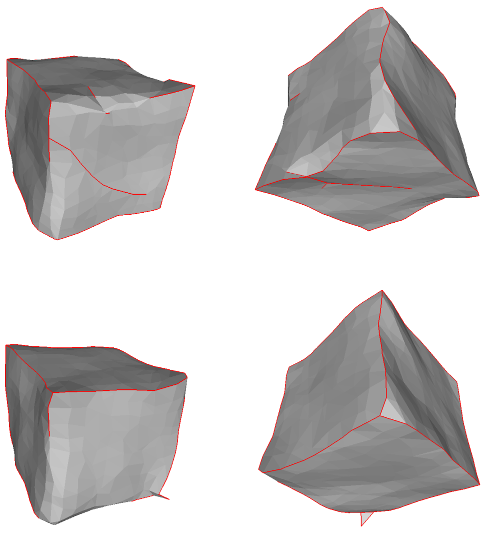
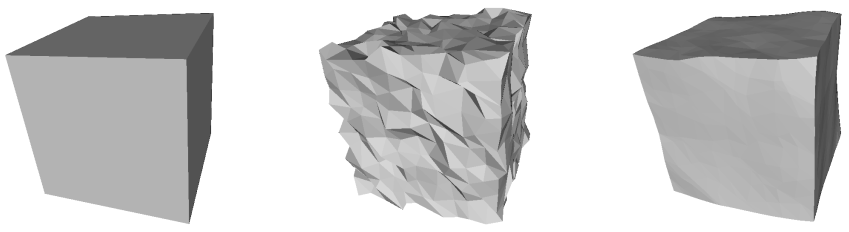


Figure 11.4: The result of two rather different processes, but both using quadratic smoothness prior coupled with the edge labeling, which includes the neighborhood support. *Top:* The edge process based on linear smoothness and support potential, but where labels are allowed to be continuous. *Bottom:* The edge process based on applying sharpness penalty to labeling that differs from the cut-off function and a neighborhood support where parallel neighboring edges are forced to have similar labels.

also prove useful when ridges get detected, and smoothing algorithm is given some more iterations to remove the low frequency noise.

In Figure 11.5 a result of using the coupled model is shown. The noise is removed and the edges preserved. However, some low frequency oscillations are still present, and the ridges of the model are not straight. The likelihood potential for the noise-free image is rather high, and we can conclude that the data term contributes to the low frequency noise.



distances:
$d_H = 0.66$ (0.58 av.e.l.)
$d_{ME} = 0.19$ (0.16 av.e.l.)

distances:
$d_H = 0.31$
$d_{ME} = 0.04$

potentials/model	noise-free	noisy	smoothed
likelihood	46	0	16
smoothness	0	334	9
sharpness	-63	-127	-52
support	-85	-41	-79
total	-102	166	-106

Figure 11.5: Smoothing the noisy cube using the coupled model. *Top left:* Testing model, $10 \times 10 \times 10$ cube. *Top center:* Cube corrupted with Gaussian noise, $\sigma_{noise} = 0.2$ average edge length. *Top right:* Cube smoothed using the quadratic smoothness prior and stochastic continuous labeling based on linear potentials. Parameters used: data weight $\alpha = 0.5$, vertex process temperature $T_V = 0$, sampling step $\sigma_N = 2\sigma_T = 0.1$ average edge length. Neighborhood support weight $\beta = 0.5$, sharpness threshold $\phi_0 = 60^\circ$, support threshold $\theta_0 = 90^\circ$, edge process temperature $T_E = 0.05$, no annealing (k=1). *Middle:* Hausdorff distance and mean error between the noise-free model and the remaining two models. *Bottom:* The potentials for all three models, where likelihood potential has the noisy mesh as reference, and edge labels are rounded for better comparison.

Figures 11.6 and 11.7 show the smoothing of the corrupted fandisk model. Almost all feature edges are successfully preserved, apart from a very subtle feature edge, which was commented next to Figure 8.1. The low frequency oscillations have again not been totaly removed, especially from the more noisy mesh. Allowing more than just 100 iterations would probably help, even though the plots of the updates and potentials implies convergence.

However, the fandisk model has 13 000 faces, and smoothing it was a true test of the algorithms efficiency. Almost an hour was needed for 100 iterations. An implementation focused on efficiency would certainly improve running time, but smoothing large meshes would still be time demanding.

11.3 Why Not Just Thresholded?

Yet again a question about the role of the edge labeling is addressed here. Edge labeling allows a control over the sharpness of the preserved edges. Could the same be achieved by using a well chosen smoothness prior? A small experiment has been conducted to answer this question.

The results on the Figure 11.8 demonstrate the control achieved by changing the sharpness threshold ϕ_0. Feature edges of different sharpness get labeled and preserved. This allows a precise use of the surface prior.

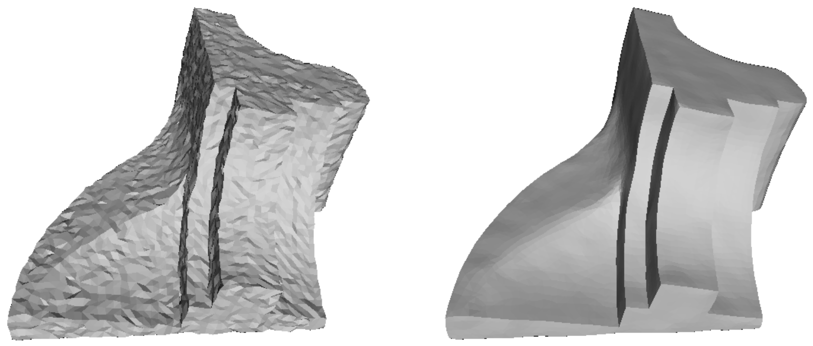
In the Figure 11.9 the thresholded smoothness prior is applied to the same model. The results exhibit the same control of the ridge sharpness — by changing the threshold we influence the number of rounded edges. So, it *is* possible to obtain similar results by using only smoothness prior, but after more than double iterations.

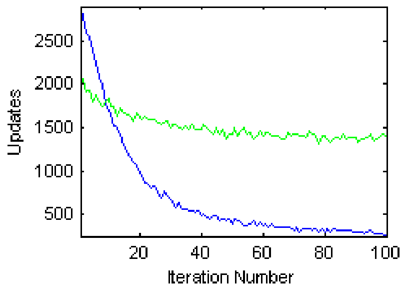
However, when comparing the quality of the results one can see that the explicit edge labeling produces nicer result, with less artifacts. The thresholded smoothness prior has even left an inverted triangle.

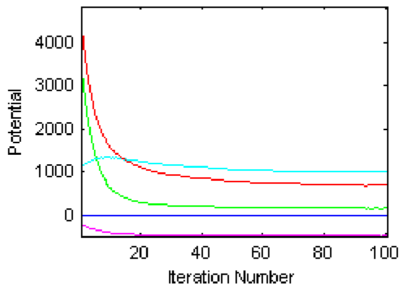
11.4 Conclusion

In this chapter I used the coupled vertex and edge process to smooth the mesh, while detecting and preserving sharp ridges of the mesh surface. The presented results show considerable denoising of the mesh surface, but small low-frequency oscillations would often still be present in the smoothed mesh. To remove the low-frequency oscillations (i.e., preform mesh fairing) the data term needs to be adjusted carefully. Adaptively changing the size of the sampling step could probably also contribute to the removal of the small oscillations.

The questions about the role of the edge labeling process and the role of the neighborhood support are addressed in this chapter. It has been shown that both of those processes help preserve sharp surface ridges. However, as we increase the complexity of the smoothing algorithm, the number of smoothing parameters grows, and it gets increasingly difficult to estimate the parameters or to predict the







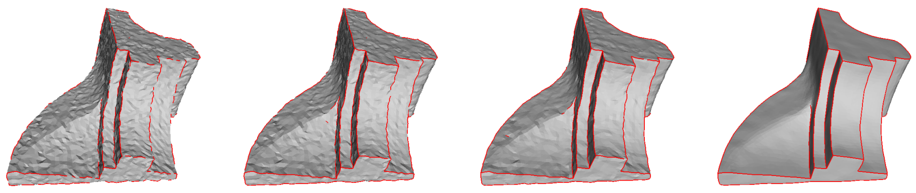


Figure 11.6: Reconstructing the fandisk model corrupted by Gaussian noise $\sigma_{\text{noise}} = 0.1$ average edge length. Model used: quadratic smoothness prior and stochastic continuous edge labeling based on linear potentials. Parameters used: data weight $\alpha = 0.1$, vertex process temperature $T_V = 0$, sampling step $\sigma_N = 0.05$ average edge length, $\sigma_T = 0.02$ average edge length. Neighborhood support weight $\beta = 0.5$, sharpness threshold $\phi_0 = 80°$, support threshold $\theta_0 = 90°$, edge process temperature $T_E = 0.05$, no annealing (k=1). *Top:* Corrupted model and reconstructed model (100 iterations). *Middle left:* The number of updated vertices (blue) and edges (green) over time. *Middle right:* Potentials over time: blue — likelihood potential, green — smoothness potential, cyan — sharpness potential, magenta — support potential, red — total potential. *Bottom:* Edge labeling for iterations 1 (based only on sharpness), 5, 10 and 100.

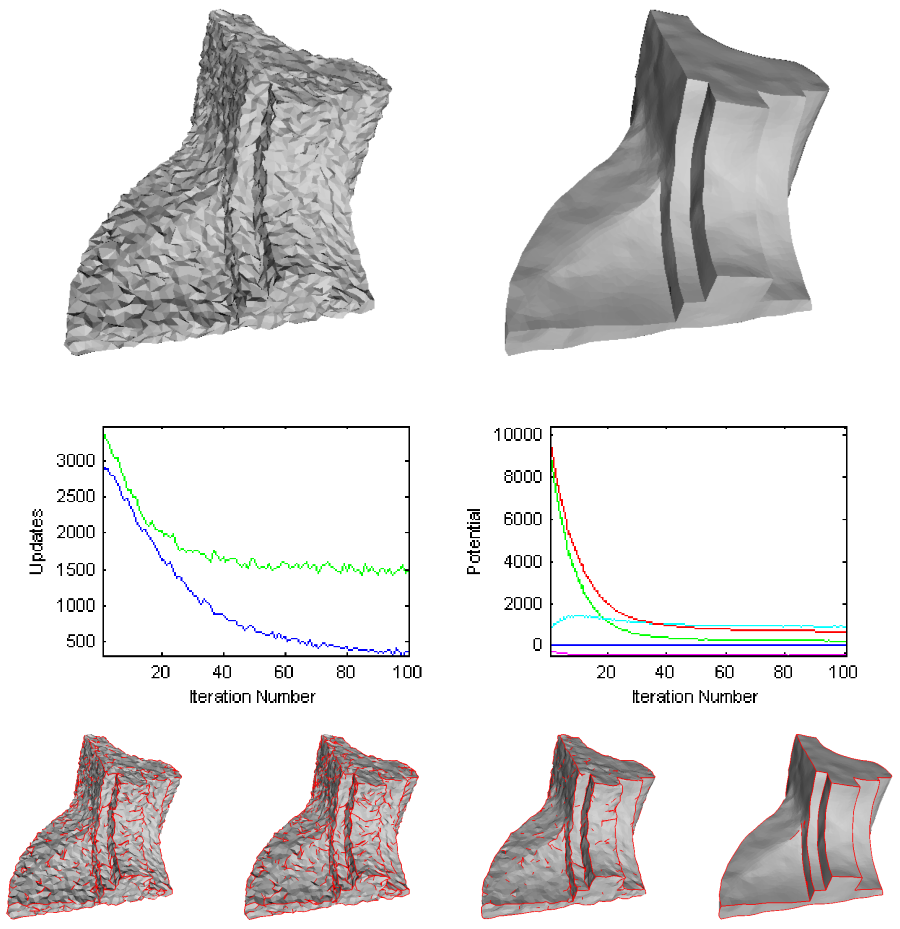


Figure 11.7: Reconstructing the fandisk model corrupted by Gaussian noise $\sigma_{\text{noise}} = 0.2$ average edge length. The model and the parameters the same as in Figure 11.6, apart from $\phi_0 = 70^\circ$, $\alpha = 0.2$. *Top:* Corrupted model and reconstructed model (100 iterations). *Middle:* Updates over time (blue — vertices, green — edges), potentials over time (blue — likelihood, green — smoothness, cyan — sharpness, magenta — support, red — total). *Bottom:* Iterations 1, 5, 15 and 100.

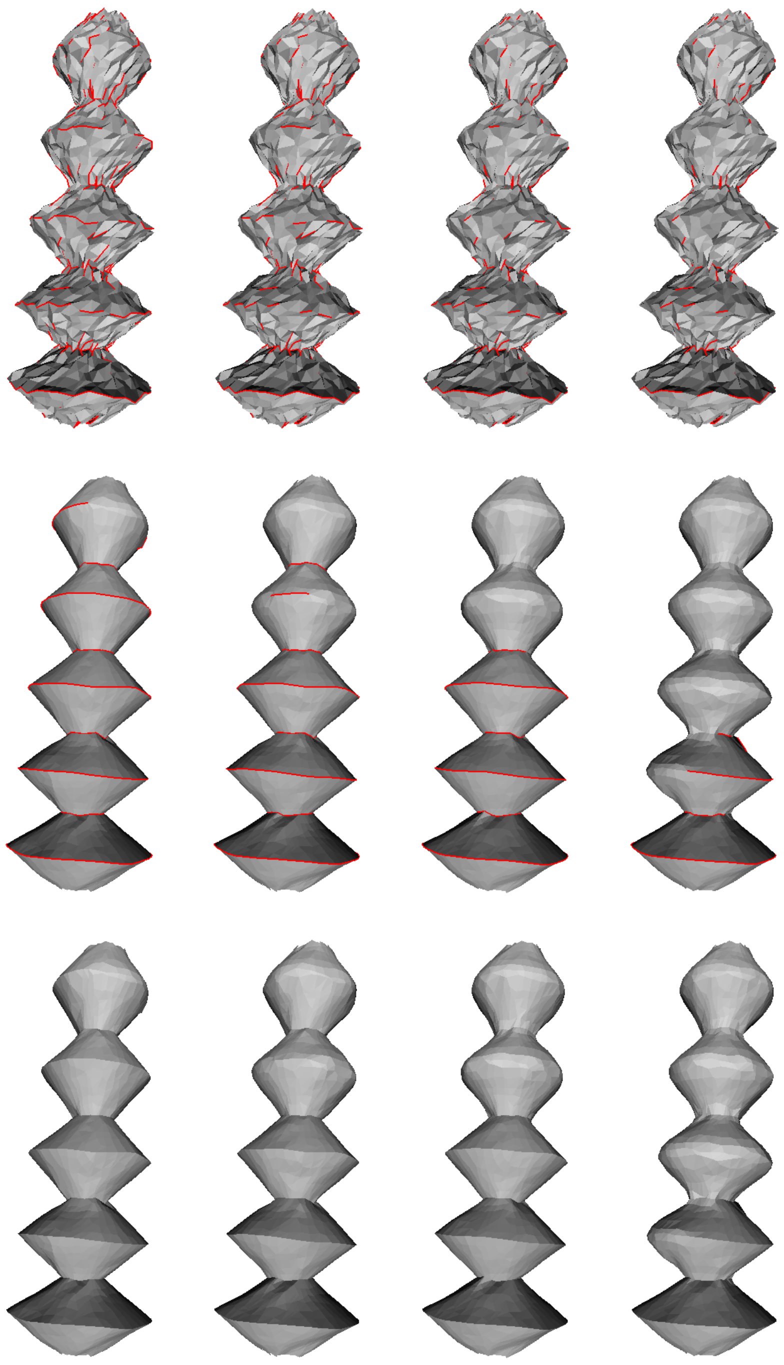


Figure 11.8: Sharpness control for the coupled model. Sharpness threshold ϕ_0 determines which edges are labeled as ridge and which edges get smoothed. The edgy model has edges from 68° to 108° in steps of 5°. *Left to right:* Sharpness threshold put to 85°,95°,105° and 115°. *Top to bottom:* Initial edge labeling (based only on sharpness), labeling after 100 iterations, and a final reconstruction.

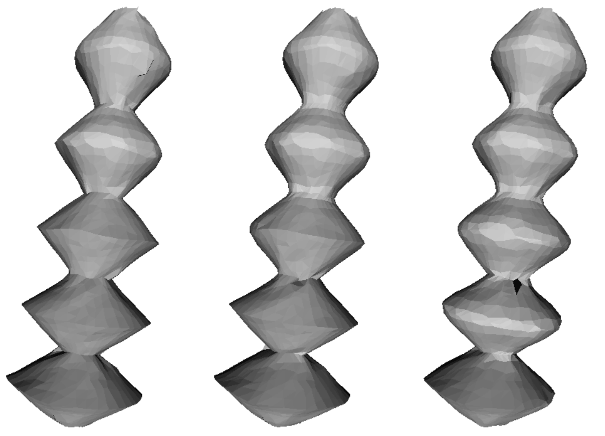


Figure 11.9: Sharpness control for the smoothing without edge labeling, but with the thresholded smoothness potential. Smoothness threshold put to 65°,75° and 85° (left to right), the results after 300 iterations.

behavior of the smoothing algorithm. Some of the common pitfalls (e.g., detecting the ridge in the noise) has also been shown.

Still, even though the explicit edge labeling produces nicer result, using only smoothness potential might be sufficient for some applications, especially where the automatic smoothing is needed. In cases where we want more control in describing surface prior it would be beneficial to use the coupled model.

Chapter 12

Future Work

The future work following from this thesis can roughly be divided it in three groups. First, I suggest a number of small improvements to the algorithm implemented here. Second, a functional mesh smoother could be developed, where we drop the assumptions about the correct topology, and consider the input mesh as triangulation of a point cloud obtained from a 3D scanner. Third, a number of extensions to both the smoothness prior and to the edge labeling are possible, which would allow describing priors for other types of surfaces. In this chapter I briefly mention the different possibilities.

12.1 Improvements of The Existing Algorithm

Some of the possible improvements have already been discussed, mostly regarding the optimization scheme. In Section 6.7.1 I suggest the improvements of the random sampling algorithm regarding the shape of the used Gaussian distribution and the size of the update step.

More importantly, the benefit of the simulated annealing should be investigated, and some deterministic optimization method should be tested for comparison.

As for the edge labeling, the more efficient priors should be devised, probably using larger neighborhood than here.

12.2 A Functional Mesh Smoother

The focus of this thesis was to investigate the performance of the developed smoothness priors with the idea of using it for smoothing the mesh. In practical applications the input mesh will be obtained the 3D scanners, and the assumptions about correct topology and triangulation will be invalid. For example, there is in general no guarantee that the tip of the ridge will get sampled or that the mesh edges lie along the surface features. A smoothing system should therefore take those facts in consideration. Likewise, a care should be taken to ensure the quality of the output mesh (e.g., the shape of the triangular faces).

In a practical smoothing system focus should be put on surface to surface smoothing. The likelihood function developed here, which measures the displacement of the vertices, is therefore not an optimal choice. The distance between surfaces should be measured instead, as briefly mentioned in Section 6.5.

12.3 Extensions of the Model

The basic model developed here could be extended in different ways. The prior for piecewise-quadratic surfaces can be developed, by requiring the curvature to be constant between the ridges. As for the ridge detection, an additional prior forcing the detected ridges to be straight or smoother could significantly assist the mesh fairing process. A mechanism for removal of unwonted spurious ridge-edge structures is yet another likely improvement.

It could be possible to use the piecewise-smoothness prior for segmenting the mesh in smooth parts, which could then allow for another representation of the underlying surface.

Chapter 13

Conclusion

The primary aim of this Master's thesis was to investigate the use of MRF theory in formulating priors on surface meshes. A few different priors have been presented here, showing indeed that the desired results are produced in this setting.

It is pretty straightforward to define MRF on triangular meshes using the mesh connectivity. However, when implementing a smoothing algorithm a lot of details had to be dealt with.

In case of the vertex process, the developed priors are easy to understand and produce a desired result. The analysis and the experimental results show that the quadratic potential rounds the sharp ridges, square root potential preserves sharp ridges, and thresholded potential makes it possible to control the sharpness of the ridges that one wants preserved. Still, there are many open questions about the optimization for the vertex process. The Metropolis sampler with simulated annealing was used to avoid making any assumptions about the energy function. However, the results indicate that the gradient-based method could also be applied successfully, which would surely improve the efficiency of the algorithm.

In case of the edge process, it is not easy to formulate the neighborhood support in a simple and straightforward way. The sharpness of the edges is constrained by the mesh connectivity and the role of the neighborhood support is not obvious in this setting. It is demonstrated that neighborhood support contributes to good edge labeling, but almost just as good a result is obtained without the use of neighborhood support. On the other hand, it is observed that the explicit edge labeling helps detect the ridges on the surface.

In this work some light is shed on the behavior of the MRF priors on triangular meshes. After the promising preliminary results, it would surely be interesting to see a practical application that uses MRF on surfaces. This is probably just a matter of time.

Appendix A

Box-Müller Transformation

Box-Müller transformation [Box and Muller, 1958] is a popular method of generating pseudo-random samples from a normal (Gaussian) distribution. The method generates pairs of independent normally distributed random numbers, given a source of uniformly distributed random numbers.

Let u_1 and u_2 be two independent and uniformly distributed random numbers, $u_1, u_2 \sim \mathrm{U}[0,1]$, where $u_1 \neq 0$. The numbers

$$z_1 = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$$

$$z_2 = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$$

are then independent and normally distributed random numbers, $z_1 \sim \mathrm{N}[0,1]$ and $z_2 \sim \mathrm{N}[0,1]$.

Appendix B

Hausdorff Distance

One of the simplest distortion measurements for 3D models is Hausdorff distance, a generic technique used to define a distance between two nonempty sets.

To define the Hausdorff Distance between two surfaces S and S' we start with defining point-to-surface distance between a point p belonging to surface S and a surface S' as

$$d(p,S') = \min_{p' \in S'} \|p - p'\| \ ,$$

where $\|.\|$ is the Euclidean norm. From this definition, the Hausdorff distance between S and S' is given as

$$d(S,S') = \max_{p \in S} d(p,S') \ .$$

As illustrated on Figure B.1, the Hausdorff distance is in general not symmetrical, so it is convenient to introduce the symmetrical Hausdorff distance

$$d_s(S,S') = \max\left(d(S,S'), d(S',S)\right) \ .$$

When surfaces S and S' are represented as triangle meshes, $d(p,S')$ can be computed analytically for any point $p \in S$, as the minimum of the distances between p

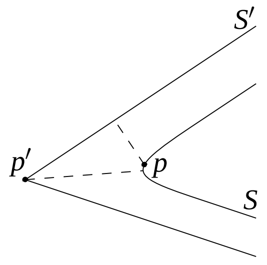


Figure B.1: Hausdorff distance is generally not symmetrical. In this case $d(S,S') = d(p,S') < d(p',S) = d(S',S)$.

and all the triangles that make the surface S'. However, to obtain the maximum for $p \in S$ it is necessary to resort to sampling of the triangles that make the surface S [Aspert *et al.*, 2002].

Instead of sampling the surface S, a fairly good approximation of the Hausdorff distance is obtained if one considers only the vertices v of the surface S. This leads to vertex-to-surface Hausdorff distance

$$d_v(S,S') = \max_{v \in V} d(p,S')$$

where V is the set of vertices of the triangle mesh that represents the surface S. In the similar way as above, the symmetric version of the vertex-to-surface Hausdorff distance is defined.

The mean and the root-mean-square error can also be approximated in the vertex-to-surface manner as

$$d_m(S,S') = \frac{1}{|V|} \sum_{v \in V} d(v,S')$$

$$d_{rms}(S,S') = \sqrt{\frac{1}{|V|} \sum_{v \in V} d(v,S')^2}$$

where $|V|$ is the number of vertices in the mesh that makes the surface S.

Bibliography

[Aspert *et al.*, 2002] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. Mesh: Measuring errors between surfaces using the Hausdorff distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume I, pages 705 – 708, 2002.

[Attene *et al.*, 2005] Marco Attene, Bianca Falcidieno, Jarek Rossignac, and Michela Spagnuolo. Sharpen&bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):181–192, 2005.

[Bærentzen and Aanæs, 2005] Jakob Andreas Bærentzen and Henrik Aanæs. Signed distance computation using the angle weighted pseudo-normal. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):243–253, may 2005.

[Botsch *et al.*, 2006] Mario Botsch, Mark Pauly, Christian Rössl, Stephan Bischoff, and Leif Kobbelt. Geometric modeling based on triangle meshes. SIGGRAPH Course Notes, 2006.

[Box and Muller, 1958] George Edward Pelham Box and Mervin Edgar Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29:610–611, 1958.

[Desbrun *et al.*, 1999] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Desbrun *et al.*, 2000] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Anisotropic feature-preserving denoising of height fields and images. In *Proceedings of Graphics Interface*, pages 145–152, 2000.

[Diebel and Thrun, 2005] James R. Diebel and Sebastian Thrun. An application of Markov random fields to range sensing. In *Proceedings of Conference on Neural Information Processing Systems*, Cambridge, MA, 2005. MIT Press.

[Diebel *et al.*, 2006] James R. Diebel, Sebastian Thrun, and Michael Brünig. A Bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics*, 25(1):39–59, 2006.

[Dyn *et al.*, 2001] Nira Dyn, Kai Hormann, Sun-Jeong Kim, and David Levin. Optimizing 3D triangulations using discrete curvature analysis. In *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 135–146, Nashville, TN, USA, 2001. Vanderbilt University.

[Fleishman *et al.*, 2003] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 950–953, New York, NY, USA, 2003. ACM Press.

[Frost and Heineman, 1997] Richard Frost and Philip Heineman. Simulated annealing: A heuristic for parallel stochastic optimization. In *PDPTA '97: Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, 1997.

[Geman and Geman, 1984] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(332):721–741, 1984.

[Hammersley and Clifford, 1971] John Michael Hammersley and Peter Clifford. Markov field on finite graphs and lattices. 1971.

[Hartelius and Carstensen, 2003] Karsten Hartelius and Jens Michael Carstensen. Bayesian grid matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2), February 2003.

[Jiao and Alexander, 2005] Xiangmin Jiao and Phillip J. Alexander. Parallel feature-preserving mesh smoothing. In *International Conference on Computational Science and Its Applications (4)*, pages 1180–1189, 2005.

[Jones *et al.*, 2003] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 943–949, New York, NY, USA, 2003. ACM Press.

[Kirkpatrick *et al.*, 1983] Scott Kirkpatrick, Jr. C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Li, 2001] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Verlag, Tokyo, second edition, 2001.

[Sun *et al.*, 2003] Jian Sun, Heung-Yeung Shum, and Nan-Ning Zheng. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[Szeliski and Tonnesen, 1992] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. In *SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 185–194, 1992.

[Tasdizen *et al.*, 2002] Tolga Tasdizen, Ross Whitaker, Paul Burchard, and Stanley Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *VIS '02: Proceedings of the Conference on Visualization 2002*, pages 125–132, Washington, DC, USA, 2002. IEEE Computer Society.

[Taubin, 1995] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 351–358, New York, NY, USA, 1995. ACM Press.

[Willis *et al.*, 2004] Andrew Willis, Jasper Speicher, and David B. Cooper. Surface sculpting with stochastic deformable 3d surfaces. In *ICPR '04: Proceedings of the 17th International Conference on Pattern Recognition*, volume 2, pages 249–252, Washington, DC, USA, 2004. IEEE Computer Society.

[Winkler, 2003] Gerhard Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*. Springer Verlag, 2003.