Computational Differential Geometry Tools for Surface Interrogation, Fairing, and Design

Dissertation

zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Shin Yoshizawa

Saarbrücken
2006

Datum des Kolloqiums: 18.12.2006
Dekan der Naturwissenschaftlich-Technischen Fakultät I:
Prof. Dr. Thorsten Herfet

Mitglieder des Prüfungsausschusses:
Vorsitzender: Prof. Dr. Philipp Slusallek
1. Gutachter: Prof. Dr. Hans-Peter Seidel
2. Gutachter: Prof. Dr. Alexander G. Belyaev
Akademischer Mitarbeiter: Dr. Hendrik P. A. Lensch

Abstract

This thesis presents a set of new mesh processing methods which are based on computational differential geometry techniques. The underlying idea of the methods consists of using proper discrete approximations of differential surface properties. The methods developed in the thesis contribute to the areas of curvature feature detection, mesh parameterization, fair mesh generation, mesh denoising, and free-form and variational mesh deformations. Comparisons of the developed methods with several state-of-the-art techniques and algorithms are done. The results of numerous numerical experiments demonstrate significant advantages of the proposed methods over conventional techniques. Applications of the methods are discussed and demonstrated.

The main contributions of the thesis are as follows:

Similarity-based Mesh Denoising. A new, powerful, and high quality feature preserving mesh/soup denoising technique and a new scheme for comparing different mesh/soup smoothing methods are proposed. The technique is based on a similarity-weighted averaging procedure and a new and robust similarity measuring scheme.

Fair Mesh Generation via Elastica. A new numerical scheme for generating fair meshes is developed. Applications to shape restoration are considered. The scheme is build upon a discrete approximation of Willmore flow. A tangent speed component is introduced to the discrete Willmore flow in order to improve the quality of the evolving mesh and to increase computational stability.

Fast and Robust Detection of Feature Lines on Meshes. A new, fast, and robust crest line detection method is developed. Applications to feature-adaptive mesh simplification and segmentation are considered. A novel thresholding scheme and a simple new formula for computing directional curvature derivatives are also introduced.

Fast Low-Stretch Mesh Parameterization. A new, fast, simple, and valid low-stretch mesh parameterization scheme and its application for efficient remeshing are proposed by using a moving mesh approach. The scheme is based on a weighted quasi-conformal parameterization which equalizes the local stretch distribution. Particularly, the scheme does not generate regions of undesirable high anisotropic stretch.

Free-Form Skeleton-driven Mesh Deformations. A new and powerful approach for generating natural-looking large-scale mesh deformations is proposed. An interesting feature of the approach consists of preserving original shape thickness. New self-intersection fairing schemes are also developed. Multiresolutional and variational extensions of the approach are considered.

Kurzzusammenfassung

Diese Dissertation stellt neue Bearbeitungsmethoden für Dreiecksnetze vor, die auf Techniken der rechnergestützten Differentialgeometrie basieren. Die zugrundeliegende Idee dieser Methoden ist, geeignete diskrete Näherungen für analytische Flächeneigenschaften zu verwenden. Die Methoden, die in dieser Dissertation entwickelt werden, stellen einen Beitrag zu folgenden Gebieten dar: Erkennung von Flächencharakteristika, Parametrisierung von Dreiecksnetzen, Erzeugung von ästhetischen Dreiecksnetzen, Entfernen von Rauschen in Dreiecksnetzen und Deformation von Dreiecksnetzen für freie Gestaltung mit Variationsmethoden. Vergleiche der entwickelten Methoden mit aktuellen Techniken und Algorithmen werden angestellt. Die Ergebnisse der zahlreichen numerischen Experimente zeigen eine hohe Leistung der vorgeschlagenen Methoden. Anwendungen der Methoden werden besprochen und vorgeführt. Die Hauptbeiträge der Dissertation sind folgende:

Ähnlichkeitsbasiertes Entfernen von Rauschen in Dreiecksnetzen.
Eine neue, leistungsfähige Technik zum Entfernen von Rauschen in Dreiecksnetzen mit und ohne Konnektivität mit qualitativ hochwertiger Bewahrung von Flächencharakteristika und ein neues Schema für das Vergleichen unterschiedlicher Dreiecksnetz-Glättungsmethoden werden vorgeschlagen. Die Technik basiert auf einer nach Ähnlichkeit gewichteten Mittelung und einem neuen und robusten Schema zur Messung von Ähnlichkeit.

Erzeugung von ästhetischen Dreiecksnetzen mit Elastica.
Ein neues, numerisches Schema für das Erzeugen ästhetischer Dreiecksnetze wird entwickelt. Anwendungen zur Gestalt-Rekonstruktion werden betrachtet. Das Schema gründet auf einer diskreten Näherung des Willmore-Flusses. Eine Tangentialgeschwindigkeitskomponente wird im diskreten Willmore-Fluss eingeführt, um die Qualität des entstehenden Dreiecksnetzes zu verbessern und die Berechnungsstabilität zu erhöhen.

Schnelles und robustes Erkennen von charakteristischen Linien auf Dreiecksnetzen.
Eine neue, schnelle und robuste Methode zum Erkennen von Kammlinien wird entwickelt. Anwendungen auf Dreiecksnetzvereinfachung und -segmentierung unter Berücksichtigung von Flächencharakteristika werden betrachtet. Ein neues Schwellwert-Schema und eine einfache neue Formel für das Berechnen von Richtungsableitungen von Krümmung werden auch eingeführt.

Schnelle Parametrisierung mit geringer Streckung von Dreiecksnetzen.
Ein neues, schnelles, einfaches und gültiges Schema zur Parametrisierung mit geringer

Streckung von Dreiecksnetzen und seine Anwendung für effizientes Neuvernetzen werden vorgeschlagen, indem man eine ”moving mesh” Methode verwendet. Die Methode basiert auf einer gewichteten quasi-konformen Parametrisierung, die die lokale Streckung gleichmäßig verteilt. Insbesondere erzeugt die Methode keine Regionen unerwünscht hoher anisotroper Streckung.

Freiform, Skelett-kontrollierte Deformation von Dreiecksnetzen.
Eine neuer und leistungsfähiger Ansatz für das Erzeugen natürlich wirkender, großmaßstäblicher Deformationen von Dreiecksnetzen wird vorgeschlagen. Eine interessante Eigenschaft des Ansatzes ist das Bewahren der ursprünglichen Dicke des Körpers. Außerdem werden neue Techniken zum Glätten von Selbst-Überschneidungen entwickelt. Erweiterungen um Auflösungs-Hierarchien und Variationsverfahren des Ansatzes werden betrachtet.

Acknowledgements

I wish to express my most sincere appreciation to Prof. Dr. Hans-Peter Seidel and Prof. Dr. Alexander G. Belyaev for their excellent advice and diligent efforts to guide and support me through my Ph.D study. Also, I would like to thank them deeply for their encouragement and discussions which have became a big energy, motivation, and vitality for me, not only in this thesis but my life in Germany.

I also would like to express my gratitude for Prof. Dr. Philipp Slusallek and Dr. Hendrik Lensch who kindly agree to be examination committee members for my Ph.D defense and carefully read this thesis and gave many useful advice.

I am grateful to Max-Planck Institut für Informatik: Computer Graphics Group, International Max-Planck Research School for Computer Science, and AIM@SHAPE Network of Excellence project for their financial supports during this project.

I would like to thank Dr. Yutaka Ohtake for fruitful discussions about my crest line detection method and many useful geometry processing and programming advice. Discussions with him have been inspiring me a lot about new geometry processing idea and approaches.

I would like to thank Dr. Hugues Hoppe for a helpful discussion about my mesh parameterization method.

Many special thanks to my current and former colleagues of MPI Informatik: Computer Graphics Group for their helpful comments, discussions, and helping my life in Germany. Especially, I would like to acknowledge the following colleagues who helped me so much. Our group secretaries Ms. Sabine Budde and Ms. Conny Liegl have been always kindly helping me about formal procedures regarding to my life in MPII and Germany. Dr. Hitoshi Yamauchi and Dr. Takehiro Tawara had helped me to be accustomed to live in Germany as a foreign country and they also gave me many useful UNIX related techniques. I wish thank Dr. Ioannis Ivrissimtzis, Dr. Jens Vorsatz, and Dr. Christian Rössl for their useful comments and discussions regarding to subdivision, remeshing, and multiresolution methods. Discussions about mesh parameterization techniques with Mr. Rhaleb Zayer who has been a good roommate in my office have been interesting and useful for me. Besides many helpful geometry processing discussions with Mr. Torsten Langer and Mr. Oliver Schall, they helped me to translate the abstract and summary of this thesis to German language. Of course, I am deeply appreciate all other my colleagues of our computer graphics group.

Finally, I wish to thank my family and friends for their support and encouragement.

However, all mistakes of this thesis that remain are my own.

Contents

1

Introduction

We are witnessing an explosion in the use of digital multi-media: sound, image, video, and digital 3D geometry. Rapid advances in 3D shape acquisition technologies are forcing fast and impressive development of Digital Geometry Processing [SS01], a new research area whose goal is to build new mathematical and computational tools needed for efficient processing of 3D geometry information.

Modern surface digitizing devices can yield millions of 3D point locations distributed over the surface of an object being digitized. Usually the collected points are then converted into a dense triangle mesh, a digital surface representation convenient for further shape processing stages including smoothing, interrogating, editing, parameterizing, remeshing, decimating, fitting with curved surface patches, etc.

In this thesis, we deal with piecewise-smooth surfaces approximated by dense triangle meshes and develop new theoretical and computational tools for mesh interrogating, fairing, and editing. Extensive use of differential geometric concepts is a common denominator of the presented mesh processing techniques. The proposed methods are first designed for processing smooth surfaces and then adapted for dealing with dense meshes.

The main results described in this thesis are presented in the works of the author [YB02, YBS02, YBS03, YBS04, YBS05b, YBS05a, YBS06b, YBS06c, YBS06a]. The organization and main contributions of the thesis are as follows.

Chapter 2. Similarity-based mesh denoising. A new similarity-based mesh denoising method developed in [YBS06b] is described.

Chapter 3. Fair mesh generation via elastica. A novel mesh fairing and restoration scheme [YB02] build upon the classical Willmore flow is presented.

Chapter 4. Fast and robust detection of feature lines on meshes. A new technique for fast and robust detection of salient curvature extrema on surfaces approximated by dense triangle meshes [YBS05a] is discussed. Applications to feature-sensitive mesh simplification and segmentation problems are considered.

Chapter 5. Fast low-stretch mesh parameterization. A moving mesh approach adapted to mesh parameterization and remeshing problems [YBS04, YBS05b] is presented and discussed.

Chapter 6. Free-from skeleton-driven mesh deformations. A powerful approach for feature-preserving free-form shape deformations [YBS02, YBS03, YBS06c, YBS06a] is described.

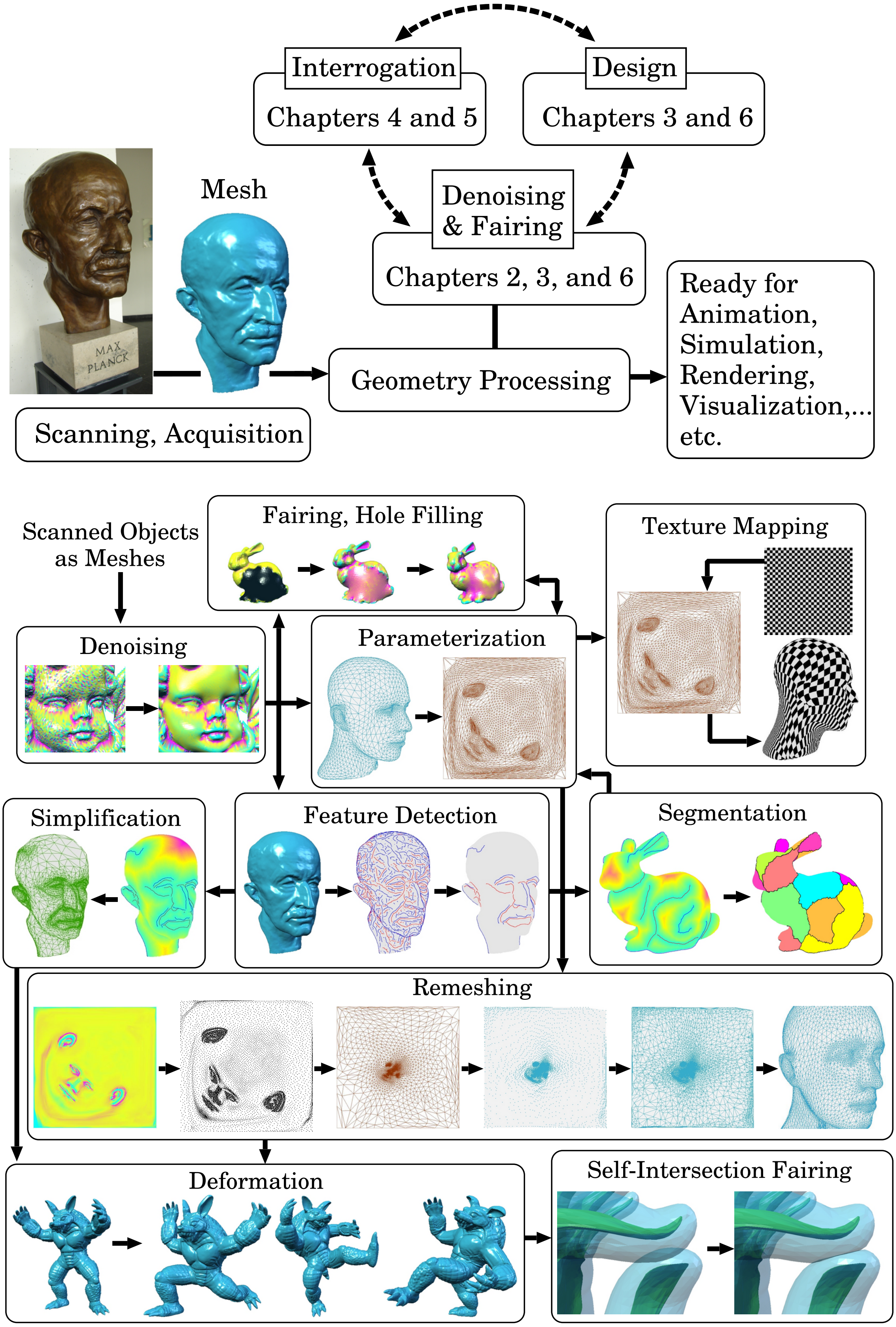


Figure 1.1: Proposed digital geometry processing.

As demonstrated in Fig. 1.1, all these topics are closely connected with each other within various digital shape processing pipelines.

In the rest of this chapter, we provide the reader with short descriptions of each of the above-mentioned thesis contributions.

1.1 Similarity-based Mesh Denoising

Real-world signals do not exist without noise. While recent developments of digital recording and scanning technologies allows us to generate digital data with a relatively high signal-to-noise ratio, denoising digital images and their 3D geometry counterparts, polygonal meshes and point clouds, remains to be an active and important area of research. A common approach to digital signal denoising consists of using linear and nonlinear diffusion/convolution processes. In signal and image processing, denoising techniques are usually based on a Fourier-based analysis and, hence, are nicely adapted for processing signals with regular structure. In geometric modeling, we usually deal with irregular data and, therefore, straightforward adaption and use of signal and image processing denoisng techniques is not possible. A typical approach to mesh smoothing is based on diffusion-like mesh evolutions [Tau95, DMSB99, HP04] and can be reformulated in terms of weighted averaging of mesh vertex positions. Similar to the PDE[1]-based strategy in adaptive image smoothing [PM90, Wei98] where weights depend on the image gradient, the weights in mesh smoothing schemes should reflect variations of mesh normal field in order to achieve an edge-preserving effect.

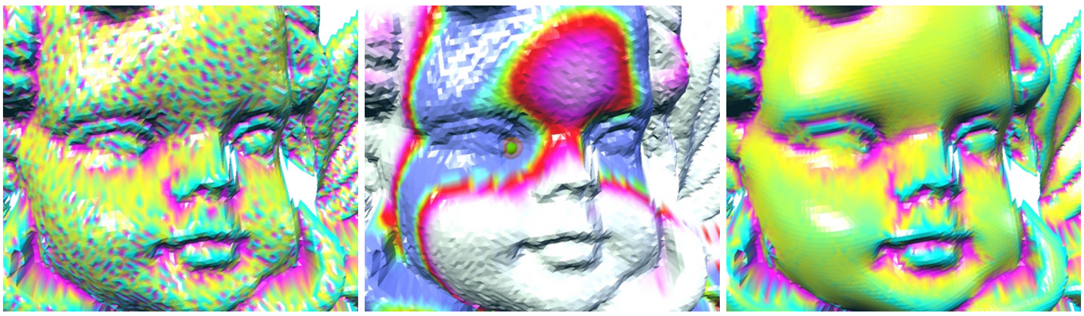


Figure 1.2: Denoising via similarity-weighted averaging. Left: an input noisy scanned mesh colored by mean curvature. Center: coloring by similarity. A mesh vertex is chosen at the left corner of the right eye of the original mesh. The mesh is colored according to a similarity with the shape of the model at the chosen vertex. The similarity increases from white to blue. The vertices with higher similarity values have a stronger contribution to the new (smoothed) position of the chosen vertex. Right: the mesh is smoothed by our similarity-based denoising method colored by mean curvature.

In Chapter 2, we follow [YBS06b] and describe a new and powerful shape denoising technique for processing surfaces approximated by triangle meshes and soups. Our approach is inspired by a recent non-local image denoising scheme proposed by Buades, Coll, and Morel [BCM05a] and naturally extends bilateral mesh smoothing methods [FDCO03, JDD03]. The main idea behind the approach is very simple. A new position of vertex P of a noisy mesh is obtained as a weighted mean of mesh vertices Q with nonlinear weights reflecting a similarity

[1] Partial Differential Equation

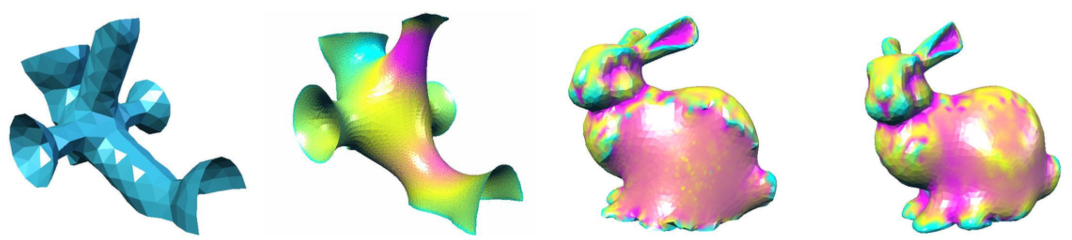
between local neighborhoods of P and Q. The use of similarity weights suppresses smoothing effect over local patterns consisting of the neighborhoods of P and Q (pattern-preserving). We demonstrate that our technique outperforms recent state-of-the-art smoothing methods in terms of quality. Also, a new scheme for comparing different mesh/soup denoising methods is suggested. Figure 1.2 illustrates our similarity-based mesh denoising method.

1.2 Fair Mesh Generation via Elastica

Surface fairing, generating free-form surfaces satisfying aesthetic requirements, is important for many computer graphics and geometric modeling applications. A common approach for fair surface design consists of minimization of a fairness measure which penalizes large curvature values and curvature oscillations. The aesthetic surfaces are usually modeled by solutions of geometric PDEs which are derived from minimizing the fairness measures, e.g. the membrane (squared surface gradient), surface bending (squared normal curvature), and minimum variation curvature (squared gradient curvature) energies.

Variational approaches have been became popular in geometric modeling since 90's (see references in [Yos01]) because of developing fast computers and robust numerical methods for PDE solving. The so-called elastica surface [HKS92] is a natural extension of the Euler's elastica curve [Eul44] where the corresponding fairness measure is the surface bending energy. The corresponding surface evolution whose speed is chosen to minimize the bending energy is the so-called Willmore flow [BS05]. The linearizations of the bending energy such as thin plate splines and biharmonic radial basis functions are often applied in CAGD (Computer Aided Geometric Design) [Far02] and scattered data interpolations [BN92, CBC+01].

Surface evolution techniques have been applied for fair shape modeling [YB02, XPB06], image processing [Wei98], smoothing [Tau95, DMSB99, HP04], fluid mechanics and grid generation [Set96, Lis04], feature extraction and recognition of shape and image [Set96], and many other applications. In Chapter 3, we follow [YB02] and describe a numerical approach for fair surface modeling via geometric surface evolutions of triangle meshes. Chosen the speed function of the evolution properly minimizing the surface bending energy, the evolving surface converges to a desired shape: a discrete elastica. A tangent speed component is introduced to improve the quality of the evolving mesh and to increase computational stability. Figure 1.3 illustrates how our method can be used in various geometric modeling applications.



(a) (b) (c) (d)

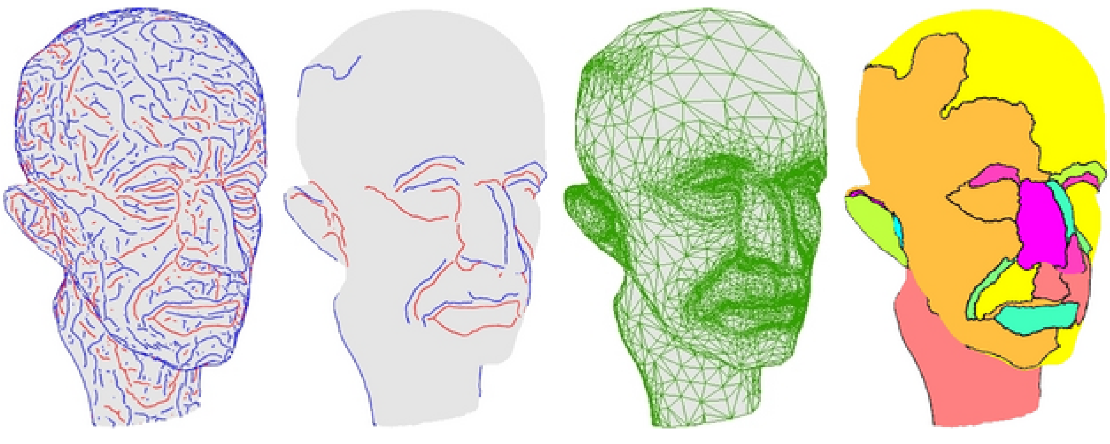
Figure 1.3: Generating fair triangle meshes with discrete elastica. (a): An initial mesh outlined a complex tubular object. (b): A discrete elastica surface (mesh) obtained from the initial mesh. (c): The Stanford bunny model with a large part of the mesh removed and then triangulated. (d): The modified part of the bunny is restored as a discrete elastica. Coloring by the mean curvature is employed to demonstrate a high quality of the generated meshes.

1.3 Fast and Robust Detection of Feature Lines on Meshes

Surface creases, curves on a surface along which the surface bends sharply, are important shape descriptors. They can be intuitively defined as loci of sharp variation points of the surface normals. Mathematically the surface creases can be described via extrema of the surface principal curvatures along their corresponding lines of curvature. Various subsets of such curvature extrema have been thoroughly studied in connection with research on classical differential geometry and singularity theory [Koe90, Por01], quality control of free-form surfaces [Hos92], face pattern analysis [HGY$^+$99], and many other areas of engineering, geographical, geological, medical, and computer sciences. See recent papers [OBS04, YBS05a, HPW05] and Chapter 4 of this thesis for more or less extensive literature surveys.

Practical extraction of curvature extrema is a difficult computational task because it requires a good estimation of high-order surface derivatives. In Chapter 4, we follow [YBS05a] and describe an accurate and efficient method for detecting salient curvature extrema on surfaces approximated by dense triangle meshes. Our approach combines a local polynomial fitting procedure with a new thresholding scheme and allows us to achieve a fast and accurate detection of curvature extrema lines on models with complex geometry.

We are mainly interested in detecting ridge-valley structures on surfaces and demonstrate the power of our approach by dealing with the so-called crest lines, probably the most salient line features on a smooth surface. The crest lines can be considered as a natural generalization of image edges to surfaces and are defined as the loci of points where the magnitude of the largest (in absolute value) principal curvature attains a maximum along its corresponding line of curvature [MBF92]. Thus, provided with a surface orientation, we distinguish the convex crest lines (ridges) and concave ones (valleys).



(a) (b) (c) (d)

Figure 1.4: (a): The crest lines detected, no filtering is applied. (b) Our novel thresholding scheme allows us to keep the most salient ridges and valleys while eliminating less significant crest lines and spurious lines resulting from noise. (c) Our feature-sensitive mesh decimation procedure keeps a higher mesh density near the most important crest lines. (d) Our feature-sensitive mesh segmentation scheme takes into account salient ridges and valleys.

Figure 1.4 demonstrates typical results obtained using our approach. The left images show the crest lines (the ridges and valleys are colored in blue and red, respectively) detected on a detail mesh approximating a surface with complex geometry. Notice how well the most salient ridges and valleys are detected. The right images illustrate how the ridges and valleys can be used for feature sensitive mesh simplification and segmentation.

1.4 Fast Low-Stretch Mesh Parameterization

A surface parameterization process consists of a surface decomposition into a set of patches and establishing one-to-one mappings between the patches and reference domains. Numerous applications of surface parameterization in computer graphics and geometric modeling include texture mapping, shape morphing [Ale02], surface reconstruction and repairing [AUGA05], and grid generation [Lis04].

We deal with a planar parameterization for a triangle mesh approximating a smooth surface, a bijective mapping between the mesh and a triangulation of a planar polygon. An excellent survey of recent advances in mesh parameterization is given in [FH04], see also references therein. While various algorithms are developed for mesh parameterization approaches based on solid mathematical theories (e.g., conformal mappings), effective computational schemes for generating low-stretch mesh parameterization [SSGH01] have not yet been proposed. Generating mesh parameterization with low distortion measured via the stretch error of [SSGH01] and similar quasi-isometry type error metrics [SGSH02, TSS$^+$04, ZMT05] is important in many applications. Besides the mesh parameterization procedures of [SSGH01, SGSH02] often generate regions of high anisotropic stretch, consisting of slim triangles. Such the regions on a parameterized and textured mesh look like cracks and we call them *parameter cracks*. The left image of Figure 1.5 demonstrates an appearance of such parameter cracks on the textured Mannequin Head model parameterized by the stretch minimization method from [SSGH01].

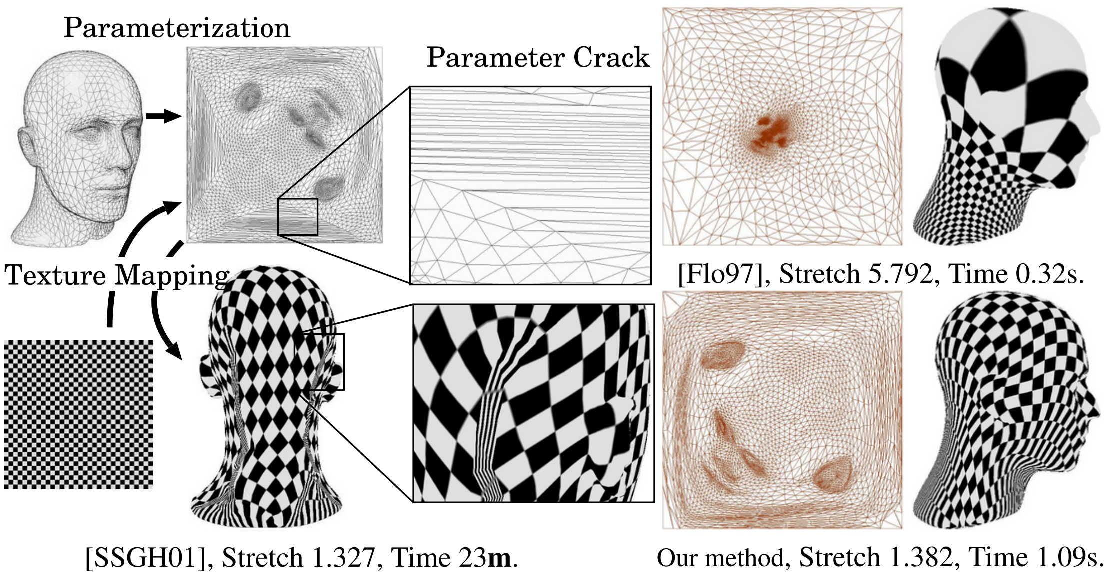


Figure 1.5: Fast low-stretch parameterization. Left: parameter cracks on textured Mannequin Head model parameterized by the stretch minimization method of Sander et al. [SSGH01]. Top-Right: a quasi-conformal parameterization by Floater [Flo94]. Bottom-Right: our fast low-stretch parameterization.

In Chapter 5, we follow [YBS04, YBS05b] and propose to use a moving mesh approach which resembles a popular grid adaption technique in computational mechanics. Our approach is used for generating low-stretch mesh parameterizations. Instead of minimizing nonlinear stretch distortions directly, we equalize the local stretch distribution over the parameter domain by optimizing the parameterization gradually. At each improvement step, we optimize the parameterization generated at the previous step by minimizing a weighted quadratic energy where the weights are chosen in order to minimize the parameterization stretch. This optimization procedure does not generate triangle flips if the boundary of the parameter domain is a convex polygon. Moreover already the first optimization step produces a high-quality mesh parameterization. We compare our parameterization procedure with several state-of-art mesh parameterization methods and demonstrate its speed and high efficiency in parameterizing large and geometrically complex models. Our method is significantly faster than the conventional low-stretch parameterization schemes [SSGH01, SGSH02], and does not generate parameter cracks because of our stretch equalizing strategy. Figure 1.5 shows the parameterized meshes of the Mannequin Head model via conventional schemes (Left and Top-Right images) and our method (Bottom-Right image).

We also propose a novel remeshing scheme based on two parameterizations which equipped with different mapping characteristics such as a low-stretch map for sampling new vertices and a quasi-conformal map for triangulation of the sampled vertices.

1.5 Free-Form Skeleton-driven Mesh Deformations

Generating natural-looking deformations of complex shapes has multiple applications in CAGD, computer animation, and geometric modeling. Since the pioneering works [Bar84, SP86], developing fast, efficient, and intuitive methods for local and global free-form shape deformations is a subject of intensive research. See, for example, recent works [BPGK06, vFTS06, HSL+06]. Recently skeleton-based global shape deformations drew considerable attention [LCF00, SK00b, CGC+02] because they are well-suited for large-scale shape deformations and, therefore, can be used in numerous applications in the computer game and digital movie industries.

Bloomenthal [Blo02] proposed to use the medial axis of Blum [Blu67] as intermediate control interface in order to obtain natural-looking deformations by preserving original shape thickness (distance to the medial axis). In Chapter 6, we follow [YBS03, YBS06c, YBS06a] and present new schemes for free-form skeleton-driven global mesh deformations. First a skeletal mesh, a Voronoi-based approximation of the medial axis, is extracted from a given mesh. Next the skeletal mesh is modified by free-form deformations. Then a desired global shape deformation is obtained by reconstructing the shape corresponding to the deformed skeletal mesh. We develop mesh fairing procedures allowing us to avoid possible global and local self-intersections of the reconstructed mesh.

Figure 1.6 represents our basic free-form skeleton-driven mesh deformation process described in Section 6.3. In Section 6.5, the reconstructing and fairing procedures are extended to a variational approach called *discrete differential coordinates* [Sor05]. We combine a skeleton-driven mesh deformation technique with discrete differential coordinates in order to create natural-looking global shape deformations. In particular, our variational skeleton-driven deformation framework works well for bending, twisting, and other complex large-scale deformations. Finally, using a multiresolution surface representation [LMH00] improves the speed and robustness of our approach. The resulting deformations via the variational extension described in Section 6.5 are demonstrated in Figure 1.7.

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

Figure 1.6: A free-form skeleton-driven mesh deformation. (a): The original hand mesh, its skeletal mesh, and control points to be used to deform the skeletal mesh. (b): A deformed skeletal mesh. (c): Folds and protrusions are observed in the deformed mesh. (d): The folds and protrusions are removed by the mesh evolutions proposed in Section 6.3.1; however global and local self-intersections are still presented. (e): The global and local self-intersections are eliminated by our fairing scheme proposed in Section 6.3.2.

Figure 1.7: Examples of variational skeleton-driven mesh deformations.

2

Similarity-based Mesh Denoising

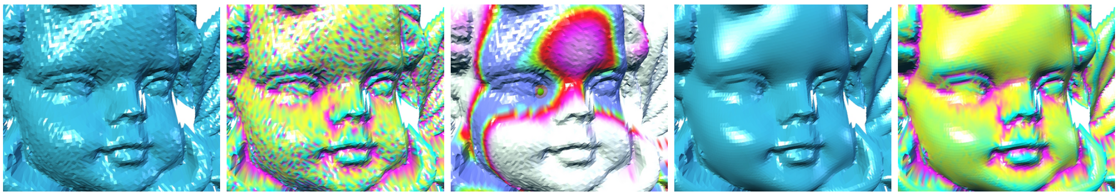
Recent advances in digital recording technologies dramatically increase the use of digitized real-world signals which usually contain noise. Consequently, developing denoising methods has been an active and important area of research.

In signal and image processing, denoising techniques based on a Fourier analysis and its extensions (Wavelets) [SN96] and PDEs [Wei98] are popular and well studied. These techniques are nicely adapted for processing regular structures as images. See [BCM05b] for a recent review of image denoisng methods. In geometric modeling, we usually deal with irregular data such as polygonal meshes and point clouds. Therefore, new ideas and approaches are required for efficient denoising of irregular data.

Since seminal works of Taubin [Tau95, Tau01] and Desbrun et al. [DMSB99], many mesh smoothing techniques have been proposed in computer graphics and geometric modeling. Recent advances in developing feature preserving smoothing techniques include diffusion-driven methods [TWBO03, HP04, LP05], projection-based approaches [FCOS05, OBA05], and the so-called bilateral mesh filtering schemes [FDCO03, JDD03, CT03]. The latter were inspired by image processing techniques based on spatial-tonal normalized convolutions [Weu94, AW95, SB97, TM98] which in their turn can be considered as generalizations of the Yaroslavsky neighborhood filter [Yar85].

Very recently, the so-called *Non-Local means* (or *NL-means*) concept, a natural and elegant extension of the image bilateral filtering paradigm, was proposed by Buades, Coll, and Morel [BCM05a, BCM05b, BCM06]. The NL-means method was inspired by the famous Texture-Synthesis-by-Example approach of Efros and Leung [EL99]. The method and its applications and extensions are currently a subject of intensive research in image and video processing [KOJ05, MS05]. The basic idea behind NL-means is very simple: for a given pixel, its new (denoised) intensity value is computed as a weighted average of the other image pixels with weights reflecting the similarity between local neighborhoods of the pixel being processed and the other pixels. A similar idea was independently proposed in [BM05] where it was used for video enhancement purposes.

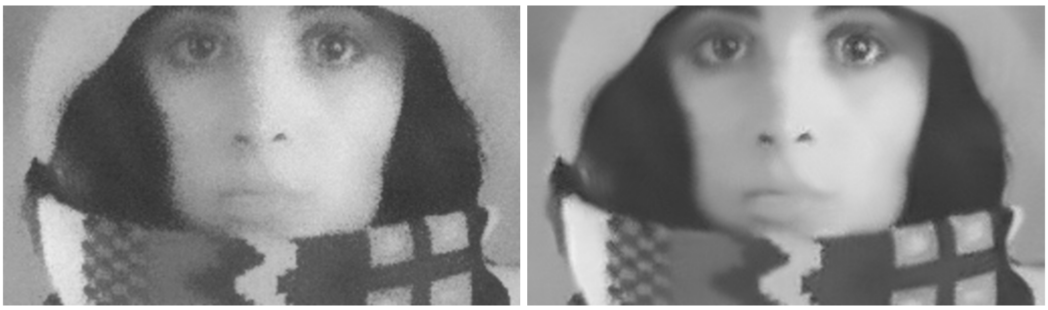
In this Chapter, we follow [YBS06b] and present a new mesh smoothing method based on the NL-means concept. The developed method has a number of important advantages over the main state-of-the-art mesh denoising techniques. Since only vertex positions and corresponding normals are used in our denoising procedure, our method can be applied for not only watertight meshes but also triangle soups and point clouds with normals. Fig. 2.1 illustrates the idea and potential of our NL-means mesh smoothing method. We also suggest a new scheme for comparing different mesh/soup denoising methods.



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

Figure 2.1: Denoising Angel model with Non-Local means. (a): Original noisy mesh (flat-shading is used). (b): Original noisy mesh colored by mean curvature; the curvature map helps us to identify surface defects and roughness. (c): Coloring by similarity. A mesh vertex is chosen at the left corner of the right eye of the original mesh. The mesh is colored according to a similarity with the shape of the model at the chosen vertex. The similarity increases from white to blue. The vertices with higher similarity values have a stronger contribution to the new (smoothed) position of the chosen vertex. (d): Mesh is smoothed by the similarity-based method developed in this thesis (flat-shading is used). (e): Smoothed mesh colored by mean curvature; the curvature map indicates high quality of the smoothed surface.

(a) (b) (c)



(d) (e)

Figure 2.2: (b): "Trui" image corrupted by noise. (d): Smoothed by bilateral filtering; (a): The difference between the original noisy and smoothed images contains important image structures. (e): Smoothed by NL-means filter of Buades, Coll, and Morel; (c): The difference between the noisy and NL-smoothed images contains much less features of the original image. Thus the NL-means filter does a much better denoising job than the bilateral filter.

2.1 Image Filtering with NL-means.

Consider a gray-scale image $I(\mathbf{x})$ defined over a bounded domain Ω (which is usually a rectangle). The NL-means filter is defined by

$$J(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \int_{\Omega} w(\mathbf{x}, \mathbf{y}) I(\mathbf{y})\, d\mathbf{y}, \tag{2.1}$$

where the convolution kernel $w(\mathbf{x}, \mathbf{y})$ is given by

$$\exp\left\{ -\frac{1}{h^2} \int G_a(|\mathbf{t}|)\, |I(\mathbf{x}-\mathbf{t}) - I(\mathbf{y}-\mathbf{t})|^2\, d\mathbf{t} \right\}, \tag{2.2}$$

and measures a similarity between neighborhoods of pixels $\mathbf{x}$ and $\mathbf{y}$, $C(\mathbf{x}) = \int_{\Omega} w(\mathbf{x}, \mathbf{y})\, d\mathbf{y}$ is a normalizing factor, and $G_a(\cdot)$ is a Gaussian kernel of standard deviation a. Here h and a are filtering parameters.

In practice, integration in (2.2) is performed over a a fixed-size small window. The typical window size varies from 5×5 to 9×9.

A pictorial explanation of the NL-means method is given in Fig. 2.3.

While the NL-means method is slow, it substantially outperforms the bilateral scheme and other similar filters. The advantages of the NL-means method are especially manifested by processing images with complex texture patterns. We compare the NL-means and bilateral filters in Fig. 2.2.

Buades, Coll, and Morel also suggested a simple and convenient technique for evaluating the quality of image smoothing methods [BCM05a, BCM05b]. The idea is to consider and visualize the difference between the original noisy image $I(\mathbf{x})$ and its smoothed version $J(\mathbf{x})$. If the difference $I(\mathbf{x}) - J(\mathbf{x})$ does not contain geometric structures of the original image $I(\mathbf{x})$ and looks like a random signal, one can conclude that the tested smoothing method removes noise and do not destroy image features. (Of course, similar SNR-based techniques are widely used in image processing, see, for example, [GSZ05, MN03] and references therein.) In Fig. 2.2, we apply the Buades et al. image difference technique to demonstrate that the NL-means filter substantially outperforms bilateral filtering in preserving salient image structures.

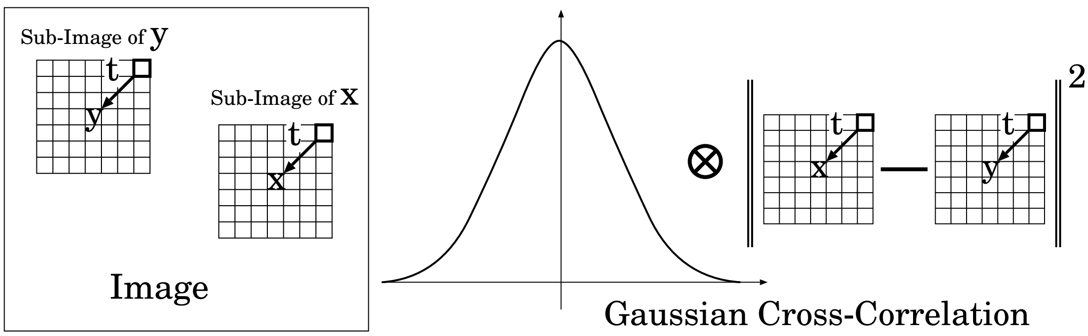


Figure 2.3: We measure similarity $w(\mathbf{x}, \mathbf{y})$ between two image windows centered at $\mathbf{x}$ and $\mathbf{y}$ by convolving the squared difference between the windows with a Gaussian kernel.

2.2 Mesh Filtering with NL-means

Given a triangle mesh $\mathcal{M}$, consider a mesh vertex $\mathbf{x}$ and denote by $\Omega_\sigma(\mathbf{x})$ the 2σ-neighborhood of $\mathbf{x}$ on $\mathcal{M}$: $\Omega_\sigma(\mathbf{x}) = \{\mathbf{y} \in \mathcal{M} : |\mathbf{x} - \mathbf{y}| \leq 2\sigma\}$. We use bilateral mesh smoothing flow of [FDCO03] as a basis of our method and denoise $\mathcal{M}$ by updating repeatedly the position of each mesh vertex $\mathbf{x}$:

$$\mathbf{x}^{n+1} = \mathbf{x}^n + k(\mathbf{x}^n)\mathbf{n}_{\mathbf{x}}^n, \tag{2.3}$$

where $\mathbf{n}_\mathbf{x}$ is the unit normal at $\mathbf{x}$,

$$k(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \int\limits_{\Omega_{\sigma_2}(\mathbf{x})} w(\mathbf{x}, \mathbf{y}) I(\mathbf{y})\, dS_{\mathbf{y}}, \tag{2.4}$$

$$C(\mathbf{x}) = \int\limits_{\Omega_{\sigma_2}(\mathbf{x})} w(\mathbf{x}, \mathbf{y})\, dS_{\mathbf{y}}, \tag{2.5}$$

$$I(\mathbf{y}) = \langle \mathbf{n}_\mathbf{x}, \mathbf{y} - \mathbf{x} \rangle, \tag{2.6}$$

$$w(\mathbf{x}, \mathbf{y}) = \exp\left\{-\, D(\mathbf{x}, \mathbf{y}) / (2\sigma_1^2)\right\}. \tag{2.7}$$

Here $S_\mathbf{y}$ stands for the area element of $\mathcal{M}$ at $\mathbf{y}$, $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$, and $D(\mathbf{x}, \mathbf{y})$ is a similarity kernel.

Similarity Kernel. The main difficulty of extending the NL-means approach to meshes consists of defining an appropriate similarity kernel $D(\mathbf{x}, \mathbf{y})$.

Consider mesh vertices $\mathbf{w} \in \Omega_{\sigma_3}(\mathbf{x})$, $\mathbf{z} \in \Omega_{\sigma_3}(\mathbf{y})$, and $\mathbf{y} \in \Omega_{\sigma_2}(\mathbf{x})$ as shown in the left-top image of Fig. 2.4.

First we choose a pair of unit tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ in the tangent plane of each mesh vertex $\mathbf{x}$ (the tangent plane at mesh vertex $\mathbf{x}$ is the plane passing through $\mathbf{x}$ and orthogonal to mesh normal $\mathbf{n}_\mathbf{x}$). Let us define a translation vector $\mathbf{t}$, a mesh counterpart of the image translation vector $\mathbf{t}$ in (2.2), by

$$\mathbf{t} = -\,(u_\mathbf{z}, v_\mathbf{z}) = -\,(\langle \mathbf{t}_1, \mathbf{z} - \mathbf{y} \rangle, \langle \mathbf{t}_2, \mathbf{z} - \mathbf{y} \rangle).$$

Now let use radial basis functions (RBFs) to build a local approximation of the mesh in a neighborhood of $\mathbf{x}$. Let $(u_\mathbf{w}, v_\mathbf{w}, w_\mathbf{w})$ be the local coordinates of mesh vertex $\mathbf{w}$ w.r.t the basis $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n}_\mathbf{x})$. The local RBF approximation near $\mathbf{x}$ is given by

$$F_\mathbf{x}(u, v) = p(u, v) + \sum_{\mathbf{w} \in \Omega_{\sigma_3}(\mathbf{x})} \lambda_\mathbf{w} \Phi(\sqrt{u^2 + v^2}), \tag{2.8}$$

where $\Phi(\rho) = \rho^2 \log(\rho)$, $p(u, v)$ is a linear polynomial and RBF coefficients $\{\lambda_\mathbf{w}\}$ are obtained by solving a system of linear equations

$$F_\mathbf{x}(u_\mathbf{w}, v_\mathbf{w}) = w_\mathbf{w}, \qquad \sum_{\mathbf{w} \in \Omega_{\sigma_3}(\mathbf{x})} \lambda_\mathbf{w} p(u_\mathbf{w}, v_\mathbf{w}) = 0.$$

We approximate $I(\mathbf{x} - \mathbf{t})$ corresponding to $I(\mathbf{y} - \mathbf{t})$ by $F_\mathbf{x}(u_\mathbf{z}, v_\mathbf{z})$, as seen in Fig. 2.4. Finally we define the similarity kernel $D(\mathbf{x}, \mathbf{y})$ by

$$D(\mathbf{x}, \mathbf{y}) = \int\limits_{\Omega_{\sigma_3}(\mathbf{y})} G_{\sigma_3}(|\mathbf{t}|)\, |F_\mathbf{x}(u_\mathbf{z}, v_\mathbf{z}) - I(\mathbf{y} - \mathbf{t})|^2\, d\mathbf{t}, \tag{2.9}$$

where $I(\mathbf{y} - \mathbf{t}) = \langle \mathbf{n}_\mathbf{x}, \mathbf{z} - \mathbf{x} \rangle$ and $G_\sigma(\cdot)$ is a Gaussian kernel.

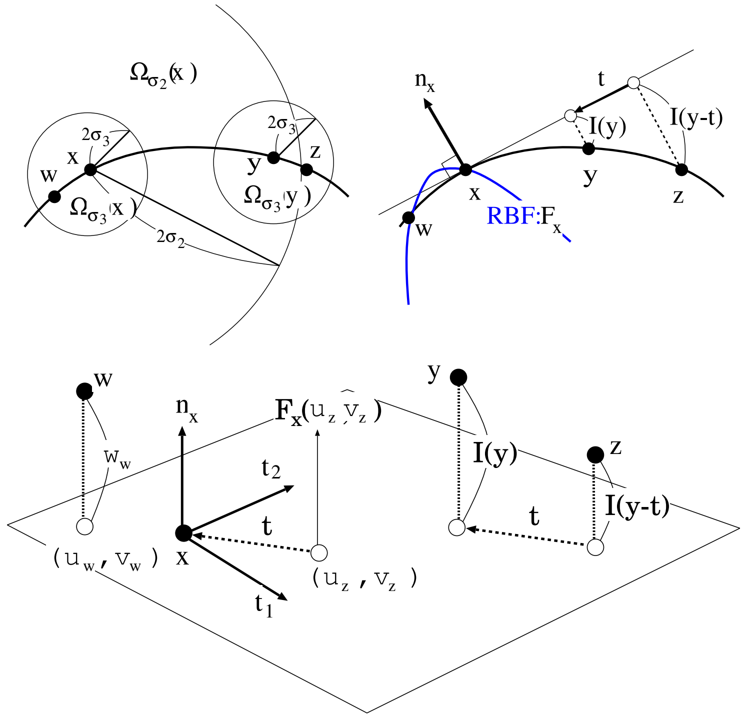


Figure 2.4: Neighbor and local coordinates for RBF.

2.3 Results and Discussion of Mesh Denoising

In our numerical experiments, we use gcc 3.3.5 C++ compiler on a 1.7GHz Pentium 4 computer with 1GB of RAM. We use the N. Max weights [Max99] for computing the mesh normals.

Parameters. Four user-specified parameters are used in our method:

1. σ_1, the standard deviation of the similarity kernel (2.7);
2. σ_2 the size of the integration domain in (2.4) and (2.5);
3. σ_3, the size of the similarity domain in (2.9);
4. n, the number of iterations of (2.3).

Similar to [JDD03], we make the parameters σs proportional to the average edge length e of the evolving mesh $\mathcal{M} = \mathcal{M}^n$:

$$\sigma_i = \eta_i e, \quad i = 1, 2, 3.$$

Ideally σ_1 represents the noise deviation, therefore similar to [FDCO03] it could be chosen as a standard deviation of the heights of vertices $\mathbf{y}$ for either a user-specified flat region or an average standard deviation of an entire mesh. The other two coefficients η_2 and η_3 are constant for the most of models, similar to the image case [BCM05b, BCM06, KOJ05]. According to our experiments, setting $\eta_2 = \{1.0, 2.0\}$ and $\eta_3 = \{0.75, 1.0\}$ leads to good results.

Quality Evaluation and Comparison. We have implemented three recent state-of-the-arts mesh denoising techniques: the Anisotropic Mean Curvature Flow (AMCF) [HP04] and Bilateral Mesh Filters [FDCO03] and [JDD03]. In our implementation of AMCF, the weight for ij edge is given by $G_\sigma(k_{ij})h_{ij}$, where h_{ij} is a cotangent-based weight associated with ij and k_{ij} is a directional curvature [LP05]. In our experiments, for both these methods we try to choose parameter settings producing the best results.

We use two visualization schemes to compare the techniques with our method. The first scheme consists of coloring by the mean curvature. The second scheme measures the difference between the original and smoothed meshes. More precisely, we visualize the differences in the positions of the corresponding vertices of the meshes $|\mathbf{x}_k^{\text{noisy}} - \mathbf{x}_k^{\text{smoothed}}|$.

We use three models in our comparison: a noisy Fandisk model (Fig. 2.5), a noisy Dragonhead model (Fig. 2.6), and the Angel model (Fig. 2.10). For these models, Table 2.1 presents timing results and parameter settings used for our method and our implementations of methods of [HP04] and [FDCO03].

Fig.	Method	n	η_1	η_2	η_3	Time
2.5	[HP04]	3	10	25		1.2s
	[FDCO03]	3	0.25	1	1	0.8s
	our	3	0.4	1	0.75	13.2s
2.9	[HP04]	1	2×10^4	100		14.7s
	[FDCO03]	2	1.5	4	1	126s
	our	3	0.35	1	0.75	606s
2.10	[DMSB99]	2	0.15			2.7s
	[JDD03]	2	0.25	1	0.75	16.4s
	[BO01]	10	10.0	2	5	134s
	[HP04]	1	100	25		1.67s
	[FDCO03]	2	0.25	1	0.75	3.7s
	our	2	0.25	1	0.75	64.5s

Table 2.1: Parameter setting and timing results. Here n stands for the number of iterations. For MCF [DMSB99], the step-size parameter is equal to $\eta_1 e$. For nonlinear normal diffusion [BO01], the step-size parameter is equal to $\eta_1 e$, $\eta_2 e$ and $\eta_3 e$ denote the spatial size of summing normals and the size σ of the Gaussian kernel, respectively. For AMCF [HP04], $\eta_1 e$ and $\eta_2 e$ denote the step-size (implicit scheme) and the size σ of the Gaussian kernel, respectively. For bilateral filterings [JDD03] and [FDCO03], the deviation of the hight Gaussian kernel is equal to $\eta_1 e$, the integration domain size is given by $\eta_2 e$, and the deviation of the spatial Gaussian kernel is set equal to $\eta_3 e$. Here e denotes the average edge length of the evolving mesh $\mathcal{M} = \mathcal{M}^n$.

As seen in Fig. 2.8 our method outperforms its rivals in restoring sharp edges and low-curvature regions. In addition, the max-norm and average errors produced by the method and measured w.r.t. the original clean Fandisk mesh are substantially smaller than those of the Anisotropic Mean Curvature Flow [HP04] and Bilateral Mesh Filter [FDCO03]. Fig. 2.9 demonstrates that our method delivers the best performance according the entropy of the differences between the original (noisy) and smoothed models. It also indicates that the method preserves fine geometric features better than two its competitors. Fig. 2.10 shows that our method produces lowest oversmoothing to compare with the five other smoothing techniques.

Finally in Fig. 2.11 we demonstrate how our method handles triangle soups. Denoising a complex Gargoyle model (about 98 K triangles) by our method is rather slow (five iterations took 31 minutes) but the result is worth seeing.

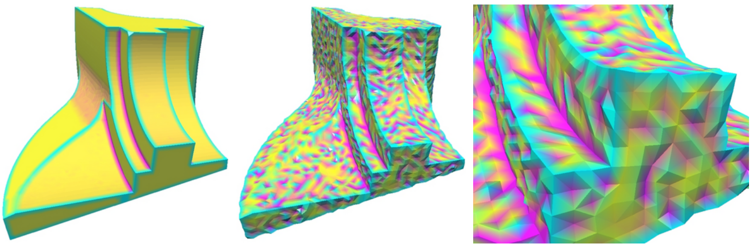


Figure 2.5: Left: initial Fandisk model colored by mean curvature. Center and Right: noisy Fandisk (Gaussian noise with $\sigma = 0.1e$ is added).

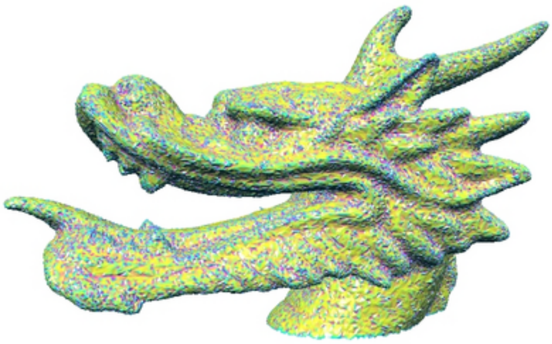


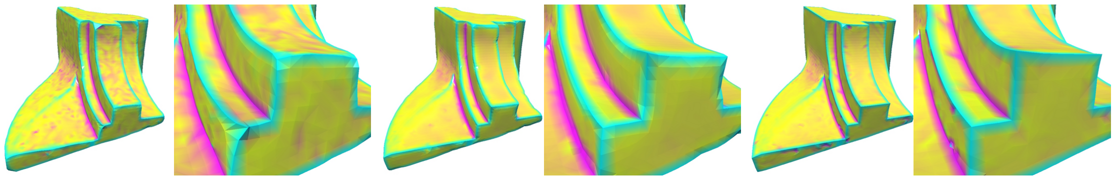
Figure 2.6: Noisy Dragon-head model (Gaussian noise with $\sigma = 0.2e$ is added) from [JDD03] is colored by mean curvature.

Figure 2.7: Left: mean curvature profile palette. Right: this palette is used for visualizing the differences in vertex positions of noisy and smoothed meshes.

Complexity. The average computational complexity of our method is given by $O(V_\mathbf{y} V_\mathbf{w} V_\mathbf{z} V + V \log V)$ where V is the number of vertices of $\mathcal{M}$, $V_\mathbf{y}$, $V_\mathbf{w}$, and $V_\mathbf{z}$ are the average numbers of vertices of local patches $\Omega_{\sigma_2}(\mathbf{x})$, $\Omega_{\sigma_3}(\mathbf{x})$, and $\Omega_{\sigma_3}(\mathbf{y})$. Retrieving $2\sigma_2$-neighborhood of $\mathbf{x}$ requires $O(\log V)$ operations by using a kd-tree, and evaluating the similarity kernel (2.9) is done using $O(V_\mathbf{w} V_\mathbf{z})$ operations for each pair $\mathbf{x}$ and $\mathbf{y}$.

At the first glance, $O(V_\mathbf{y} V_\mathbf{w} V_\mathbf{z} V + V \log V)$ looks too large. However $V_\mathbf{y}$, $V_\mathbf{w}$, $V_\mathbf{z}$ are the number of vertices in local neighborhoods of mesh vertices $\mathbf{y}$, $\mathbf{w}$, $\mathbf{z}$ used in our method. For a typical uniformly dense mesh, we have $V_\mathbf{y} \approx 20\eta_2$ and $V_\mathbf{w} \approx 20\eta_3 \approx V_\mathbf{z}$. If η_2 is large, a fast implementation of RBFs [BN92] should be used.

Although the influence of each parameter σ_1, σ_2 and σ_3 is clear, an optimal selection of all of them is not trivial. Further work is required for a deeper understanding correlations between these parameters.



Ave error = 0.021 | Ave error = 0.0067 | Ave error = 0.0048

Max error = 0.11 | Max error = 0.069 | Max error = 0.03

Figure 2.8: Smoothing noisy Fandisk model ($V = 6474$, $F = 12944$). Mean curvature coloring enhances surface defects and roughness of the smoothed meshes which can not be recognized by human eyes if we use a flat/smooth shading. Left: Anisotropic Mean Curvature Flow [HP04] is used. Middle: Bilateral Mesh Filter [FDCO03] is applied. Right: our method is employed.

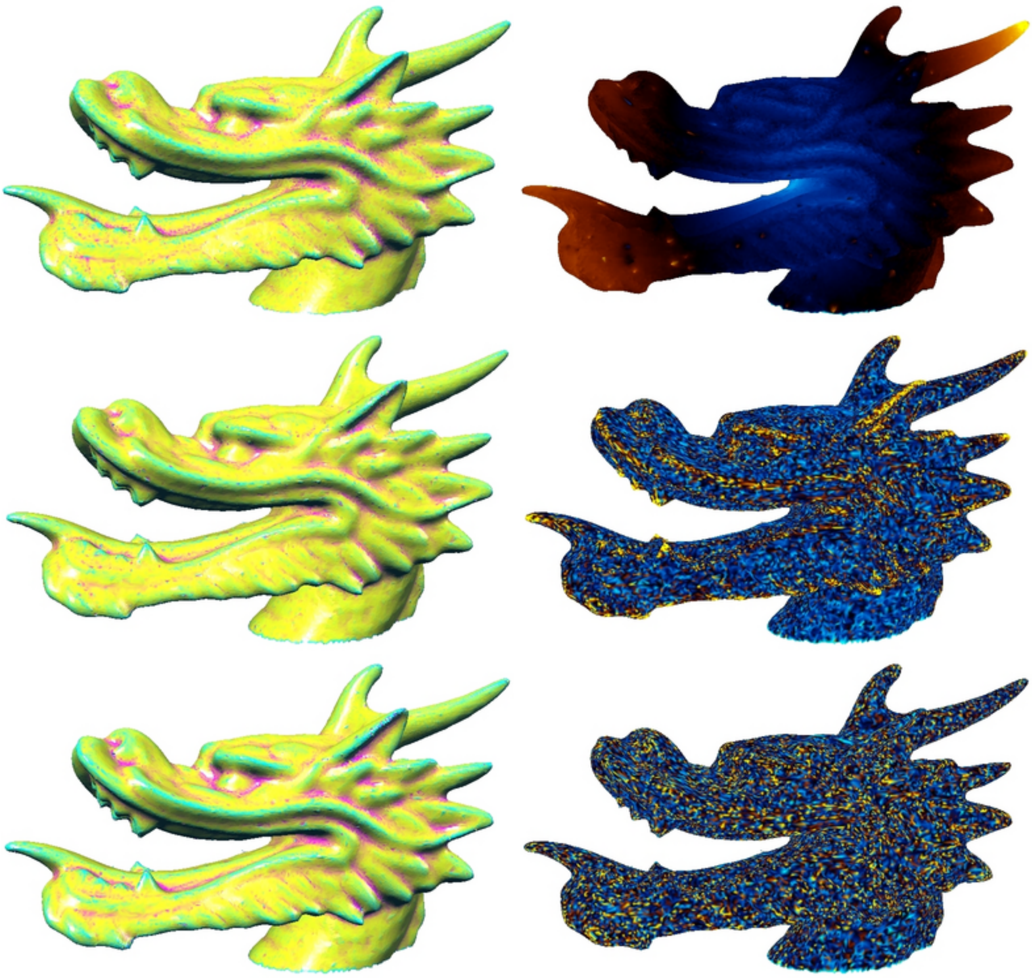
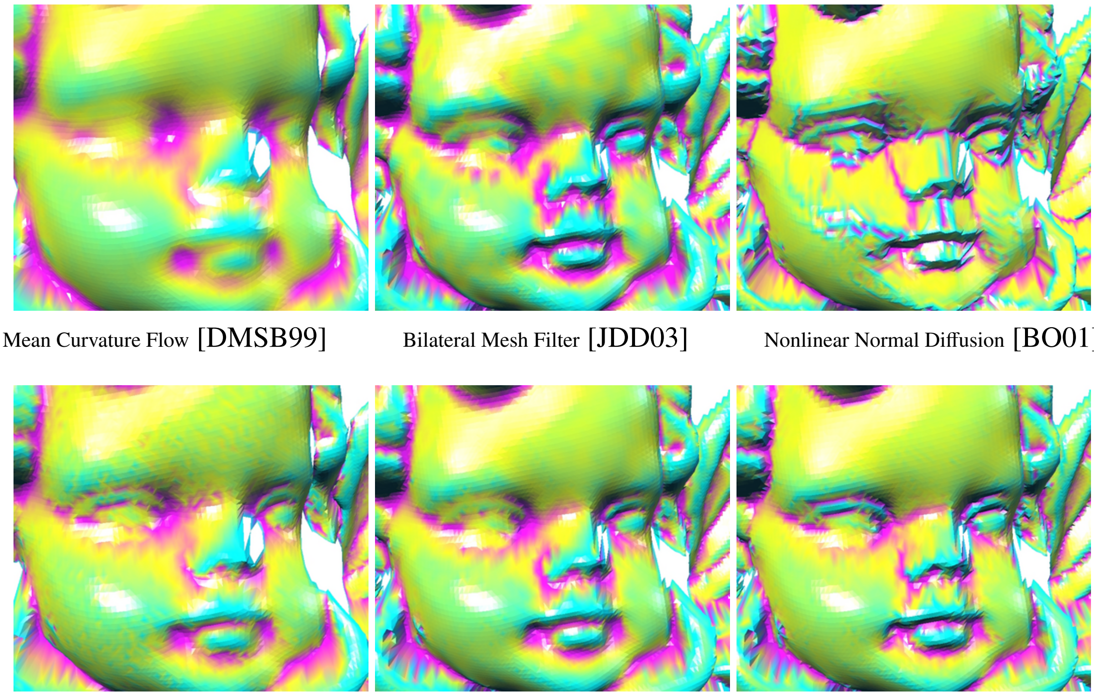


Figure 2.9: Smoothing noisy Dragon-head model ($V = 100056$, $F = 199924$). Top: Anisotropic Mean Curvature Flow [HP04] is used. Middle: Bilateral Mesh Filter [FDCO03] is applied. Bottom: our method is employed. Left: coloring by mean curvature indicates that our method outperforms its rivals in preserving fine surface features. Right: our method delivers the best performance according the entropy of the difference between the original (noisy) and smoothed models.



Mean Curvature Flow [DMSB99] Bilateral Mesh Filter [JDD03] Nonlinear Normal Diffusion [BO01]

Anisotropic Mean Curvature Flow [HP04] Bilateral Mesh Filter [FDCO03] Our Method [YBS06b]

Figure 2.10: Smoothing noisy Angel model ($V = 24566$, $F = 48090$). Our method produces lowest oversmoothing to compare with five other smoothing techniques.

2.4 Summary of Mesh Denoising

We have extended the recent NL-means image filtering approach [BCM05a, BCM05b, BCM06] to the 3D meshes and triangle soups approximating piecewise smooth surfaces. The extension is far from being straightforward, since the original NL-means approach relies heavily on the image structure regularity. We think we have found a simple and elegant solution to the problem by employing local RBF approximations.

Recently semi-local similarity-based shape descriptors received a considerable attention in connection with shape matching, retrieval, and modeling applications [BIT04, GCO05, GGGZ05, SACO04, ZG04a] which are too expensive for practical mesh smoothing. The local RBF approach we use in this Chapter is much simpler.

We have demonstrated that our method outperforms other recent state-of-the-art smoothing techniques which are among best up-to-date mesh denoising schemes.

Finally we have suggested a new way to compare different mesh/soup denoising methods. We believe that statistical analysis (entropy measurements, etc.) of the difference between the original (noisy) and smoothed datasets will lead to developing new surface denoising techniques and new principles for a fair comparison of existing ones.

The source code of our method is available on the Web for evaluation [YBS06b].

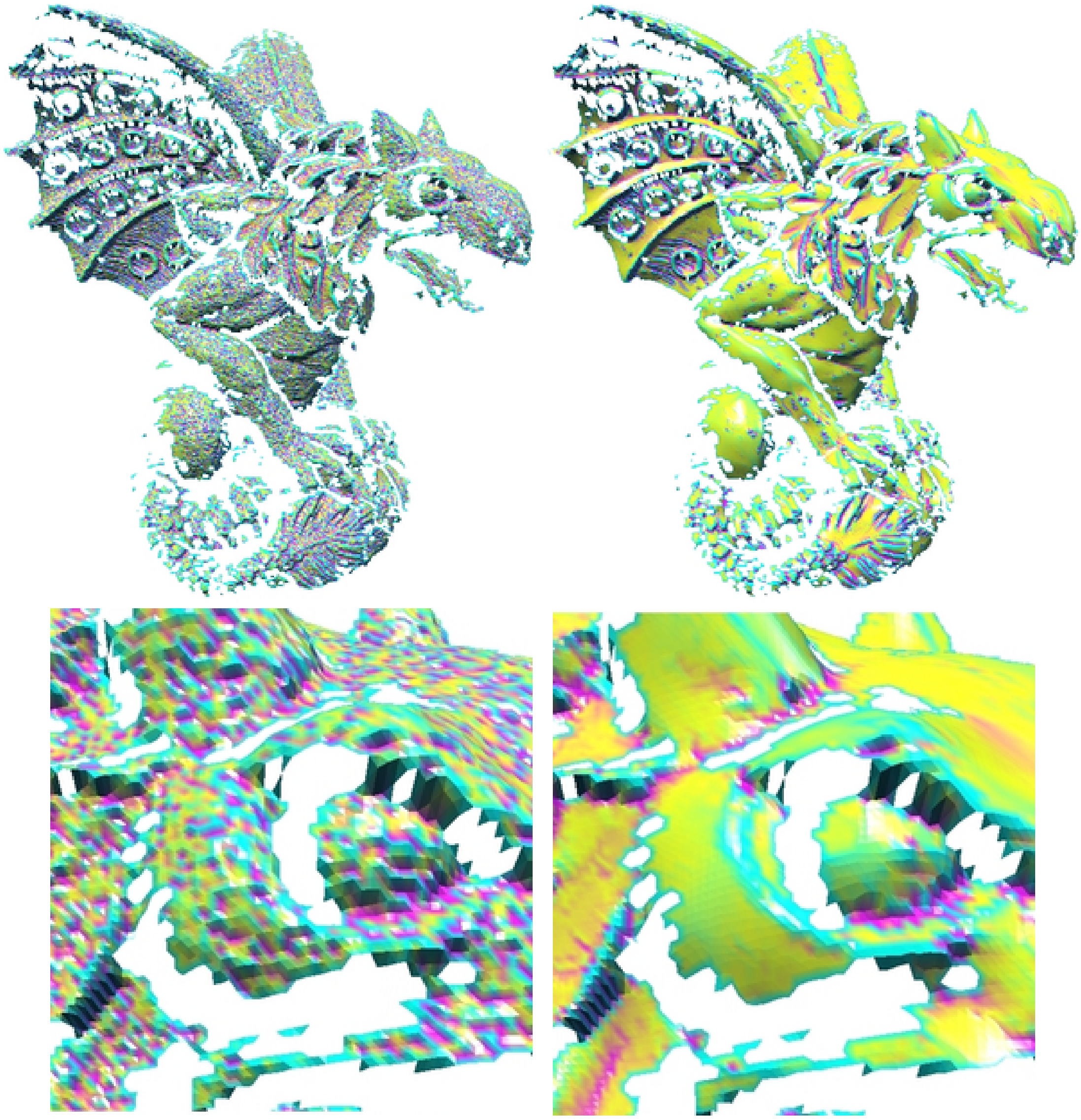
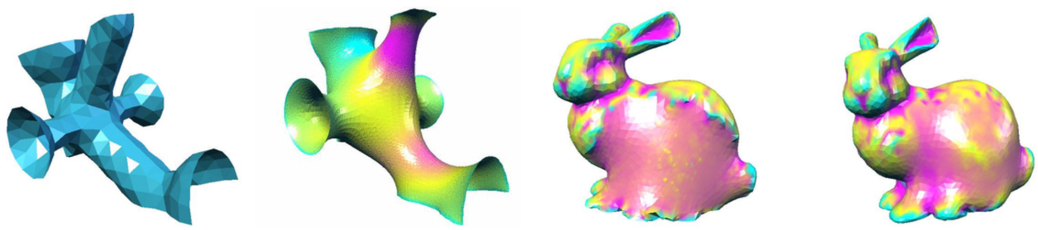


Figure 2.11: Denoising a complex Gargoyle model ($V = 54907$, $F = 97769$) by our method with $\{\eta_1, \eta_2, \eta_3\} = \{0.28, 2, 1\}$. Left: original data colored by mean curvature. Right: smoothed data colored by mean curvature; noise is gently removed and fine geometric features are accurately preserved.

3

Fair Mesh Generation via Elastica



(a) (b) (c) (d)

Figure 3.1: Generating fair triangle meshes with discrete elastica. (a): An initial mesh outlined a complex tubular object. (b): A discrete elastica surface (mesh) obtained from the initial mesh. (c): Bunny model with a large part of the mesh removed and then triangulated. (d): The modified part of the Bunny is restored as a discrete elastica. Coloring by the mean curvature is used to demonstrate a high quality of the generated meshes.

Variational shape fairing consists of generating shapes satisfying certain aesthetic requirements. It is usually achieved via minimization of fairness measures penalizing large curvature values and curvature oscillations [MS92, Gre94, WW92, WW94, SK01, CDD$^+$04]. See also recent works [YB02, BS05, XPB06] and references therein for fair shape generation via geometric surface flows. A popular surface fairing measure used in various computer graphics and geometric modeling applications is the so-called *total curvature* functional [HKS92]

$$\iint (k_{\max}^2 + k_{\min}^2)\, dA \tag{3.1}$$

Here $k_{\max}$ and $k_{\min}$ are the surface principal curvatures, and dA is the surface area element. The total curvature (3.1) approximates the elastic bending energy of a thin plate [HKS92]. Let us call the surfaces minimizing (3.1)

$$\iint (k_{\max}^2 + k_{\min}^2)\, dA \to \min \tag{3.2}$$

elastica surfaces because they generalize the elastica curves [Eul44] of *Leonhard Euler* (1707 - 1783). See also [BHN96] for a good literature review and for a very effective method to approximate the elastica curves by polylines.

The Euler-Lagrange equation corresponding to (3.2) is given by

$$\triangle_{\mathbf{S}}(H) + 2H(H^2 - K) = 0, \tag{3.3}$$

where H and K are the mean and Gaussian curvatures, respectively, and $\triangle_{\mathbf{S}}(\cdot)$ is the Laplace-Beltrami operator introduced by *Eugenio Beltrami* (1835 - 1900) [Str88][page 160]. See [GH96][pages 82-85] for a derivation of (3.3).

In this Chapter, we represent an approach for approximating elastica surfaces by triangle meshes. Our approach to minimize the total curvature functional (3.1) can be considered as a combination of the steepest descent method for (3.2) with finite differencing (approximating a smooth surface by a triangle mesh). A preliminary version of the approach was developed in [Yos01].

Consider a family of smooth surfaces $\mathcal{S}(t, u, v)$, where u, v parameterize the surface and t parameterizes the family. We suppose t to be independent of u, v. Let us assume that the family evolves according to the following evolution equation

$$\frac{\partial \mathcal{S}(t, u, v)}{\partial t} = F\,\mathbf{N}, \qquad \mathcal{S}(0, u, v) = \mathcal{S}^{(0)}(u, v), \tag{3.4}$$

where $\mathbf{N}(t, u, v)$ is the unit normal vector for $\mathcal{S}(t, u, v)$, F is a speed function. The family parameter t can be considered as the time duration of the evolution. The gradient-descent flow, also called Willmore flow [BS05], for (3.2) is given by (3.4) with

$$F \equiv -\triangle_{\mathbf{S}}(H) - 2H(H^2 - K). \tag{3.5}$$

If a surface evolved by (3.4), (3.5) converges to a limit surface $\mathcal{S}(\infty, u, v)$, as $t \to \infty$, then it is an elastica since the Euler-Lagrange equation (3.3) is satisfied for that limit surface. In [HKS92] the Willmore flow was applied to closed triangulated polygonal surfaces via the surface evolver [Bra92].

We approximate the evolution (3.4), (3.5) by a discrete evolution of triangle meshes and use discrete analogues of the Laplace-Beltrami operator and Gaussian and mean curvatures.

One of the important contributions of our method consists of adding to a discrete version of (3.4) a special tangent speed component used to improve the quality of the evolving mesh and to increase computational stability.

Figure 3.1 illustrates how our method can be used in various geometric modeling applications. The two left images demonstrate an initial triangle mesh approximating a tubular object and a discrete elastica obtained from that initial mesh by a discrete approximation of (3.4), (3.5). The two right images show how a large missed part of a complex mesh (Bunny) can be restored by a discrete elastica surface. Coloring by the mean curvature demonstrates a high quality of the generated meshes.

3.1 Discrete Willmore Flow

To solve (3.4) numerically, we first approximate the time derivative term in (3.4) by its forward difference approximation

$$\frac{\partial \mathcal{S}(t, u, v)}{\partial t} \approx \frac{\mathcal{S}(t + \tau, u, v) - \mathcal{S}(t, u, v)}{\tau}, \quad \tau \ll 1.$$

Thus we approximate (3.4) by a discrete evolution process

$$\mathcal{S}(t + \tau, u, v) = \mathcal{S}(t, u, v) + \tau F \mathbf{N}(t, u, v), \tag{3.6}$$

where the speed function F is defined by (3.5). Then the surface $\mathcal{S}(t, u, v)$ is approximated by a triangle mesh and discrete approximations to the Laplace-Beltrami operator, Gaussian and mean curvatures, and other geometric attributes are considered. Thus the discrete evolution of surfaces (3.6) is approximated by a mesh updating process

$$\mathcal{P}_i^{(k+1)} = \mathcal{P}_i^{(k)} + \tau^{(k)} F_i^{(k)} \mathbf{N}_i^{(k)}, \tag{3.7}$$

where the points $\{\mathcal{P}_i^{(k)}\}$ form a mesh $\mathcal{M}^{(k)}$ obtained after k steps of the process from an initial mesh $\mathcal{M}^{(0)}$ approximating $\mathcal{S}^{(0)}(u, v)$ and $\mathbf{N}_i^{(k)}$ is the unit mesh normal at $\mathcal{P}_i^{(k)}$. Here the unit mesh normal $\mathbf{N}$ at vertex $\mathcal{P}$ is computed as the normalized weighted sum of of the normals of the incident triangles, with weights equal to the areas of the triangles.

Since (3.4), (3.5) is a fourth-order partial differential equation, (the term $\triangle_{\mathbf{S}}(H)$ involves fourth-order surface derivatives) we choose the step-size $\tau^{(k)}$ in (3.7) proportional to the squared area of the smallest triangle of $\mathcal{M}^{(k)}$. More precisely, we set $\tau^{(k)} = A_k^2/150$, where A_k is the minimal triangle area among the all triangles of $\mathcal{M}^{(k)}$.

Tangential Drift for Equalization of Mesh Triangles. Note that (3.7) is similar to an explicit finite difference scheme for a parabolic partial differential equation and, therefore, may be unstable if step-size $\tau^{(k)}$ is not small enough in a comparison with mesh triangles. Thus we can expect that a better stability of the discrete mesh evolution process can be achieved if the mesh triangles which are close to equilateral triangles and have almost the same size.

Our mesh triangle equalization technique consists of adding a tangent speed vector to (3.7). Note that adding a tangent speed component to (3.4) affects only the surface parameterization. Therefore instead of (3.7) we consider

$$\mathcal{P}_i^{(k+1)} = \mathcal{P}_i^{(k)} + \tau^{(k)} F_i^{(k)} \mathbf{N}_i^{(k)} + \epsilon^{(k)} \mathbf{T}_i^{(k)}, \tag{3.8}$$

where $\mathbf{T}_i^{(k)}$ is a vector orthogonal to $\mathbf{N}_i^{(k)}$ and attached at $\mathcal{P}_i^{(k)}$, $\epsilon^{(k)}$ is a small positive parameter.

At an inner mesh vertex $\mathcal{P}$ let us consider the so-called umbrella-operator [Tau95, KCVS98] defined by

$$\mathcal{U}(\mathcal{P}) = \sum_i w_i \overrightarrow{\mathcal{P}Q_i}, \tag{3.9}$$

where summation is taken over all neighbors of $\mathcal{P}$, w_i are positive weights. The geometric idea behind the umbrella operator is illustrated in Figure 3.2.

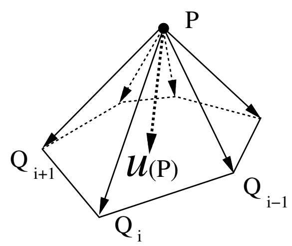


Figure 3.2: Umbrella operator associated with a mesh vertex $\mathcal{P}$ is defined as a weighted average of the neighbor vectors, see (3.9).

In [OBB00] it was proposed to use the tangent component of $\mathcal{U}_0$, the umbrella operator with equal weights, for mesh regularization. The tangent component of the bi-umbrella operator $\mathcal{U}_0^2 = \mathcal{U}_0 \circ \mathcal{U}_0$ was used in [WDSB00] for similar purposes.

Following [Yos01] we use the tangent component of an area weighted bi-umbrella operator $\mathcal{U}^2_{\text{area}}$:

$$\mathbf{T} = -\left[\mathcal{U}^2_{\text{area}} - (\mathcal{U}^2_{\text{area}} \cdot \mathbf{N})\mathbf{N}\right], \tag{3.10}$$

where

$$\mathcal{U}_{\text{area}}(\mathcal{P}) = \frac{1}{2An}\sum_{i=1}^{n} a_i \left(\frac{\overrightarrow{\mathcal{P}Q_i}}{|\overrightarrow{\mathcal{P}Q_i}|} + \frac{\overrightarrow{\mathcal{P}Q_{i+1}}}{|\overrightarrow{\mathcal{P}Q_{i+1}}|} \right),$$

where a_i is the area of the triangle $Q_i\mathcal{P}Q_{i+1}$, n is the number of neighboring vertices for $\mathcal{P}$, $A = \sum_{i=1}^{n} a_i$ is the total area of the triangles adjacent to $\mathcal{P}$.

If $\mathcal{P}$ is a boundary vertex, we set $\mathcal{U}_{\text{area}}(\mathcal{P}) = 0$.

According to our numerical experiments, setting $\epsilon^{(k)} = 12A_k$ produces good results. Here A_k be the minimal triangle area among the triangles of the evolving mesh $\mathcal{M}^{(k)}$.

Figures 3.3 and 3.4 demonstrates equalizing mesh triangles by (3.8) with the tangent component defined by (3.10) and $\tau^{(k)} = 0$. Notice how well the proposed procedure of mesh equalization preserves the shape approximated by the original mesh. The bi-umbrella operator is a better choice than the single umbrella operator in tangential smoothing, see images (b) of Figure 3.4. Especially the tangential smoothing based on the single umbrella operator does not regularize the mesh where the mesh consists of saddle points.

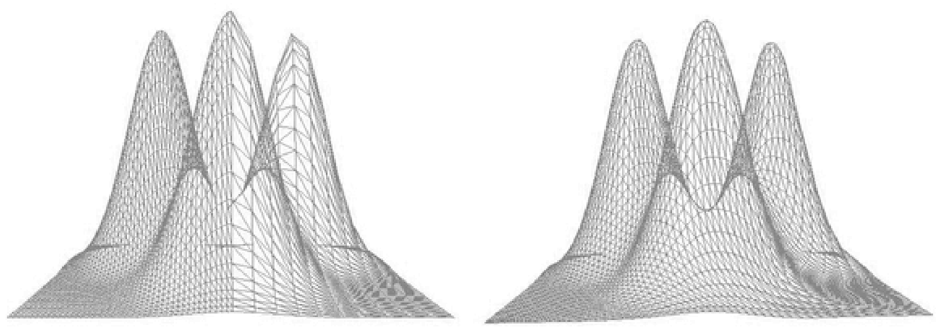


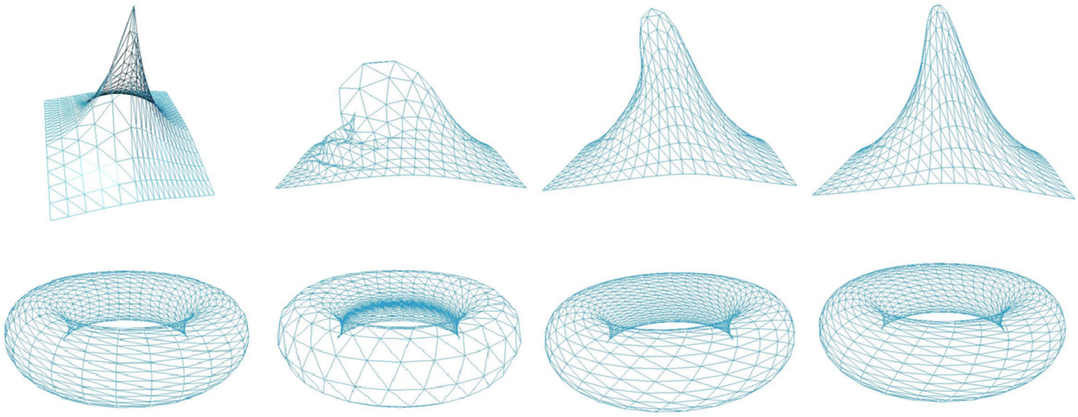
Figure 3.3: Left: a mesh consisted of two parts with different sampling rates. Right: tangential mesh evolution (3.8) with $\tau^{(k)} = 0$, (3.10) was used to equalize the mesh triangles.

The mesh boundary vertices are treated in a similar but more complex way since they are allowed to move along the boundary of $\mathcal{S}(u, v)$ only. For implementation details see [Yos01].

Approximation of Laplace-Beltrami Operator and Curvatures. Recently a very efficient approximation of the Laplace-Beltrami operator for a surface approximated by a triangle mesh was introduced by Pinkall and Polthier [PP93] in geometric modeling, see also [MDSB02]. A discrete Laplace-Beltrami operator $\triangle_{\mathbf{S}}(\mathcal{P})$ at a mesh vertex $\mathcal{P}$ is defined by

$$\triangle_{\mathbf{S}}(\mathcal{P}) = \frac{3}{A}\sum_{i=1}^{n}(\cot\ \alpha_i + \cot\ \beta_i)(Q_i - \mathcal{P}), \tag{3.11}$$

where A is the total area of the triangles adjacent to $\mathcal{P}$, α_i and β_i are the angles $\angle\mathcal{P}Q_{i-1}Q_i$ and $\angle\mathcal{P}Q_{i+1}Q_i$, respectively.



(a) (b) (c) (d)

Figure 3.4: Comparisons for tangential smoothing schemes. (a): Initial mesh. (b): Mesh evolution based on the tangent component of umbrella operator. (c): Mesh evolution based on the tangent component of bi-umbrella operator. (d): Tangential mesh evolution (3.8) with $\tau^{(k)} = 0$, (3.10) was used to equalize the mesh triangles.

Given a smooth surface $\mathcal{S}$ and a triangle mesh $\mathcal{M}$ approximating the surface, we use a standard angle-deficit approximation for the Gaussian curvature

$$K = \frac{3}{A}(2\pi - \sum_{i=1}^{M} \varphi_i),$$

where φ_i is the angle between $\mathcal{P}Q_i$ and $\mathcal{P}Q_{i+1}$.

Since for a smooth surface $\triangle_{\mathbf{S}}(\mathcal{S}) = 2H\mathbf{N}$ [Str88], a discrete approximation of the mean curvature H can be derived from the above discrete approximation of the Laplace-Beltrami operator

$$H = \frac{1}{2}\mathbf{N} \cdot \triangle_{\mathbf{S}}(\mathcal{P}).$$

This approximation works very well in many applications [DMSB99, MDSB02].

Although $H^2 - K$ is always positive for a smooth surface, it is not necessary true for discrete approximations of the Gaussian and mean curvatures. A standard approach to cope with this problem is to detect the mesh vertices where a discrete approximation of $H^2 - K$ is negative and set it equal to zero at those vertices.

However this approach is not acceptable to us since the term $H^2 - K$ is presented in (3.5) and it is not desired to have it discontinuous.

Let D denote the set of those mesh vertices for which $H \neq 0$ and $H^2 - K < 0$. We first compute

$$\lambda = \min_{D} \sqrt{\frac{H^2}{K}}.$$

Then we re-scale the mean curvature $H \to H/\lambda$ for the all vertices of D.

Since the quality of the mesh is improved during the evolution (3.8), $\lambda \to 1$ as $k \to \infty$.

Subdivision. In order to accelerate the mesh evolution process (3.8) we start from a coarse mesh and perform the linear one-to-four mesh subdivision when (3.8) is close to its steady-state. Figure 3.5 show various stages of approximating an elastica surface via combining (3.8) with subdivision.

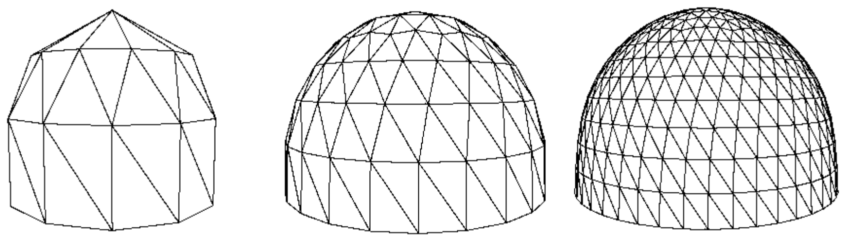


Figure 3.5: Starting from a coarse mesh evolved by (3.8), linear one-to-four mesh subdivision is used when (3.8) is close to its steady-state.

3.2 Numerical Experiments of Discrete Willmore Flow

Mesh Fairing. We compare the discrete Willmore flow (3.8), (3.5) with the bilaplacian flow

$$\mathcal{P}_i^{(k+1)} = \mathcal{P}_i^{(k)} - \tau\,\mathcal{U}_0^2(\mathcal{P}_i^{(k)}),$$

and a mesh evolution (3.8) by the Laplacian of mean curvature flow with speed F equal to

$$F = -\triangle_{\mathbf{S}}(H) \tag{3.12}$$

(various numerical approaches to the Laplacian of mean curvature flow were developed in [SK00a, SK01]).

Figures 3.8, 3.9, and 3.10 demonstrate various stages of mesh fairing by the bilaplacian flow, the Laplacian of mean curvature flow, and the discrete Willmore flow, respectively. The mesh shown in Figure 3.1 (a) is used as the initial mesh. The fairing processes are also combined with subdivision. These figures and Figure 3.11 demonstrate the superiority of the discrete Willmore flow (3.8), (3.5) over the bilaplacian flow and the Laplacian of mean curvature flow. Coloring by the mean curvature is used to visualize the geometric quality of the meshes.

Shape Restoration via Willmore Flow. When a real-world object is digitized by a range finder, a part of shape information may be lost because of specular reflection effects, object self-occlusion, etc. The Willmore flow can be used to restore missed shape parts [YB02, CDD$^+$04, XPB06].

Figure 3.6 demonstrates the Bunny having a large part of its flank removed and then triangulated. The discrete Willmore flow is applied to the triangles filled the hole. The result is presented in Figure 3.7. Notice a high quality of the restored part of the Bunny.

3.3 Summary of Discrete Willmore Flow

We presented a numerical approach for generating high quality, nice-looking shapes via the discrete Willmore flow. Contributions of our method include adding a tangential speed component to the Willmore flow for increasing computational stability of the flow and combining the mesh evolution approach with mesh refinement. Applications of the proposed numerical approach to mesh fairing and shape restoration were demonstrated.

Combining the developed approach with the automatic dynamic connectivity method [KBS00] and using implicit numerical schemes for the Willmore flow (3.4), (3.5) constitute themes for future research.

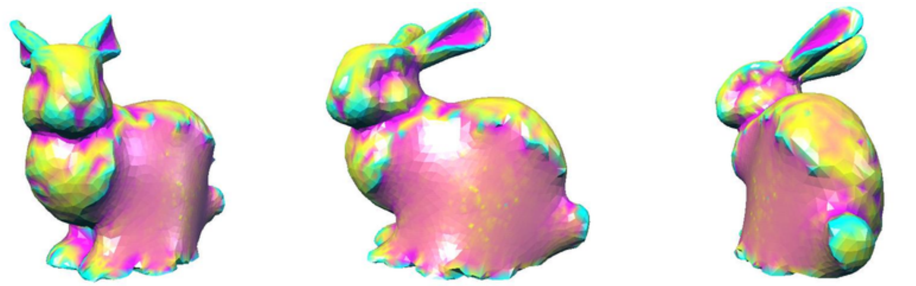


Figure 3.6: Bunny with a large part of its flank removed and then triangulated.

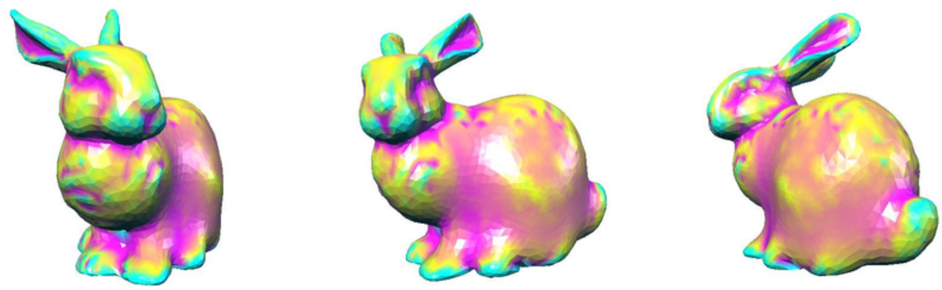


Figure 3.7: The Bunny flank is restored by the discrete Willmore flow.

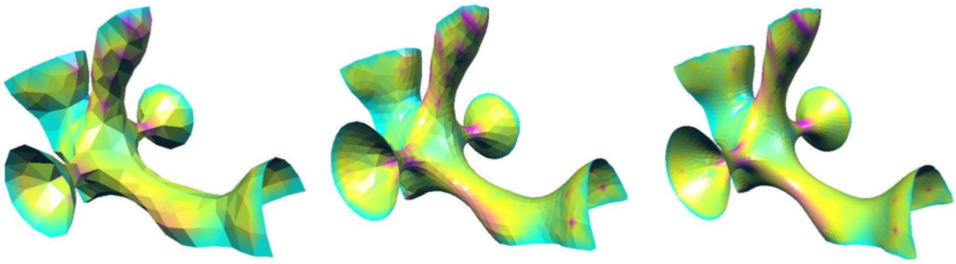


Figure 3.8: Mesh fairing by bilaplacian flow.

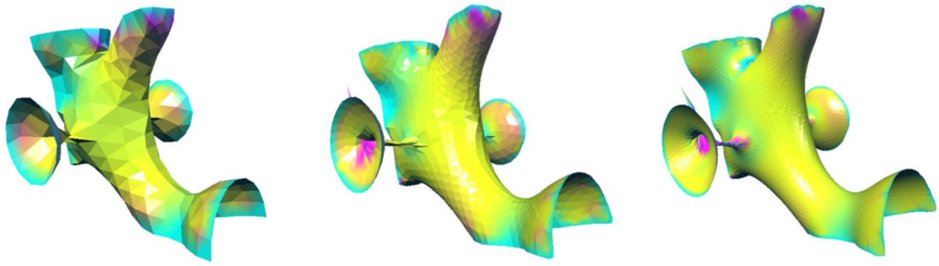


Figure 3.9: Mesh fairing by the Laplacian of mean curvature flow.

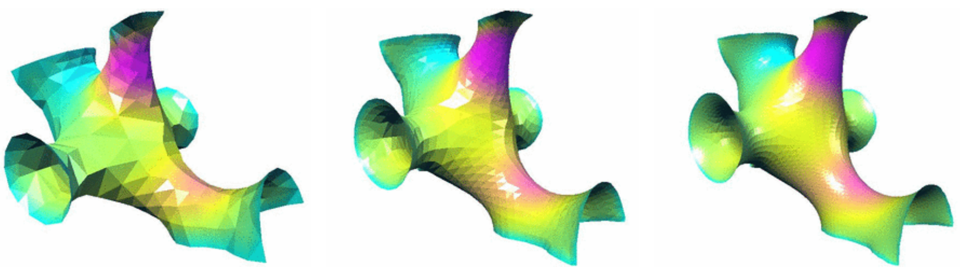


Figure 3.10: Mesh fairing by discrete Willmore flow.

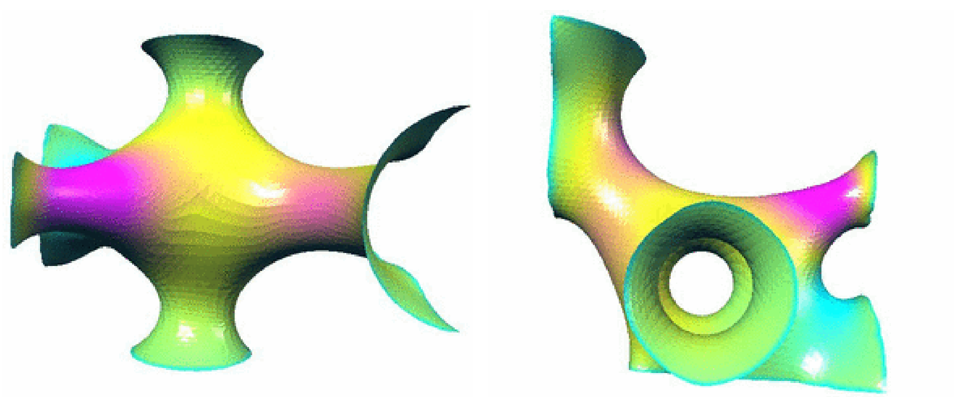


Figure 3.11: Discrete Willmore flow produces high quality shapes.

4

Fast and Robust Detection of Feature Lines on Meshes

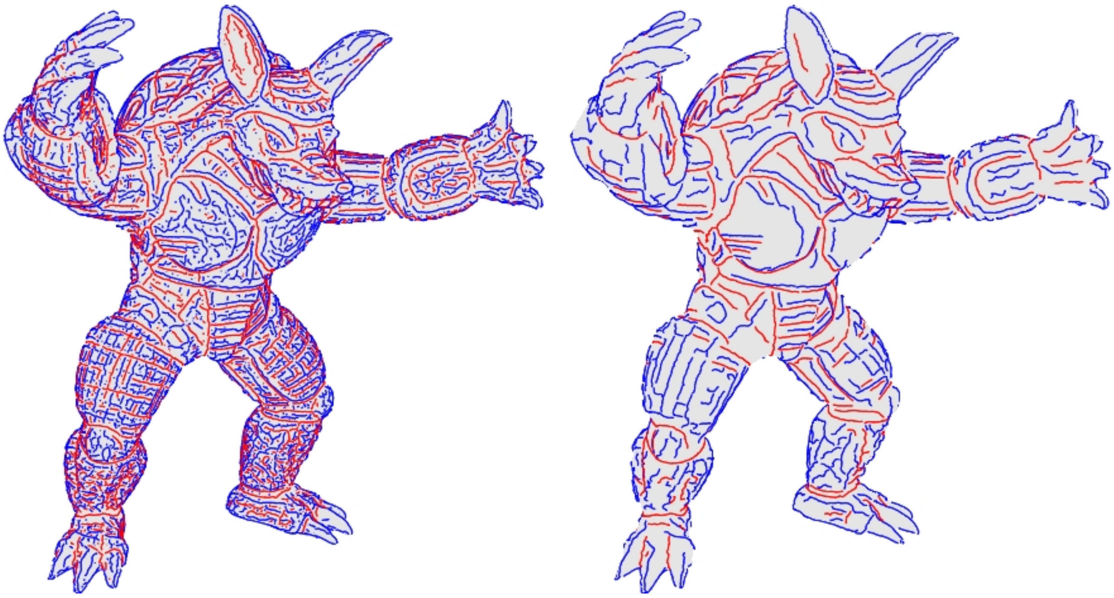


Figure 4.1: Detected crest lines. Changing the fitting neighbor size and filtering threshold gives us wide variety of salient surface features for example from highly detailed crest lines (left image) to large scale crest lines (right image).

Surface creases, curves on a surface along which the surface bends sharply can be intuitively defined as loci of sharp variation points of the surface normal. Mathematically the sharp variation points of the surface normals are described via extrema of the surface principal curvatures along their corresponding lines of curvature. These curvature extrema, called also ridges and crest lines, have been thoroughly studied in connection with research on classical differential geometry and singularity theory. Such curvature extremum curves first appeared in the research of the optics of the human eye [Gul04] by *Allvar Gullstrand* (1862 - 1930). He received the Nobel Prize for medicine or physiology in 1911 [Gul11]. Since then the ridges and their subsets have numerous applications in human perception [HR85], quality control of free-form surfaces [Hos92], free-form shape deformations [IFP95], image and data analysis [Ebe96], reverse engineering [HDW98], image reconstruction and registration [LFM96, GPA97], analysis and registration of anatomical structures [GM98], face pattern analysis and recognition [HGY$^+$99], mesh segmentation and flattening [SF04], mesh simplification [WB01, YBS05a], geomorphology [LS01], and non-photorealistic rendering [IFP95, DFRS03]. See also references therein. The so-called crest lines are formed by the perceptually salient ridge points and consist of the surface points where the magnitude of the largest (in absolute value) principal curvature attains a maximum along its corresponding line of curvature [MBF92].

Developing methods for fast and accurate detection of feature lines on polygonal surfaces is currently a subject of intensive research [YBS05a, HPW05, KK05]. Numerous ridges and crest lines detection techniques have been proposed for analytical surfaces (see for example, ridges on explicit [TG96], parametric [Mor96], and implicit [BPK98] surfaces) and images [MB95, BLBK03]. Practical detection of the crest lines and other types of curvature extrema on polygonal and point-sampled surfaces is a difficult computational task because it requires a high-quality estimation of the curvature tensor and curvature derivatives. In general, global fitting methods do a better job in estimating high-order surface derivatives and lead to more accurate detection of curvature extrema [KMW96, KLML96, OBS04, KK05] than the local estimation schemes. On the other hand, the local schemes are much faster and often demonstrate a quite satisfactory performance [Gué93, SF03, SF04, CP04a, YBS05a, HPW05].

In this Chapter, we follow [YBS05a] and describe a fast and robust method for detecting surface creases on surfaces approximated by dense triangle meshes. Our procedure for detecting the crest lines combines local polynomial fitting based on a modification of the method of [GI04], a finite difference scheme/test proposed in [OBS04] and used for curvature maxima/minima identification, and a careful thresholding based on the MVS functional of Moreton and Sequin [MS92]. Our method is fast since we estimate necessary surface derivatives via local polynomial fitting. For example, for the Igea model consisting more than $200K$ triangles it takes only nine seconds for estimating the curvature tensor and curvature derivatives and four seconds for detecting crest lines on a standard 1.7 GHz Pentium 4 PC. Our approach is capable of achieving high quality results comparable with those obtained via global fitting procedures [OBS04]. Figures 4.1 and 4.3 show crest line patterns found on simple and complex geometrical models for various values of a user-specified parameter which controls the strength of detected crest lines.

Applications of the crest lines for adaptive mesh simplification and feature-guided mesh segmentation are also discussed in Sections 4.8 and 4.9, respectively.

4.1 Differential Geometry Background of Curvature Extrema

We describe differential geometry background of the curvature extremum curves (ridges and crest lines) through their connections with *focal sets*, *medial axis*, and *Dupin's cyclides* by using singularity analysis. Then we explain why practical detection of crest lines is difficult, and show thresholding based on the MVS functional as a robust approach to tackle the difficulty. Figure 4.3 illustrates the relationships between a surface (curve), its one of focal set (evolute curve), its focal set singularity called focal rib (evolute cusp), and medial axis.

Curvature Extremum Sets. Let us consider curvature extremum curves on a surface $\mathbf{S}(u, v)$ where they are loci of principal curvature extrema along lines of curvature. Let $k_{\max}$ and $k_{\min}$ be the maximum and minimum principal curvatures of $\mathbf{S}(u, v)$, and $\mathbf{t}_{\max}$ and $\mathbf{t}_{\min}$ be the corresponding principal directions, respectively. In [Hos92], Hosaka derived the differential equations of the curvature extremum curves. His equations are characterized by the following four equations:

$$\frac{\partial k_{\max}}{\partial \mathbf{t}_{\max}} = 0, \quad \frac{\partial k_{\max}}{\partial \mathbf{t}_{\min}} = 0, \quad \frac{\partial k_{\min}}{\partial \mathbf{t}_{\min}} = 0, \quad \frac{\partial k_{\min}}{\partial \mathbf{t}_{\max}} = 0.$$

Here, we are interested in only two sets of curvature extremum curves on $\mathbf{S}(u, v)$ such that they are loci of principal curvature extrema w.r.t. corresponding principal directions. Denote

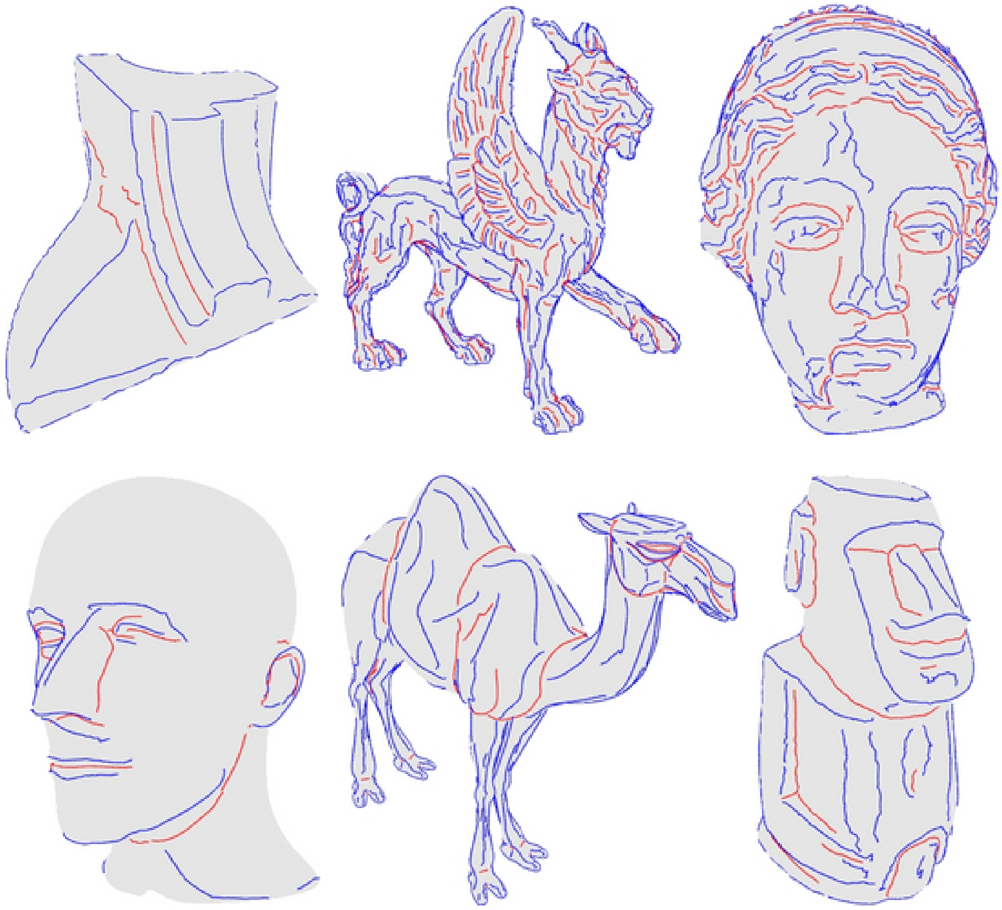


Figure 4.2: Crest lines detected on various triangle meshes. A scale-independent parameter T defined by (4.10) is used to keep the most visually important features: $T = 2.4$ for the Fan model, $T = 1.0$ for the Feline model, $T = 2.7$ for the Igea model, $T = 3.2$ for the Mannequin Head model, $T = 0.9$ for the Camel model, and $T = 2.3$ for the Moai model. For all the model one-ring neighborhood polynomial fitting is used for estimating the curvature tensor and curvature derivatives.

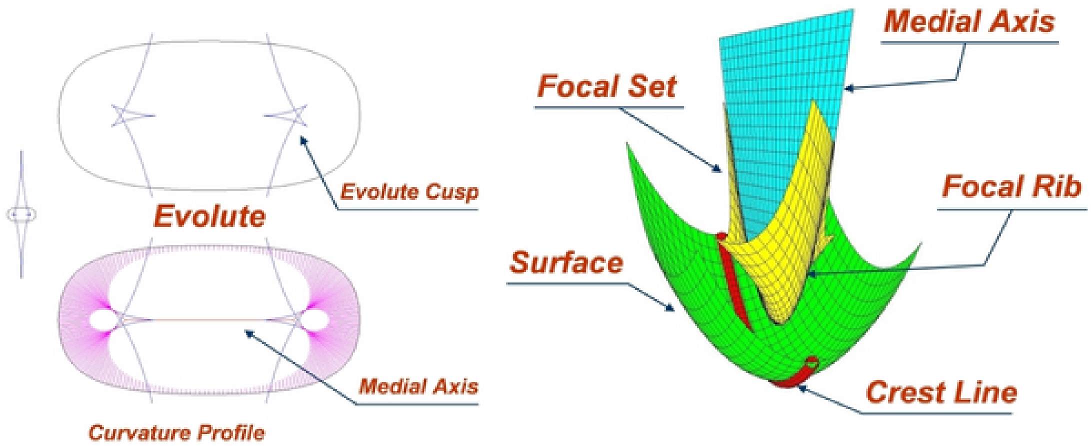


Figure 4.3: Curvature extrema, focal sets, and medial axis.

by $e_{\max}$ and $e_{\min}$ the derivatives of the principal curvatures along their corresponding curvatures directions:

$$e_{\max} = \frac{\partial k_{\max}}{\partial \mathbf{t}_{\max}} \quad \text{and} \quad e_{\min} = \frac{\partial k_{\min}}{\partial \mathbf{t}_{\min}} \tag{4.1}$$

Following [Thi96] let us call $e_{\max}$ and $e_{\min}$ the *extremality coefficients*. The extremality coefficients are not defined at the umbilical points ($k_{\max} = k_{\min}$) since the principal directions are undefined there. The surface creases considered in this Chapter are formed by the closure of points on $\mathbf{S}(u, v)$ where one of the extremality coefficients vanishes [Yui89]. According to this definition, the umbilical points belong to the surface creases. In [Por01, HGY+99, CP04b, CP05] the curvature extremum patterns in small vicinities of umbilical points are analyzed.

Crest Lines. In previous literature, definitions of curvature extremum curves, ridges, and crest lines are sometime mixed. According to [MBF92, OBS04, YBS05a], we define the ridges, ravines, and crest lines as follows. Ridge (ravine) points are characterized by positive (negative) maxima (minima) of maximum (minimum) principal curvatures w.r.t. maximum (minimum) principal directions:

$$\begin{array}{rlll} \text{Ridge:} & k_{\max} \geq 0, & e_{\max} = 0, & \frac{\partial e_{\max}}{\partial \mathbf{t}_{\max}} < 0, \\ \text{Ravine:} & k_{\min} \leq 0, & e_{\min} = 0, & \frac{\partial e_{\min}}{\partial \mathbf{t}_{\min}} > 0. \end{array}$$

The crest lines consist of perceptually salient ridge points. We distinguish convex and concave crest lines. The convex crest lines are given by

$$k_{\max} > |k_{\min}|, \qquad e_{\max} = 0, \qquad \frac{\partial e_{\max}}{\partial \mathbf{t}_{\max}} < 0,$$

while the concave crest lines are characterized by

$$k_{\min} < -|k_{\max}|, \qquad e_{\min} = 0, \qquad \frac{\partial e_{\min}}{\partial \mathbf{t}_{\min}} > 0.$$

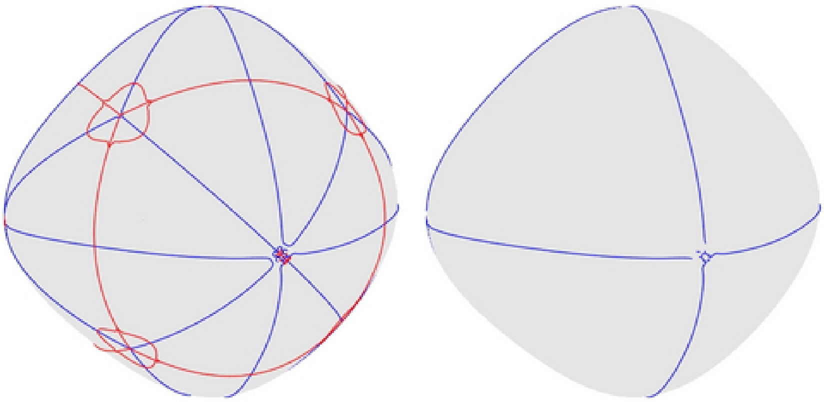
Figure 4.4 demonstrates the examples of two sets of curvature extremum curves and their subsets (ridge, ravine, and crest line). The convex and concave crest lines are dual w.r.t. the surface orientation as well as ridges and ravines: changing the orientation turns the convex crest lines (ridges) into concave one (ravines) and vice versa. According to the above definitions, one particular difference between the ridge-ravine and the crest lines is that the ridges and ravines can be intersected but the convex and concave crest lines can not, see images (g) and (h) of Figure 4.4.

4.2 Focal Sets

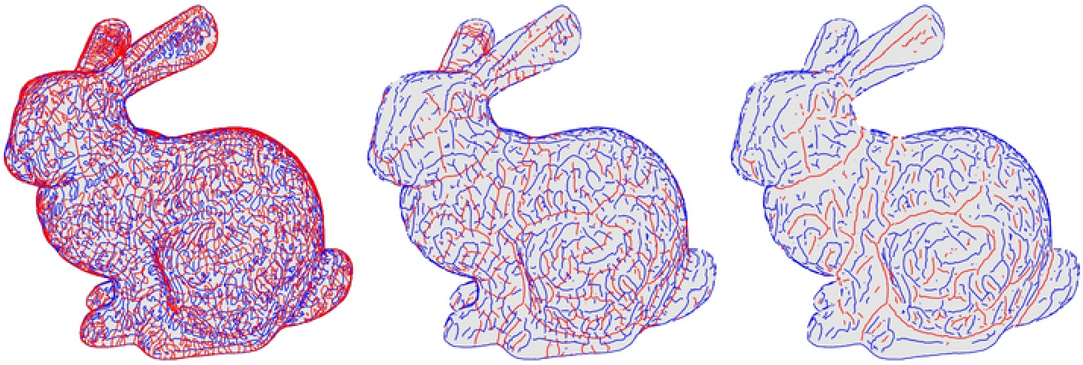
For a given surface $\mathbf{S}(u, v)$, a focal set $\mathbf{f}(u, v)$ which is the 3D analog of the evolute of a planar curve is defined by

$$\mathbf{f}(u, v) = \mathbf{S}(u, v) + R(u, v)\mathbf{n}(u, v), \tag{4.2}$$

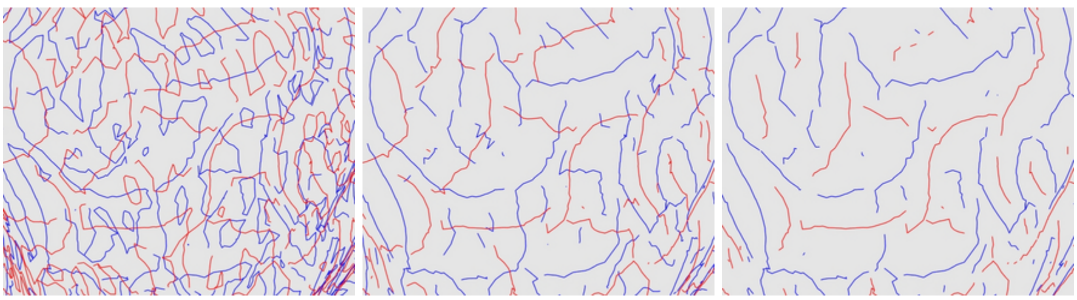
where $R(u, v)$ is equal to either $1/k_{\max}$ or $1/k_{\min}$ and $\mathbf{n} = \mathbf{n}(u, v)$ is the unit normal of $\mathbf{S}(u, v)$. In [HH92, HHS+92, HHS95] the focal sets are studied for shape interrogation purposes. The focal sets $\mathbf{f}_{\max}$ and $\mathbf{f}_{\min}$ consists of two sheets corresponding to $k_{\max}$ and $k_{\min}$ have singularities. The singularities of the focal sets consist of space curves are the so-called *focal ribs* called also cuspidal edges [Por87, Por01].



(a) (b)



(c) (d) (e)



(f) (g) (h)

Figure 4.4: Curvature extremum curves characterized by the conditions $e_{\max} = 0$ and $e_{\min} = 0$ are visualized in (a) and (c). Here (b) and (d) represent ridge-ravine subsets of (a) and (c), respectively. The crest lines of Bunny is given in (e). Here the ridges (ravines) and convex (concave) crest lines are equivalent in the octahedron mesh (b). The images (f), (g), and (h) represent the magnifications of same parts of (c), (d), and (e), respectively.

Proposition 1 *A curvature extremum curve point at $P = \mathbf{S}(u_0, v_0)$ corresponds a point on the focal rib at $\mathbf{f}(u_0, v_0)$; focal sets degenerate to the space curve iff $e_{\max} = 0$, $e_{\min} = 0$, or $k_{\max} = k_{\min}$.*

Consider the principal coordinate system at a surface point $P = \mathbf{S}(u_0, v_0)$ where the directions of basic tangents $\mathbf{S}_u = \frac{\partial \mathbf{S}(u,v)}{\partial u}$ and $\mathbf{S}_v = \frac{\partial \mathbf{S}(u,v)}{\partial v}$ coincide with the principal directions $\mathbf{t}_{\max}$ and $\mathbf{t}_{\min}$, respectively, at P. Assume $\{\mathbf{t}_{\max}, \mathbf{t}_{\min}, \mathbf{n}\}$ forms orthogonal basis at P and locally choose the arc length parameterization of the lines of curvature: $|\mathbf{t}_{\max}| = |\mathbf{t}_{\min}| = 1$. According to the classical formula of Rodrigues, we obtain

$$d\mathbf{n} + k\, d\mathbf{S} = \mathbf{n}_u du + \mathbf{n}_v dv + k(\mathbf{S}_u du + \mathbf{S}_v dv) = \nabla\mathbf{n} \cdot \mathbf{t} + k\, \nabla\mathbf{S} \cdot \mathbf{t} = 0.$$

In such principal coordinate system, we can express the partial derivatives of the normal $\mathbf{n}_u$ and $\mathbf{n}_v$ by

$$\mathbf{n}_u = -k_{\max}\mathbf{S}_u \quad \text{and} \quad \mathbf{n}_v = -k_{\min}\mathbf{S}_v,$$

where $(du, dv) = (\text{constant}, 0)$ leads $k = k_{\max}$ and $\mathbf{t} = \mathbf{t}_{\max}$, and $(du, dv) = (0, \text{constant})$ leads $k = k_{\min}$ and $\mathbf{t} = \mathbf{t}_{\min}$. The focal set degenerates iff the oriented area element of $\mathbf{f}(u, v)$ vanishes. It gives

$$\begin{aligned} \mathbf{f}_u \times \mathbf{f}_v &= (\mathbf{S}_u + R_u\mathbf{n} + R\,\mathbf{n}_u) \times (\mathbf{S}_v + R_v\mathbf{n} + R\,\mathbf{n}_v) && (4.3)\\ &= A\mathbf{n}(1 - R\,k_{\max})(1 - R\,k_{\min}) - \mathbf{S}_u R_u(1 - R\,k_{\min}) - \mathbf{S}_v R_v(1 - R\,k_{\max}) = \mathbf{0} && (4.4) \end{aligned}$$

where $\mathbf{f}_u$ and $\mathbf{f}_v$ are the partial derivatives of $\mathbf{f}(u, v)$, $A = |\mathbf{S}_u \times \mathbf{S}_v|$, and $R = 1/k_{\max}$ or $1/k_{\min}$. Thus the $k_{\max}$-branch of $\mathbf{f}(u, v)$ degenerates at $\mathbf{f}(u_0, v_0)$ if $e_{\max} = 0$, the $k_{\min}$-branch of $\mathbf{f}(u, v)$ degenerates at $\mathbf{f}(u_0, v_0)$ if $e_{\min} = 0$, and both the branches degenerate at common point $\mathbf{f}(u_0, v_0)$ if $k_{\max} = k_{\min}$ (or $e_{\max} = 0 = e_{\min}$).

A singularity analysis of the focal sets has been a common tool for investigating the behavior of various types of curvature extrema. The equation (4.4) was derived in [ABK94], and applied in [YBS05a] for practical detection of the curvature extremum curves.

Generalized Offset Surfaces. The above singularity analysis $\mathbf{f}_u \times \mathbf{f}_v = \mathbf{0}$ of the focal set (4.2) can be directory applied to the generalized offset surfaces where $R(u, v)$ of (4.2) is a graph of function defined on $\mathbf{S}(u, v)$.

Theorem 1 (Offset Singularity) *Let $\mathbf{n}(u, v)$, $k_{\max}$, and $k_{\min}$ be the unit normal, maximum and minimum principal curvatures of a surface $\mathbf{S}(u, v)$, and $R(u, v)$ is a graph of function defined on $\mathbf{S}(u, v)$. Iff*

$$\begin{cases} R_u(1 - R\,k_{\min}) &= 0\\ R_v(1 - R\,k_{\max}) &= 0\\ (1 - R\,k_{\max})(1 - R\,k_{\min}) &= 0 \end{cases}$$

where $(u, v) = (u_0, v_0)$ then the generalized offset surface

$$\mathbf{g}(u, v) = \mathbf{S}(u, v) + R(u, v)\mathbf{n}(u, v) \tag{4.5}$$

has a singularity at (u_0, v_0).

For example when a classical offset surface $R(u, v) = \text{constant}$ has the singularities iff $R = 1/k_{\max}$ or $R = 1/k_{\min}$.

4.3 Medial Axis

The medial axis was proposed by Blum for 2D shape perception and recognition purposes [Blu67]. In 3D, the medial axis has been intensively studied in computational geometry through connection with the Voronoi diagram and surface reconstructions [ABK98, ACK01a, DZ02, DG03, MAVdF05], meshing and finite element generations [Owe98, ACSYD05], CAD [WF00, Sur03], solid modeling [BBGS99, BL99], shape deformation tasks [Blo02, YBS03], motion planning [Lat91], and many other applications.

Mathematically the medial axis is defined as loci of centers of maximal empty balls for a bounded figure $\mathcal{F}$. The maximal empty ball, also called *medial ball* [ACK01b], is completely contained in no other empty ball. The medial axis together with this associated radius function is called the *medial axis transform*. The sharp boundaries of medial axis are called the *skeletal edges*, see [ABOK94] for classifications of points on the medial axis.

Consider shrink wrapping of a boundary $\partial\mathcal{F}$ to the medial axis: two-sided medial axis. The mathematical description of the two-sided medial axis first appeared in [SPW96] through analysis of topological structure of the medial axis; there exists a continuous mapping between the medial axis and its corresponding boundary $\partial\mathcal{F}$ [Wol92], see also [ABE06] for a survey of topological analysis for the medial axis. Practical usage of the two-sided medial axis was first proposed recently for feature detection of meshes [HBK02], and later it was applied to mesh deformations [YBS03, YBS06c, YBS06a].

Consider a bounded 3D figure $\mathcal{F}$ whose boundary $\partial\mathcal{F}$ is a smooth closed surface $\mathbf{S}(u, v)$. Consider a point $\mathbf{S}(u^0, v^0) \in \partial\mathcal{F}$. Let $r(u^0, v^0)$ be the radius of the inner medial ball for which $\mathbf{S}(u^0, v^0)$ is a tangency point. See [SPW96] for mathematical construction of $r(u, v)$. The parametric representation of the medial axis is given by $\mathbf{m}(u, v) = \mathbf{S}(u, v) + r(u, v)\mathbf{n}(u, v)$ which is a particular case of the generalized offset surface (4.5) with $R(u, v) = r(u, v)$.

According to Theorem 1, if $\mathbf{m}(u, v)$ degenerates at $\mathbf{m}(u_0, v_0)$ then $r(u, v)$ is equal to either $1/k_{\max}$ or $1/k_{\min}$ at $(u, v) = (u^0, v^0)$. This immediately gives us either $e_{\max} = 0$ or $e_{\min} = 0$ at $\mathbf{S}(u^0, v^0)$. Thus, the medial ball boundary of radius $r = 1/k_{\max}$ ($r = 1/k_{\min}$) at (u^0, v^0) coincides with the osculating sphere of radius $1/k_{\max}$ ($1/k_{\min}$) at (u^0, v^0). Consequently, $\mathbf{m}(u_0, v_0)$ with $r = 1/k_{\max}$ ($r = 1/k_{\min}$) belongs the focal rib $\mathbf{f}_{\max}(u_0, v_0)$ ($\mathbf{f}_{\min}(u_0, v_0)$).

Proposition 1 and the above analysis of the medial axis indicate that the skeletal edges belong to focal ribs. This fact is well-known in 2D [Ley87, CCM97] and 3D [YL90, ABK94, BAK97]. A geometric description of focal ribs which belong to the skeletal edges was given in [BY01]. Proposition 1 also leads a relationship between the curvature extremum curves and a special family of surfaces called Dupin's cyclides.

4.4 Dupin's Cyclides

The Dupin's cyclides were introduced by the French geometer *Pierre Charles Francois Dupin* (1784 - 1873) at the beginning of 19th century while he was still an undergraduate at the Ecole Polytechnique in Paris. Since then the Dupin's cyclides have been intensively studied in connection with various shape modeling tasks. See, for example [CDH89] for a short historical survey of the Dupin's cyclides and their usage and [FG04] for recent applications of the Dupin's cyclides in geometric modeling as a CAGD primitive. The family of Dupin cyclides includes spheres, cylinders, cones, and tori, see Figure 4.5.

The Dupin's cyclides are characterized by the condition $e_{\max} = 0 = e_{\min}$. Here $e_{\max}$ and $e_{\min}$ are the extremality coefficients defined in (4.1). It means that lines of curvature are all straight

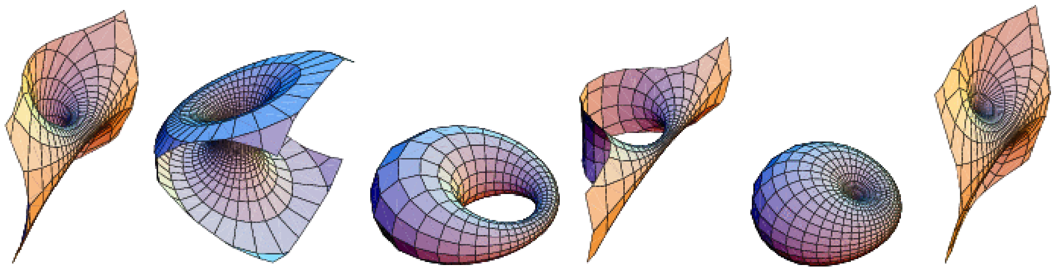


Figure 4.5: Cyclide examples from www.mathworld.wolfram.com.

lines or circular arcs [Pin86]. From $e_{\max} = 0 = e_{\min}$ and Proposition 1, the focal set of the Dupin's cyclides degenerates *everywhere* to the focal ribs which form space curves include the isolated-points. In fact this is an another definition of the Dupin's cyclides [Hir90]. The medial axis of the Dupin's cyclides degenerates to a set of space curves. A sphere and a plane can be considered as degenerated Dupin's cyclides whose focal sets and medial axis are isolated points; the focal point of a plane is located at infinity. There are no salient surface creases on the Dupin's cyclides although their extremality coefficients vanish.

Let us consider a surface point where extremality coefficients at the point satisfy the condition

$$|e_{\max}|^2 + |e_{\min}|^2 = 0. \tag{4.6}$$

Notice that the left-hand side of (4.6) is the integrand of the so-called MVS functional introduced in [MS92] for fair surface design purposes. For a generic surface, focal ribs always go through the focal set singularities corresponding to the umbilics of the surface. Now we can conclude that a generic surface region where the left hand-side of (4.6) is small is close to a part of a Dupin cyclide.

A practical detection of the ridges and their subsets is extremely difficult in those surface regions which are slightly perturbed Dupin cyclide patches and where, therefore, the left hand-side of (4.6) is close to zero. Such regions may contain many spurious ridges (and crest lines). Thus it seems natural to use the left hand-side of (4.6) as a measure for selecting geometrically important crest lines.

4.5 Estimating Surface Derivatives

Given a mesh $\mathcal{M}$ approximating a smooth surface $\mathcal{S} = \mathbf{S}(u, v)$, in order to achieve a fast and accurate estimation of the principal curvatures and their derivatives a bivariate polynomial is fitted locally to each mesh vertex. To date, two polynomial fitting strategies are used for estimating surface derivatives at a mesh vertex. According to one strategy, it is assumed that the surface normal at vertex is preliminary estimated. It leads to the so-called adjacent-normal cubic approximation method [GI04]. The second strategy [CP03] does not assume that the mesh normal is already given. According to our numerical experience, if the vertex normal is approximated appropriately, the first strategy leads to a better estimation of the surface curvatures and their derivatives at the vertex.

In our numerical experiments we use the following enhancement of adjacent-normal cubic approximation method. For each mesh vertex $\mathbf{p} \in \mathcal{M}$ its one-link neighborhood is considered and a new vertex $\mathbf{p}'$ is obtained as the arithmetic mean of the centroids of the mesh triangles adjacent to **p**. These new vertices $\{\mathbf{p}'\}$ form a new mesh $\mathcal{M}'$ which is smoother than $\mathcal{M}$. Now for each vertex $\mathbf{p}' \in \mathcal{M}'$ its unit normal is estimated via Nelson Max's method [Max99]. Then a cubic polynomial

$$h(x, y) = \frac{1}{2}\left(b_0 x^2 + 2b_1 xy + b_2 y^2\right) + \frac{1}{6}\left(c_0 x^3 + 3c_1 x^2 y + 3c_2 xy^2 + c_3 y^3\right) \qquad (4.7)$$

is fitted in the least-square sense [GI04] to $\mathbf{p}'$ and a set of its neighboring vertices. That set of neighbors of $\mathbf{p}'$ is obtained from the k-link neighborhood of $\mathbf{p}'$ by removing those vertices whose normals make obtuse angles with the normal at $\mathbf{p}'$. In practice we use $k = 1, 2, 3, 4$. Next the curvature tensor and extremality coefficients are expressed via derivatives of local cubic polynomial $h(x, y)$. Finally these curvature attributes are assigned to the original vertices $\{\mathbf{p}\}$ of mesh $\mathcal{M}$.

We have also derived an elegant formula for an extremality coefficient at a surface point where $\mathcal{S}$ is locally approximated by (4.7)

$$e = \partial k/\partial \mathbf{t} = \begin{pmatrix} t_1^2 \\ t_2^2 \end{pmatrix}^T \begin{pmatrix} c_0 & c_1 \\ c_2 & c_3 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}. \qquad (4.8)$$

Here $\mathbf{t} = (t_1, t_2)^T$ is the principal direction corresponding to a principal curvature k. Because of its simplicity, (4.8) leads to a significant reduction of computational time, see [TG96, MAM97] for comparison with the traditional long formulae of $e_{\max}$ and $e_{\min}$.

To prove (4.8) let us consider the well-known formula for an extremality coefficient $e = \partial k/\partial \mathbf{t}$ for a surface given in implicit form $F(\mathbf{x}) = 0$, $\mathbf{x} = (x_1, x_2, x_3)$, (see, for example, [Por01, Exercise 11.8] and also [MBF92] where a small mistake in the final formulas for the curvature derivatives is made)

$$e = \nabla k \cdot \mathbf{t} = \frac{F_{ijl} t_i t_j t_l + 3k F_{ij} t_i n_j}{|\nabla F|}, \qquad (4.9)$$

where F_{ij} and F_{ijl} denote the second and third partial derivatives of $F(\mathbf{x})$, respectively, $\mathbf{t} = (t_1, t_2, t_3)$ is the principal direction corresponding to a principal curvature k, $\mathbf{n} = (n_1, n_2, n_3)$ is the unit surface normal, and the summation over repeated indices is implied. In our case, $F = z - h(x, y)$ and at the origin of coordinates $\mathbf{n} = (0, 0, 1)$ and $\mathbf{t} = (t_1, t_2, 0)$. Thus, since the polynomial $h(x, y)$ does not contain linear terms, at the origin of coordinates (4.9) simplifies into

$$e = F_{ijl} t_i t_j t_l$$

and (4.8) immediately follows.

Figure 4.6 compares the sets of crest lines detected on a 3D text mesh via the straightforward polynomial fitting (the top image) and the enhanced adjacent-normal cubic approximation method (we use $k = 1$ in this example). Figures 4.8 and 4.9 demonstrate how our procedure to estimate surface derivatives is effective comparing with another smoothing method and other normal estimation methods.

Although our scheme for estimating surface derivatives seems complicated, it leads to highly effective crest line detection procedure which only slightly depends on the mesh connectivity and triangle aspect ratios. In Figure 4.7 we compare the patterns of the crest lines detected on the original Stanford bunny mesh and on the mesh obtained via an implicitization of the bunny model and then polygonizing using Bloomental's method [Blo94]. Despite the fact that the new bunny mesh contains many sliver triangles and has irregular connectivity, the patterns of the crest lines found on the meshes are remarkably similar.

Figure 4.6: Crest lines detected on 3D text. Top: polynomial fit without preliminary estimation of mesh normals is used. Bottom: the enhanced adjacent-normal cubic approximation method is employed for estimating surface curvatures and their derivatives. In both the cases preliminary smoothing $\mathbf{p} \to \mathbf{p}'$ was applied.

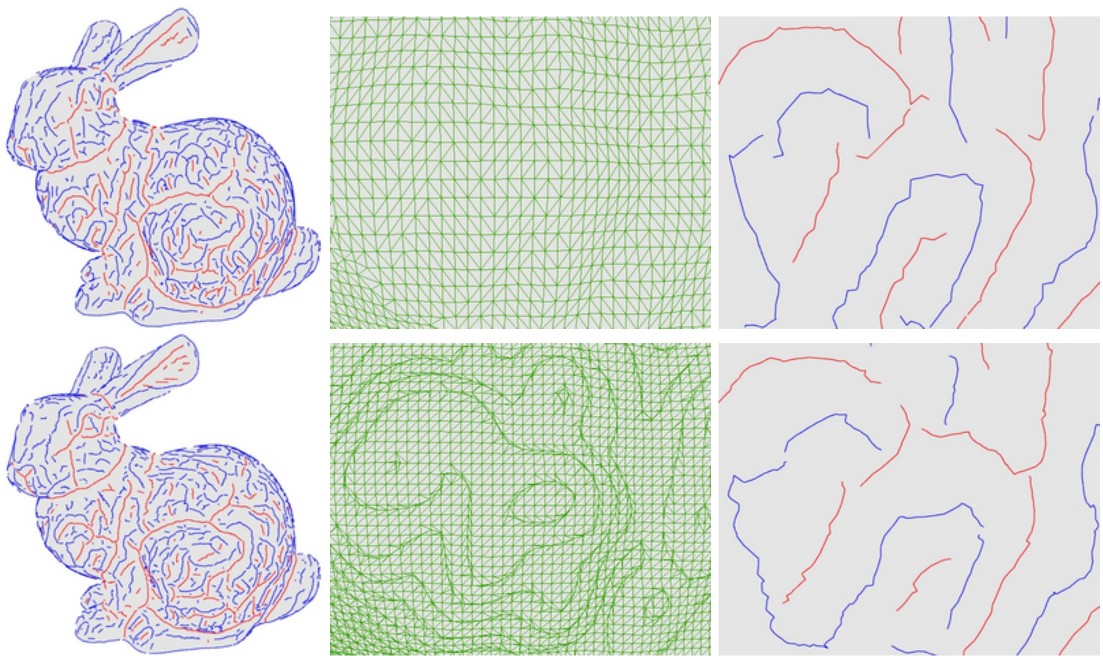


Figure 4.7: Patterns of crest lines and mesh triangles for two bunny models. Top: original Stanford bunny mesh with 69,451 triangles is used. Bottom: another bunny mesh with 279,984 triangles is used. The necessary surface derivatives are estimated via the enhanced cubic polynomial fitting with $k = 1$ for the original Stanford bunny mesh and $k = 3$ for the remeshed bunny since the latter is more than three times bigger than the original one.

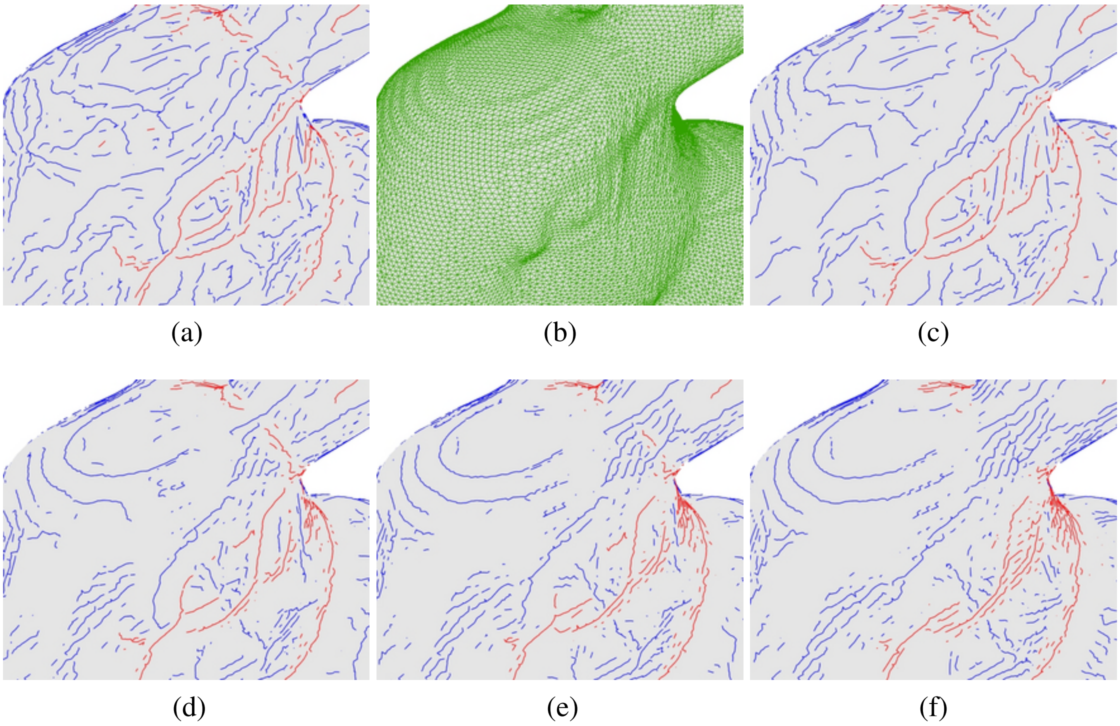


Figure 4.8: Smoothing effects for crest line detection. (a): Crest lines detected on (b) without any smoothing. (b): The irregular bunny mesh generated by an implicitization of the Stanford bunny model and then polygonizing using Bloomental's method [Blo94]. (c): Crest lines detected on (b) with our preliminary smoothing $\mathbf{p} \rightarrow \mathbf{p}'$. The images (d), (e), and (f) represent the crest lines detected on (b) with the semi-implicit mean curvature flow proposed in [DMSB99] where the time step parameters are 0.25, 0.5, and 1.0, respectively with one iteration. One-ring neighborhood polynomial fitting is used for all the models ($k = 1$).

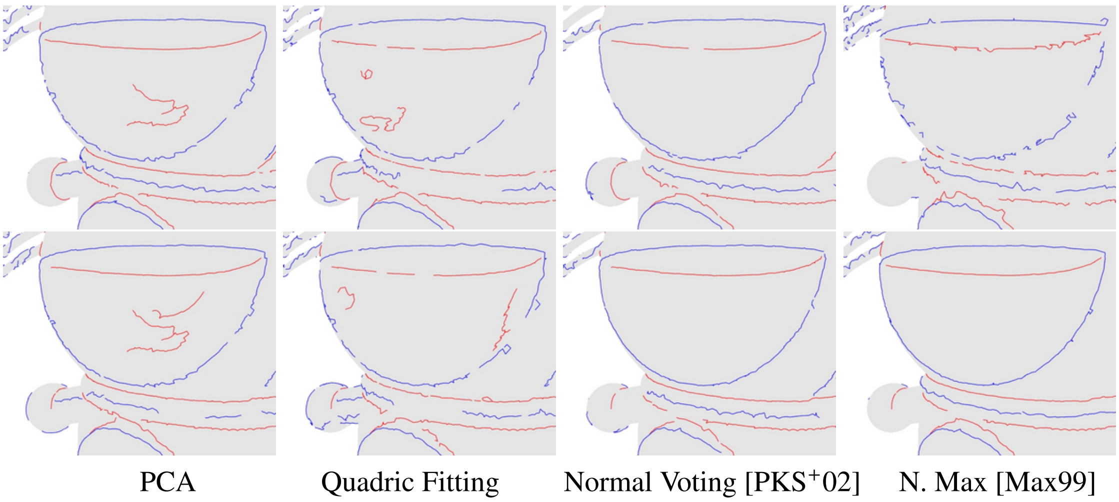


Figure 4.9: Normal estimations and smoothing. Top and bottom images represent crest lines detected on Robot Cat model without smoothing and with smoothing, respectively by using different vertex normal approximation methods where bottom-right image corresponds our result. One-ring neighborhood polynomial fitting is used for all the models ($k = 1$).

4.6 Tracing and Thresholding Crest Lines

Once the curvature tensor and extremality coefficients are estimated at each vertex of $\mathcal{M}$, we inspect the edges $\mathcal{M}$ and check whether they contain curvature maxima and minima. We detect the crest line vertices and connect them together following the procedure proposed in [OBS04] with one small, but important, addition. It turns out that the procedure may generate several close disconnected crest lines in situations similar to those shown in the left image of Figure 4.10. In order to reduce the fragmentation of the crest lines we inspect the mesh vertices and their one-ring neighborhoods. For each one-ring vertex neighborhood containing crest line end-points we connect two end-points if $\alpha \leq \pi/3$, $\beta \leq \pi/3$, $\gamma \leq \pi/2$, where α, β, and γ are the angles between the end-segments and the segment connecting the end-points, as seen the right image of Figure 4.10, see also Figure 4.11.

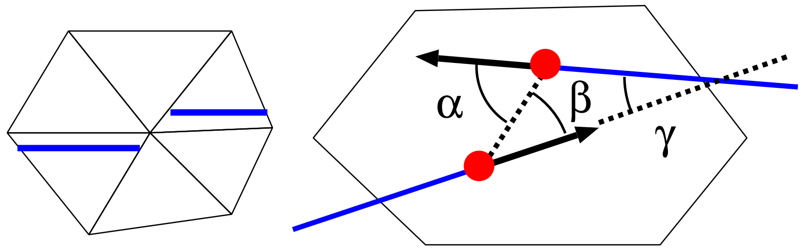


Figure 4.10: Left: a situation when we may want to connect the crest lines (shown in bold) together. Right: angles α, β, and γ generated by crest line end-segments and the segment connecting crest line end-points are used to measure when gap-jumping is necessary.

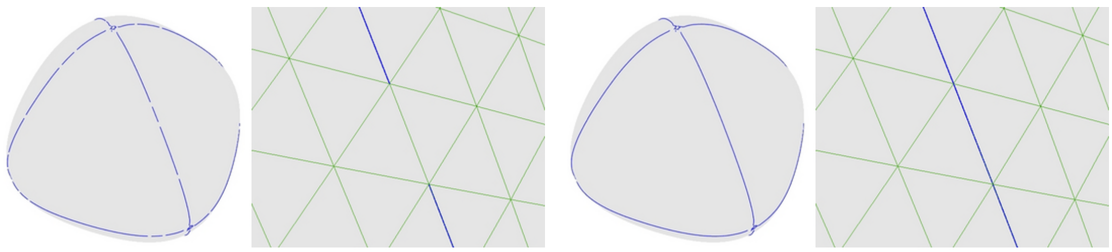


Figure 4.11: Left two images represent the close disconnected crest lines. Right two images demonstrate the extracted crest lines after connecting the gaps by using the angles α, β, and γ defined in Figure 4.10.

Although increasing neighborhood size for polynomial fitting gives us much smooth crest lines as shown in 4.15, the spurious ridges and crest lines can not be removed without over-smoothing of the crest lines by changing the size. As we mentioned before, the sum of squared extremality coefficients is very appropriate for measuring saliency of the crest lines. In practice we use the following scale-independent quantity to measure the strength of a crest line

$$T = \int ds \cdot \int \sqrt{|e_{\max}|^2 + |e_{\min}|^2}\, ds, \tag{4.10}$$

where the integrals are taken over the crest line. This thresholding parameter involves third-order surface derivatives and is more complex than that used in [OBS04] where the integral of a principal curvature along a feature line was used. On the other hand, thresholding with (4.10) is simpler than the thresholding scheme proposed in [CP04a] where a second-order curvature derivative is used for filtering out spurious ridges and crest lines.

We use a linear interpolation scheme for estimating the cyclideness

$$C = \sqrt{|e_{\max}|^2 + |e_{\min}|^2} \tag{4.11}$$

at crest line vertex $\mathbf{v}$ located on mesh edge $[\mathbf{p}, \mathbf{q}]$:

$$C(\mathbf{p}) = \frac{a\,C(\mathbf{p}) + b\,C(\mathbf{q})}{a+b},$$

where $a = |e_{\max}(\mathbf{q})|$, $b = |e_{\max}(\mathbf{p})|$ for the convex crest lines and $a = |e_{\min}(\mathbf{q})|$, $b = |e_{\min}(\mathbf{p})|$ for the concave ones. Now the integrals in (4.10) are estimated by a simple trapezoid approximation similar to that used in [OBS04].

Roughly speaking, cyclideness (4.11) measures how far a surface region is from being a part of a Dupin cyclide. If $\mathbf{x}$ lies on a convex (concave) crest line, then $e_{\max}(\mathbf{x}) = 0$ and $C(\mathbf{x}) = |e_{\min}(\mathbf{x})|$ ($e_{\min}(\mathbf{x}) = 0$ and $C(\mathbf{x}) = |e_{\max}(\mathbf{x})|$).

At the first glance, it looks that (4.10) does not affect umbilical regions. In fact it does: by continuity cyclideness (4.11) vanishes at the isolated umbilics. A small perturbation of an umbilic region creates a non-umbilical region containing isolated umbilics. Further, as it was shown in [BAK97], the crest lines do not pass through the generic (typical) umbilics.

Figure 4.12 demonstrates how our crest line filtering scheme works for a model with spherical and cylindrical regions. Notice how well the crest lines detected at the mesh parts approximated those regions are filtered out.

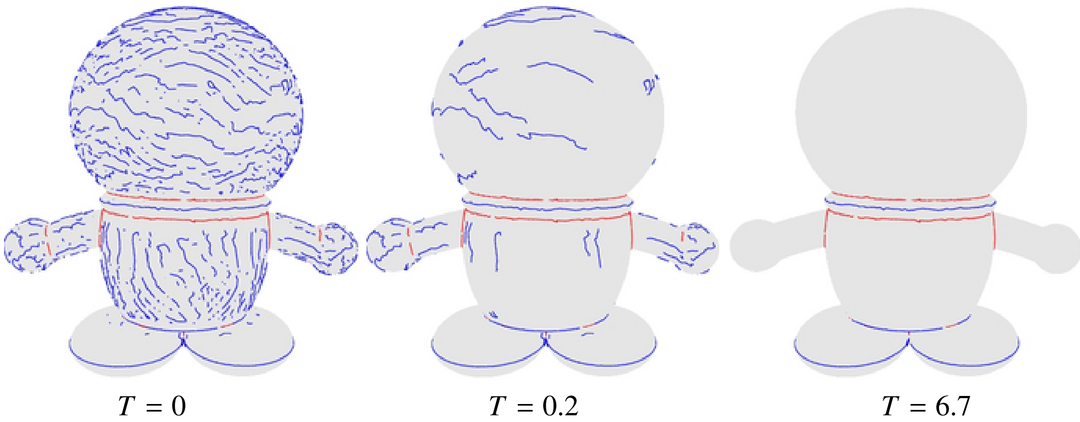


Figure 4.12: Detecting crest lines for a model containing spherical and cylindrical regions for various values of threshold T. For each mesh vertex, its three-ring neighborhoods ($k = 3$) is used for local polynomial fitting of Robot Cat.

Figure 4.13 exposes detecting crest lines on a more complex model containing flat, cylindrical, and slightly curved regions and small features. Increasing T allows us to remove inessential crest lines while preserving salient ones. The figure also demonstrates how the size of vertex neighborhoods used for polynomial fitting affects the crest line detection procedure (see also Figure 4.15). A larger neighborhood leads to smoother approximation of the mesh and, therefore, allows us to disregard the crest lines located in slightly convex/concave regions. See also Figure 4.2 where one-ring neighborhood polynomial fitting is used for all the models. By using simple triangulation for an image, our method also can be applied to the image, see Figure 4.14.

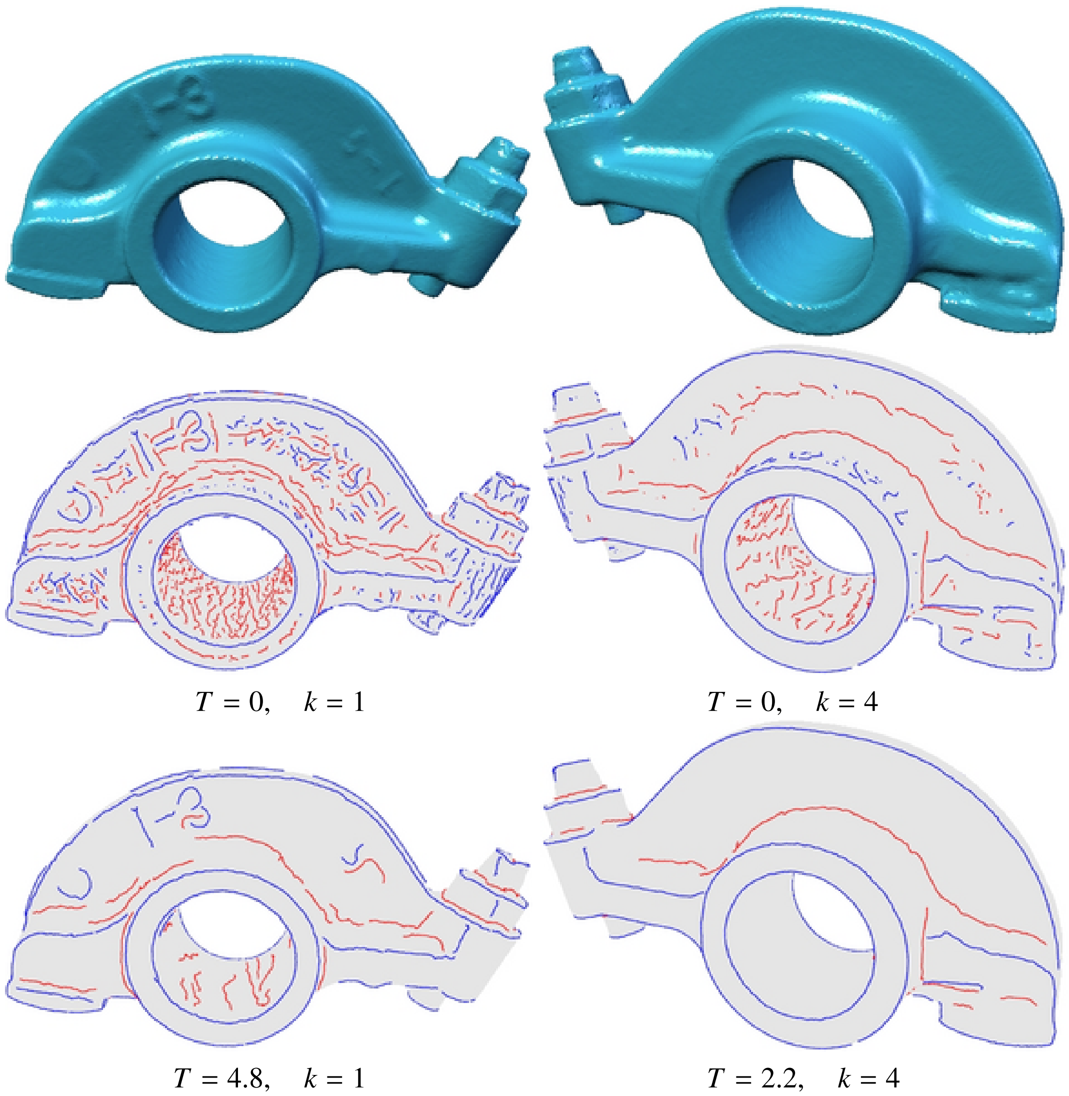


Figure 4.13: Crest lines detected on a mechanical part model with different values of threshold T and vertex neighborhood size k used for local polynomial fitting.

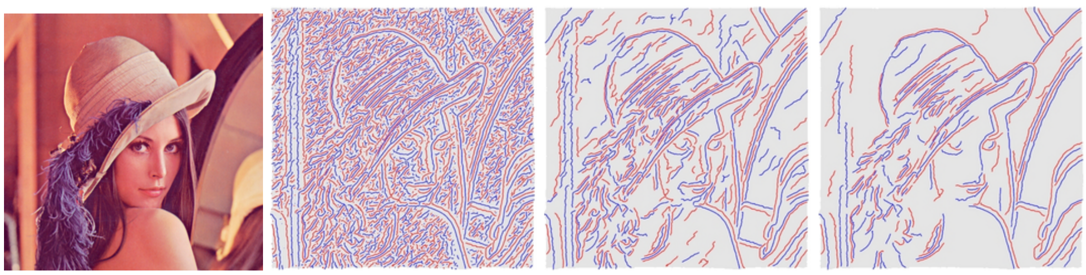


Figure 4.14: The crest lines on Lena are found with three-ring neighborhood fitting ($k = 3$) with $T = \{0, 8, 37\}$, respectively where the Lena image is triangulated for our method.

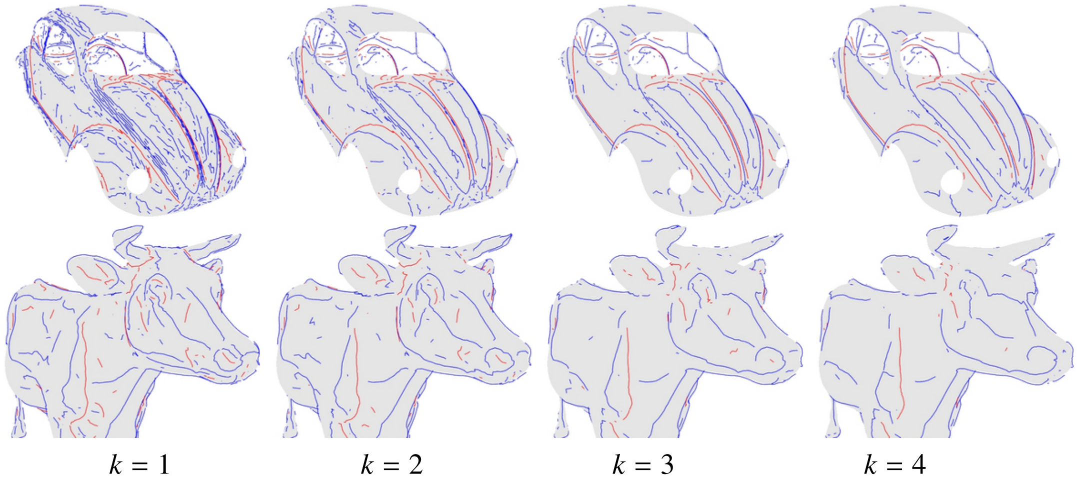


Figure 4.15: Crest lines detected on Car and Cow models where neighborhood sizes for polynomial fitting are equal to $k = \{1, 2, 3, 4\}$ from left to right images, respectively.

4.7 Numerical Experiments of Crest Line Detection

All examples presented in this Chapter are computed by using gcc 2.95 C++ compiler on a standard 1.7 GHz Pentium 4 PC with 512 MB RAM. As demonstrated in Figure 4.16, our method is fast and processes about $20K/k$ triangles per second for k-ring neighborhood polynomial fitting and estimating the curvature tensor and curvature derivatives. The crest line tracing stage at the method is faster than the estimation stage although the former depends on geometric complexity of models. The method is robust. The results of our crest line detection procedure depend only slightly on the quality of the mesh, as demonstrated in Figure 4.7.

Our method is capable of achieving high quality results in detecting salient curvature extrema to compare with schemes based on global fitting procedures. In Figures 4.17 and 4.18 we give a visual comparison of our method with that developed in [OBS04] and with the exact detection of the crest lines on analytical waving surface $\mathbf{r}(u, v) = [u \cos v, u \sin v, \cos u]$. The mesh we used to approximate the waving surface is not dense: it consists of less than 5K triangles only. Nevertheless the max-norm error estimates for the extemality coefficients are reasonably good: 0.48 for $e_{\max}$ and 0.56 for $e_{\min}$, see Table 4.1.

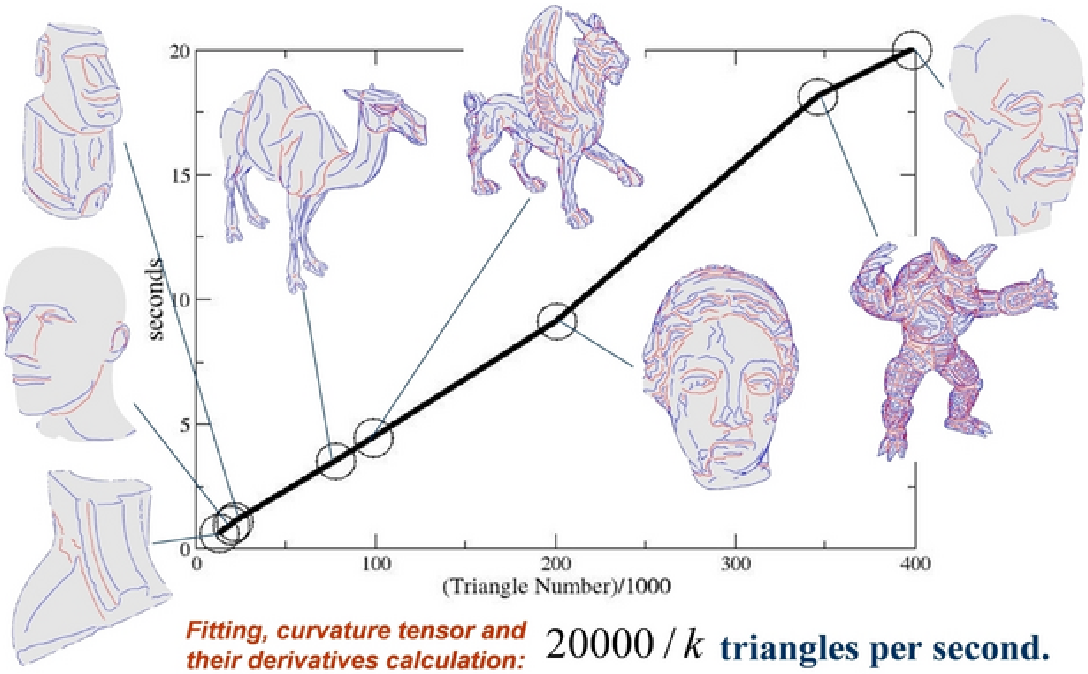


Figure 4.16: Timings. Our method requires linear computational complexity which is relatively low compared with global implicit fitting methods. Also our curvature derivatives formula (4.8) dramatically reduces actual computational time.

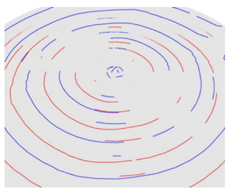
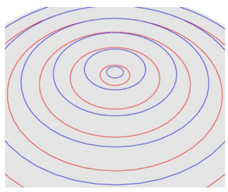
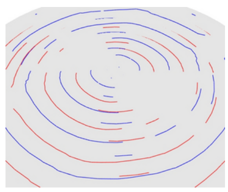
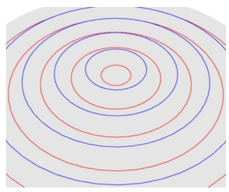
	Without Smoothing				With Smoothing			
	Without Normal		With Normal		Without Normal		With Normal	
								
	L^2	L^∞	L^2	L^∞	L^2	L^∞	L^2	L^∞
$e_{\max}$	0.306	0.926	0.116	0.415	0.307	0.926	0.145	0.479
$e_{\min}$	0.299	0.926	0.141	0.852	0.298	0.926	0.126	0.563

Table 4.1: Numerical error comparison with the exact detection of the crest lines on analytical waving surface $\mathbf{r}(u, v) = [u \cos v, u \sin v, \cos u]$ where one-ring neighborhood polynomial fitting with and no filtering is used for all the models ($k = 1$). Here L^2 and L^∞ errors of $e_{\max}$ and $e_{\min}$ approximated by our method (most right image and errors) are measured for the analytical surface with (without) use of our preliminary smoothing $\mathbf{p} \to \mathbf{p}'$ and normals for polynomial fitting.

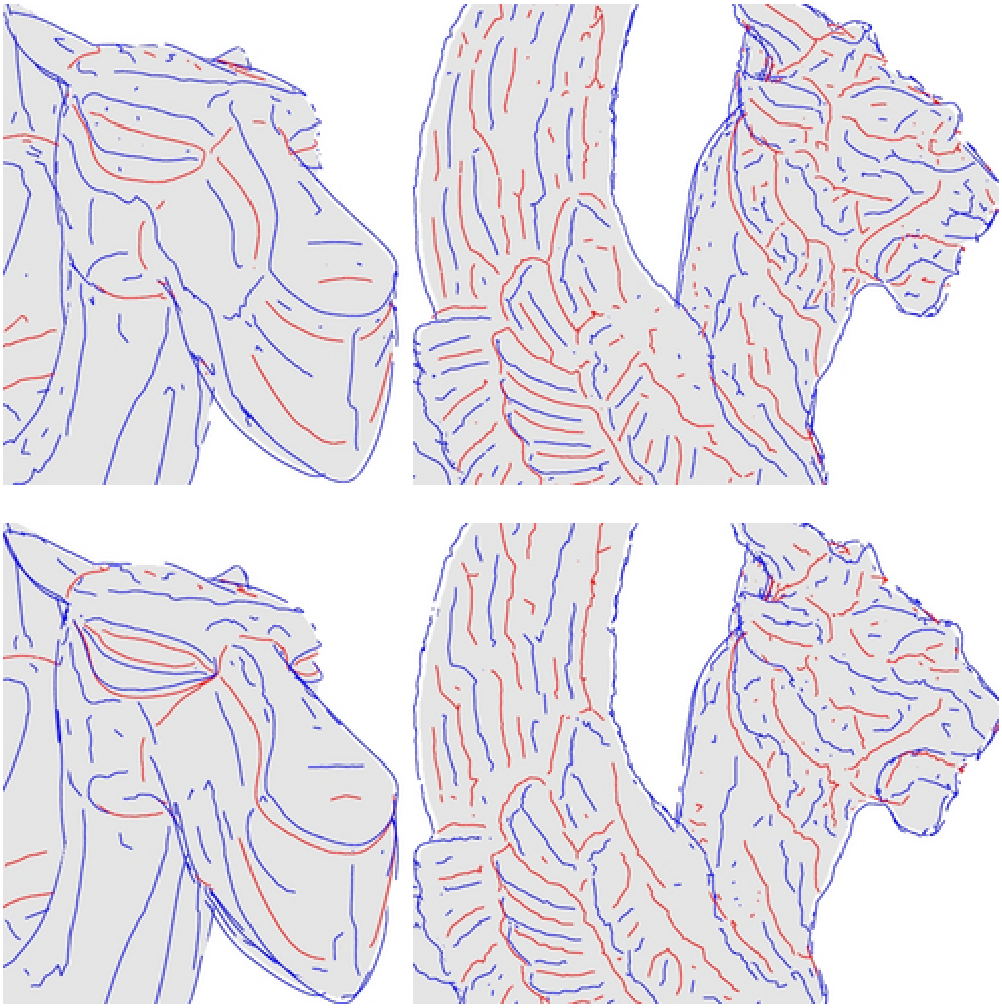


Figure 4.17: Comparison with global fitting method. Top: Crest lines detected using a global implicit fitting method [OBS04]. Bottom: Crest lines detected using the method of this paper, one-ring neighborhood fitting is used. In both the cases, no filtering is applied.

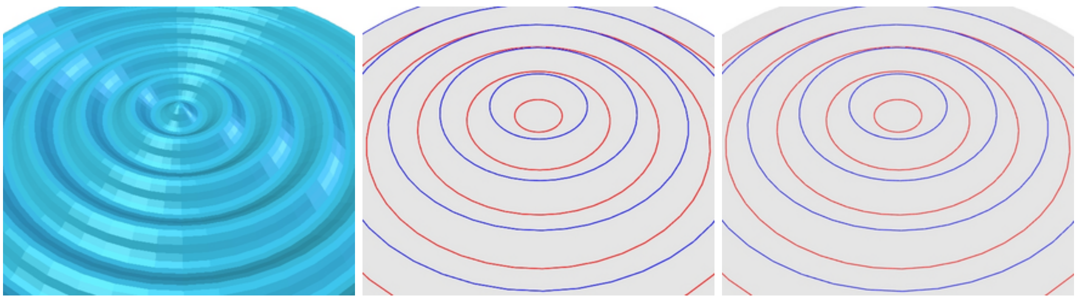


Figure 4.18: Comparison with exact crest lines. Left: Input a simple analytical surface. Center: Exact crest lines on the analytical surface. Right: The crest lines detected with our method, one-ring neighborhood fitting is employed.

4.8 Crest Lines and Mesh Simplification

In this section, we present a quadric-based mesh simplification procedure guided by the distance field from crest lines. Our use of crest lines for adaptive mesh simplification purposes is inspired by recent work [KG03]. Since crest lines on a mesh are important shape features, it is natural to simplify the mesh aggressively far from the most salient crest lines and preserve the mesh in a vicinity of them.

Given a set of feature lines (crest lines, in our case) on surface $\mathcal{S}$, following [LPRM02] for a surface point $\mathbf{p} \in \mathcal{S}$ we consider $d(\mathbf{p})$ the geodesic distance between $\mathbf{p}$ and the closest feature line (crest line) point. Let $\max(d)$ be the maximum of the geodesic distances $d(\mathbf{p})$ over all points of $\mathcal{S}$. We introduce a scale-independent weighted distance function

$$F(d) = \left(\frac{d}{\max(d)} + \epsilon\right)^{\eta}, \tag{4.12}$$

where ϵ is a regularization parameter (in all our experiments we use $\epsilon = 0.1$) and η is a positive user-specified parameter which is used to control a degree of influence of the crest lines.

Figure 4.19 describes our feature sensitive mesh simplification framework.

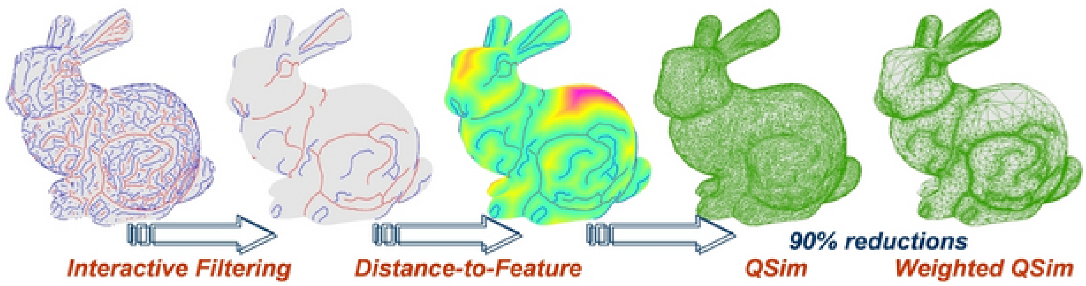


Figure 4.19: Feature sensitive mesh simplification framework. Right two images show the results of the 90%-decimated Stanford bunny models via QSim [GH97] and our weighted QSim ($\eta = 6$), respectively.

Once the crest lines are detected and filtered, we compute a discrete feature distance d_i for each triangle $T_i \in \mathcal{M}$. Let us define the distance between two triangles T_j and T_i of $\mathcal{M}$ sharing a common edge as the sum of distances between the triangle centroids and the edge midpoint. To compute $\{d_i\}$ we use a variant the Floyd-Warshall all-pairs shortest path algorithm.

Figure 4.22 visualizes the distance fields computed on the Max-Planck and Stanford bunny meshes.

Similar to [KG03] a weighted quadric error metric $w_j Q(T_j)$ is assigned to each triangle T_j of mesh $\mathcal{M}$, where $Q(T_j)$ is the standard Garland-Heckbert QEM [GH97]. We set $w_j = 1/F(d_j)$ and control the degree of influence of crest lines via parameter η in (4.12). Figures 4.20 and 4.21 present the Max-Planck and Stanford bunny meshes with their eye region 90%-decimated for various values of η. The detected crest lines are those shown in the most-right images of Figure 4.22. The mesh density is changing smoothly according to geodesic distance to the crest lines.

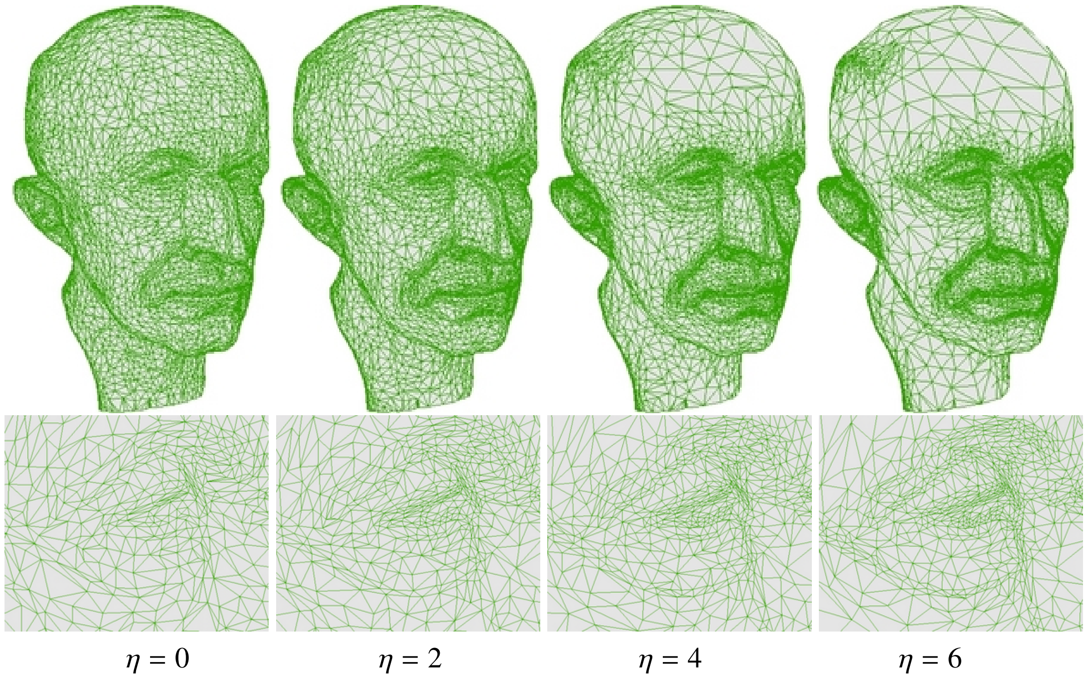


Figure 4.20: Max-Planck mesh and its eye part 90%-decimated for various values of η. The left image ($\eta = 0$) shows the result of the standard Garland-Heckbert decimation procedure. The original mesh, its crest lines, filtered crest lines, and its distance field are visualized in the top images of Figure 4.22.

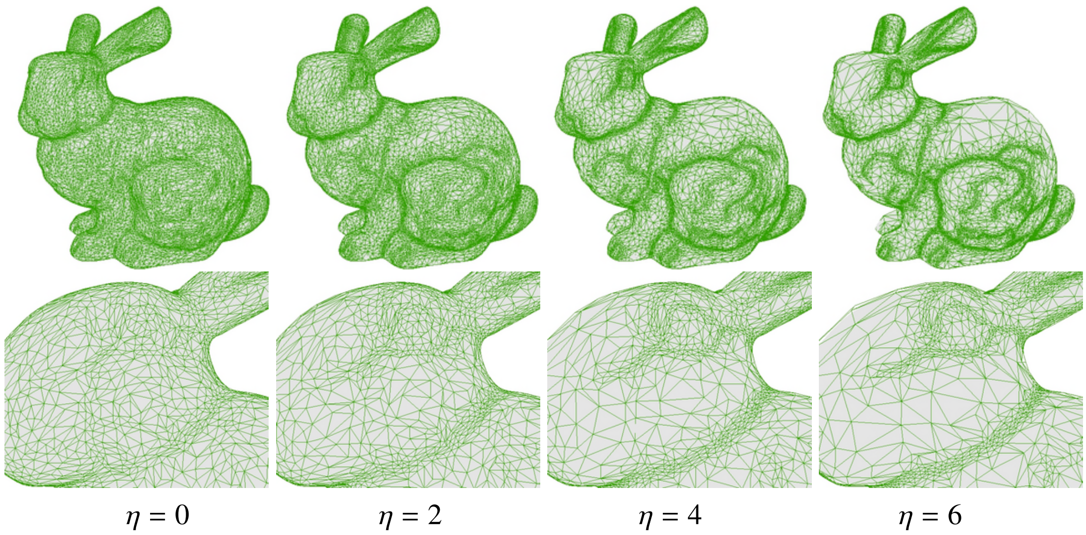


Figure 4.21: Stanford bunny (remeshed model used in Figure 4.7) mesh and its eye part 90%-decimated for various values of η. The left image ($\eta = 0$) shows the result of the standard Garland-Heckbert decimation procedure. The original mesh, its crest lines, filtered crest lines, and its distance field are visualized in the bottom images of Figure 4.22.

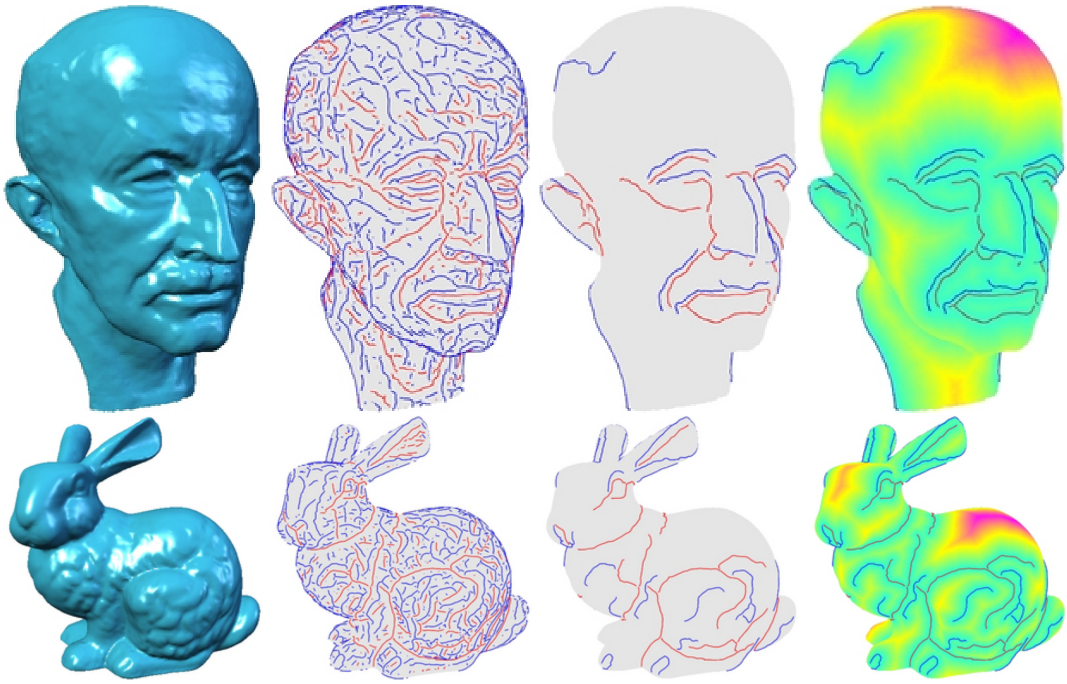


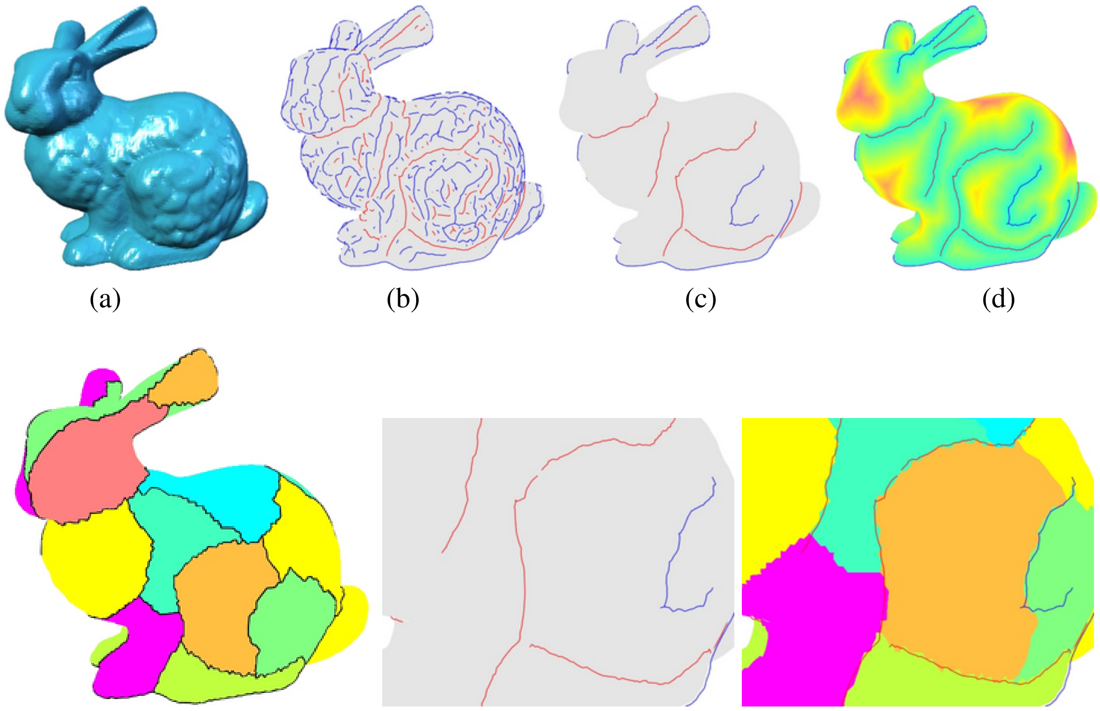
Figure 4.22: Distance from salient crest lines is visualized for Max-Planck $T = 40$ and Stanford bunny $T = 18.9$ (remeshed model used in Figure 4.7) meshes. The crest lines are found with three-ring neighborhood fitting ($k = 3$) for both the models.

4.9 Crest Lines and Mesh Segmentation

In this section, we develop a framework for mesh segmentations guided by using the distance field (4.12) as a weight function of a region growing algorithm. Region growing techniques are widely used in mesh segmentations [LPRM02, SWG⁺03, CSAD04]. Figure 4.23 demonstrates the mesh segmentation result of conformal atlas generation (CAG) proposed in [LPRM02]. Here our crest lines are employed for feature extraction phase of [LPRM02] in Figure 4.23. Although the CAG with our crest lines produces nice segmentation results in natural objects as shown in Figure 4.23, the hierarchical face clustering (HFC) [GWH01] and variational shape approximation (VSA) [CSAD04] generate much better segmentations for shapes constructed by a set of planar regions. See Figure 4.24 for an example of segmenting mechanical objects. In order to partition natural and mechanical objects well by a unified approach, we propose a novel weighting scheme for VSA. It is a Lloyd partitioning type region growing algorithm. The generated segments form a centroidal Voronoi diagram on the mesh.

Denote C_k be a segment which is a set of connected triangles. Let ∂C_k be the boundary triangles of C_k. We use the so-called $L^{2,1}$ error metric of VSA which is a weighted difference between normals of C_k and a triangle T_j. Here triangle T_j does not belong to C_k but shares at least a common edge with a triangle $T_i \in \partial C_k$. A local weight $w_{ij} = 2/(F(T_i)+F(T_j))$ is assigned to each pair of triangles. Then our error metric of the VSA proxy is defined by $w_{ij}L^{2,1}(T_j, C_k)$: $T_i \in \partial C_k$ where $F(\cdot)$ is given by the equation (4.12). This weighting method depends on order of growing which is different from the weighting scheme suggested in [CSAD04]. Similar as the previous section, we control the degree of influence of crest lines via parameter η in (4.12). Figure 4.25 describes our feature-guided mesh segmentation framework.

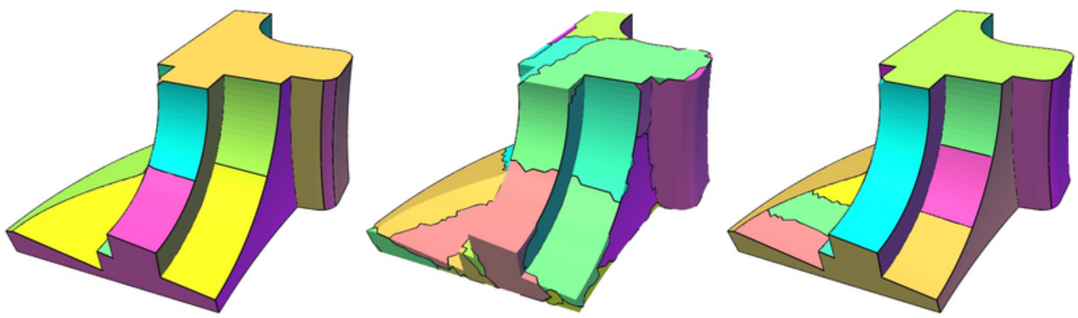
Figures 4.26 and 4.27 present the Stanford bunny and Max-Planck meshes with 25 and 80 segments for various values of η, and also comparisons with HFC and CAG (using our crest lines). The detected crest lines are those shown in the top images of Figures 4.23 and 4.22, respectively. The segmentations are changing and adapting smoothly according to geodesic distance to the crest lines.



(a) (b) (c) (d)

(e) (f) (g)

Figure 4.23: Segmentations on natural object. The original Stanford bunny mesh (a) is partitioned into 25 segments (e) by using our extracted and filtered crest lines (b) and (c) ($k = 3$ and $T = 41$) for feature extraction phase of [LRPM02]. The corresponding distance filed is visualized in (d). Images (f) and (g) represent the magnifications of (c) and (e), respectively.



HFC [GWH01] CAG [LPRM02] VSA [CSAD04]

Figure 4.24: Segmentations on mechanical object (25 segments). Hierarchical face clustering [GWH01] and Variational shape approximation [CSAD04] are much powerful methods to partition a mesh into a set of planar segments compared with conformal atlas generation [LRPM02].

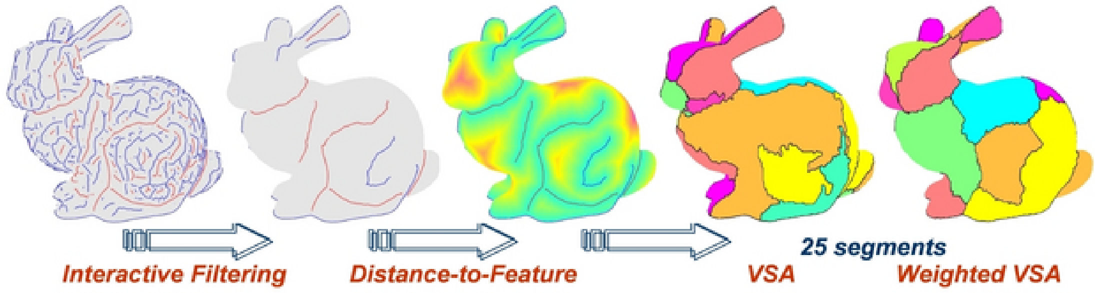


Figure 4.25: Feature-guided mesh segmentation framework. Right two images show the results of the 25 segmented Stanford bunny models via VSA [CSAD04] and our weighted VSA ($\eta = 6$), respectively.

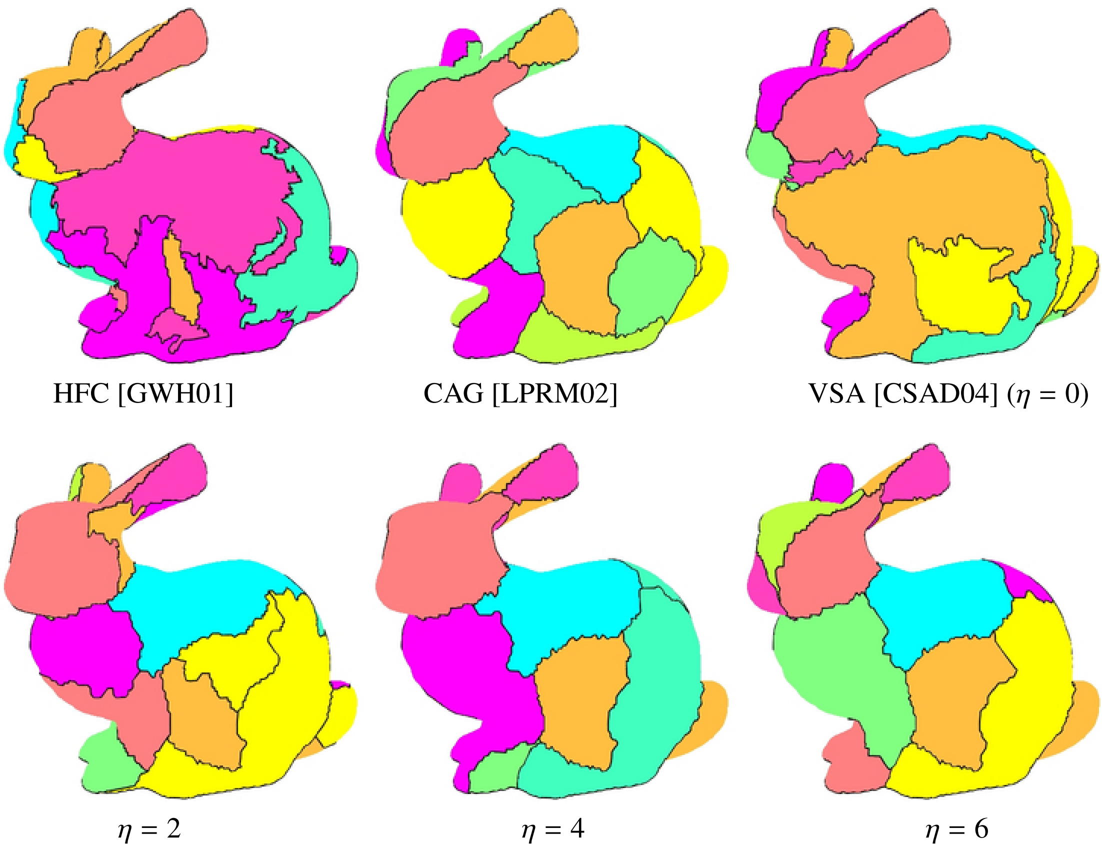
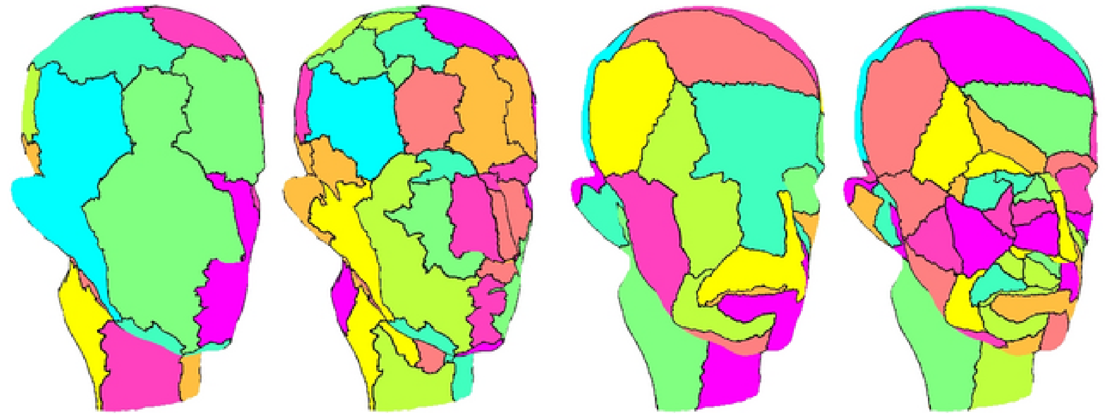
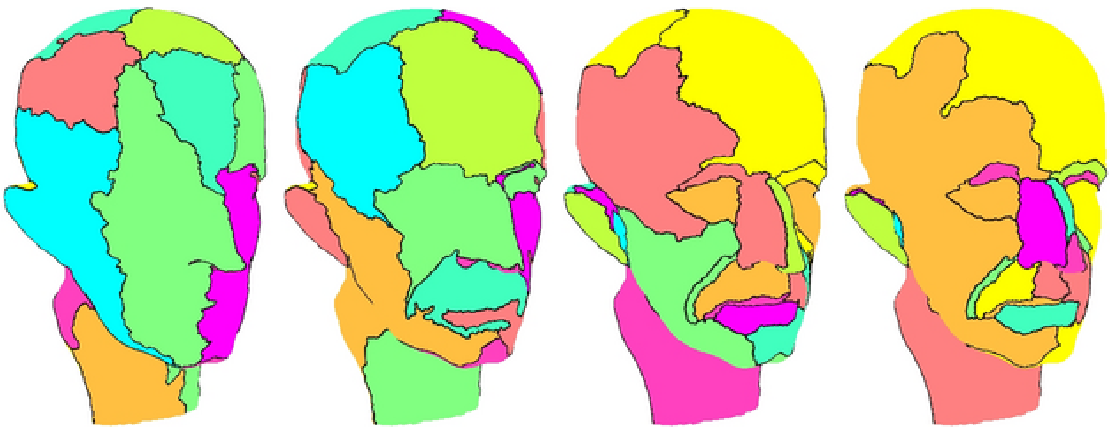


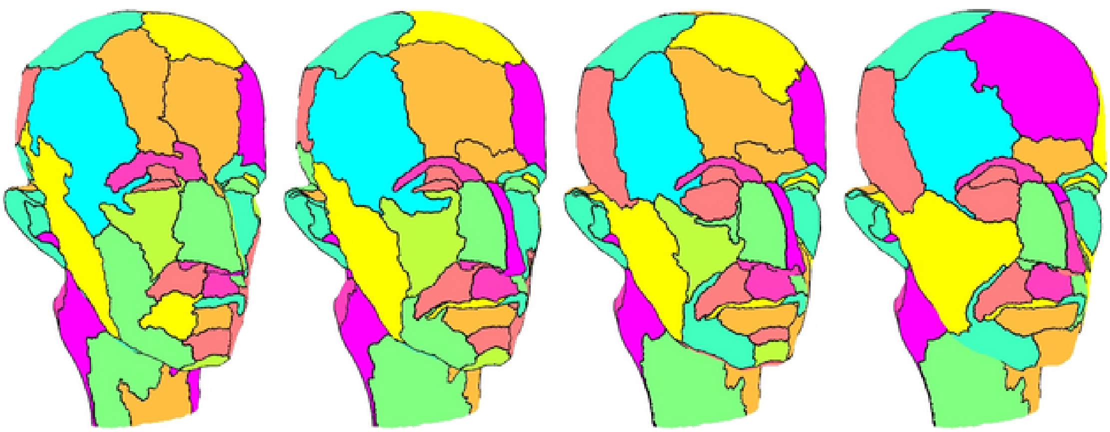
Figure 4.26: Feature-guided mesh segmentations. The 25 segmented original Stanford bunny mesh for various values of η. The top-right image ($\eta = 0$) shows the result of the standard variational shape approximation procedure. The original mesh, its crest lines, filtered crest lines, and its distance field are visualized in the top images of Figure 4.23. The top-left and top-center images show the 25 segmented original Stanford bunny mesh via HFC [GWH01] and CAG [LPRM02], respectively. Our filtered crest lines shown in the image (c) of Figure 4.23 are employed for the CAG.



HFC [GWH01], 25 and 80 segments. CAG [LPRM02], 25 and 80 segments.



VSA [CSAD04] ($\eta = 0$) $\eta = 2$ $\eta = 4$ $\eta = 6$



VSA [CSAD04] ($\eta = 0$) $\eta = 2$ $\eta = 4$ $\eta = 6$

Figure 4.27: Feature-guided mesh segmentations. The 25 (middle images) and 80 (bottom images) segmented Max-Planck meshes for various values of η. The middle-left and bottom-left images ($\eta = 0$) show the results of the standard variational shape approximation procedure. The original mesh, its crest lines, filtered crest lines, and its distance field are visualized in the top images of Figure 4.22. The top-left and top-right images represent the resulting segmentations via HFC [GWH01] and CAG [LPRM02], respectively.

4.10 Summary of Salient Feature Detection

We have presented a fast and robust method for detecting salient curvature extrema on surfaces approximated by dense triangle meshes. The method is based on approximating principal curvatures and their derivatives by the novel local polynomial fitting procedure. Contributions of our method include a new curvature derivative formula (4.8) and the smart thresholding based on cyclideness in order to remove spurious ridges and crest lines. The results of our crest line detection procedure depends only slightly on the quality of the mesh. Our method is capable of achieving high quality results in detecting salient curvature extrema to compare with schemes based on global fitting procedures.

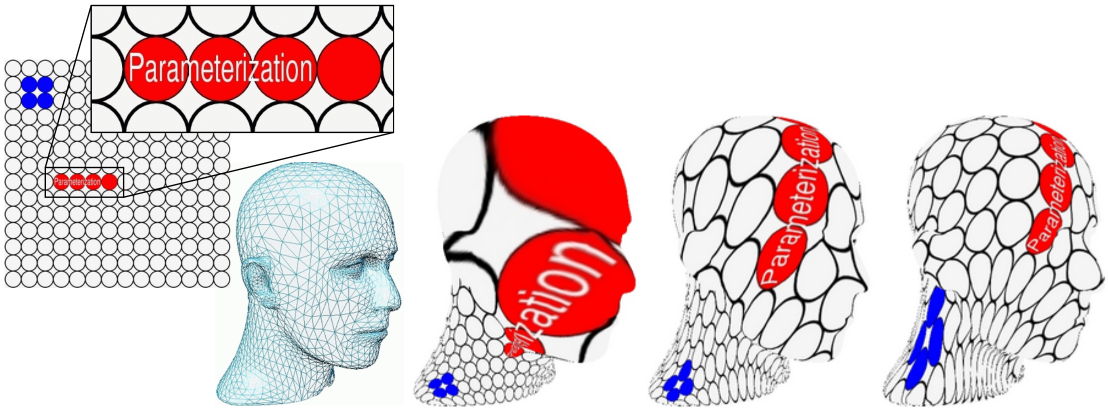
Our filtering scheme for removing unessential crest lines is based on interesting relationships between Dupin cyclides, focal sets, curvature extrema, and variational functionals. We use cyclideness (4.11) as the main ingredient of our filtering scheme and measure the strength of crest lines by scale-independent quantity (4.10). Thus long but weak crest lines are preferred to strong but short ones. Of course, different filtering procedures can be also used instead of that based on (4.10). Similar manual thresholding schemes were also used in [OBS04, CP04a]. Manual filtering is hardly avoidable for complex geometry surfaces, since the crest lines are local surface features while saliency-based thresholding should take into account global surface shape.

The source code of our method is available on the Web for evaluation [YBS05a].

Applications of the crest lines for adaptive mesh simplification and feature-guided mesh segmentation are also considered by using geodesic distance to the crest lines.

5

Fast Low-Stretch Mesh Parameterization



(a) (b) (c) (d)

Figure 5.1: Texture mapping of the Mannequin Head model with three mesh parameterizations used in our method. (a): Texture and model. (b): Floater's shape preserving parameterization [Flo97] is used as an initial mesh parameterization. (c): After a single optimization pass. (d): Our optimal low-stretch parameterization.

Map projections have been studied and known since before Geography [Pto91] of Greek geographer *Claudius Ptolemaeus* (Ptolemy, about 100-170 A.D.) who already introduced iso-lines: longitude and latitude for projecting a sphere onto a flat domain. Although famous projections which include orthogonal, stereographic, Mercator's, and Lambert's were developed for drawing the earth on planar maps in order to travel around the world, it became possible to investigate mappings via sophisticated mathematics after developing differential and Riemannian geometries by *Carl Friedrich Gauß* (1777 - 1855) and *Georg Friedrich Bernhard Riemann* (1826 - 1866), see Chapters 5-2 and 5-3 of [Str88]. Conformal and equiareal mappings preserve angles and areas, respectively. If a mapping is conformal and equiareal then the mapping is isometric, i.e. it preserves distances, areas, and angles. However the only developable surfaces are isometric to the plane.

Instead of non-planar meshes, we deal with a planar parameterization for a triangle mesh approximating a smooth surface, a bijective mapping between the mesh and a triangulation of a planar polygon. A non-planar mesh can be always decomposed into a set of planar meshes by using the mesh segmentation methods proposed in Section 4.9, the conventional charting [LPRM02], partitioning [CSAD04, YGZS05, JKS05], or cutting [GGH02, GWY03, WGMY05] techniques. Our goal is to generate low-distortion planar mesh parameterizations. An excellent survey of recent advances in mesh parameterization is given in [FH04], see also references therein. While various algorithms are developed for mesh parameterization approaches based on solid mathematical theories (e.g., conformal mappings), effective computational schemes for generating practically important low-stretch mesh parameterization [SSGH01] (and also similar stretch-based mesh parameterizations [SGSH02, TSS$^+$04, ZMT05]) have not yet been proposed.

In this Chapter, we follow [YBS04, YBS05b] and present a simple and fast method for generating low-stretch mesh parameterizations. Our approach is based on a moving mesh approach, a popular grid adaption technique in computational mechanics. Instead of minimizing the nonlinear stretch energy of [SSGH01], we improve the parameterization gradually from an initial parameterization by equalizing local stretches over the mesh. The proposed optimization procedure does not generate triangle flips if the boundary of the parameter domain is a convex polygon. Moreover already the first optimization step produces a high-quality mesh parameterization. Figure 5.1 shows the three stages of our mesh parameterization method: generating an initial parameterization, our single-pass low-stretch parameterization, and the optimal low-stretch parameterization. We compare our parameterization procedure with several state-of-art mesh parameterization methods and demonstrate its speed and high efficiency in parameterizing large and geometrically complex models. Also application to efficient remeshing is developed in Section 5.4 by using two mesh parameterizations: quasi-conformal and low-stretch.

5.1 Mapping Distortions and Computational Difficulties

Consider a surface $\mathbf{S}(u, v) \in \Re^3$ topologically equivalent to a disk and given parametrically by $\mathbf{S}(u, v) = (x(u, v), y(u, v), z(u, v))$. The Jacobian matrix corresponding to the mapping $f : (u, v) \to \mathbf{S}(u, v)$ is given by a 2x3 matrix $J = (\mathbf{S}_u, \mathbf{S}_v)$. The Jacobian J determines all the first-order geometric properties of the parameterization f, including the area, angle, and length distortions caused by the mapping f.

Denote by $\Gamma(u, v)$ and $\gamma(u, v)$ the maximal and minimal singular values of J. Then Γ^2 and γ^2 are the eigenvalues of the metric tensor

$$J^T J = \begin{bmatrix} E & F \\ F & G \end{bmatrix}.$$

It is convenient to use Γ and γ for measuring various properties of $\mathbf{p}$. For example, if $\Gamma(u, v) = \gamma(u, v)$, the parameterization is conformal and mapping f preserves angles.

Since the conformal mappings are well understood mathematically, discrete approximations of harmonic and conformal mappings are widely used for mesh parameterization purposes [HG99, HAT+00, SU01, LPRM02, GY03, KSS06] in computer graphics and [TWM85, Lis04] in grid generation. See [Ahl66, Dur04] for mathematical theory of the harmonic, conformal, and quasi-conformal mappings. Since the pioneering works [EDD+95, Flo97], the so-called convex combination methods are insensibly studied [Gus02, DMA02, Flo03] because we can construct the valid one-to-one mapping by simply solving a sparse system of linear equations. However conformal mappings often produce high stretch regions where texture mappings have severe undersampling artifacts.

In computer graphics, [MYV93] first proposed to use a linear combination of angle and area error terms to define the mapping distortion, see also [FSD99, AHTK99] where similar distortion metrics were employed to flatten meshes approximating medical surfaces. Numerous discrete parameterization papers have been published also in grid generation based on a linear combination of angle and area/volume related terms [Res68, Hua01, CHR03].

Stretch Distortion. It is natural to measure the local stretch of mapping f by $\sqrt{(\Gamma^2 + \gamma^2)/2} = \sqrt{(E + G)/2}$. Stretch minimizing mesh parameterization was first proposed by Sander et al. [SSGH01]. It has been developed and employed for many other parameterization techniques which include signal-specialized parameterizations [SGSH02, TSS+04], a spherical parameterization [PH03], charting [SCOGL02, ZSGS04], and geometry images [GGH02, SWG+03, CHCH06]. See also [ZMT05] where the Green-Lagrange tensor is used to measure the stretch. In [DMK03] authors employed a multiplication of angle and area error terms, and latter [FSD05] indicated that the stretch distortion can be represented by a multiplication of the Dirichlet energy and area error term. These distortion metrics of quasi-isometry type parameterizations are highly nonlinear and difficult to obtain optimal solutions. While the stretch minimization approach proposed in [SSGH01] and further developed in [SGSH02, ZMT05, TSS$^+$04] leads to generating high-quality mesh parameterizations, the computational procedure used in these methods for stretch minimization is time consuming. Besides the mesh parameterization procedure of [SSGH01, SGSH02] often generates regions of high anisotropic stretch, consisting of slim triangles. Such the regions on a parameterized and textured mesh look like cracks and we call them *parameter cracks*. Figure 5.2 demonstrates an appearance of such parameter cracks on the textured Mannequin Head model parameterized by the stretch minimization method from [SSGH01].

In [PH03] the authors propose to add a regularization term to the stretch energy in order to avoid parameter cracks. The term depends on two user-specified parameters. Besides minimizing the resulting energy does not produce a minimal stretch parameterization.

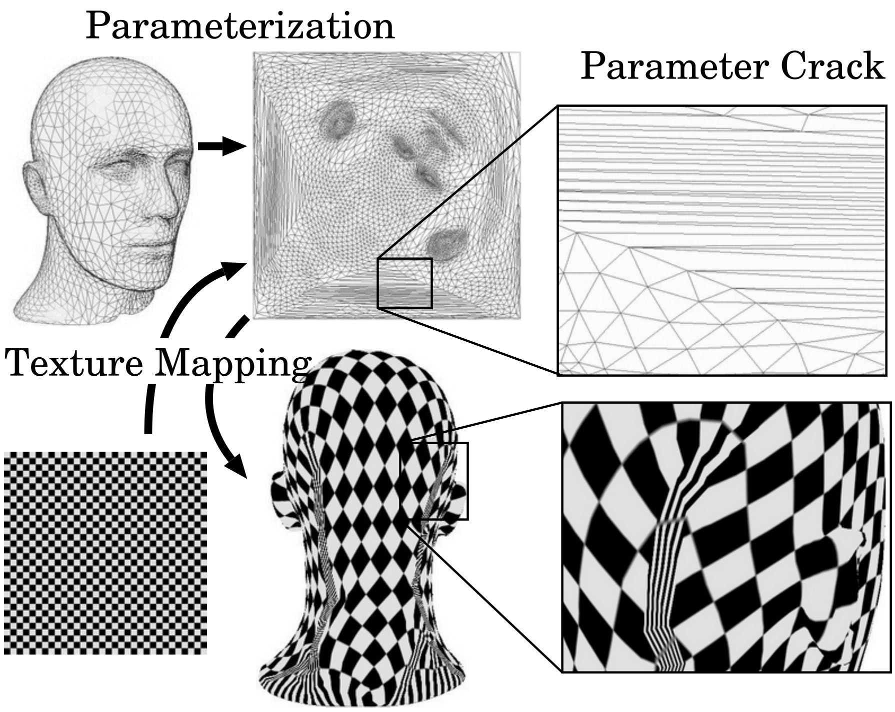


Figure 5.2: Parameter cracks on textured Mannequin Head model parametrized by the stretch minimization method of Sander et al [SSGH01].

Given a triangle mesh, we first construct an initial mesh parameterization as mapping and then improve the parameterization gradually: at each improvement step we optimize the parameterization generated at the previous step. The optimization is achieved by minimizing a weighted quadratic energy with positive weights chosen to minimize the parameterization stretch. Thus the single optimization step is fast since it is based on solving a sparse system of linear equations. A few optimization steps are enough to obtain the optimal low-stretch mesh parameterization, therefore, our method is extremely faster than the nonlinear optimization methods include hierarchical algorithms [HGC99, FH02, DMK03, RL03, Kan04, SLMB05]. Besides if the boundary of the parameterization domain forms a convex polygon, triangle flips never happen [Flo97] according to Tutte's theorem [Tut63].

Our method can be considered as an error redistribution (diffusion) procedure applied to local stretches. The error redistribution (also known as the moving mesh method or r-method) is a powerful mesh adaption technique in computational mechanics (see, for example, [LTZ01, CHR03, Lis04] and references therein). It has become popular after seminal works of De Boor [DB73] and Babuška and Rheiboldt [BR78]. The general idea behind the approach is extremely simple let us move mesh vertices to positions where they are mostly needed. Obviously this leads to error equalization w.r.t. a user-specified error measure (energy) often called a monitor function in computational mechanics studies, see [BS82, CD85, Hua01] for adaptive zoning, grading functions, and mesh redistribution. Error equalization resembles a diffusion process and can be governed by a system of partial differential equations [Win67, Win81, HH03]. In the geometric modeling field, it generalizes Laplacian smoothing and similar ideas were used for mesh parameterization purposes [YBS04, YBS05b] and optimizing texture maps [SDS02, BTB02]. Our error equalization technique of [YBS04, YBS05b] has been successfully employed for not only the mesh parameterizations [ZRS05a, ZRS05b, YYSZ06] but also texture mapping (geometry image) [WGMY05, CHCH06], morphing [SK04], meshing point clouds [ZG04b], cloth simulation [WTY05], and feature extraction [NNS06] in computer graphics and geometry processing.

We compare our low-stretch mesh parameterization procedure with several state-of-art mesh parameterization methods and demonstrate its speed and high efficiency in parameterizing large and geometrically complex models. Besides we show how our mesh parameterization approach can be combined with the interactive geometry remeshing scheme of Alliez et al. [AMD02] in order to achieve fast and high quality remeshing.

5.2 Fast Low-Stretch Mesh Parameterization

Given a parametrized triangle mesh $\mathcal{M} \in \Re^3$, consider a mesh triangle $T = \langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle \in \mathcal{M}$ and its corresponding triangle $U = \langle \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \rangle$ in the parametric plane $\Re^2_{u,v}$. Triangles $\{U\}$ define a planar mesh $\mathcal{U} \in \Re^2_{u,v}$ and the parameterization of $\mathcal{M}$ is given by one-to-one mapping between meshes $\mathcal{U}$ and $\mathcal{M}$. The correspondence between the vertices of T and U uniquely defines an affine mapping $P : U \to T$. Let us denote by $\Gamma(T)$ and $\gamma(T)$ the maximal and minimal eigenvalues of the metric tensor induced by the mapping [SSGH01, ZMT05]. As we mentioned above, quantity

$$\sigma(U) = \sqrt{(\Gamma^2 + \gamma^2)/2}$$

characterizes the stretch of mapping P.

For each vertex $\mathbf{u}_i$ in the parameter domain let us define its stretch $\sigma_i = \sigma(\mathbf{u}_i)$ by

$$\sigma_i = \sqrt{\sum A(T_j)\sigma(U_j)^2 \Big/ \sum A(T_j)} \tag{5.1}$$

where $A(T)$ denotes the area of triangle T and the sums are taken over all triangles T_j surrounding mesh vertex $\mathbf{p}_i$ corresponding to $\mathbf{u}_i$.

Our method to build a low stretch mesh parameterization consists of several steps. First we construct an initial mesh parameterization using the Floater approach [Flo97]: the boundary vertices of mesh $\mathcal{M}$ are mapped into the boundary vertices of $\mathcal{U}$ which form a polygon in the parameter plane $\mathbb{R}^2_{u,v}$ and for each inner vertex $\mathbf{p}_i$ of $\mathcal{M}$ its corresponding vertex $\mathbf{u}_i$ inside the polygon is selected such that the following local quadratic energy

$$E(\mathbf{u}_i) = \sum\nolimits_j w_{ij} \|\mathbf{u}_j - \mathbf{u}_i\|^2, \tag{5.2}$$

achieves its minimal value. Here $\{\mathbf{u}_j\}$ are vertices corresponding to the mesh one-link neighbors of $\mathbf{p}_i \in M$ and $\{w_{ij}\}$ are positive weights. Now the optimal positions for $\mathbf{u}_i$ are found by solving a sparse system of linear equations

$$\sum\nolimits_j w_{ij}\left(\mathbf{u}_j - \mathbf{u}_i\right) = 0. \tag{5.3}$$

This computationally simple procedure produces a valid parameterization of mesh $\mathcal{M}$ and avoids triangle flips if the boundary of $\mathcal{U}$ is a convex polygon [Tut63, Flo97].

Notice that modifying weights $\{w_{ij}\}$ in quadratic energy (5.2) and, consequently, in (5.3) modifies the mesh parameterization. Thus one can improve the mesh parameterization initially determined by (5.3) with weights $\{w_{ij}^{\text{old}}\}$ via selecting better weights $\{w_{ij}^{\text{new}}\}$. In our mesh optimization procedure, we exploit this simple observation and choose weights $\{w_{ij}^{\text{new}}\}$ such that vertices $\{\mathbf{u}_j\}$ are moved toward locations where they are mostly needed.

Let us estimate local stretch $\sigma_i = \sigma(\mathbf{u}_i)$ for each inner vertex $\mathbf{u}_i$ in the parametric plane. We redistribute the local stretches by assigning

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} \big/ \sigma_j \tag{5.4}$$

in (5.2). The new positions of $\{\mathbf{u}_i\}$ are now found by solving (5.3).

We can think about vertices $\{\mathbf{u}_i\}$ and corresponding energies (5.2) in terms of a mass-spring system. For an area preserving parameterization, if a high (low) stretch is observed at $\mathbf{u}_i$, that is $\sigma_i > 1$ ($\sigma_i < 1$), we relax (strengthen) the springs connected with $\mathbf{u}_i$ by solving (5.3) with new weights (5.4). It works similarly for a general parameterization.

Our idea to diffuse the local stretches iteratively by (5.1), (5.3), (5.4) resembles mesh moving techniques discussed in the previous section.

We start from an initial parameterization $\mathcal{U}^0 = \{\mathbf{u}_i^0\}$ and then improve it gradually: $\mathcal{U}^{h+1} = \{\mathbf{u}_i^{h+1}\}$ is obtained from $\mathcal{U}^h = \{\mathbf{u}_i^h\}$ by solving (5.3) with weights w_{ij}^{h+1} defined by

$$w_{ij}^{h+1} = w_{ij}^h \big/ \sigma\left(\mathbf{u}_j^h\right).$$

We select w_{ij}^0 as the shape preserving weights proposed by Floater [Flo97]. The boundary vertices of the evolving mesh $\mathcal{U}^h$, $h = 0, 1, 2, \ldots$, remain fixed. When solving (5.3) with $w_{ij} = w_{ij}^{h+1}$ numerically we use $\mathcal{U}^h$ as the initial guess for the numerical solver we employ.

We use the L^2 stretch metric of Sander et al. [SSGH01]

$$E_s^h = E_s(\mathcal{U}^h) = \sqrt{\sum A(T)\sigma(U^h)^2 \big/ \sum A(T)}, \tag{5.5}$$

where the sums are taken over all the triangles T of mesh $\mathcal{M}$, to define a stopping criterion. Namely, if $E_s^{h+1} \geq E_s^h$ we consider $\mathcal{U}^{\text{opt}} = \{\mathbf{u}_i^h\}$ as an optimal low stretch mesh parameterization.

Besides $\mathcal{U}^{\text{opt}}$ we also consider $\mathcal{U}^1 = \{\mathbf{u}_i^1\}$, the mesh parameterization obtained after one step of our optimization procedure since, according to our experiments, already the first step dramatically improves the parameterization quality.

We also can vary the strength of stretch redistribution (diffusion) step (5.4) by using the weights $\{\sigma_i^\eta\}$, $0 < \eta \leq 1$, instead of $\{\sigma_i\}$ in (5.4):

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} \Big/ \sigma_j^\eta. \tag{5.6}$$

Using (5.6) with $\eta < 1$ slows down the stretch minimization process but, on the other hand, often improves the mesh parameterization quality. The influence of exponent η in (5.6) is demonstrated in Figure 5.3 for our single-step parameterization $\mathcal{U}^1$. Choosing smaller values for η leads to a less aggressive stretch minimization.

In the next section, we compare $\mathcal{U}^1$ and $\mathcal{U}^{\text{opt}}$ with results produced by conventional mesh parameterization schemes.

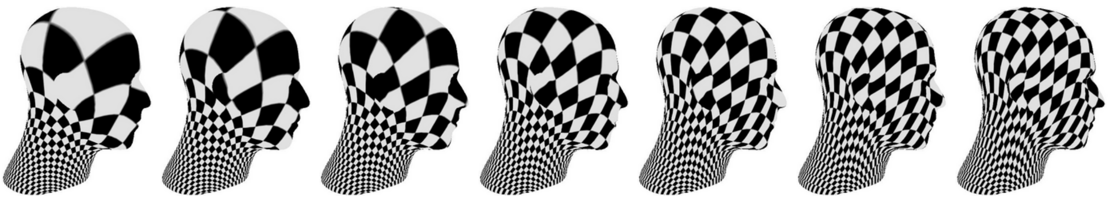


Figure 5.3: Choosing smaller values for η leads to a less aggressive stretch minimization. From left to right: $\mathcal{U}^1$ parameterization of Mannequin Head with $\eta = \{0, 0.1, 0.2, 0.4, 0.6, 0.8, 1\}$.

5.3 Low-Stretch Parameterization: Results and Comparisons

Computing. All the examples presented in this Chapter are computed by using gcc 2.95 C++ compiler on a 1.7GHz Pentium 4 computer with 512MB RAM. To solve a system of linear equation $\mathbf{Ax} = \mathbf{b}$ we use PCBCG [PTVF88] with the maximum number of iterations equal to 10^4 and the approximation error $|\mathbf{Ax} - \mathbf{b}| / |\mathbf{b}|$ set to 10^{-6}. Note that we can also use the recent developments of the sparse direct solvers e.g. [Dav04] instead of PCBCG.

Error Metrics. To evaluate the visual quality of a parameterization we use the checkerboard texture shown in the bottom-left image of Figure 5.2. For a quantitative evaluation of various mesh parameterization methods we employ L^2 stretch metric (5.5) and consider edge, angle, and area distortion error functions defined below. To measure the edge distortion error we use

$$\sum \left| \frac{|\mathbf{p}_i - \mathbf{p}_j|}{\sum |\mathbf{p}_i - \mathbf{p}_j|} - \frac{|\mathbf{u}_i - \mathbf{u}_j|}{\sum |\mathbf{u}_i - \mathbf{u}_j|} \right|,$$

where the sums are taken over all the edges of meshes $\mathcal{M}$ and $\mathcal{U}$. The angle distortion error is defined by

$$\frac{1}{3F} \sum_j \sum_{i=1}^{3} \left|\theta_{j,i} - \phi_{j,i}\right|,$$

where the sums are taken over all the angles $\theta_{j,i}$ and $\phi_{j,i}$ of the triangles of meshes $\mathcal{M}$ and $\mathcal{U}$, respectively, and F is the total number of triangles (faces) of $\mathcal{M}$. The area distortion is measured by

$$\sum \left| A(T_j) / \sum A(T_j) - A(U_j) / \sum A(U_j) \right|,$$

where the sums are taken over all the triangles of meshes $\mathcal{M}$ and $\mathcal{U}$.

Comparison and Evaluation. We have implemented a number of conventional mesh parameterization methods and compared them with our low stretch technique:

(a)	Eck et al. harmonic map [EDD+95]
(b)	Floater's shape preserving parameterization [Flo97]
(c)	Desbrun et al. intrinsic parameterization [DMA02]
(d)	Sander et al. stretch minimizing parameterization [SSGH01]
(e)	Our single-step parameterization $\mathcal{U}^1$
(f_h)	Our optimal parameterization $\mathcal{U}^{\text{opt}}$

The subindex h in (f_h) in the bottom row of the above table shows the total number of optimization steps (5.3), (5.4) needed to generate $\mathcal{U}^{\text{opt}}$.

Tables 5.1-5.12 and Figures 5.14-5.16, and 5.4 present qualitative and visual comparisons of the above mesh parameterization schemes tested on various models topologically equivalent to a disk. The unit square is used as the parameter domain and for each models its the boundary vertices are fixed on the boundary of the square. The errors and computational times measured in seconds (s) and sometimes in minutes (**m**) and hours (**h**) are given.

For the intrinsic parameterization method [DMA02], we use the equal blending of the Dirichlet and Authalic energies for all the models, except for the Fish model (Table 5.11) where we use only the Dirichlet energy in order to avoid triangle flips.

Our single-step mesh parameterization procedure (generating $\mathcal{U}^1$) is only slightly slower than the fast Floater and Eck et al. parameterization methods and faster than the intrinsic parameterization of Desbrun et al. [DMA02]. Besides $\mathcal{U}^1$ demonstrates competitive results in minimizing the stretch, edge, area, and angle distortions.

Our optimal mesh parameterization procedure is also fast enough and sometimes achieves better results in stretch minimizing than the probabilistic minimization of Sander et al. [SSGH01] which is very slow. Moreover, by contrast with [SSGH01], $\mathcal{U}^{\text{opt}}$ does not generate parameter cracks (see Figure 5.4) because (5.3) acts like a diffusion process. Besides, if a very low stretch parameterization is needed, $\mathcal{U}^{\text{opt}}$ can be used as an initial parameterization for [SSGH01].

Figure 5.5 shows $\mathcal{U}^{\text{opt}}$ parameterization of the Mannequin Head model when the parameter domain has boundaries of various shapes. The left images show the parameterization and corresponding texture mapping results when the boundary is the unit circle. The right images demonstrate similar results when the boundary of the parameter domain was obtained as the so-called natural boundary for the conformal parameterization of [DMA02] (similar free boundary schemes such as [KGG05, Wan05] also can be used). Notice that the stretch distortions near the boundary are substantially reduced in the latter case. Although mesh parameterization with free boundary conditions can be achieved by angle based flattening [SU01] and its hierarchical extension [SLMB05], they do not produce low-stretch parameterizations as our method.

In Figure 5.17 mesh parameterizations $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$ are evaluated and compared using the checkerboard texture. Sometimes $\mathcal{U}^{\text{opt}}$ does not produce the best visual result because of high anisotropy and $\mathcal{U}^1$ is preferable. Finally, in Figure 5.18 we analyze how the stretch distribution over a complex geometry model is changing during the optimization process $\mathcal{U}^0 \rightarrow \mathcal{U}^1 \rightarrow \mathcal{U}^{\text{opt}}$. The top row of images presents the model (a decimated Max-Planck bust model) and results of checkerboard texture mapping with $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$. The four remaining images of the model show the stretch distribution over the model for $\mathcal{U}^0$, $\mathcal{U}^1$, and

$\mathcal{U}^{\text{opt}}$ parameterizations. The images demonstrate how well our stretch minimization procedures minimize and equalize the stretch. It is interesting to notice that near the mesh boundary the optimized meshes have large area and angle distortions (the same effect is observed in all the other tested models) but relatively low stretch distortions. One can hope that an appropriate relaxation of boundary conditions will reduce those area and angle distortions while maintaining low stretch.

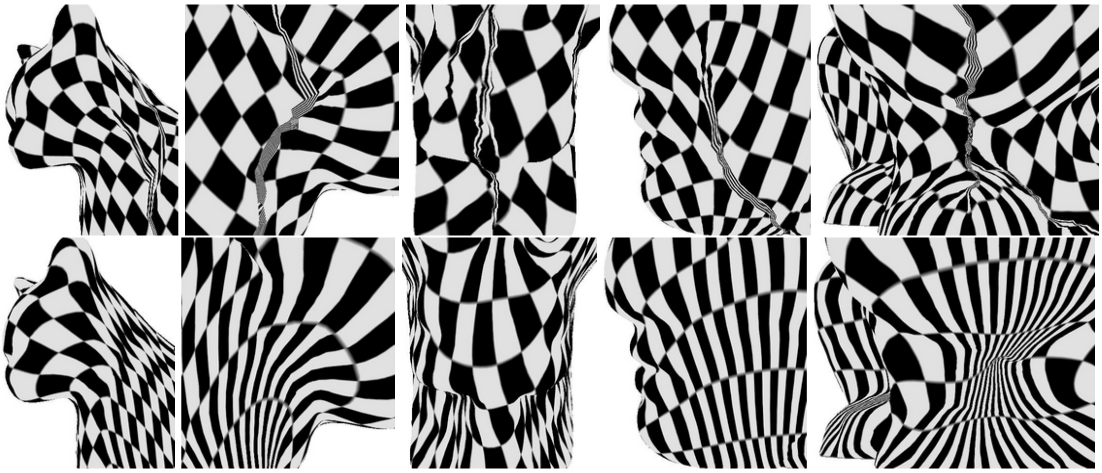
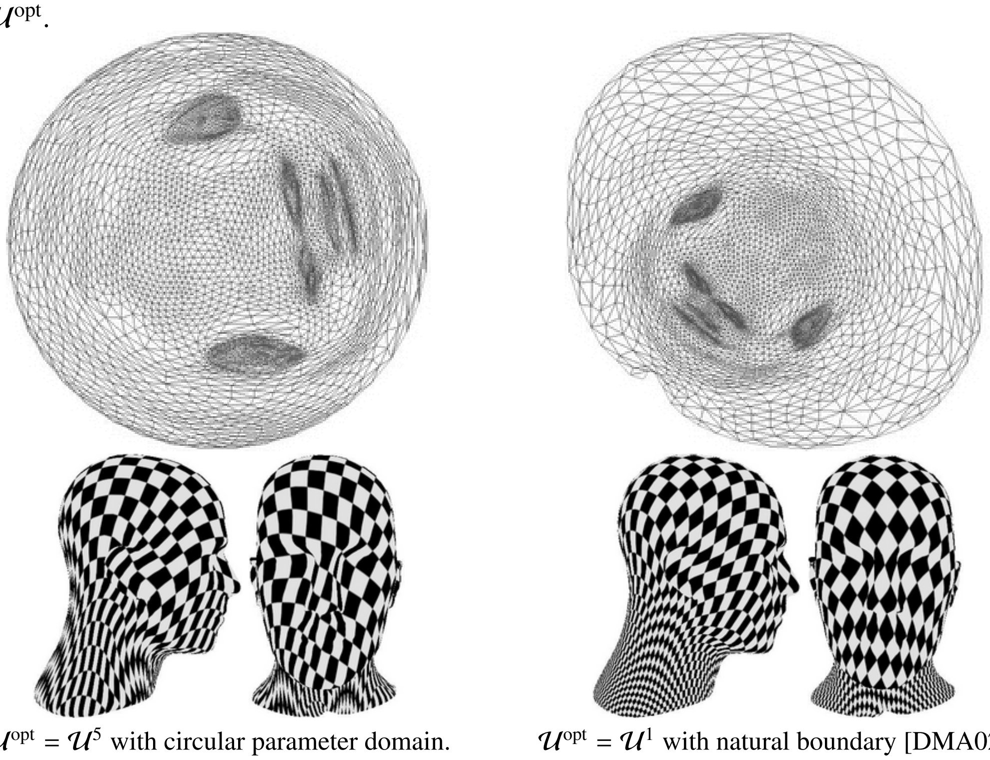


Figure 5.4: Parameter cracks on various models textured with checkerboard texture. The images of the upper row demonstrate parameter cracks generated by the stretch-minimization method of Sander et al. The images of the bottom row show the same parts of the models parameterized by our $\mathcal{U}^{\text{opt}}$.



$\mathcal{U}^{\text{opt}} = \mathcal{U}^5$ with circular parameter domain.
Time: 1.51 s, Stretch: 1.34,
Edge: 0.43, Angle: 0.47, Area: 0.4.

$\mathcal{U}^{\text{opt}} = \mathcal{U}^1$ with natural boundary [DMA02].
Time:1.67 s, Stretch: 1.68,
Edge: 0.5, Angle: 0.37, Area: 0.9.

Figure 5.5: Using various parameter domains for $\mathcal{U}^{\text{opt}}$.

5.4 Application to Remeshing

In the right columns of Figures 5.14-5.16 and in Figure 5.13 we demonstrate how our mesh parameterization technique can be used for fast and high quality remeshing of complex surfaces. We have chosen the interactive geometry remeshing scheme of Alliez et al. [AMD02] and implemented its main steps, see Figure 5.6:

1. Create a mesh parameterization.
2. Compute area, curvature, and control maps using hardware accelerated OpenGL commands.
3. Sample points by applying an error diffusion to the control map.
4. Connect the points using the Delaunay triangulation.
5. Use the parameterization to map the points into 3D.

A conformal mesh parameterization is the best choice for the described remeshing scheme.

It is clear that the remeshing quality depends on the size of an image used for the hardware assisted acceleration: the bigger size, the better result as demonstrated in Figure 5.6. On the other side, the image size is restricted by the graphics card memory. It turns out that a high quality remeshing can be obtained even for a relatively small image size. Let us assume that we have two parameterizations of a 3D mesh: a conformal parameterization and an area-preserving one. Then let us use the area-preserving parameterization for computing the control map and resampling the points via an error diffusion process. Finally, the points are mapped from the area-preserving parameterization to the conformal one and are connected using the Delaunay triangulation.

The above remeshing modification has one drawback: it requires two parameterizations, conformal and area-preserving. However since our low-stretch parameterization $\mathcal{U}^{\mathrm{opt}}$ has nice area-preserving properties and the initial Floater's parameterization $\mathcal{U}^0$ is close to a conformal one, we use $\mathcal{U}^0$ and $\mathcal{U}^{\mathrm{opt}}$ instead of the conformal and area-preserving parameterizations in the above modification of the interactive geometry remeshing scheme of Alliez et al. So we use $\mathcal{U}^{\mathrm{opt}}$ for resampling and then map the sampled points to $\mathcal{U}^0$, and apply the Delaunay triangulation on $\mathcal{U}^0$. Figure 5.7 describes our double-parameterization remeshing framework. Figures 5.8 and 5.9 demonstrate sampling efficiency and triangulation quality, respectively.

The right images of rows (a)-(c) of Figures 5.14-5.16 demonstrate results of the single-parameterization remeshing scheme if the discrete harmonic map parameterization [EDD$^+$95], Floater's shape preserving parameterization [Flo97], and intrinsic discrete conformal parameterization are used, respectively. The right images of rows (d)-(f) of Figures 5.14-5.16 present our experiments with the double-parameterization remeshing scheme. We set Floater's parameterization $\mathcal{U}^0$ as a substitute of a conformal parameterization and used $\mathcal{U}^0$ as an initial parameterization to generate the stretch-minimizing parameterization of Sander et al. [SSGH01] and $\mathcal{U}^1$ and $\mathcal{U}^{\mathrm{opt}}$. These low-stretch parameterizations were used as substitutes of an area-preserving parameterization.

Figure 5.13 presents remeshed Max-Planck bust and Stanford bunny models obtained by the remeshing schemes based on (from left to right) $\left\{\mathcal{U}^0\right\}$, $\left\{\mathcal{U}^0,\mathcal{U}^1\right\}$, and $\left\{\mathcal{U}^0,\mathcal{U}^{\mathrm{opt}}\right\}$ parameterizations. Here using $\{\mathcal{U}',\mathcal{U}''\}$ parameterizations means that we use $\mathcal{U}'$ as a substitute of a conformal parameterization and $\mathcal{U}''$ as a substitute of an area preserving one. Notice that the double-parameterization remeshing scheme with $\left\{\mathcal{U}^0,\mathcal{U}^{\mathrm{opt}}\right\}$ yields the best results.

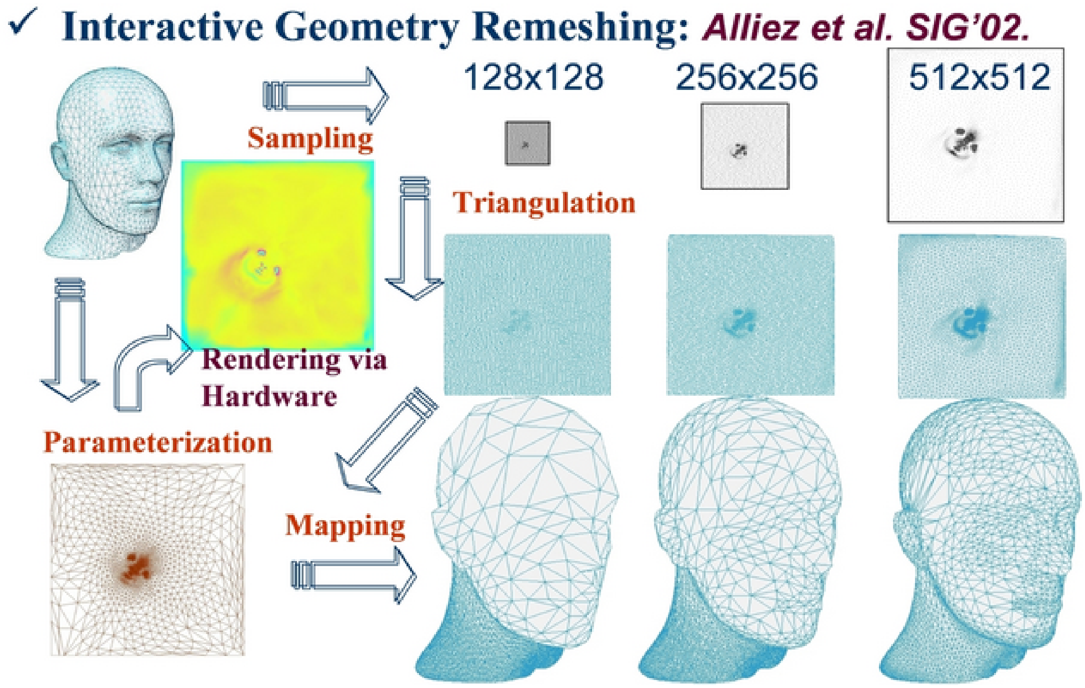


Figure 5.6: Remeshing framework of Alliez et al. [AMD02]. Resamplings on the conformal parameterization are demonstrated for 128x128, 256x256, and 512x512 image sizes.

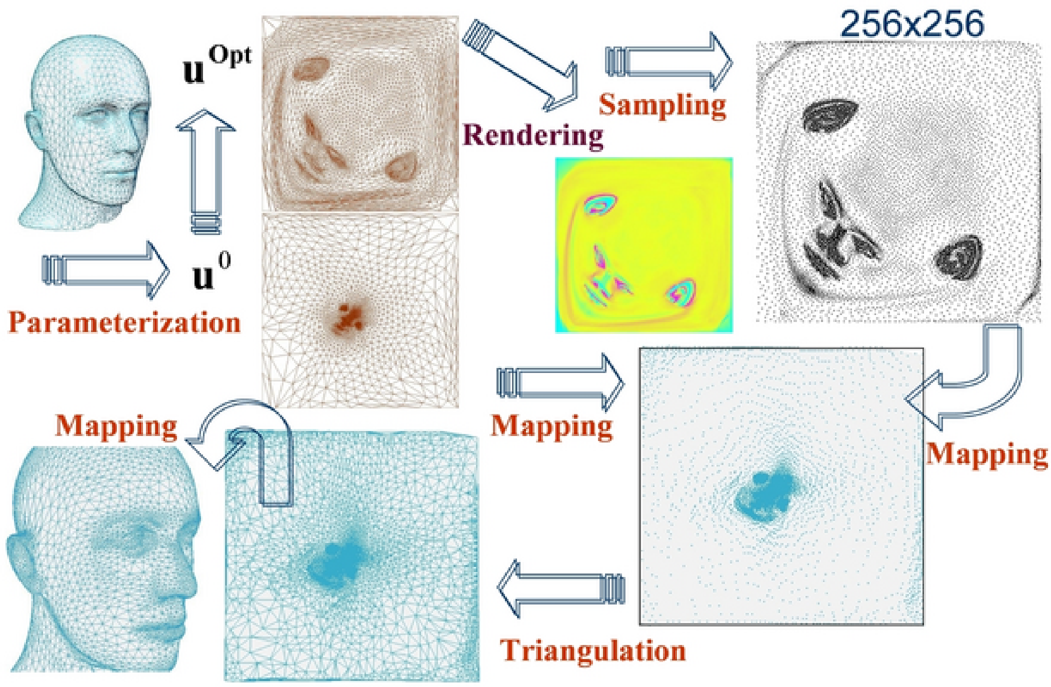


Figure 5.7: Double-parameterization remeshing framework: $\mathcal{U}^{\text{opt}}$ and $\mathcal{U}^0$ are employed for resampling and triangulation, respectively. Resampling on $\mathcal{U}^{\text{opt}}$ and triangulation on $\mathcal{U}^0$ are demonstrated for the 256x256 image size: the resulting 3D mesh is much better than even 512x512 image case of the conformal parameterization, see Figure 5.6. See also Figures 5.8 and 5.9 for advantages of our scheme.

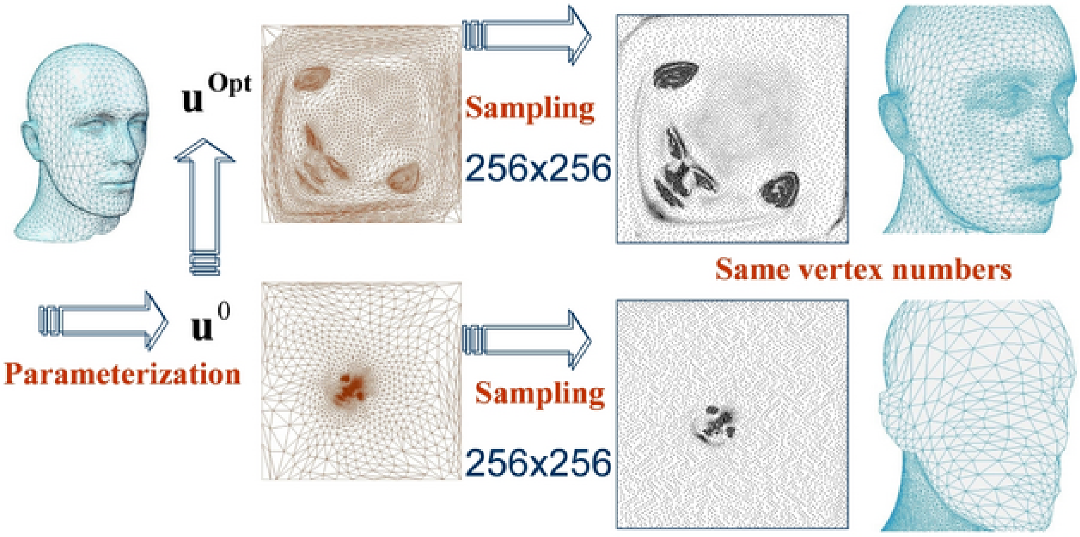


Figure 5.8: Sampling advantage of our double-parameterization remeshing. It is obvious that our low-stretch parameterization gives us efficient sampling rate for a fixed image size (256x256 is used in this Figure).

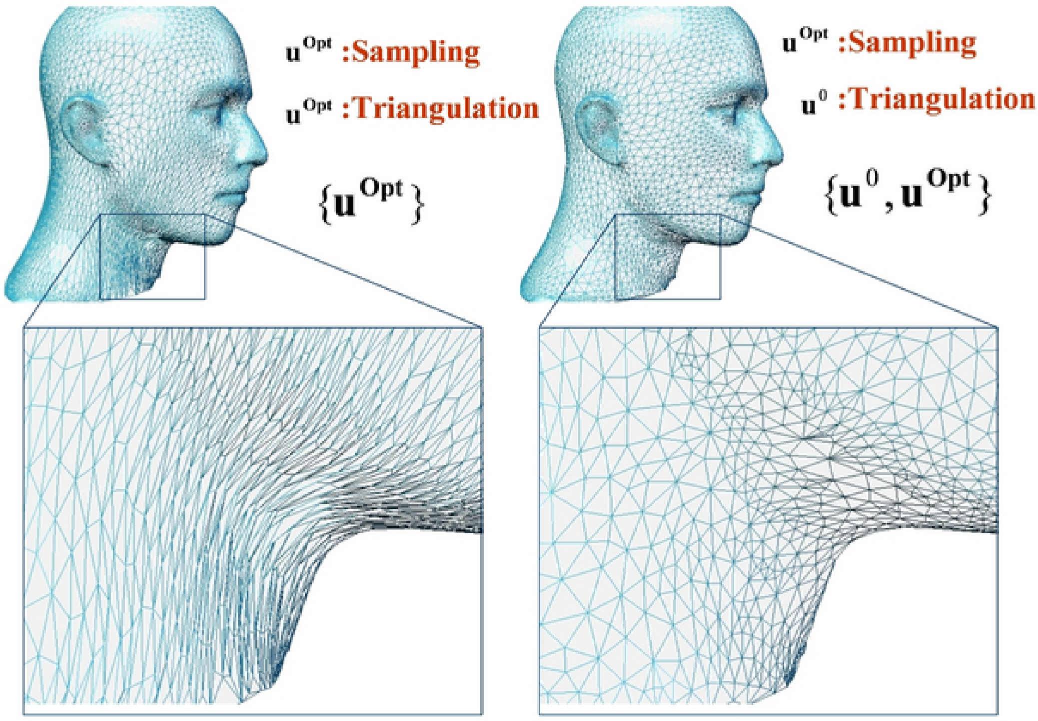


Figure 5.9: Remeshing quality advantage of our double-parameterization remeshing. Left: only $\mathcal{U}^{\text{opt}}$ is used. Right: both $\left\{\mathcal{U}^0, \mathcal{U}^{\text{opt}}\right\}$ are employed. Our double-parameterization remeshing avoids angle distortion cased by $\mathcal{U}^0 \rightarrow \mathcal{U}^{\text{opt}}$ because the Delaunay triangulation used to triangulate the sampled points maximizes minimum angle of the triangles. Since $\mathcal{U}^0$ is close to conformal, quality of 2D triangulation is preserved in 3D.

5.5 Discussion of Low-Stretch Mesh Parameterization

The final result of our mesh optimization method depends on the choice of initial weights $\left\{\mathbf{u}_i^0\right\}$. In particular we found out that selecting Floater's shape preserving weights [Flo97] leads to a very effective stretch minimization procedure. Even better results are often obtained if the so-called cotangent weights [PP93, DMA02] are used for generating the initial parameterization $\mathcal{U}^0$. However since cotangent weights are not necessary positive, using them may generate triangle flips.

One interesting situation when the choice of shape preserving weights is not very appropriate consists of parameterizing meshes with multiple boundaries, see the left image of Figure 5.10 for such a mesh topologically equivalent to a sphere with holes. One solution to create a good initial parameterization of such a mesh consists of the following. Let us choose one hole (the biggest one) as the outer hole and the remaining holes as inner holes. Let us triangulate the inner holes and then use the shape preserving weights. Alternatively, for each edge $[\mathbf{x}_i, \mathbf{x}_j]$ of an inner hole, according to the right image of Figure 5.10, we can compute angles needed to generate either the mean value weights [Flo03]

$$\frac{\tan(\theta_{ij}/2) + \tan(\phi_{ij}/2)}{|\mathbf{x}_i - \mathbf{x}_j|}$$

or cotangent weights

$$\cot(\alpha_{ij}) + \cot(\beta_{ij})$$

and use either of these sets of weights for generating the initial parameterization $\mathcal{U}^0$.

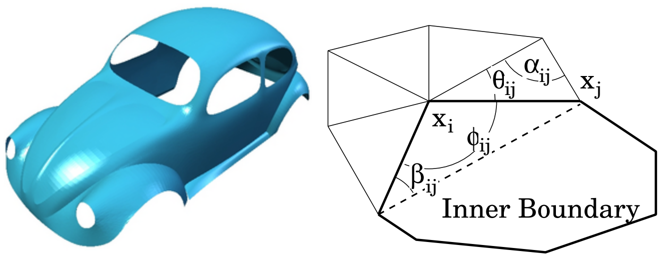


Figure 5.10: Left: a mesh with multiple boundaries. Right: the angles needed to define the cotangent and mean value weights for boundary vertices.

This technique as well as the virtual boundary method of Lee et al [LKL02] is developed for dealing with mesh parameterizations defined over non-convex parameter domains. In contrast to [LKL02] our approach is especially designed for processing meshes with holes. The use of the virtual boundary method [LKL02] for meshes with holes would require a nontrivial hole filling procedure (see, for example, [Lie03]) as a preprocessing step.

Figures 5.11 and 5.12 demonstrate the power of this our technique and show parameterizations $\mathcal{U}^0$ and $\mathcal{U}^{\text{opt}}$ obtained for the Car model.

5.6 Summary of Low-Stretch Mesh Parameterization

We have presented a fast and powerful method for generating low-stretch mesh parameterizations and demonstrate its applicability to high quality texture mapping and remeshing. Our method is much faster than the stochastic stretch minimization procedure of Sander et al. [SSGH01] (note that their more recent coarse-to-fine stretch optimization procedure [SGSH02] is significantly faster than that of [SSGH01] but still slower than ours) and often produces better quality results. In particular, it does not generate parameter cracks. Our approach has been already employed and extended for not only the mesh parameterizations [ZRS05a, ZRS05b, YYSZ06] but also texture mapping (geometry image) [WGMY05, CHCH06], morphing [SK04], meshing point clouds [ZG04b], cloth simulation [WTY05], and feature extraction [NNS06].

Our approach is heuristic. Although it has much in common with mesh moving techniques widely used in computational mechanics and often justified mathematically, at present we are not able to support our approach by rigorous mathematical results. In future we would be glad to justify the effectiveness of our approach rigorously. Also future research includes extending our method to spherical mesh parameterizations [GY02, GGS03] and global mesh parameterizations [GY03, GGT06, TACSD06, RLL+06]. The source code of our method is available on the Web for evaluation [YBS04].

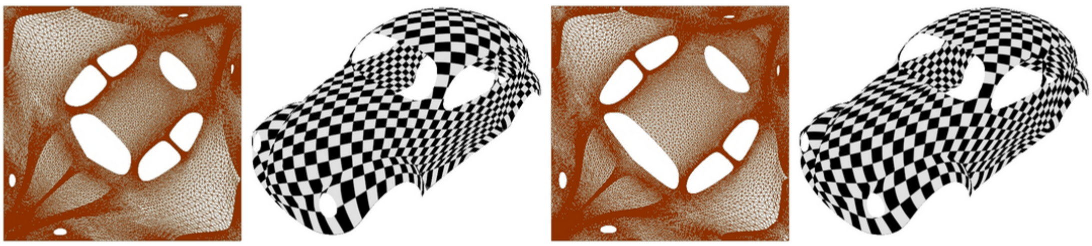


Figure 5.11: Left: the cotangent (harmonic) weights are used to generate $\mathcal{U}^0$; stretch L^2 error = 1.495, stretch L^∞ error = 360.4. Right: $\mathcal{U}^{\text{opt}} = \mathcal{U}^1$; stretch L^2 error = 1.178, stretch L^∞ error = 20.13.

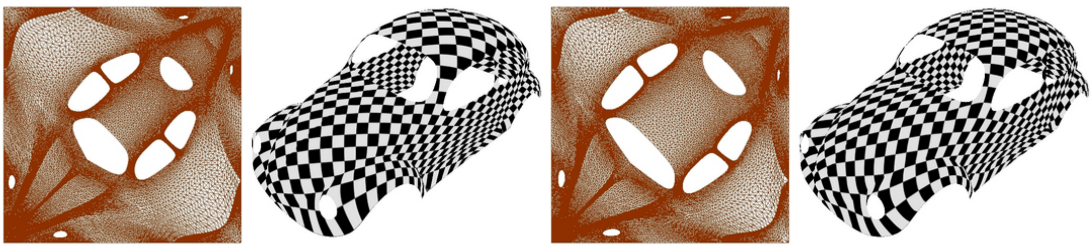


Figure 5.12: Left: the mean value weights are used to generate $\mathcal{U}^0$; stretch L^2 error = 1.395, stretch L^∞ error = 172.7. Right: $\mathcal{U}^{\text{opt}} = \mathcal{U}^1$; stretch L^2 error = 1.181, stretch L^∞ error = 21.37.

	time	Stretch	Edge	Angle	Area
(a)	0.06 s	6.6507	0.9918	0.125	1.4032
(b)	0.06 s	5.9171	0.9635	0.1995	1.3801
(c)	0.12 s	6.2751	0.9778	0.1619	1.3931
(d)	80.91 s	1.375	0.5162	0.2952	0.5232
(e)	0.08 s	1.6691	0.5084	0.3717	0.8836
(f_3)	0.16 s	1.4084	0.4814	0.4479	0.4165

Table 5.1: Mannequin Head model: V = 689, F = 1355

	time	Stretch	Edge	Angle	Area
(a)	0.21 s	1.9708	0.4935	0.0969	0.8455
(b)	0.17 s	1.8084	0.4648	0.1568	0.8409
(c)	0.33 s	1.8511	0.4753	0.1189	0.84
(d)	213 s	1.172	0.2996	0.2239	0.3043
(e=f_1)	0.3 s	1.2057	0.2862	0.2881	0.3179

Table 5.2: Cat Head model: V = 1856, F = 3660

	time	Stretch	Edge	Angle	Area
(a)	0.37 s	6.6617	0.9971	0.0685	1.4036
(b)	0.32 s	5.7921	0.9599	0.1807	1.3733
(c)	0.76 s	6.1295	0.9784	0.1209	1.3886
(d)	23 **m**	1.3279	0.5393	0.2744	0.4956
(e)	0.5 s	1.6425	0.5073	0.3838	0.8717
(f_3)	1.09 s	1.382	0.4748	0.4132	0.3832

Table 5.3: Refined Mannequin Head model: V = 2732, F = 5420

	time	Stretch	Edge	Angle	Area
(a)	1.23 s	13.306	0.7563	0.1041	1.0207
(b)	0.87 s	11.729	0.6976	0.2545	0.9526
(c)	1.81 s	12.266	0.7232	0.176	0.9795
(d)	1 **h**	1.3408	0.4955	0.3477	0.4227
(e)	1.5 s	1.7643	0.4551	0.3735	0.4676
(f_3)	3.44 s	1.4791	0.4661	0.5226	0.3613

Table 5.4: Cat model: V = 5649, F = 11168

	time	Stretch	Edge	Angle	Area
(a)	3.16 s	18.027	1.2288	0.0361	1.692
(b)	2.29 s	15.941	1.2074	0.1441	1.6373
(c)	17.4 s	16.933	1.2157	0.0857	1.6618
(d)	57.5**h**	1.3257	0.7021	0.2501	0.5436
(e)	4.18 s	2.2037	0.6249	0.372	1.1899
(f_3)	9.22 s	1.5392	0.5623	0.4905	0.6217

Table 5.5: Decimated Max-Planck bust model: V = 9462, F = 18866

	time	Stretch	Edge	Angle	Area
(a)	12.9 s	1.5348	0.3025	0.1313	0.5063
(b)	6.21 s	1.485	0.3412	0.1748	0.5651
(c)	25.8 s	43.947	0.7602	0.3622	1.0085
(d)	4.5 **h**	1.2226	0.2833	0.1934	0.4338
(e)	17.9 s	1.2105	0.2477	0.2112	0.3876
(f_3)	42.6 s	1.1718	0.24	0.2636	0.2375

Table 5.6: Fandisk model: V = 9919, F = 19617

	time	Stretch	Edge	Angle	Area
(a)	5.55 s	9179549	1.6037	0.0915	1.7599
(b)	4.24 s	1120318	1.5049	0.3491	1.7175
(c)	21.1 s	231989	1.5494	0.2707	1.7387
(d)	39.7 **h**	7635.3	1.1442	0.3544	0.8435
(e)	6.99 s	313.64	0.9883	0.6341	1.4739
(f_8)	33.2 s	3.5688	0.8522	0.8253	0.7897

Table 5.7: Half-of-Dragon model: V = 13927, F = 27782

	time	Stretch	Edge	Angle	Area
(a)	12.4 s	9462.1	0.9729	0.0704	1.5132
(b)	8.95 s	181.05	0.9983	0.3852	1.5725
(c)	90.7 s	320.53	0.9845	0.2281	1.5425
(d)	43.4 **h**	1.6816	0.7193	0.2917	0.6665
(e)	14.7 s	3.3929	0.5041	0.6184	0.8078
(f_3)	32.3 s	2.884	0.6399	0.7747	0.5344

Table 5.8: Dragon Head model: V = 23929, F = 47783

	time	Stretch	Edge	Angle	Area
(a)	11.2 s	3.4799	0.7924	0.0542	1.3399
(b)	8.46 s	4.676	0.8678	0.1627	1.3664
(c)	93.8 s	34.621	0.8104	0.1831	1.3525
(d)	18.6 **h**	1.3092	0.4603	0.2265	0.5492
(e)	15.2 s	1.4373	0.4166	0.3446	0.6868
(f_2)	27.2 s	1.304	0.385	0.3923	0.4123

Table 5.9: Igea model: $V = 24720$, $F = 49301$

	time	Stretch	Edge	Angle	Area
(a)	17.9 s	712.33	0.7097	0.0797	1.098
(b)	13.2 s	85.181	0.7241	0.1522	1.0861
(c)	231 s	672.45	0.7062	0.2866	1.0957
(d)	55.6 **h**	1.5159	0.4982	0.3109	0.4868
(e)	22.5 s	4.7926	0.4582	0.387	0.5632
(f_6)	79.8 s	1.8755	0.6143	0.6065	0.5241

Table 5.10: Stanford Bunny model: $V = 31272$, $F = 62247$

	time	Stretch	Edge	Angle	Area
(a)	92.4 s	6.3061	0.8241	0.0445	1.3021
(b)	66.3 s	6.092	0.7752	0.1782	1.2613
(c′)	486 s	6.306	0.8241	0.0445	1.3021
(d)	120 **h**	2.5689	0.6481	0.2444	0.926
(e)	125 s	1.5683	0.4252	0.3476	0.6387
(f_2)	206 s	1.5041	0.4414	0.4678	0.3946

Table 5.11: Fish model: $V = 64982$, $F = 129664$

	time	Stretch	Edge	Angle	Area
(a)	250 s	18.207	1.2578	0.03	1.6936
(b)	204 s	18.1025	1.25	0.0512	1.6912
(c)	52.1 **m**	2.8434	1.2341	0.3068	1.6924
(e)	384 s	2.2094	0.6598	0.3698	1.2017
(f_3)	848 s	1.4926	0.5939	0.4865	0.4812

Table 5.12: Max-Planck bust model: $V = 199169$, $F = 398043$

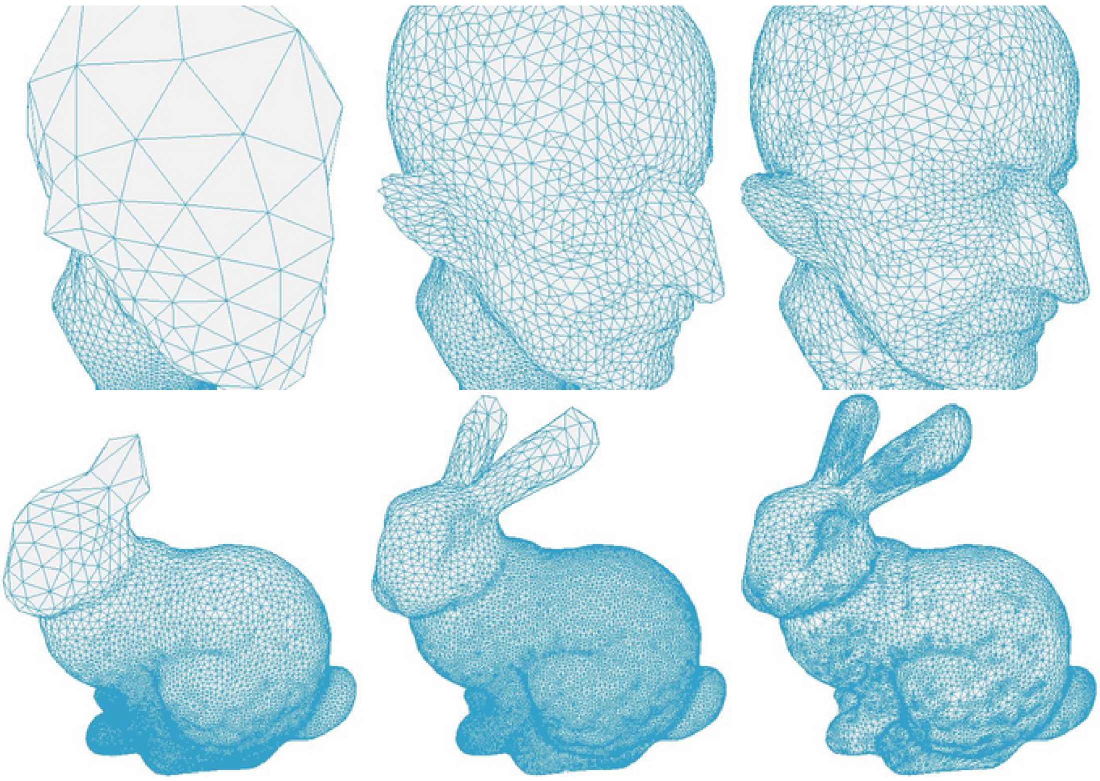
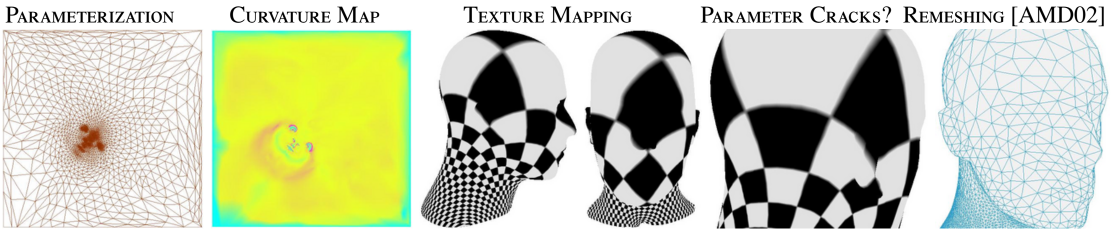
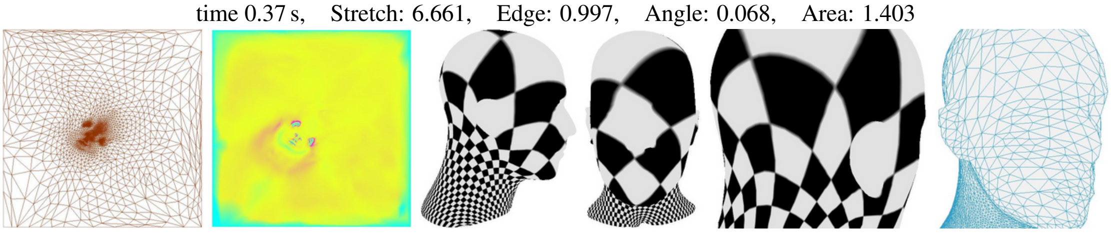


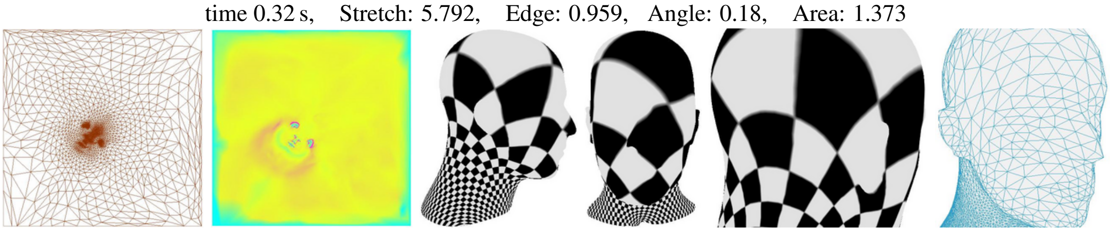
Figure 5.13: Remeshing of Max-Planck bust model (three left images) and Stanford bunny (three right images) models. For each model remeshings according to $\mathcal{U}^0$, $\{\mathcal{U}^0, \mathcal{U}^1\}$, and $\{\mathcal{U}^0, \mathcal{U}^{\text{opt}}\}$ are shown. See the text for details.



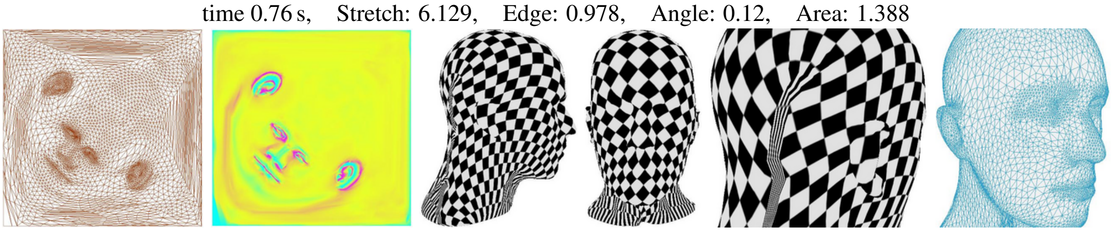
(a) Harmonic map of Eck et al. [EDD+95]:
time 0.37 s, Stretch: 6.661, Edge: 0.997, Angle: 0.068, Area: 1.403



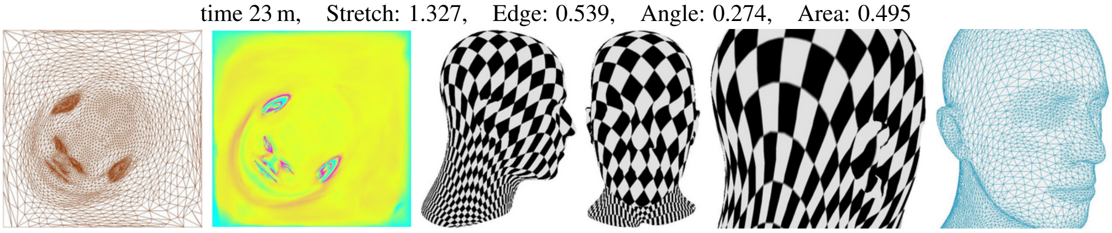
(b) Floater shape preserving weights [Flo97]:
time 0.32 s, Stretch: 5.792, Edge: 0.959, Angle: 0.18, Area: 1.373



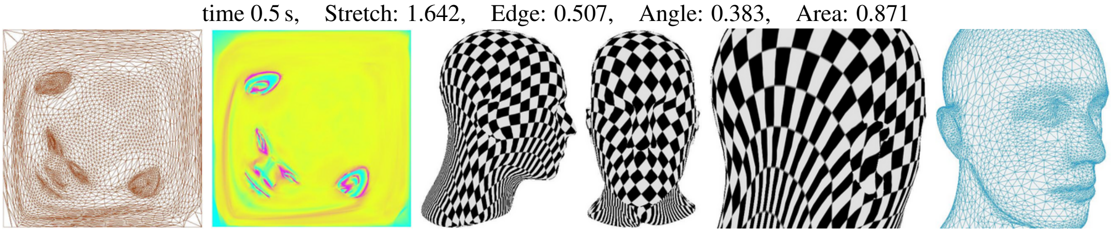
(c) Intrinsic parameterization of Desbrun et al. [DMA02]:
time 0.76 s, Stretch: 6.129, Edge: 0.978, Angle: 0.12, Area: 1.388



(d) Stretch minimization of Sander et al. [SSGH01]:
time 23 m, Stretch: 1.327, Edge: 0.539, Angle: 0.274, Area: 0.495

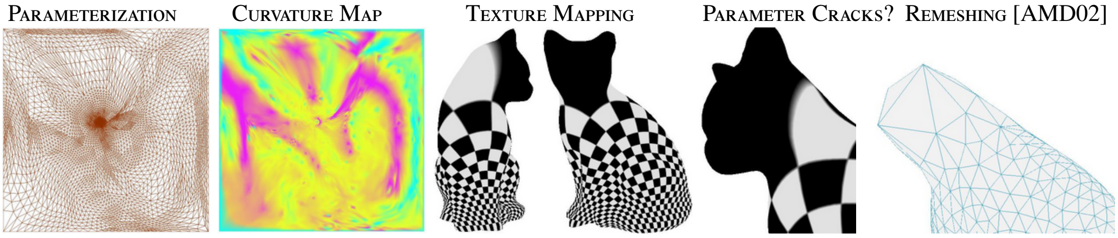


(e) Our $\mathcal{U}^1$ parameterization:
time 0.5 s, Stretch: 1.642, Edge: 0.507, Angle: 0.383, Area: 0.871

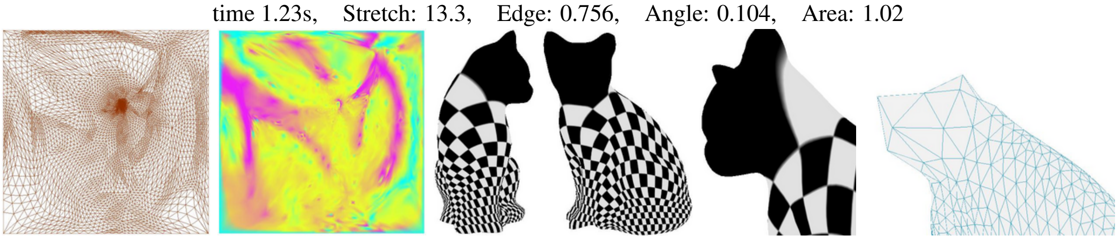


(f) Our $\mathcal{U}^{\text{opt}} = \mathcal{U}^3$ parameterization:
time 1.09 s, Stretch: 1.382, Edge: 0.4748, Angle: 0.4132, Area: 0.3832

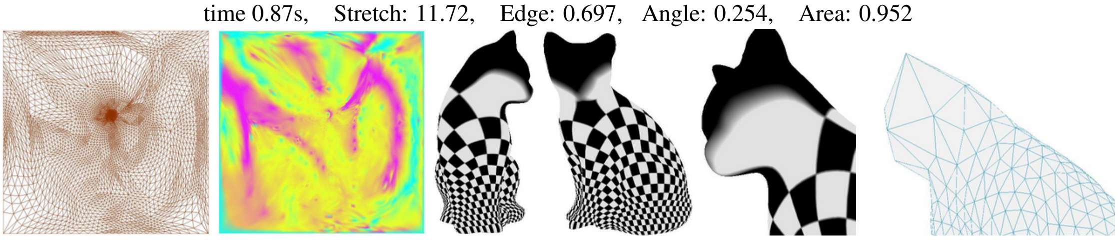
Figure 5.14: Comparison of various mesh parameterization schemes on the Mannequin Head model ($V = 2732$, $F = 5420$).



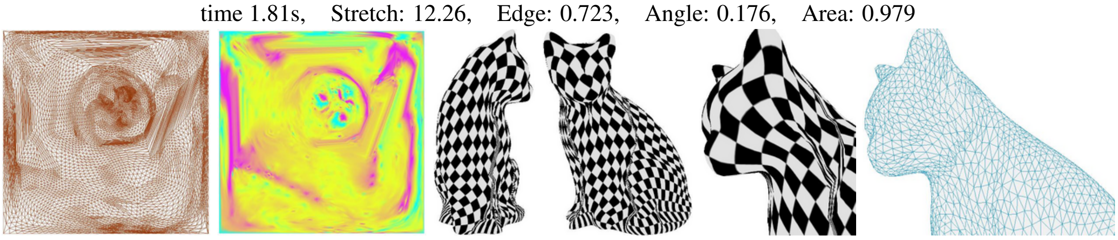
(a) Harmonic map of Eck et al. [EDD+95]:
time 1.23s, Stretch: 13.3, Edge: 0.756, Angle: 0.104, Area: 1.02



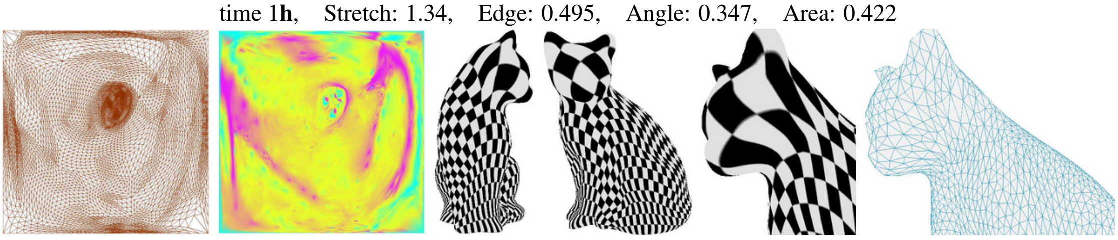
(b) Floater shape preserving weights [Flo97]:
time 0.87s, Stretch: 11.72, Edge: 0.697, Angle: 0.254, Area: 0.952



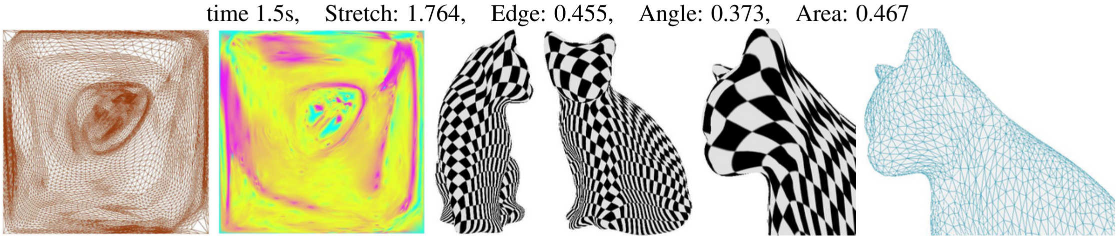
(c) Intrinsic parameterization of Desbrun et al. [DMA02]:
time 1.81s, Stretch: 12.26, Edge: 0.723, Angle: 0.176, Area: 0.979



(d) Stretch minimization of Sander et al. [SSGH01]:
time 1**h**, Stretch: 1.34, Edge: 0.495, Angle: 0.347, Area: 0.422

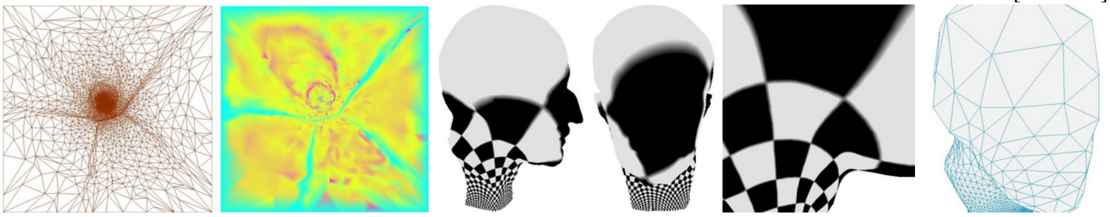


(e) Our $\mathcal{U}^1$ parameterization:
time 1.5s, Stretch: 1.764, Edge: 0.455, Angle: 0.373, Area: 0.467

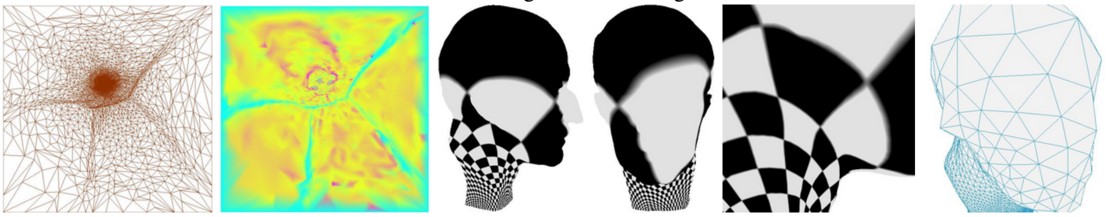


(f) Our $\mathcal{U}^{\text{opt}} = \mathcal{U}^3$ parameterization:
time 3.44s, Stretch: 1.479, Edge: 0.466, Angle: 0.522, Area: 0.361

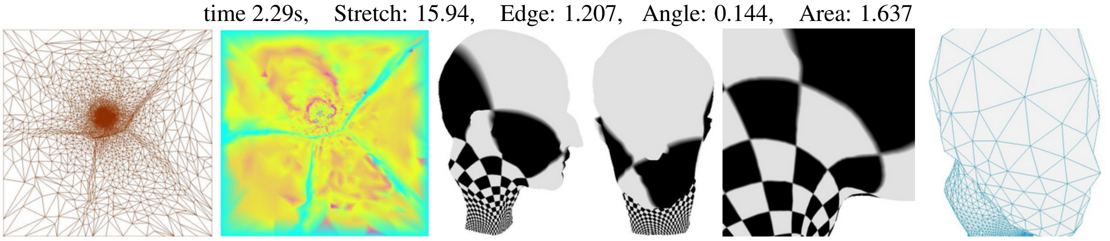
Figure 5.15: Comparison of various mesh parameterization schemes on the Cat model ($V = 5649$, $F = 11168$).



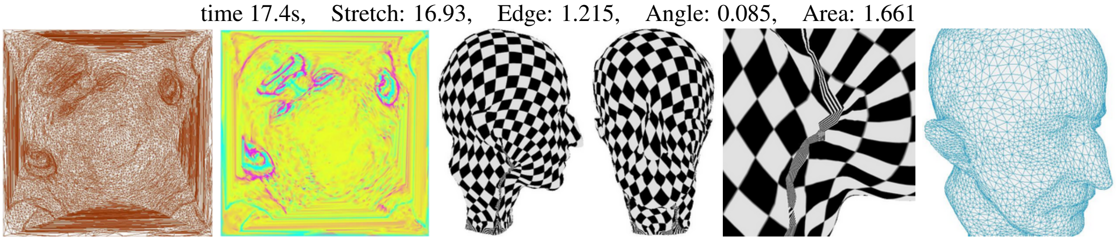
(a) Harmonic map of Eck et al. [EDD+95]:
time 3.16s, Stretch: 18.02, Edge: 1.228, Angle: 0.036, Area: 1.692



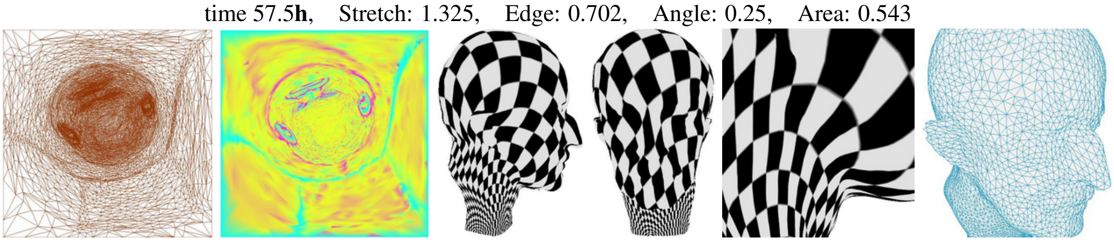
(b) Floater shape preserving weights [Flo97]:
time 2.29s, Stretch: 15.94, Edge: 1.207, Angle: 0.144, Area: 1.637



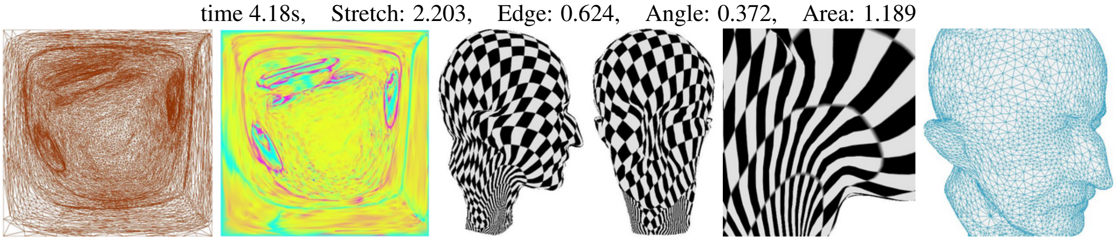
(c) Intrinsic parameterization of Desbrun et al. [DMA02]:
time 17.4s, Stretch: 16.93, Edge: 1.215, Angle: 0.085, Area: 1.661



(d) Stretch minimization of Sander et al. [SSGH01]:
time 57.5**h**, Stretch: 1.325, Edge: 0.702, Angle: 0.25, Area: 0.543



(e) Our $\mathcal{U}^1$ parameterization:
time 4.18s, Stretch: 2.203, Edge: 0.624, Angle: 0.372, Area: 1.189



(f) Our $\mathcal{U}^{\text{opt}} = \mathcal{U}^3$ parameterization:
time 9.22s, Stretch: 1.539, Edge: 0.562, Angle: 0.49, Area: 0.621

Figure 5.16: Comparison of various mesh parameterization schemes on the decimated Max-Planck bust model ($V = 9462$, $F = 18866$).

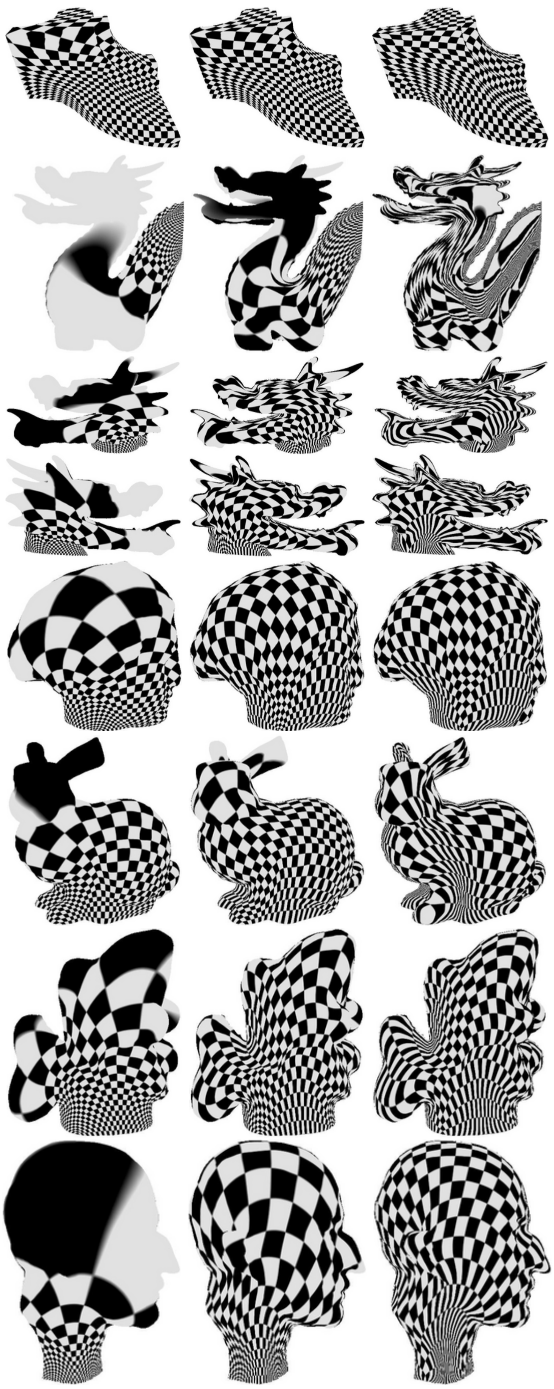


Figure 5.17: Checkerboard texture mapping with U^0 (left), U^1 (middle), and U^{opt} (right).

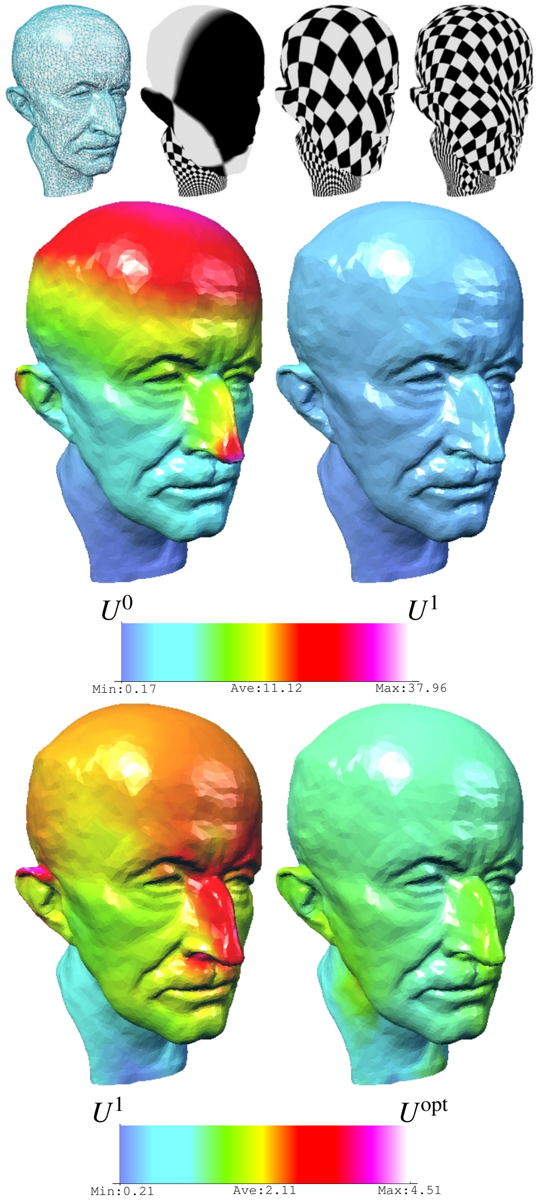


Figure 5.18: Top row: a decimated Max-Planck bust model and results of checkerboard texture mapping with $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$ parameterizations. The four remaining images of the model show the distribution of the vertex stretches over the model for $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$. Firstly coloring by stretch $\sigma \in [0.17, 37.96]$ is used to compare $\mathcal{U}^0$ and $\mathcal{U}^1$. Then the same coloring scheme on the stretch interval $[0.21, 4.51]$ is employed to compare the stretch distributions for $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$. Here the bounds of the interval are equal to the maximal and minimal stretch values.

6

Free-Form Skeleton-driven Mesh Deformations

Generating natural-looking deformations of complex shapes has multiple applications in CAGD, computer animation, and geometric modeling. Since the pioneering works [Bar84, SP86], developing fast, efficient, and intuitive methods for local and global free-form shape deformations is a subject of intensive research, see the recent works [BPGK06, vFTS06, HSL$^+$06] and references therein.

Pierre Bézier (1910 - 1999) introduced the idea of deforming shapes embedded in tensor product splines through a mapping between different configurations of spline control points [Ram94]. This idea, called space deformations, was inherited to the famous FFD (Free-Form Deformation) technique [SP86] which deforms shapes embedded in regular lattices equipped with Bézier splines. Further developments of the FFD include extensions for non-regular lattices [Coq90], direct manipulations [HHK92], arbitrary lattices [MJ96], mean value coordinates [JSW05], and many others. Besides global deformations [Bar84] as bending, twining, and tapering, space deformations can be employed without lattices. For example, displacement vectors with smooth influence functions [BR94, RNJ00, YBS02, BK03a] produce local bumping and sculpting. In [AWC04] authors proposed to decompose a local sculpting deformation into a sequence of space deformations in order to avoid self-intersections. The technique of [AWC04] is extend for constant volume deformations in [ACWK04] by using a set of swirling deformations. Many space deformation schemes are equipped with intermediate control interfaces such as lattices (FFDs), stick-figure skeletons [MTLT88, LCJ94], curves [SF98], triangle meshes [SK00b, KO03], and vector field [vFTS06] in order to control deformations intuitively and to avoid specifying many local deformations to obtain a large-scale deformation. Most of these deformations are formulated in terms of weighted blending of local coordinate frames attached to the intermediate control interfaces. Parameters of these weighted blending are usually required tedious manual adjustments to obtain natural-looking (intuitive) deformations as mentioned in [LCF00].

Recently skeleton-driven global free-form shape deformations drew a considerable attention [LCF00, SK00b, CGC$^+$02]. The skeleton-driven deformations are well-suited for large-scale shape deformations and, therefore, can be used in numerous applications in computer game and digital movie industries. Bloomenthal [BL99, Blo02] proposed to use the medial axis as skeletal control interface in order to obtain natural-looking deformations by preserving original shape thickness.

In this Chapter, we follow [YBS03, YBS06c, YBS06a] and describe new schemes for free-form skeleton-driven global mesh deformations. The basic (and very simple) idea of our skeleton-based approach to global shape deformations is sketched in Figure 6.1. Notice that usually a local shape deformation corresponds to a skeleton bifurcation (branching) while a global shape deformation corresponds to skeleton bending, as seen in Figure 6.2. First a skeletal mesh, a Voronoi-based approximation of the medial axis, is extracted from a given mesh. Next the skeletal mesh is modified by free-form deformations. Then a desired global shape deformation is obtained by reconstructing the shape corresponding to the deformed skeletal mesh. The use of the medial axis prevents the so-called *collapsing joint defects* [LCF00] which are thickness changing effects where a large bending or twisting deformation is applied via [MTLT88, LCJ94], see Figure 6.3. We develop mesh fairing procedures allowing us to avoid possible global and local self-intersections of the reconstructed mesh. The basic deformation process is described in Section 6.3. Then, the reconstructing and fairing procedures are extended to a variational framework called *discrete differential coordinates* [Sor05] in Section 6.5. Finally, since our shape representation resembles the displaced subdivision surfaces [LMH00], we enrich our mesh deformation approach by a multiscale technique. Figure 6.4 gives some impression on how our approach works.

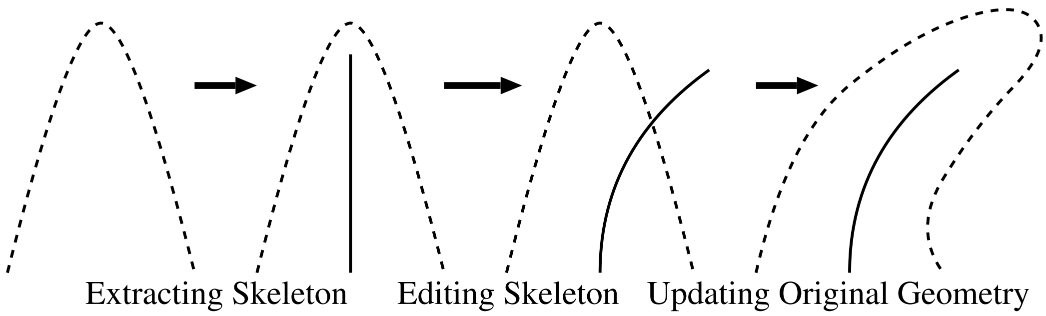


Figure 6.1: Skeleton-based shape deformation.

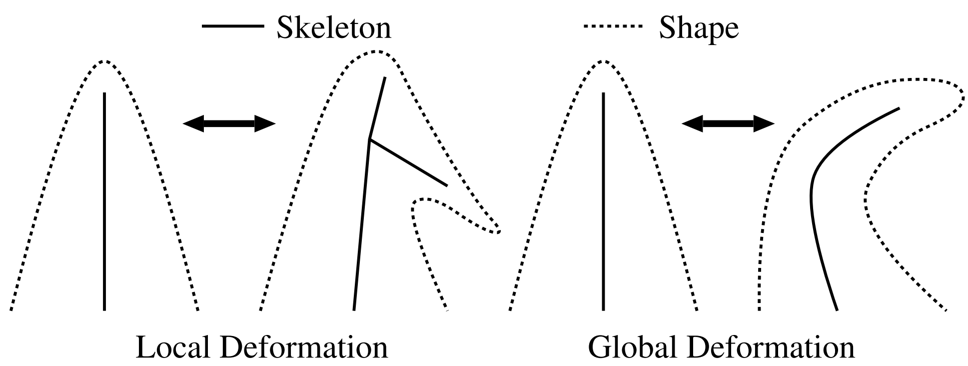


Figure 6.2: Local vs. global shape deformations. Left: local shape deformations usually produce new branches of the skeleton. Right: skeleton bendings correspond to natural-looking global shape deformations.

6.1 Voronoi-based Skeletal Mesh

We already considered the medial axis [Blu67] in Section 4.3. Here we use a discrete approximation of the medial axis, *skeletal mesh*, for the intermediate control interface as [BL99, Blo02]. Mathematically, the medial axis is defined as loci of centers of maximal empty balls for a

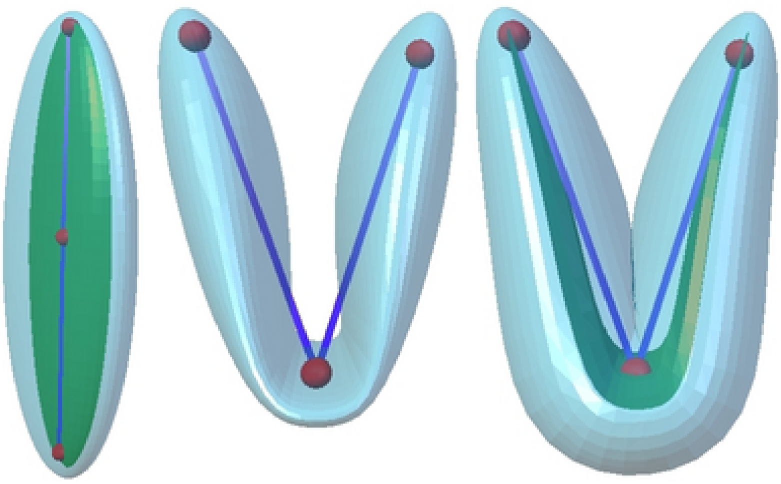
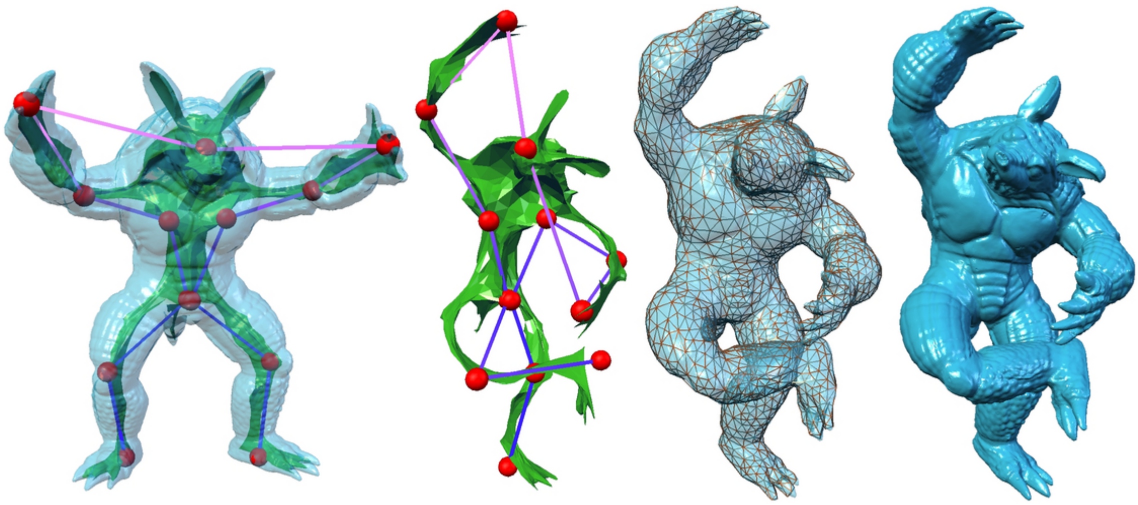


Figure 6.3: Preventing collapsing joint defects. Left: the stick-figure skeleton, the ellipsoid mesh with its skeletal mesh. Deformed meshes via the SSD [MTLT88,LCJ94] (Center) and our method (Right).



(a) (b) (c) (d)

Figure 6.4: A variational skeleton-driven deformation example. (a): Armadillo (332K triangles), its skeletal mesh (5K triangles), and the stick-figure skeleton. (b): A space deformation of the skeletal mesh. (c): A coarse deformed mesh obtained by using discrete differential coordinates from the deformed skeletal mesh. (d): A multiresolutional mesh representation gives us the final dense deformed mesh.

bounded figure $\mathcal{F}$. The maximal empty ball, also called *medial ball* [ACK01b], is completely contained in no other empty ball. The medial axis of a figure is very sensitive to small perturbations of the boundary of the figure: small perturbations of the boundary may result in large changes of the medial axis structure.

Practical extraction of the medial axis of a 3D shape is usually based on 3D Voronoi diagram techniques [TSG+97, ABK98], see also references therein. Figure 6.5 presents a 2D example demonstrating how the medial axis of a figure can be approximated by Voronoi vertices corresponding to points scattered densely over the boundary of the figure. Figure 6.5 demonstrates also how sensitive the medial axis of a figure is with respect to small perturbations of the boundary of the figure.

Recently several improvements over the basic technique developed in [ABK98] were proposed [ACK01a, DZ02, HBK02]. For our needs, we employ the approach developed recently in [HBK02] where it was proposed to approximate the medial axis of a mesh by a skeletal mesh having the same connectivity as the original mesh. The vertices of the skeletal mesh are in one-to-one correspondence with the vertices of the original triangle mesh and the skeletal mesh inherits the connectivity of the original mesh. It allows for editing the skeletal mesh by standard mesh processing tools.

Given a mesh $\mathcal{M}$, our first goal is to extract an approximate skeletal mesh $\mathcal{S}$ such that

$$\mathcal{M} = \mathcal{S} + d\mathbf{N}, \tag{6.1}$$

where $\mathbf{N}$ is the field of unit mesh normals defined at the vertices of $\mathcal{M}$ and d is the set of distances from the vertices of $\mathcal{M}$ to the corresponding vertices of $\mathcal{S}$ along $\mathbf{N}$. The shape representation (6.1) was proposed by [SPW96] for mathematical analysis of the medial axis, see Section 4.3.

The relation (6.1) is the core of our approach. It allows us to edit the original mesh $\mathcal{M}$ via modifying its skeletal mesh $\mathcal{S}$. Below we explain how to achieve a robust extraction of the skeletal mesh and build representation (6.1).

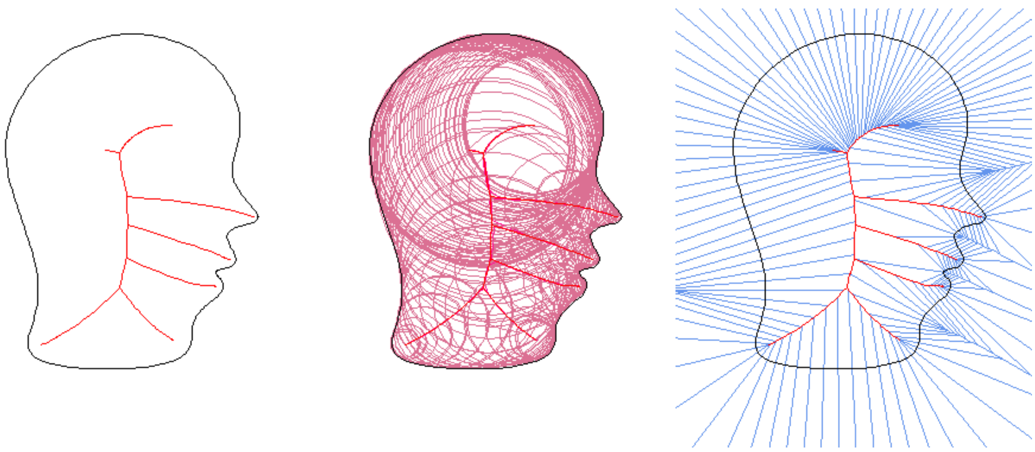


Figure 6.5: Left: a closed 2D curve and its medial axis. Middle: the medial axis is formed by the centers of all inner bitangent circles. Right: the medial axis is approximated by the vertices of the Voronoi diagram generated by points scattered densely over the curve.

Pre-Smoothing. Since the medial axis of a shape approximated by a mesh is very sensitive to the mesh quality, we first apply the bilaplacian tangent flow [WDSB00] to the mesh in order to improve the mesh quality. As demonstrated in Figures 3.3 and 3.4 of Chapter 3, the bilaplacian tangent smoothing regularizes a mesh. If the step-size of the bilaplacian tangent flow is sufficiently small, the flow keeps almost does not affect the mesh geometry while improving the aspect ratios of the mesh triangles.

Figure 6.6: Left: a cow mesh and its inner skeletal mesh, notice that the skeletal mesh intersects the cow mesh. Right: the cow mesh improved by several iterations of the bilaplacian tangent flow and its skeletal mesh improved original cow mesh via 100 bilaplacian tangent flow provides a good skeletal mesh.

This preprocessing step improves dramatically the quality of the skeletal mesh, as demonstrated in Figure 6.6. The left image shows an original cow mesh and its inner skeletal mesh. The skeletal mesh intersects the cow mesh while the true medial axis is located inside the cow model. The cow mesh in the right image is improved by several iterations of the bilaplacian tangent flow. The inner skeletal mesh of the improved cow mesh provides with a much better approximation of the true skeletal mesh.

Skeletal Mesh Extraction. In order to extract a skeletal mesh $\mathcal{S}$ from a given original mesh $\mathcal{M}$, we employed the Voronoi-based two-sided approximation of the medial axis proposed in [HBK02]. First the Voronoi cells are calculated for all vertices of $\mathcal{M}$ by using the Quickhull algorithm [BDH96]. Every vertex of $\mathcal{M}$ associates with a Voronoi cell. Consequently, one-to-one correspondences between the vertices of $\mathcal{M}$ and the associated vertices of $\mathcal{S}$ are established by assigning a linear combination of the farthest Voronoi sites in [YBS03] and the Voronoi poles [ABK98] in [YBS06c, YBS06a] as skeletal mesh vertices. The connectivity of $\mathcal{S}$ is copied from the connectivity of $\mathcal{M}$. Hence, each triangle of $\mathcal{M}$ also has a one-to-one correspondence with the associated triangle of $\mathcal{S}$. Figures 6.7 and 6.8 illustrate the skeletal meshes extracted from the given 3D meshes. The extracted skeletal mesh is a manifold approximation of the medial axis, therefore, conventional mesh processing methods can be applied to $\mathcal{S}$.

Post-Smoothing. Instead of high precision approximations of the medial axis required in surface reconstructions [ABK98, DZ02], a spike-less and non-degenerated skeletal mesh is desirable for our purpose. Similar as the pre-smoothing, the following special smoothing scheme is employed to $\mathcal{S}$ if $\mathcal{S}$ is noisy or degenerated. Assume that the normals of $\mathcal{M}$ are oriented from $\mathcal{S}$ to $\mathcal{M}$. The bilaplacian tangential smoothing is applied for any vertex of $\mathcal{S}$ such that the inner

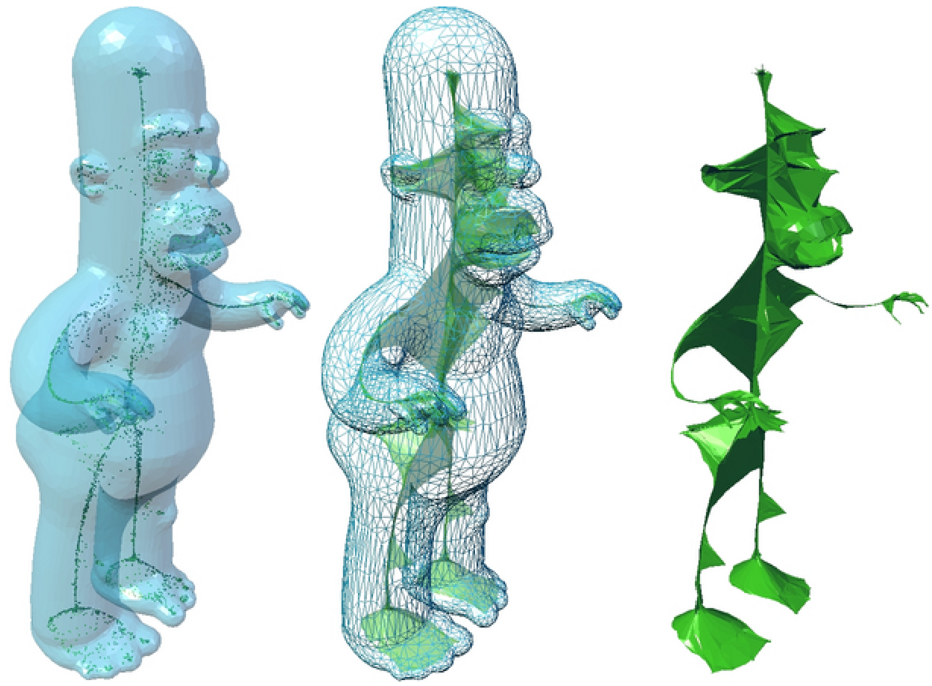


Figure 6.7: Skeletal mesh extraction. Left: Homer mesh and its Voronoi poles. Center: the triangulation of the Voronoi poles by copying Homer mesh connectivity. Right: the skeletal mesh of Homer mesh.

(a) (b) (c) (d)

Figure 6.8: A skeletal mesh is extracted by using Voronoi poles. (a): A given original triangle mesh. (b): The inner Voronoi poles of the original mesh. (c): Triangulation of the Voronoi poles by coping the original mesh connectively. (d): The skeletal mesh with its associated medial ball radius function.

product between a smoothing vector of the vertex of $\mathcal{S}$ and a one-to-one corresponding vertex normal of $\mathcal{M}$ is negative; otherwise, the vertex position of $\mathcal{S}$ is not moved by the smoothing.

6.2 Skeletal Mesh Editing

Consider a free-form deformation of the skeletal mesh $\mathcal{S}$. Our method is not restricted by any particular space deformation technique used for editing the skeletal mesh. In our implementation we use a set of free-form deformation tools developed in [YBS02].

Also following [Blo02], we have implemented the skeletal sub-space deformations (SSD) [MTLT88, LCJ94]. Let $\mathbf{y}_j$ and $\mathbf{y}_j^d$ be an original local frame origin and a deformed local frame origin, respectively. Any position $\mathbf{z} \in \mathfrak{R}^3$ is transformed according to

$$\sum_j w_j(\mathbf{y}_j^d + A_j(\mathbf{z} - \mathbf{y}_j)), \quad A_j = B_j^d B_j^{-1}, \tag{6.2}$$

where B_j and B_j^d are the original and deformed frames, respectively. The frame is usually composed by three axes (3x3 matrices). Here $\sum_j w_j = 1$ and w_j is a normalized Gaussian-like function of the distance $|\mathbf{z} - \mathbf{y}_j|$. The SSD equation (6.2) can be interpreted as weighted blending of local transformations A_j for $\mathbf{z}$ embedded in the local frame coordinate system.

To implement stick-figure skeletons, one of the frame axes can be assigned to a normalized edge of the stick-figure skeleton. The joints of stick-figure skeletons may associate with more than one frame. The initial frame origins as control points can be picked on the skeletal mesh vertices, see the image (a) of Figure 6.4. Besides the Euclidean distance $|\mathbf{z} - \mathbf{y}_j|$, we use the geodesic distances on original and skeletal meshes.

6.3 Basic Mesh Deformation Process

A direct reconstruction of a deformed mesh from the deformed skeletal mesh according to (6.1) may produce severe self-intersections of the deformed mesh. So we use a homotopy method to decompose the deformation into a sequence of L deformations connecting the original skeletal mesh $\mathcal{S}_0 = \mathcal{S}$ and deformed skeletal mesh $\mathcal{S}_d$:

$$\mathcal{S}_j = \mathcal{S}_0 + j\frac{\mathcal{S}_d - \mathcal{S}_0}{L}. \tag{6.3}$$

Now the corresponding deformations of the original mesh are computed as

$$\mathcal{M}_j = \mathcal{S}_j + \mathbf{d}\mathbf{N}_{j-1}, \quad j = 1, 2. \dots, L, \tag{6.4}$$

where $\mathbf{N}_0$ is the field of unit mesh normals for $\mathcal{M} = \mathcal{M}_0$ and $\mathbf{N}_j$ is the field of unit mesh normals for $\mathcal{M}_j$. The scalar field of displacements $\mathbf{d}$ is not changed during the deformation steps. According to our numerical experiments, the decomposition into $L = 3$ steps delivers a satisfactory combination of quality and speed.

6.3.1 Removing Folds and Protrusions

If a large skeletal mesh deformation is applied, see, for instance, Figure 6.9, the resulting deformed mesh $\mathcal{M}_d$ may have some defects, as demonstrated in the left images of Figure 6.11. In this section, we explain how to remove such defects of the deformed mesh as folds and protrusions, as seen in Figure 6.10. One possible way to avoid such mesh defects consists of reconstructing the deformed mesh $\mathcal{M}_d$ from the deformed skeletal mesh $\mathcal{S}_d$ as the envelope of medial balls [ACK01a] centered at the vertices of $\mathcal{S}_d$. However it works well only for dense meshes. So we have chosen a different approach based on mesh evolutions.

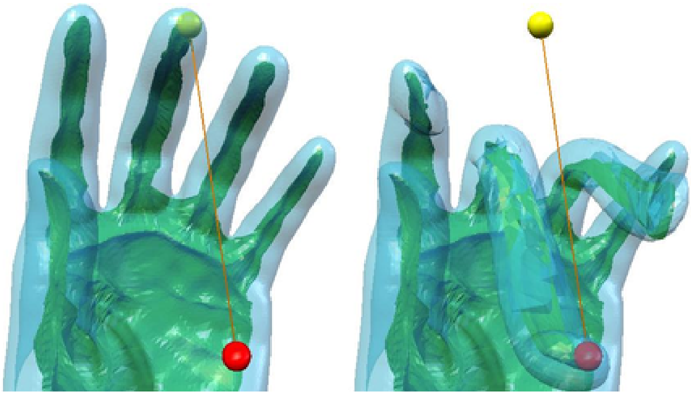


Figure 6.9: A large skeleton-based deformation of a hand model.

We consider the following mesh evolution

$$\frac{\partial \mathcal{M}}{\partial t} = -\alpha \triangle^2 \mathcal{M} - \mathbf{F} - \mathbf{V}, \quad \mathcal{M}(0) = \mathcal{M}_d, \tag{6.5}$$

where the negative bilaplacian $-\Delta^2$ and force $-\mathbf{V}$ are used for mesh relaxation and regularization purposes and force $-\mathbf{F}$ pushes the evolving mesh towards the envelope of the medial balls.

We approximate the bilaplacian operator via the bi-umbrella operator [KCVS98]. The parameter $\alpha > 0$ is not constant. Let us consider a mesh vertex $\mathbf{x}^d$ and its neighbors, compute the umbrella operators (vectors) for them, and count the number of those neighbors whose umbrella vectors form an obtuse angle with the umbrella vector at $\mathbf{x}^d$. We assign $\alpha = 0.25$ to $\mathbf{x}^d$ if the fraction that obtuse angles is less than 0.3. Otherwise we set $\alpha = 0$ at $\mathbf{x}^d$.

We want to define the force $\mathbf{F}$ such that $-\mathbf{F}$ fits the evolving mesh to the envelope of the medial balls. For each triangle T of the deformed skeletal mesh let us consider the convex hull of the medial balls centered at triangle vertices. A general approach to compute the convex hull of a set of spheres can be found in [BCD+96]. However, in our simple case, the convex hull is computed analytically: we use the fact that the convex hull can be computed as the envelop of the balls centered inside the triangle and obtained by the trilinear interpolation of the balls centered at the vertices. We describe the envelop as an implicit function. Let us define a function $w = E_T(P)$ at point P as the value of the implicit function at P. Now consider a mesh vertex $\mathbf{x}^d$, the set of mesh triangles incident with $\mathbf{x}^d$ and their centroids C_j, $j = 1, \ldots, n$. The force $\mathbf{F}$ at $\mathbf{x}^d$ is defined by

$$\mathbf{F}(\mathbf{x}^d) = \frac{1}{n} \sum_{j=1}^{n} E_{T_j}(C_j) \nabla E_{T_j}(C_j),$$

where T_j is a deformed skeletal mesh triangle corresponding to the mesh triangle with centroid C_j. Notice that the force $E \nabla E$ attracts the vertices to the zero level set of E.

The force $\mathbf{V}$ is defined as the projection of the bilaplacian vector on the plane orthogonal to $\mathbf{F}$

$$\mathbf{V} = \triangle^2 \mathcal{M} - (\triangle^2 \mathcal{M} \cdot \frac{\mathbf{F}}{|\mathbf{F}|})\frac{\mathbf{F}}{|\mathbf{F}|}.$$

As illustrated in Figure 3.4, the bilaplacian operator is a better choice than the single Laplacian operator for the tangential component.

Figure 6.10 explains why flow (6.5) eliminates mesh folds and protrusions.

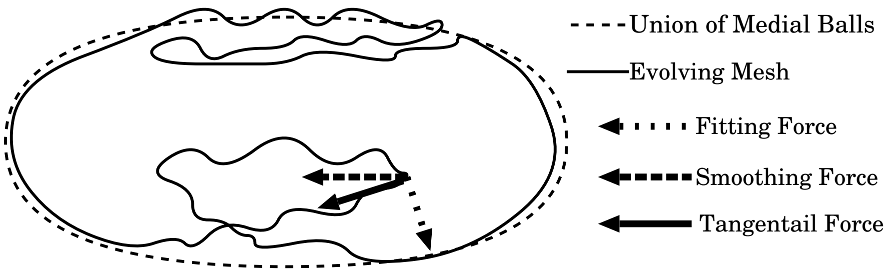


Figure 6.10: Effect of (6.5). Force $\mathbf{F}$ pushes the mesh vertices towards the envelope of the medial balls. Two other forces in the right hand-side of (6.5), tangential force $\mathbf{V}$ and smoothing force $-\alpha\triangle^2\mathcal{M}$, are used to eliminate mesh folds and protrusions.

The right images of Figure 6.11 demonstrate fixing defects of the deformed hand mesh by (6.5).

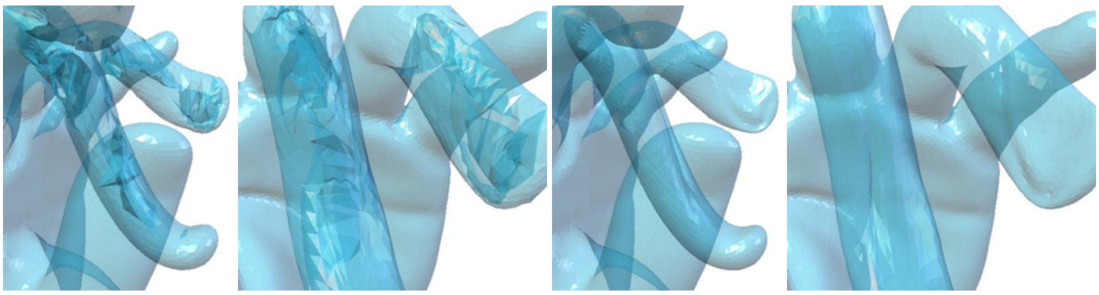


Figure 6.11: Left: zoomed parts of the deformed hand model from the right image of Figure 6.9. Right: fixing mesh defects by (6.5).

6.3.2 Eliminating Global and Local Self-Intersections

The deformed mesh still may have global and local self-intersections, as sketched in the left image of Figure 6.12.

Again we use a mesh evolution approach in order to eliminate possible global and local self-intersections of the deformed mesh. Consider a vertex $\mathbf{x}^d = (x, y, z)$ of the deformed mesh and its corresponding vertex $\mathbf{s}^d = \left(s_x, s_y, s_z\right)$ of the deformed skeletal mesh. Let us define the function

$$g(x, y, z, \mathbf{s}^d) = (x - s_x)^2 + (y - s_y)^2 + (z - s_z)^2 - d^2,$$

where d is the radius of the medial ball centered at $\mathbf{s}^d$. Now we introduce a function $f(x, y, z)$ whose zero level set $f(x, y, z) = 0$ approximates the envelope of the medial balls. Let us divide the bounding box (unit box) uniformly into $B_l \times B_m \times B_n$ voxels $G_{l,m,n}$. In practice, we use $\{B_l, B_m, B_n\} = \{20, 20, 20\}$ which gives us satisfactory results. We set

$$h_{l,m,n}(x, y, z) = \min \left\{ g(x, y, z, \mathbf{s}^d) \ : \ \mathbf{s}^d \in G_{l,m,n} \right\},$$

where the minimum is taken over all skeletal mesh vertices $\mathbf{s}^d$ that belong to the cell $G_{l,m,n}$. Then $f(x, y, z)$ is defined for $(x, y, z) \in G_{l,m,n}$ by

$$f(x, y, z) = \begin{cases} h_{l,m,n}(x, y, z) & \text{if} \quad h_{l,m,n}(x, y, z) < 0; \\ \min \{h_{l,m,n}, h_{l\pm 1,m\pm 1,n\pm 1}\} & \text{otherwise.} \end{cases}$$

The mesh evolution we use to eliminate global and local self-intersections evolves each mesh by

$$\frac{\partial \mathcal{M}}{\partial t} = -f(\mathcal{M}) \nabla f(\mathcal{M}) - W \triangle^2(\mathcal{M}). \tag{6.6}$$

Here $-f \nabla f$, the antigradient of $\frac{1}{2} f^2$, pushes the mesh vertices towards the zero level set of $f(x, y, z)$. The weight $W(\mathbf{x}^d)$ in (6.6) for a mesh vertex $\mathbf{x}^d \in \mathcal{M}$ is given by

$$W(\mathbf{x}^d) = \frac{|f(\mathbf{x}^d) - g(\mathbf{x}^d, \mathbf{s}^d)|}{\max\limits_{\mathbf{x}^d \in \mathcal{M}} |f(\mathbf{x}^d) - g(\mathbf{x}^d, \mathbf{s}^d)|},$$

where $\mathbf{s}^d$ is the skeletal mesh vertex corresponding to mesh vertex $\mathbf{x}^d$.

Similar to (6.5) the bilaplacian term in (6.6) makes the flow more stable while another term in the left hand-side of (6.6) pushes the mesh vertices towards the envelope of the medial balls centered at the vertices of the deformed skeletal mesh.

Mesh fairing with (6.6) is demonstrated in Figure 6.12 for a large-scale deformation of an ellipsoid model, see also Figure 6.13.

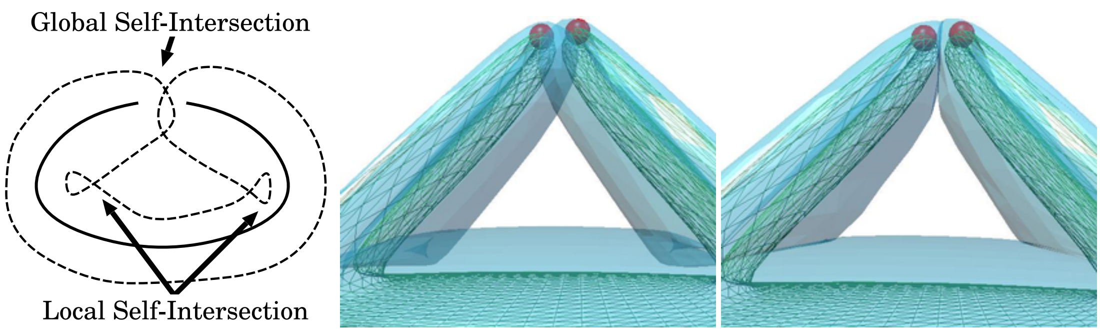


Figure 6.12: Fairing global and local self-intersections. Left: global and local self-intersections. Center: folds and protrusions are removed by (6.5), however local and global self-intersections remain. Right: removing the self-intersections by (6.6).

6.3.3 Gathering All Together

An example of our basic mesh deformation process is demonstrated in Figure 6.14 Given a mesh, first its Voronoi-based skeletal mesh is extracted. Next a free-form deformation is applied to the

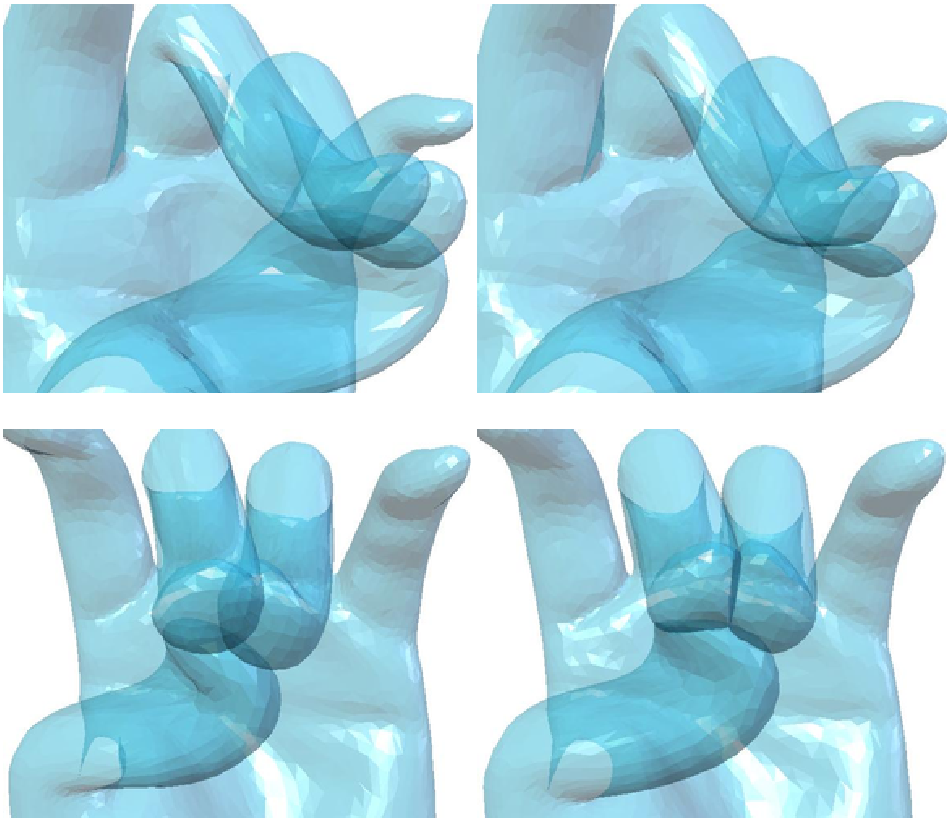


Figure 6.13: Removing self-intersections of the deformed hand mesh from Figure 6.14(d). Left: various views at a zoomed part of Figure 6.14(d). Right: corresponding views at the same parts of Figure 6.14(e). The self-intersections are gone.

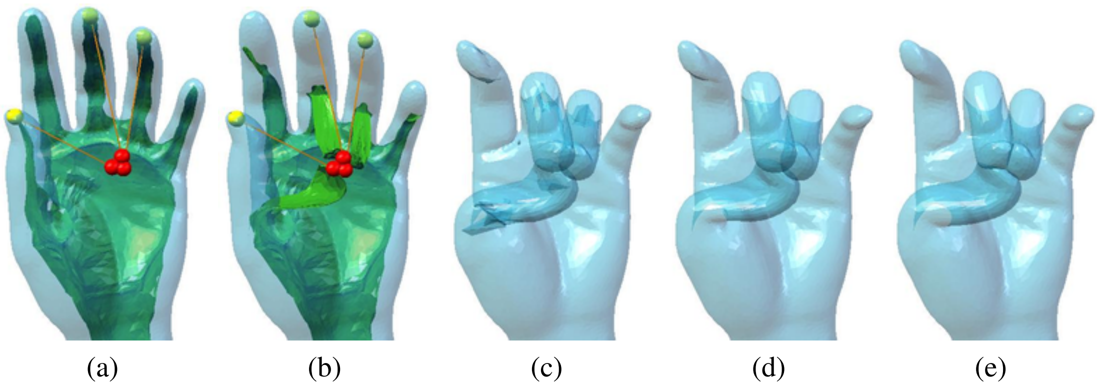


Figure 6.14: Basic mesh deformation process. (a): The original hand mesh, its skeletal mesh, and control points to be used to deform the skeletal mesh. (b): A deformed skeletal mesh. (c): Folds and protrusions are observed in the deformed mesh. (d): The folds and protrusions are removed by (6.5); however global and local self-intersections are still presented. (e): The global and local self-intersections are eliminated by (6.6).

skeletal mesh. Then a deformed mesh is reconstructed from the deformed skeletal mesh according to (6.1). We employ (6.3), (6.4) with $L = 3$ to produce the deformed mesh. Finally mesh fairing is applied. Mesh evolution (6.5) eliminates folds and protrusions and mesh evolution (6.6) removes the self-intersections.

6.4 Combining with Displaced Subdivision Surfaces

The most time consuming steps of our basic method presented in Section 6.3 are mesh fairing stages described in subsections 6.3.1 and 6.3.2. For example, for the hand model consisting of 16K triangles only, its deformation processes shown in Figure 6.13 takes approximately one and a half minutes for eliminating self-intersections, about six seconds for removing folds and protrusions, and less than one second for all the other operations (a Java3D implementation on a 1.7GHz Pentium 4 computer was used). In order to perform the deformation process in a matter of few seconds we combine it with a displaced subdivision surface representation [LMH00].

Multiresolution mesh representations are powerful tools to deform a large mesh according to deformations of its control mesh. The displaced subdivision surface representation, DSS-rep, is a compact surface representation capturing small-scale details of an original surface as a scalar displacement field over a decimated and then subdivided surface.

Given a dense mesh, first we obtain a DSS representation of the mesh: a decimated mesh and a scalar displacement field. Then we build our skeleton-based representation of the decimated mesh. The decimated mesh has much fewer vertices than the original dense mesh and do not contain small-scale details. This leads to fast and robust extraction of the Voronoi-based skeletal mesh for the decimated mesh. Moreover DSS-rep protects fine geometry features of the original mesh from being damaged by mesh evolutions (6.5) and (6.6). The mesh deformation process is now organized as follows: a free-from deformation is applied to the skeletal mesh of the decimated mesh and implies a deformation of the decimated mesh. The deformed mesh is then subdivided and, finally, a deformation of the original dense mesh is obtained from the subdivided deformed mesh by adding the scalar displacement field.

To demonstrate how the above combination of DSS-rep and the skeleton-driven mesh deformation approach described in previous sections works we used the dragon, cow, and hand models. The models are remeshed (topological noise removal, decimation, subdivision) in order to improve their quality. See Figures 6.15-6.18 for the results. Coloring by the mean curvature is used for a quality evaluation of the deformed models.

In these examples, the whole mesh deformation process takes only a few seconds without taking into account computing the DSS representation. In our current implementation, we compute the DSS representation without its most computationally expensive optimization step [LMH00]. Besides DSS-rep has to be computed only once.

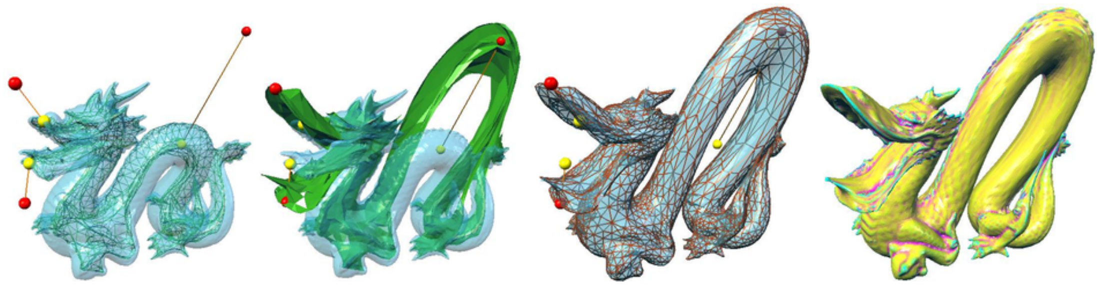


Figure 6.15: Skeleton-based deformations enriched by DSS: the dragon model has 100K triangles while its skeletal mesh consists of 6K triangles only.

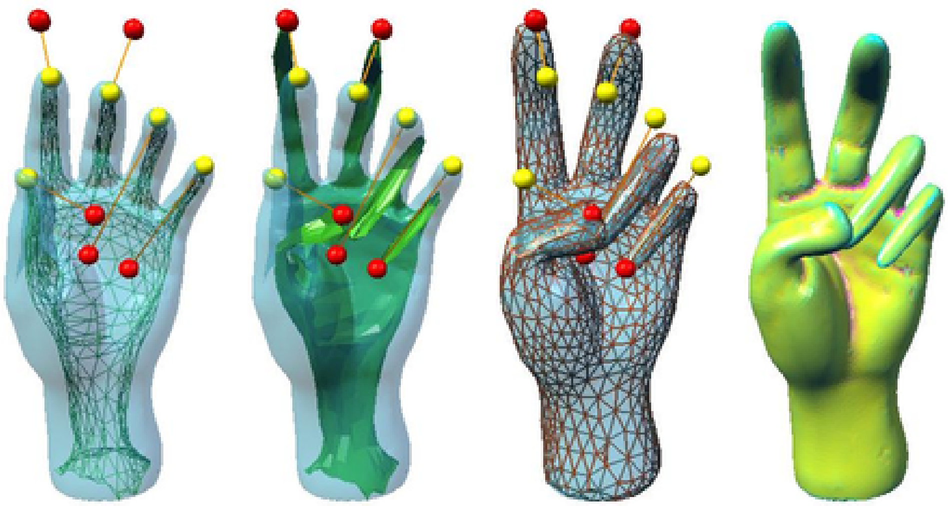


Figure 6.16: Skeleton-based deformations enriched by DSS: the hand model has 38K triangles while its skeletal mesh consists of 2K triangles only.

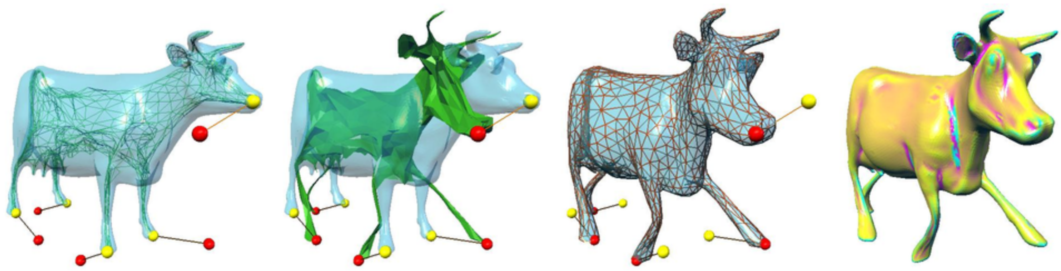


Figure 6.17: Skeleton-based deformations enriched by DSS: the cow model consists of 45K triangles while its skeletal mesh has 3K triangles only.

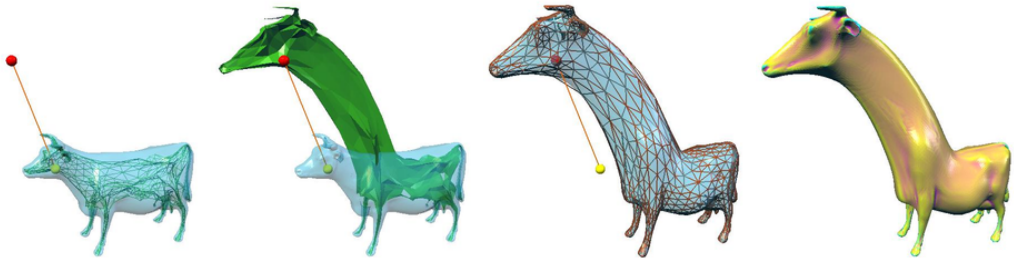


Figure 6.18: Another global deformation of the cow model.

6.5 Variational Skeleton-driven Deformation

The basic mesh deformation process proposed in Section 6.3 (also [YBS03]) preserves a shape thickness. Unfortunately, fine geometric details of the original shape are often lost during processes described in Section 6.3.1 (smoothing fold-overs [YBS03]). A similar problem arises in the space deformation method proposed in [Blo02] because of convolving the distance field.

In this section, we follow [YBS06c, YBS06a] and present a new technique for reconstructing a deformed shape according to the edited skeletal mesh. The technique consists of combining skeleton-driven mesh deformations with the so-called discrete differential coordinates.

The discrete differential coordinates of a mesh vertex are defined by a discrete Laplacian of the vertex. Each coordinate associates with a corresponding element of the Laplacian vector. Since seminal works [Ale00, LSCO+04, YZX+04], mesh deformations based on discrete differential coordinates are intensively studied because of their detail preserving ability. These methods first interpolate or propagate the user-specified affine transformations over the mesh, and then the final deformations are obtained by solving the discrete approximations of a Poisson equation. Geometric details of the mesh are embedded into a discrete Laplacian matrix, and solving the Poisson equation diffuses a deformation error over the mesh. Therefore, the fine geometric details are preserved with certain smoothness during their deformations. See a recent review of this topic [Sor05] and references therein. Since the conventional deformation techniques based on discrete differential coordinates do not preserve original shape thickness, our approach based on combining the skeleton-driven with discrete differential coordinates achieves more natural deformations than the conventional methods.

Furthermore, the use of discrete differential coordinates allows us to achieve shape preserving self-intersection fairing. We develop a new mesh evolution technique which eliminates certain self-intersections of the deformed mesh simultaneously preserving fine geometric details by minimizing the thickness and deformation errors.

Given a triangle mesh $\mathcal{M}$, consider its skeletal mesh $\mathcal{S}$ extracted from $\mathcal{M}$ by the method described in Section 6.1, and a deformed skeletal mesh $\mathcal{S}_d$. Here $\mathcal{S}_d$ is obtained from $\mathcal{S}$ by applying space deformations described in Section 6.2. The basic procedure of our variational skeleton-driven deformation technique is as follows. First, a fragmented mesh $\mathcal{M}_F$ is generated by applying local transformations to all triangles of $\mathcal{M}$ where the transformations are defined by according to the local frames attached $\mathcal{S}$ and $\mathcal{S}_d$. Then a final deformed mesh $\mathcal{M}_d$ is obtained by stitching the fragmented mesh triangles based on minimizing a deformation error. Here the error is given by a squared difference between the discrete differential coordinates of $\mathcal{M}_F$ and $\mathcal{M}_d$. In [SP04, YZX+04, ZRKS05], similar minimizing strategies are used for stitching fragmented meshes.

Let $\mathbf{x}$, $\mathbf{s}$, and $\mathbf{s}^d$ be the vertices of $\mathcal{M}$, $\mathcal{S}$, and $\mathcal{S}_d$, respectively. Since the deformation from $\mathcal{S}$ to $\mathcal{S}_d$ does not change the connectivity of $\mathcal{S}_d$, there are one-to-one correspondences between $\mathbf{x}_i \in \mathbf{x}$, $\mathbf{s}_i \in \mathbf{s}$, and $\mathbf{s}_i^d \in \mathbf{s}^d$. Consider a final deformed mesh $\mathcal{M}_d$ corresponding to $\mathcal{S}_d$. Let $\mathbf{x}^d$ be the vertices of $\mathcal{M}_d$. Recall that $\mathcal{M}_d$ is given by the shape representation (6.1): $\mathcal{M}_d = \mathcal{S}_d + d\mathbf{N}_d$ where $\mathbf{N}_d$ are the unit normals of $\mathcal{M}_d$, see Figure 6.19.

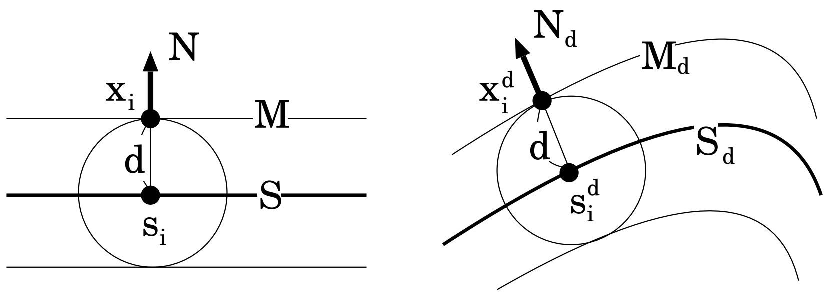


Figure 6.19: Shape representation (6.1).

The radius of medial ball whose center is located on $\mathbf{s}_i$ is given by

$$d(i) = |\mathbf{x}_i - \mathbf{s}_i|. \tag{6.7}$$

Let $\mathbf{n}_i^d \in \mathbf{N}_d$ be a unit normal of $\mathcal{M}_d$ at $\mathbf{x}_i^d \in \mathbf{x}^d$. Thus, our shape representation (6.1) can be re-written as $\mathbf{x}_i^d = \mathbf{s}_i^d + d(i)\mathbf{n}_i^d$. In Section 6.3, we approximate $\mathbf{N}_d$ by decomposing a deformation from $\mathcal{M}$ to $\mathcal{M}_d$ into a sequence of deformations. Here let us approximate $\mathbf{n}_i^d$ by the following local transformation in order to use discrete differential coordinates.

Local Transformation. Let $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\}$, $\{\mathbf{s}_i, \mathbf{s}_j, \mathbf{s}_k\}$, and $\{\mathbf{s}_i^d, \mathbf{s}_j^d, \mathbf{s}_k^d\}$ be corresponding triangles of $\mathcal{M}$, $\mathcal{S}$, and $\mathcal{S}_d$, respectively. We attach the original local frame $B_0 = (\mathbf{v}_0, \mathbf{t}_0^1, \mathbf{t}_0^2)$ to all triangles of $\mathcal{S}$ where $\mathbf{v}_0 = \mathbf{t}_0^1 \times \mathbf{t}_0^2$, $\mathbf{t}_0^1 = (\mathbf{s}_j - \mathbf{s}_i)/|\mathbf{s}_j - \mathbf{s}_i|$, and $\mathbf{t}_0^2 = (\mathbf{s}_k - \mathbf{s}_i)/|\mathbf{s}_k - \mathbf{s}_i|$. The skeletal mesh editing procedure changes the frame from B_0 to the deformed local frame B_d attached to a corresponding triangle of $\mathcal{S}_d$. Here B_d is given by the same calculation procedure of B_0 by using $\{\mathbf{s}_i^d, \mathbf{s}_j^d, \mathbf{s}_k^d\}$ instead of $\{\mathbf{s}_i, \mathbf{s}_j, \mathbf{s}_k\}$. See Figure 6.20.

Let $\mathbf{n}_i \in \mathbf{N}$ be a unit normal of $\mathcal{M}$ at $\mathbf{x}_i$. Since the displacement $\mathbf{x}_i - \mathbf{s}_i$ represents a normal vector of $\mathcal{M}$ [ACK01b] at $\mathbf{x}_i$, $\mathbf{n}_i$ is given by $\frac{\mathbf{x}_i - \mathbf{s}_i}{|\mathbf{x}_i - \mathbf{s}_i|}$. Consider a local coordinate representation of $\mathbf{N}$ on $\mathcal{S}$ such that the coordinates of $\mathbf{n}_i$ are represented by $B_0^{-1}\mathbf{n}_i$. The coordinate transformation $B_d B_0^{-1}\mathbf{n}_i$ gives us an approximation of $\mathbf{n}_i^d$, the unit normal at $\mathbf{x}_i^d$. Thus, we have

$$\mathbf{n}_i^d \approx \frac{B_d B_0^{-1}(\mathbf{x}_i - \mathbf{s}_i)}{|B_d B_0^{-1}(\mathbf{x}_i - \mathbf{s}_i)|}.$$

Here $B_d B_0^{-1}$ is calculated per triangles. Therefore, applying the following transformation (6.8) to all triangles of $\mathcal{M}$ generates a fragmented mesh $\mathcal{M}_F$. See Figure 6.21.

$$\mathbf{x}_l^T = \mathbf{s}_l^d + d(l)\frac{A(\mathbf{x}_l - \mathbf{s}_l)}{|A(\mathbf{x}_l - \mathbf{s}_l)|}, \quad A = B_d B_0^{-1}, \tag{6.8}$$

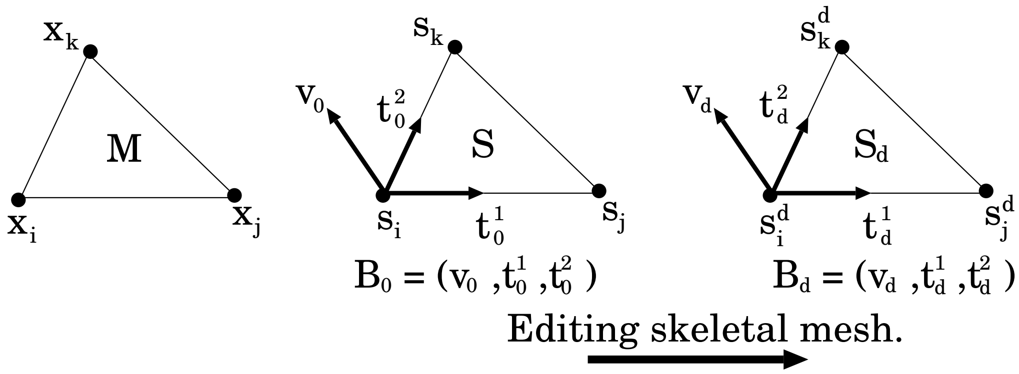


Figure 6.20: Corresponding triangles of $\mathcal{M}$, $\mathcal{S}$, and $\mathcal{S}_d$. Local coordinate frames are attached to the corresponding triangles of $\mathcal{S}$ and $\mathcal{S}_d$.

where $l = i, j, k$, $\mathbf{x}_l^T \in \mathbf{x}^T$ is the fragmented mesh vertex and $d(l)$ is the medial ball radius (6.7). Compared with similar transformations used in other deformations e.g. [SP04], the transformation (6.8) preserves shape thickness. Also the above equation (6.8) can be considered as a discrete approximation of the shape representation (6.1): $\mathcal{M}_d = \mathcal{S}_d + d\mathbf{N}_d$.

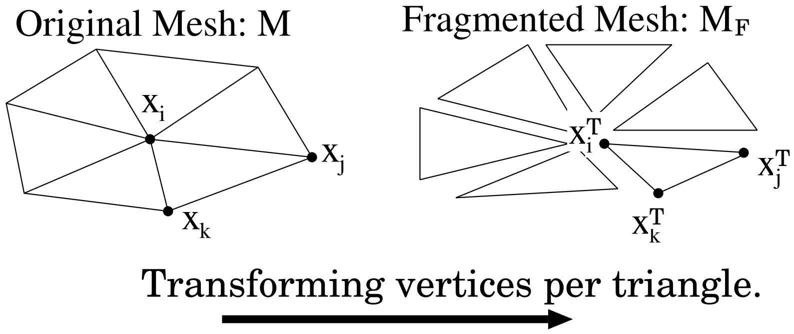


Figure 6.21: A fragmented mesh $\mathcal{M}_F$, triangle soup, is generated via (6.8).

Although the transformation (6.8) generates stretch distortions, it does not destroy fine geometric details compared with shearing transformations in $\Re^3$ as mentioned in [SCOL$^+$04] because the stretch of (6.8) is locally embedded in the skeletal mesh. In [LSCOL05] authors embed their local transformations to the original mesh in order to obtain semi-rigid deformations.

In the case of space deformations, e.g. (6.2) or [SK00b, KO03, Blo02], the final position of $\mathbf{x}_i^d$ of $\mathcal{M}_d$ is calculated by averaging the fragmented mesh vertices $\mathbf{x}^T$. Averaging the transformed vertices in Euclidean coordinates requires a large influence region which contains a lot of transformed vertices in order to generate nice deformations as indicated in [KO03]. In our approach, we calculate an average of the fragmented mesh vertices $\mathbf{x}^T$ in discrete differential coordinates where only one-link neighbor vertices are required for the blending.

Discrete Differential Coordinates. Consider two graphs $G_{\mathcal{M}}$ and $G_{\mathcal{M}_F}$ corresponding to two pairs of meshes $\{\mathcal{M}, \mathcal{S}\}$ and $\{\mathcal{M}_F, \mathcal{S}_d\}$, respectively. Since we have the one-to-one correspondence between $\mathbf{x}$ and $\mathbf{s}$, the graph structures are constructed by adding edges between $\mathcal{M}$ and $\mathcal{S}$ ($\mathcal{M}_F$ and $\mathcal{S}_d$) as illustrated in Figure 6.22. Here the vertex sets of $G_{\mathcal{M}}$ and $G_{\mathcal{M}_F}$ consist of

$\{\mathbf{x}, \mathbf{s}\}$ and $\{\mathbf{x}^T, \mathbf{s}^d\}$, respectively. Consider also an another graph $G_{\mathcal{M}_d}$ whose vertex set consists of $\{\mathbf{x}^d, \mathbf{s}^d\}$. Here $G_{\mathcal{M}_d}$ is equipped with the same connectivity of $G_{\mathcal{M}}$.

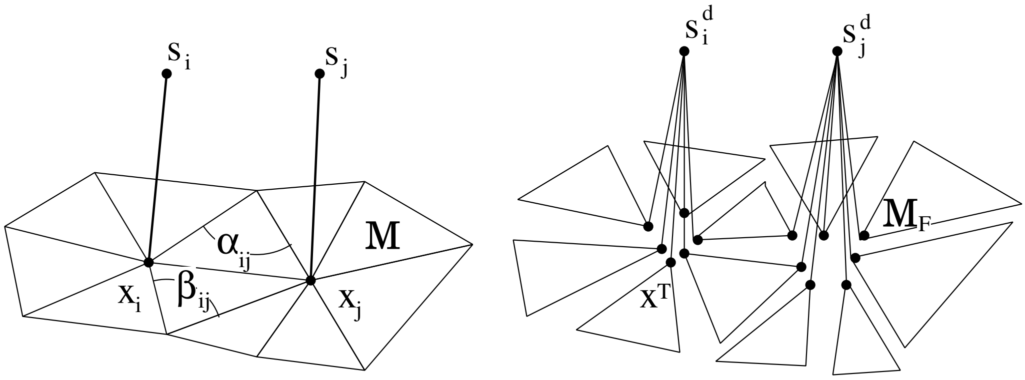


Figure 6.22: Two graphs $G_{\mathcal{M}}$ (Left) and $G_{\mathcal{M}_F}$ (Right) are considered.

Consider a weighted graph whose vertices are given by $\{v_1, v_2, ..., v_i, ..\}$. If there is the edge between v_i and v_j then they are adjacent vertices. The ij element of a graph-Laplacian matrix of the graph is defined by

$$\begin{cases} -w_{ij} & \text{if } v_j \text{ and } v_i \text{ are adjacent,} \\ \sum_{k\in N(i)} w_{ik} & \text{if } v_j = v_i, \\ 0 & \text{otherwise} \end{cases}$$

where w_{ij} is a weight associated with the edge between v_i and v_j. Here $N(i)$ is the index set of adjacent vertices of v_i. See [Chu97] for mathematical theory of a graph.

Consider the graph-Laplacian matrix for $G_{\mathcal{M}}$ where the each edge is equipped with a weight w. Here we use standard cotangent weights [PP93] $w = \cot\alpha_{ij} + \cot\beta_{ij}$ for edges between $\mathbf{x}_i$ and $\mathbf{x}_j$ where the angels α_{ij} and β_{ij} are defined in Figure 6.22. For the weight associated with the edge between $\mathbf{x}_i$ and $\mathbf{s}_i$, we use $w = 1$. Thus the graph-Laplacian matrix associated with $G_{\mathcal{M}}$ for $(\mathbf{x}, \mathbf{s})^*$ is given by

$$\begin{pmatrix} I+L_M & -I \\ -I & I \end{pmatrix}, \tag{6.9}$$

where I is the identity matrix, L_M is a mesh Laplacian matrix, and $\mathbf{a}^*$ stands for a transpose of vector $\mathbf{a}$. The ij element of L_M is given by

$$\begin{cases} -(\cot\alpha_{ij} + \cot\beta_{ij}) & j \in N(i) \\ \sum_{k\in N(i)}(\cot\alpha_{ik} + \cot\beta_{ik}) & j = i \\ 0 & \text{otherwise} \end{cases}$$

where $N(i)$ is the index set of the one-link neighborhood vertices of $\mathbf{x}_i$.

Consider the graph-Laplacian operator for $G_{\mathcal{M}_F}$ where the each edge is equipped with a weight w. We use $w = \cot\angle\mathbf{x}_i\mathbf{x}_k\mathbf{x}_j$ for the edge between $\mathbf{x}_i^T$ and $\mathbf{x}_j^T$ and $w = \cot\angle\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k$ for the edge between $\mathbf{x}_i^T$ and $\mathbf{x}_k^T$, see the right image of Figure 6.21. Also the weight for the edge between $\mathbf{x}_i^T$ and $\mathbf{s}_i^d$ is equal to $\frac{1}{T_i}$ where T_i is the number of the one-link neighborhood triangles of $\mathbf{x}_i$.

The discrete differential coordinates of $G_{\mathcal{M}_d}$ and $G_{\mathcal{M}_F}$ are the above graph-Laplacians of their vertices $(\mathbf{x}^d, \mathbf{s}^d)$ and $(\mathbf{x}^T, \mathbf{s}^d)$, respectively.

Stitching Fragmented Mesh. In order to obtain the final deformed mesh $\mathcal{M}_d$, let us minimize a difference between $G_{\mathcal{M}_d}$ and $G_{\mathcal{M}_F}$ in the discrete differential coordinates subject to the following boundary condition: the boundary of $G_{\mathcal{M}_d}$ is fixed to $\mathbf{s}^d$. This boundary condition changes the graph-Laplacian from (6.9) to

$$L_0 = \begin{pmatrix} I+L_M & -I \\ 0 & I \end{pmatrix}. \tag{6.10}$$

More precisely, $\mathbf{x}^d$ is given by solving the following sparse linear system

$$\text{minimize } |L_0\mathbf{u} - \mathbf{b}|^2 \Rightarrow \mathbf{u} = L_0^{-1}\mathbf{b}, \tag{6.11}$$

where $\mathbf{u} = (\mathbf{x}^d, \mathbf{s}^d)^*$ and $\mathbf{b} = (\mathbf{x}^f, \mathbf{s}^d)^*$ is the averaged discrete differential coordinates of $G_{\mathcal{M}_F}$. Here the i-th element of $\mathbf{x}^f$ is given by

$$\sum_{j,k \in N(i)} w_1(\mathbf{x}_i^T - \mathbf{x}_j^T) + w_2(\mathbf{x}_i^T - \mathbf{x}_k^T) + \frac{(\mathbf{x}_i^T - \mathbf{s}_i^d)}{T_i}, \tag{6.12}$$

where $w_1 = \cot \angle \mathbf{x}_i\mathbf{x}_k\mathbf{x}_j$, $w_2 = \cot \angle \mathbf{x}_i\mathbf{x}_j\mathbf{x}_k$, and T_i is the number of the one-link neighborhood triangles of $\mathbf{x}_i$.

In [ZRKS05] authors evaluate the elements of their mesh Laplacian according to the fragmented mesh coordinates. Besides L_0 is a graph-Laplacian, our angles $\angle \mathbf{x}_i\mathbf{x}_k\mathbf{x}_j$ and $\angle \mathbf{x}_i\mathbf{x}_j\mathbf{x}_k$ do not depend on the fragmented mesh coordinates. Compared with [ZHS+05] which employed a graph-Laplacian for their deformation technique, our graph structure is much simple and has a nice geometric property as the shape thickness.

Figure 6.23 illustrates the main idea and procedure of our variational skeleton-driven deformation technique described above.

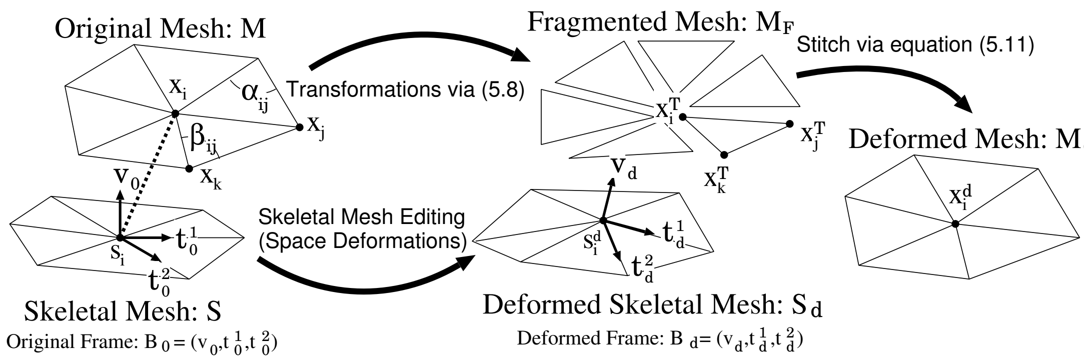


Figure 6.23: Variational skeleton-driven deformation framework.

Solving Linear System: Factorization. Due to recent developments of the direct sparse linear solvers [TCR03, Dav04], we only need factorization of L_0 once for producing L_0^{-1}. Then every deformation is obtained by updating $\mathbf{b}$ and a backward substitution of the factorized matrix with the updated $\mathbf{b}$. This gives us the linear computational complexity. Unfortunately L_0 is not a symmetric matrix. Consequently, we use the UMFPACK [Dav04] to factorize L_0 instead of the TAUCS [TCR03] which is specialized to a symmetric matrix employed in least-square systems of the Laplacian mesh deformations [LSCO+04, SCOL+04, LSCOL05]. Because $\mathbf{s}^d$ is being the fixed boundary, there are no ill-conditioned problems mentioned in [Sor05] caused by too small boundary conditions. According to our numerical experiments, L_0 is invertible as long as $\mathcal{M}$ is a non-degenerated manifold mesh.

Dense Mesh (332K triangles) Control Mesh (5.2K triangles)

	Triangle	Editing Skeleton	Update Geometry Factorization + Substitution	DSS	Total
Without DSS:	346K	94s	321s + 702s		1117s
DSS Mesh:	5.2K, 332K	**0.13s**	0.05s + **0.25s**	3.8s	4.23s

Figure 6.24: Advantages of multiresolution variational skeleton-driven deformations. The use of the multiresolution representation reduces excessive complexity of the skeletal mesh and accelerates the deformation process.

Multiresolution Representation. As described in Section 6.4, we implement the simplified version of DSS [LMH00] and combine with our variational skeleton-driven deformations. The skeletal mesh is extracted from the coarse control mesh of the DSS. The control mesh is deformed by our variational skeleton-driven deformations. Then a dense deformed mesh is obtained by using the DSS mechanism: subdividing the deformed control mesh and adding scaler displacements to the subdivided mesh normals.

Our method described in this section preserves fine geometric details, but the skeletal mesh of a textured mesh may have complex geometry. Hence, the skeletal mesh editing of the textured mesh without degenerations could become a very difficult task. Using multiresolution representations helps to reduce excessive complexity of the skeletal mesh. Figure 6.24 illustrates advantages of using the multiresolution representation for our variational skeleton-driven deformations.

6.5.1 Shape Preserving Self-Intersection Fairing

The final deformed mesh $\mathcal{M}_d$ obtained by solving the system (6.11) may contain self-intersections. Here $\mathcal{M}_d$ is an approximation of the envelope of the medial balls of $\mathcal{M}$ attached to $\mathcal{S}_d$. For a given smooth surface, the envelope of its medial balls is equivalent to the zero level set of the union of its medial balls. However the medial balls of $\mathcal{M}$ attached to $\mathcal{S}_d$ are not necessary inscribed maximal empty balls (medial balls) of $\mathcal{M}_d$ and, therefore, $\mathcal{M}_d$ may have self-intersections. In such cases, the mesh reconstructions based on the envelope and union of the attached medial balls form two different shapes, see Figure 6.25. The differences of them are exactly subsets of the envelope which we would like to eliminate.

The union of medial balls is defined by the signed distance function

$$f(\mathbf{z}) = \min_{\forall j} \left\{ |\mathbf{z} - \mathbf{s}_j^d| - d(j)) \; : \; \mathbf{z} \in \Re^3, \mathbf{s}_j^d \in \mathbf{s}^d \right\}, \tag{6.13}$$

where $d(j)$ is the medial ball radius (6.7).

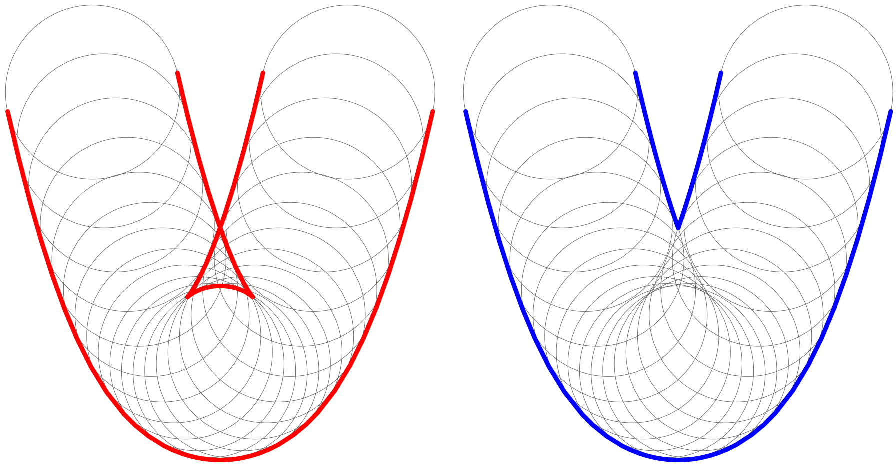


Figure 6.25: Envelope vs. Union. Left: the envelope of the balls. Right: the zero level set of the union of the balls.

Below we describe a novel self-intersection fairing method based on minimizing a self-intersection error in the discrete differential coordinates. The self-intersection error is measured by the squared distance $f^2(\mathbf{x}^d)$ from $\mathbf{x}^d$ to the zero level set of the union of the attached medial balls. Here $\mathbf{x}^d$ and f are given by the equations (6.11) and (6.13), respectively. Consider evolutions of a graph

$$\frac{\partial L_0 U(t)}{\partial t} = F(U), \tag{6.14}$$

$$F(U) = \begin{cases} f(\mathbf{z})\nabla f(\mathbf{z}) & \text{If} \quad \mathbf{z} = \mathbf{x}^d \\ 0 & \text{If} \quad \mathbf{z} = \mathbf{s}^d \end{cases}$$

where $U(t)$ is an evolving graph and $U(0) = (\mathbf{x}^d, \mathbf{s}^d)^*$ defined in previous subsection and L_0 is the graph-Laplacian defined in the equation (6.10). This evolution (6.14) is motivated by the shape preserving effect of L_0 and integrating the deformation and self-intersection errors. Note that $\mathbf{s}^d$ and L_0^{-1} are not changed by the evolution (6.14).

The equation of (6.14) is a gradient descent flow of the self-intersection error $f^2(\mathbf{x}^d)$ in differential coordinates. This gradient descent flow modifies the differential coordinates of $U(0)$. In order to preserve the fine geometric details of $\mathcal{M}$, L_0 is equipped with the original weights of $G_{\mathcal{M}}$ instead of the weights of $G_{\mathcal{M}_d}$ or $U(t)$. Hence, the evolution (6.14) minimizes not only the self-intersection error but also the deformation error simultaneously. One can find a similarity between (6.14) and Sobolev gradient flows, see [Neu83, Neu97] for mathematical theory of Sobolev gradients. In [CKPF05], a similar Sobolev gradient flow is employed for describing active contours.

The equation (6.14) is approximated by the following semi-implicit mesh evolutions.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \epsilon L_0^{-1} F(\mathbf{u}^n) \nabla F(\mathbf{u}^n), \tag{6.15}$$

where $\mathbf{u}^0 = (\mathbf{x}^d, \mathbf{s}^d)^*$.

Our evolution technique (6.15) is more robust and effective than the self-intersection fairing scheme proposed in Section 6.3.2 and [YBS03] because of our semi-implicit formulation and

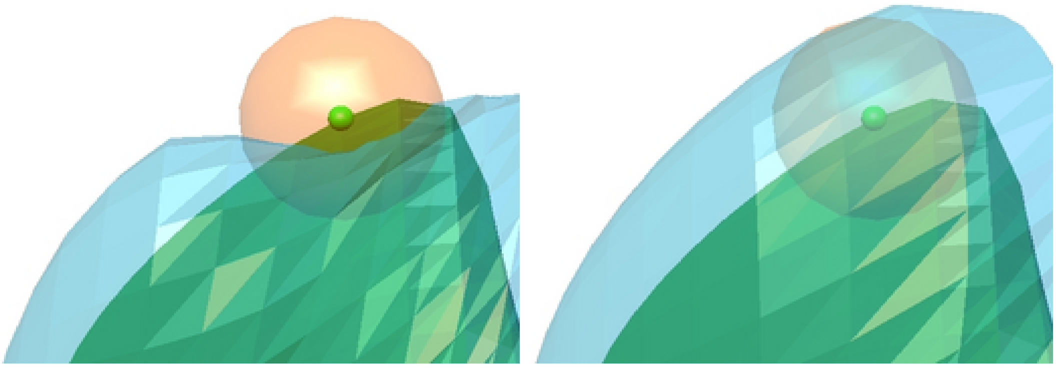


Figure 6.26: Preventing peeling skin defects. Self-intersection fairings via the mesh (Left) and graph (Right) Laplacians. The large sphere represents one of medial balls, and the small sphere indicates the corresponding medial ball center.

the shape preserving effect of L_0^{-1}. Moreover the evolution (6.15) prevents *peeling skin defects* because $\mathbf{s}^d$ plays a role of anchors during the evolutions, see Figure 6.26. This peeling skin defect is a common problem when we consider a mesh evolution on a surface where the evolving mesh may fall into a degenerated solution e.g. the spherical mesh parameterization scheme [GY02] may produce such a degenerated solution [GGS03, FSD05]. Compared with [ZHS$^+$05], the global self-intersections can be eliminated by our evolution (6.15), see Figure 6.27.

Every evolution step increases accuracy of the thickness preservation because of minimizing $f^2(\mathbf{x}^d)$. Therefore, the evolution (6.15) is useful even if there are no self-intersections in $\mathcal{M}_d$, but $f^2(\mathbf{x}^d) \neq 0$.

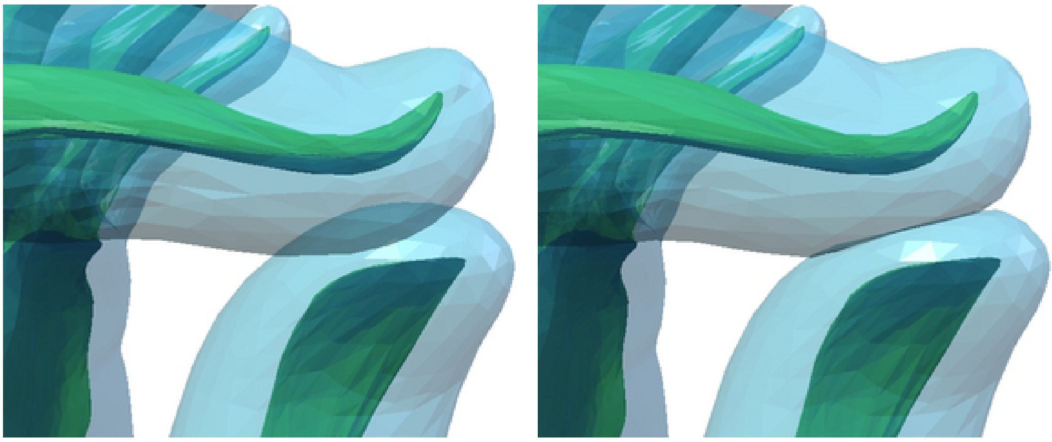


Figure 6.27: Global self-intersection fairing. Left: the deformed mesh with global self-intersections. Right: the fairing result via our evolutions (6.15).

Fast evaluation of f. Although L_0^{-1} is already computed for solving the system (6.11), evaluating the equation (6.13) could be time consuming. We accelerate the evaluation of f taking a union of a sub set of the medial balls instead of all medial balls attached to $\mathcal{S}_d$.

For an evolving graph vertex $\mathbf{u}_i^k = \{\mathbf{x}_i^k, \mathbf{s}_i^d\}$, a maximum bound of $f(\mathbf{x}_i^k)$ is given by $\|\mathbf{x}_i^k - \mathbf{s}_i^d| - d(i)|$. For every $\mathbf{u}_i^k$, consider the neighbor vertices $\mathbf{u}_j^k$ of $\mathbf{u}_i^k$ and its index set $j \in E(i)$ such that $|\mathbf{x}_j^k - \mathbf{x}_i^k| \leq d(i)$. Then the union of medial balls with its gradient is approximated by

$$f(\mathbf{x}_i^k)\nabla f(\mathbf{x}_i^k) \approx (\|\mathbf{x}_i^k - \mathbf{s}_p^d\| - d(p))(\mathbf{x}_i^k - \mathbf{s}_p^k)$$

where $p = \mathrm{argmin}_j(\|\mathbf{x}_i^k - \mathbf{s}_j^d| - d(j)|) : \forall j \in E(i)$. This vertex set $\mathbf{x}_j^k : j \in E(i)$ can be efficiently searched by using a kd-tree. We construct the kd-tree [MA06] for every evolution of (6.15) if the self-intersection fairing is necessity. Figure 6.28 describes an example of our variational skeleton-driven deformation framework.

6.5.2 Results of Variational Skeleton-driven Deformations

Our method is implemented by using the JDK 1.4 and Java3D. The UMFPACK compiled by the GNU gcc 3.3.5 is called by Java Native Interface. Figures 6.3, 6.4, 6.28, 6.30, 6.32, and 6.33 demonstrate how well the original shape thickness is preserved during our deformations.

	Ellipsoid	Camel	Homer	Gargoyle	Armadillo	Dragon
V	2.5K	37K	65K	160K	166K	263K
F	4.9K	75K	130K	320K	332K	527K
V_c		2.3K	4.1K	1.5K	2.5K	4.1K
F_c		4.7K	8.2K	5K	5.2K	8.2K
(a)		4.5s	1.6s	4.9s	25.9s	13.9s
(b)	4.5s	4.7s	6.5s	9.9s	5.1s	4.8s
(c)		2	2	3	3	3
(d)		2.8s	4.3s	26s	24.7s	19.7s
(e)		11s	8.8s	200s	161s	103s
(f)	0.08s	0.05s	0.09s	0.05s	0.05s	0.09s
(g)	0.06s	0.09s	0.14s	0.76s	0.13s	0.54s
(h)	0.05s	0.11s	0.06s	0.77s	0.13s	0.32s
(i)	0.16s	0.13s	0.29s	0.79s	0.12s	0.44s
(j)		0.61s	3.86s	4.36s	3.8s	43.8s
(k)	6.3	6.30			6.4	6.30

Table 6.1: Timings. (a): Decimation for DSS. (b): Skeletal Mesh Extraction. (c): Subdivision Level. (d): Subdivision for DSS. (e): Displacement Sampling for DSS. (f): Factorization of L_0. (g): Skeletal Mesh Editing. (h): Transformation (6.8) and Solving (6.11). (i): Self-Intersection Fairing (6.15) (1 step). (j): DSS Reconstruction. (k): Corresponding Figure Numbers.

Timing. Table 6.1 represents timings which are measured on a 1.7 GHz Pentium 4 with 1 GB RAM computer. Here V and F are the vertex and triangle numbers of $\mathcal{M}$, and V_c and F_c are the vertex and triangle numbers of the DSS's coarse control mesh, respectively. The interactive mesh deformations are achieved for the control meshes, see (g) and (h) of Table 6.1.

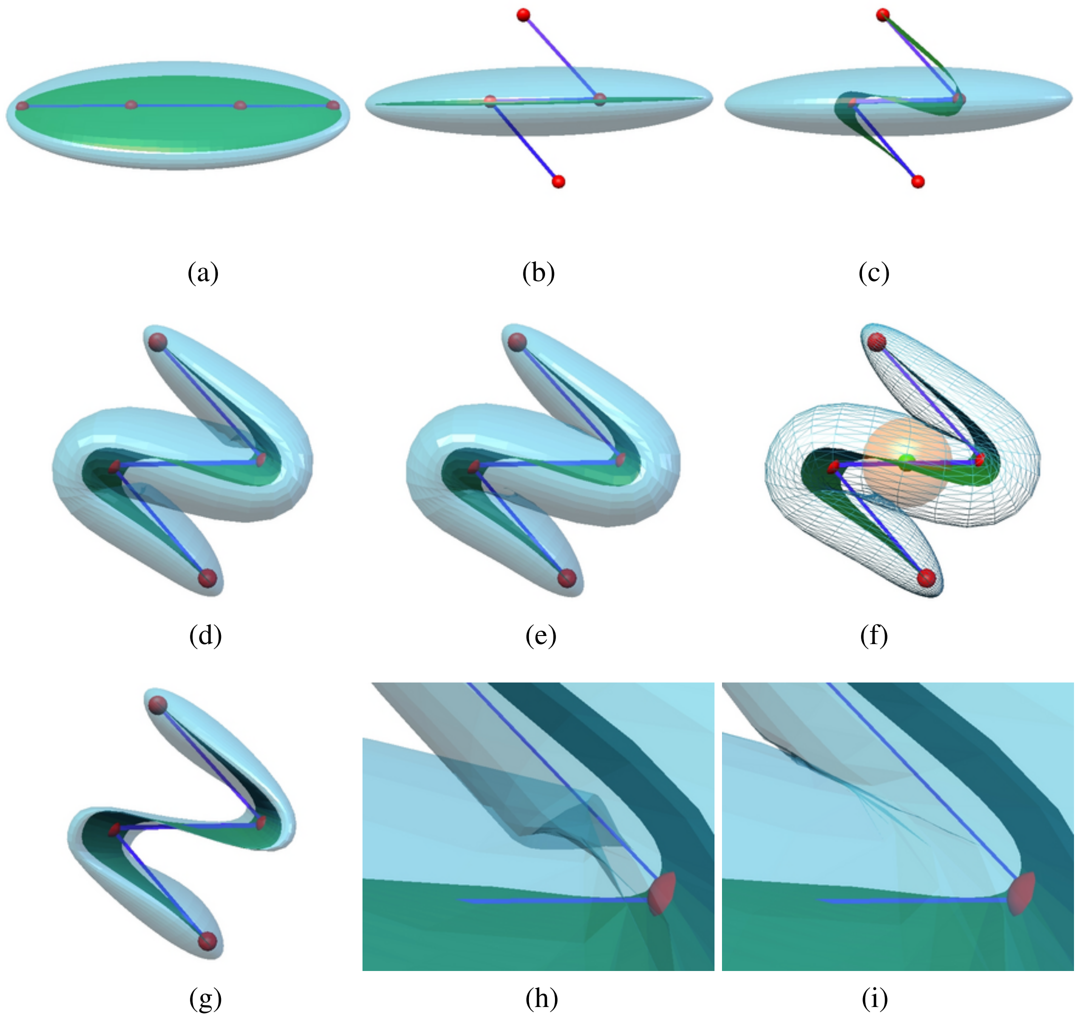


Figure 6.28: An example of variational skeleton-driven deformation framework. (a): Ellipsoid, its skeletal mesh, and control stick-figure skeletons. (c): The skeletal mesh is edited by using a space deformation [MTLT88,LCJ94] according to change of the stick-figure skeletons described in (b). (d): Our deformation result via (6.11). (e): Our evolution result via (6.15). The resulting evolution eliminates self-intersections but also restores the original shape thickness as shown in (f). Here a medial ball is represented by the large sphere, and the small sphere corresponds its center on the skeletal mesh. See also (h) and (i) which are zoom images of (d) and (e), respectively. (g): A deformation result via only using a space deformation [MTLT88,LCJ94] with the stick-figure skeletons of (b).

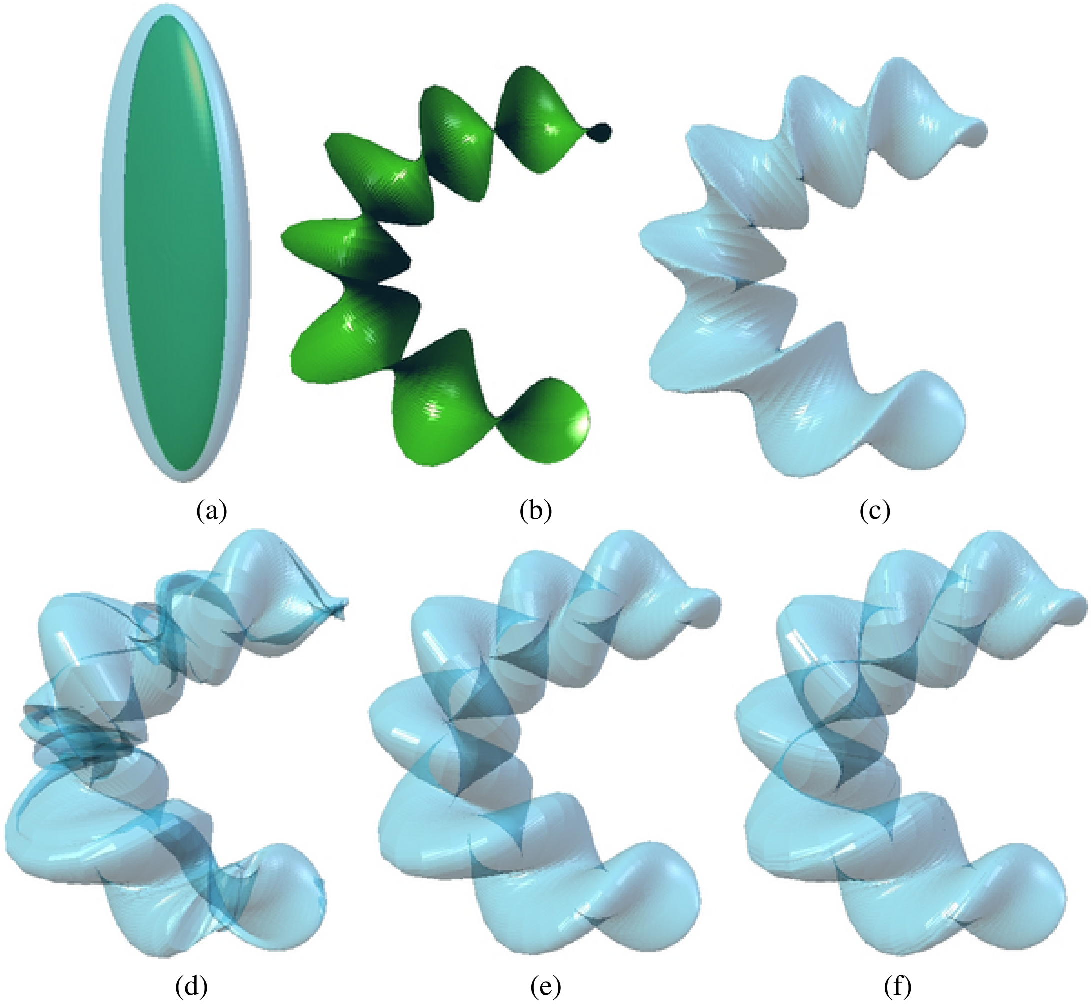


Figure 6.29: Comparison with conventional skeleton-driven deformations. Here f is given by the equation (6.13), and summation $\sum$ is carried out over all mesh vertices. (a): Ellipsoid and its skeletal mesh, V: 9.8K, F: 19.6K. (b): Edited skeletal mesh. (c): SSD [MTLT88,LCJ94], time: 2.3s, $\sum f^2$: 20.2. (d): Homotopy method [YBS03] (Section 6.3) without DSS, time: 2.7s, $\sum f^2$: 19.6. (e): Weighted blending [Blo02], time: 14.8s, $\sum f^2$: 11.8. (f): Our variational skeleton-driven deformation method without DSS, time 2.9s, $\sum f^2$: 3.8. By using our evolution (6.15) $\sum f^2$ becomes 0.2.

The multiresolution mesh representation described in Section 6.4 significantly improves the computational time of our approach. For example, deforming Armadillo (V:173K, F:346K) without the DSS requires the following timings: (b):356s, (f):321s, (g):94s, (h):423s, and (i):279s where alphabets are explained in the caption of Table 6.1.

Comparison. We have implemented the conventional skeleton-driven deformations: weighted blending [Blo02] and homotopy method [YBS03] (Section 6.3) to compare with our approach. Our implementation of weighted blending [Blo02] is summing up the local frame changes within its medial ball radius instead of all local frames. The homotopy method suffers from fold-overs especially twisting, see Figure 6.29. Our approach outperforms [Blo02, YBS03] in terms of the both speed and quality.

Besides the conventional deformation methods based on discrete differential coordinates which include the most recent ones [LSCOL05, ZRKS05, ZHS+05] do not have the skinning ability. The original shape thickness is not preserved during their deformations. We believe that the shape thickness is a natural and intuitive measurement than local volumes [BK03b, ZHS+05, BPGK06] or global volume [ACWK04, vFTS06] because volume-preserving shape deformations include large shearing effects in $\mathfrak{R}^3$ which may not produce natural-looking mesh deformations as indicated in [SCOL+04].

Discussion. Our approach depends on the skeletal mesh $\mathcal{S}$. Consequently, if a part of $\mathcal{S}$ is degenerated to a space curve or a point (e.g. when $\mathcal{M}$ belongs to Dupin's cyclides) then we may need another definition of local frames B_0 and B_d for the degenerated part. A possible solution would be that a frame axis defined from a degenerated point to a neighbor skeletal vertex position is employed to construct the frame.

The graph-Laplacian L_0 defined by the equation (6.10) does not contain the angle information $\angle \mathbf{s}_i \mathbf{x}_i \mathbf{x}_j$ or $\angle \mathbf{s}_i \mathbf{x}_i \mathbf{x}_k$. Further work is required for deeper understanding of discretization effects of L_0.

Our self-intersection fairing (6.15) does not work appropriately if $\mathcal{M}_d$ intersects $\mathcal{S}_d$ because $\nabla F(U)$ may be oriented to an opposite direction of our desired gradient direction in such cases. Adding smoothing components to the fairing (6.15) as the self-intersection fairing described in Section 6.3.2 would help such cases.

6.6 Summary of Free-Form Skeleton-driven Deformations

We have developed new approaches to free-from skeleton-based mesh deformations. Using the medial axis as the base of our technique allows us to generate natural-looking large-scale mesh deformations by preserving the original shape thickness. The main features of our approaches are using Voronoi-based skeletal mesh, applying mesh evolutions for skeletal mesh fairing, and combining skeleton-based mesh deformations with the discrete differential coordinates and the DSS mesh representation. All this makes it possible to produce global mesh deformations of satisfactory quality.

As demonstrated, our skeleton-based approach works well for deforming objects composed of elongated parts. The approach has limitations in processing objects of spherical-like shapes because their skeletal meshes are usually very complex. Another limitation of our method consists in deforming models with sharp edges and corners since the mesh evolutions (6.5) and (6.6) may destroy the mesh sharp features. However the shape preserving mesh evolutions (6.14) provide us satisfactory results as demonstrated in the previous sections.

We have successfully achieved to integrate advantages of skeleton-driven deformations and discrete differential coordinates. Preserving the original shape thickness and fine geometric details allows us to generate natural-looking complex mesh deformations.

One of promising future work would be applying discrete differential coordinates to the skeletal mesh editing phase by using a single-sided skeletal mesh.

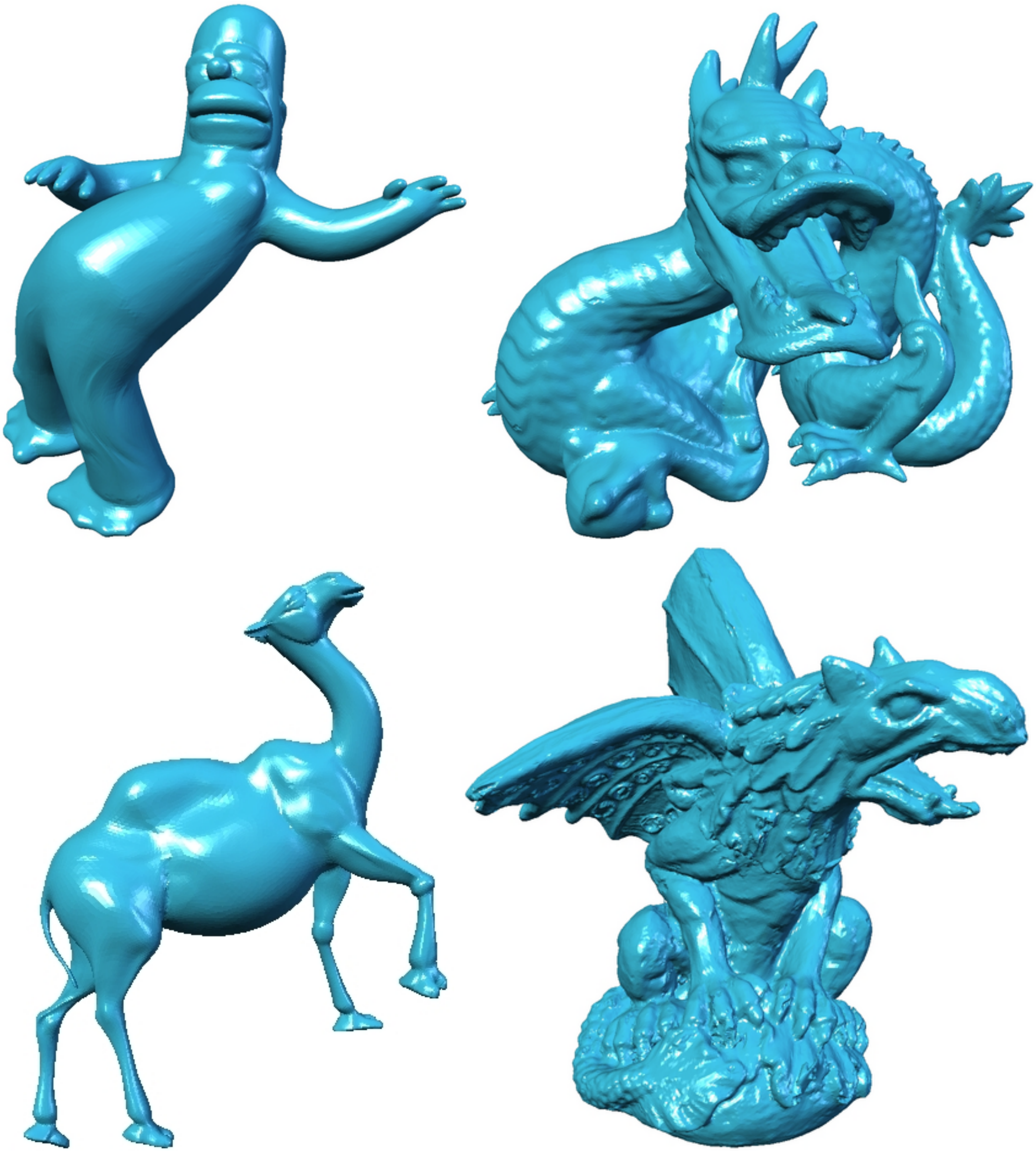


Figure 6.30: Deformation examples. The initial models are represented in Figure 6.31.

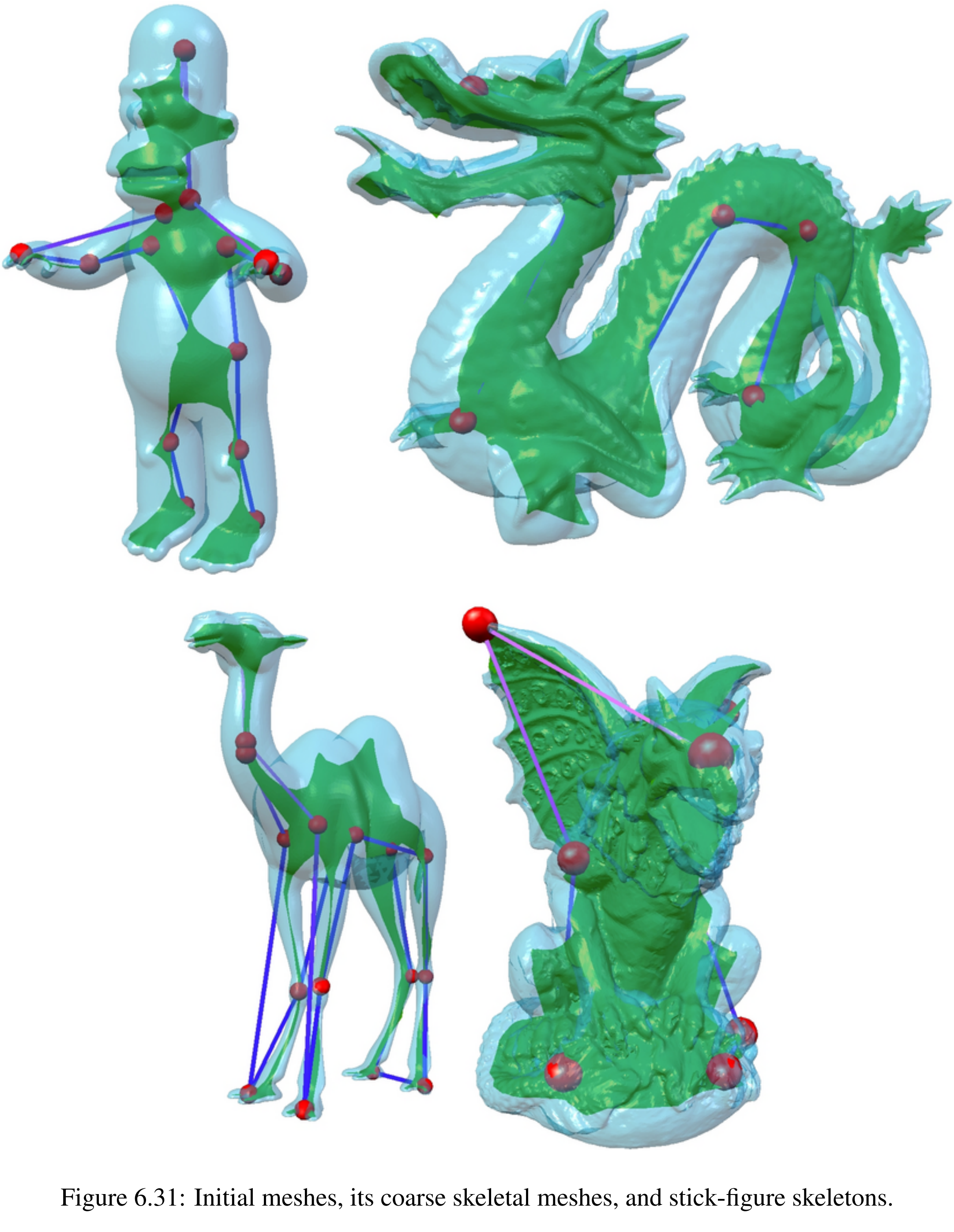


Figure 6.31: Initial meshes, its coarse skeletal meshes, and stick-figure skeletons.

Figure 6.32: Armadillo's gymnastics. Initial mesh, its coarse skeletal mesh, and the stick-figure skeleton are represented in the image (a) of Figure 6.4.

Figure 6.33: Thickness visualization of our deformations. Top-left: original Stanford Armadillo model consisting of 332K triangles. Top-right: the control mesh approximates the medial axis with its associated distance field. Middle/Bottom-left: different large-scale deformations generated using our method. Middle/Bottom-right: modified control meshes with their associated distance fields. Generating these deformations is rather fast: it takes only 0.3 seconds for deforming the coarse control mesh (5.2K triangles) and 3.8 seconds for a multiresolutional reconstruction of the deformed dense mesh.

7

Conclusion

In this thesis, new approaches for surface interrogating, fairing, and designing are developed. The approaches are based on computational differential geometry. They are first designed for processing smooth continuous surfaces and then adapted for dealing with triangulated polygonal surfaces. This our strategy to start from differential geometry concepts and then develop and use their proper discrete analogs turned out to be quite successful and led us to the following contributions in the geometric modeling area.

A new and powerful mesh/soup denoising technique is described in Chapter 2. Our technique is based on similarity-weighted averaging, therefore, high quality denoising results are achieved by preserving features (local shape patterns). A new scheme for comparing different mesh/soup denoising methods is also suggested.

In Chapter 3, a novel mesh fairing and restoration scheme is presented. Our scheme is build upon a discrete approximation of Willmore flow. A tangent speed component is introduced to the discrete Willmore flow in order to improve the quality of the evolving mesh and to increase computational stability.

A new technique for fast and robust detection of salient curvature extrema on meshes is proposed in Chapter 4. Our technique consists of a novel local polynomial fitting procedure, a new curvature derivatives formula, and a smart thresholding scheme. Applications to feature-sensitive mesh simplification and partition problems are also demonstrated.

In Chapter 5, a powerful moving mesh approach is described to a mesh parameterization problem. Our approach equalizes local stretches over a mesh by solving a few sparse systems of linear equations. Consequently, our method is significantly faster than the conventional method and capable of achieving high quality mesh parameterizations. Moreover the generated mesh parameterizations do not have both high anisotropic distortions and triangle flips. Application to a remeshing problem is also considered by using a new double parameterization scheme.

A powerful approach for feature-preserving free-form shape deformations is proposed in Chapter 6. Our approach can generate natural-looking large-scale deformations by preserving original shape thickness. Mesh fairing procedures for removing possible global and local self-intersections are also developed. Finally, we combine our skeleton-driven deformation method with the variational approach and multiresolution representation.

In spite of good performance of the developed methods and approaches, there are many directions for their further improvements. The following future work directions look promising.

For mesh denoising, introducing a new similarity measurement by using Fourier transform of the meshes would be an interesting and challenging topic for future research.

Incorporating multi-scale and multiresolution analysis for our crest line detection procedure may decrease fragmentation of detected feature lines.

In this thesis, we considered planar mesh parameterizations. Extending our moving mesh approach to the conformal gradient field of [GY03, TACSD06] may contribute to generate low-stretch non-planar mesh parameterizations.

Applying discrete differential coordinates for editing skeletal meshes may improve our skeleton-driven deformation framework.

Bibliography

[ABE06] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. Stability and computation of medial axes — a state-of-the-art report. In B. Hamann T. Moeller and B. Russell, editors, *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Springer-Verlag, 2006.

[ABK94] E. V. Anoshkina, A. Belyaev, and T. L. Kunii. Detection of ridges and ravines based on caustic singularities. *Int. J. of Shape Modeling*, 1(1):13–22, 1994.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of ACM SIGGRAPH*, pages 415–421, 1998.

[ABOK94] E. V. Anoshkina, A. Belyaev, O. G. Okunev, and T. L. Kunii. Ridges and ravines: A singularity approach. *Int. J. of Shape Modeling*, 1(1):1–11, 1994.

[ACK01a] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *6th ACM Symposium on Solid Modeling*, pages 249–260, 2001.

[ACK01b] N. Amenta, S. Choi, and R. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2-3):127–153, 2001.

[ACSYD05] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):362–371, July 2005. Proceedings of ACM SIGGRAPH.

[ACWK04] A. Angelidis, M.-P. Cani, G. Wyvill, and S. King. Swirling-sweepers: Constant-volume modeling. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 10–15, Washington, DC, USA, 2004. IEEE Computer Society.

[Ahl66] L. V. Ahlfors. *Lectures on Quasiconformal Mappings*. Van Nostrand-Reinhol, Princeton, New Jersey, 1966.

[AHTK99] S. Angenent, S. Haker, A. Tannenbaum, and R. Kikinis. On area preserving mappings of minimal distortion. In T. Djaferis and I. Schick, editors, *System Theory: Modeling, Analysis, and Control, Holland: Kluwer*, pages 275–287, 1999.

[Ale00] M. Alexa. As-rigid-as-possible shape interpolation. *Proceedings of ACM SIGGRAPH*, pages 157–164, 2000.

[Ale02] M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.

[AMD02] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. In *Proceedings of ACM SIGGRAPH*, pages 347–354, 2002.

[AUGA05] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. Technical report, AIM@SHAPE Network of Excellence, 2005.

[AW95] V. Aurich and J. Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995, 17. DAGM-Symposium*, pages 538–545, London, UK, 1995. Springer-Verlag.

[AWC04] A. Angelidis, G. Wyvill, and M.-P. Cani. Sweepers: Swept user-defined tools for modeling by deformation. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 63–73, 2004.

[BAK97] A. Belyaev, E. V. Anoshkina, and T. L. Kunii. Ridges, ravines, and singularities. In A. T. Fomenko, and T. L. Kunii, *Topological Modeling for Visualization*, pages 375–383. Springer, 1997. Ch. 18.

[Bar84] A. H. Barr. Global and local deformations of solid primitives. *Proceedings of ACM SIGGRAPH*, pages 21–130, 1984.

[BBGS99] R. Blanding, C. Brooking, M. Ganter, and D. Storti. A skeletal-based solid editor. In *Proceedings of 5th ACM Symposium on Solid Modeling and Applications*, pages 141–150, New York, NY, USA, 1999. ACM Press.

[BCD+96] J.-D. Boissonnat, A. Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d. *Comp. Geom. Theory Appl.*, 6:123–130, 1996.

[BCM05a] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR'05)*, pages 60–65, 2005.

[BCM05b] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation (SIAM interdisciplinary journal)*, 4(2):490–530, 2005.

[BCM06] A. Buades, B. Coll, and J. M. Morel. Neighborhood filters and PDE's. *Numerische Mathematik*, 2006.

[BDH96] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[BHN96] A. M. Bruckstein, R. J. Holt, and A. N. Netravali. Discrete elastica. In *Discrete Geometry for Computer Imagery, Lecture Notes in Computer Science 1176*, pages 59–72, Lyon, France, November 1996.

[BIT04] P. Bhat, S. Ingram, and G. Turk. Geometric texture synthesis by example. In *Second Eurographics Symposium on Geometry Processing*, 2004.

[BK03a] G. H. Bendels and R. Klein. Mesh forging: editing of 3d-meshes using implicitly defined occluders. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 207–217, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[BK03b] M. Botsch and L. Kobbelt. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum*, 22(3):483–491, 2003. Eurographics 2003 issue.

[BL99] J. Bloomenthal and C. Lim. Skeletal methods of shape manipulation. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 44–47, 1999.

[BLBK03] I. A. Bogaevski, V. Lang, A. Belyaev, and T. L. Kunii. Color ridges on implicit polynomial surfaces. In *GraphiCon 2003 Proceedings*, pages 161–164, September 2003.

[Blo94] J. Bloomenthal. An implicit surface polygonizer. In *Graphics Gems IV*, pages 324–349. Academic Press Professional, Inc., 1994.

[Blo02] J. Bloomenthal. Skinning: Medial-based vertex deformation. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 147–151, 2002.

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Symposium on Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

[BM05] E. P. Bennett and L. McMillan. Video enhancement using per-pixel virtual exposures. *ACM Transactions on Graphics*, 24(3):845–852, 2005. Proceedings of ACM SIGGRAPH.

[BN92] R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: I. *Computers Math. Applic.*, 24(12):7–19, 1992.

[BO01] A. Belyaev and Y. Ohtake. Nonlinear diffusion of normals for crease enhancement. In *Proceedings of Vision Geometry X, SPIE Annual Meeting*, pages 42–47, 2001.

[BPGK06] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 11–20, 2006.

[BPK98] A. Belyaev, A. A. Pasko, and T. L. Kunii. Ridges and ravines on implicit surfaces. In *Proceedings of Computer Graphics International 1998*, pages 530–535, 1998.

[BR78] I. Babuška and W. C. Rheiboldt. A posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Eng.*, 12:1597–1615, 1978.

[BR94] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, 1994.

[Bra92] K. A. Brakke. The Surface Evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[BS82] J. U. Brackbill and J. S. Saltzman. Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys.*, 46:342–368, 1982.

[BS05] A. I. Bobenko and P. Schröder. Discrete willmore flow. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 101–110, 2005.

[BTB02] L. Balmelli, G. Taubin, and F. Bernardini. Space-optimized texture maps. *Computer Graphics Forum*, 21(3):411–420, 2002. Eurographics 2002 issue.

[BY01] A. Belyaev and S. Yoshizawa. On evolute cusps and skeleton bifurcations. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 134–141, 2001.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of SIGGRAPH*, pages 67–76, New York, NY, USA, 2001. ACM Press.

[CCM97] H. I. Choi, S. W. Choi, and H. P. Moon. Mathematical theory of medial axis transform. *Pacific Journal of Mathematics*, 181(1):57–88, 1997.

[CD85] G. F. Carey and H. T. Dinh. Grading functions and mesh redistribution. *SIAM J. Numer. Anal.*, 22(5):1028–1040, 1985.

[CDD+04] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu. A finite element method for surface restoration with smooth boundary conditions. *Comput. Aided Geom. Des.*, 21(5):427–445, 2004.

[CDH89] V. Chandru, D. Dutta, and C. M. Hoffmann. On the geometry of Dupin cyclides. *The Visual Computer*, 5(5):277–290, October 1989.

[CGC+02] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. In *Proceedings of ACM SIGGRAPH*, pages 586–593, New York, NY, USA, 2002. ACM Press.

[CHCH06] N. A. Carr, J. Hoberock, K. Crane, and J. C. Hart. Rectangular multi-chart geometry images. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 181–190, 2006.

[CHR03] W. Cao, W. Huang, and R. D. Russell. Approaches for generating moving adaptive meshes: location versus velocity. *Appl. Numer. Math.*, 47(2):121–138, 2003.

[Chu97] F. R. K. Chung. *Spectral graph theory*. American Mathematical Society, 1997. Regional Conference Series in Mathematics, number 92.

[CKPF05] G. Charpiat, R. Keriven, J.-P. Pons, and O. Faugeras. Designing spatially coherent minimizing flows for variational problems based on active contours. In *ICCV '05: Proceedings of the Tenth IEEE International Conferenc e on Computer Vision*, pages 1403–1408, Washington, DC, USA, 2005. IEEE Computer Society.

[Coq90] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proceedings of ACM SIGGRAPH*, pages 187–196, New York, NY, USA, 1990. ACM Press.

[CP03] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 177–187, Aachen, Germany, June 2003.

[CP04a] F. Cazals and M. Pouget. Ridges and umbilics of a sampled smooth surface: a complete picture gearing toward topological coherence. Rapport de Recherche RR-5294, INRIA, September 2004.

[CP04b] F. Cazals and M. Pouget. Smooth surfaces, umbilics, lines of curvatures, foliations, ridges and the medial axis: a concise overview. Rapport de Recherche RR-5138, INRIA, March 2004.

[CP05] F. Cazals and M. Pouget. Differential topology and geometry of smooth embedded surfaces: selected topics. *Int. J. of Computational Geometry and Applications*, 15(5):511–536, 2005.

[CSAD04] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, 2004.

[CT03] P. Choudhury and J. Tumblin. The trilateral filter for high contrast images and meshes. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 186–196, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Dav04] T. A. Davis. Umfpack - an unsymmetric-pattern multifrontal method with a column preordering strategy. *ACM Transactions on Mathematical Software*, 30(2):196–199, 2004.

[DB73] C. De Boor. Good approximation by splines with variable knots ii. In *Conference on the Numerical Solution of Differential Equations, Lecture Notes in Mathematics. No. 363*, pages 12–20, 1973.

[DFRS03] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855, 2003. Proceedings of ACM SIGGRAPH.

[DG03] T. K. Dey and S. Goswami. Tight cocone: a water-tight surface reconstructor. In *8th ACM Symposium on Solid Modeling and Applications*, pages 127–134, 2003.

[DMA02] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002. Eurographics 2002 issue.

[DMK03] P. Degener, J. Meseth, and R. Klein. An adaptable surface parameterization method. In *Proceedings of 12th International Meshing Roundtable*, pages 201–213, 2003.

[DMSB99] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of ACM SIGGRAPH*, pages 317–324, 1999.

[Dur04] P. Duren. *Harmonic Mappings in The Plane*. Cambridge University Press, Edinburgh, Cambridge, UK, 2004. Series: Cambridge Tracts in Mathematics (No. 156).

[DZ02] T. K. Dey and W. Zhao. Approximate medial axis as a voronoi subcomplex. In *Proceedings of 7th ACM Symposium on Solid Modeling and Applications*, pages 356–366, Saarbrüken, Germany, June 2002.

[Ebe96] D. H. Eberly. *Ridges in Image and Data Analysis*. Kluwer, 1996.

[EDD+95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzl. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 173–182, 1995.

[EL99] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision (ICCV'99)*, pages 1033–1038, 1999.

[Eul44] L. Euler. Additamentum 'de curvis elasticis'. In *Methodus Inveniendi Lineas Curvas Maximi Minimive Probprietate Gaudentes*, Lausanne, 1744.

[Far02] G. Farin. A history of curves and surfaces in cagd. In G. Farin, J. Hoschek, and M.-S. Kim, editors, *Handbook of Computer Aided Geometric Design*, pages 1–22. Elsevier Science, 2002.

[FCOS05] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24(3):544–552, 2005. Proceedings of ACM SIGGRAPH.

[FDCO03] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics*, 22(3):950–953, 2003. Proceedings of ACM SIGGRAPH.

[FG04] S. Foufou and L. Garnier. Dupin cyclide blends between quadric surfaces for shape modeling. *Computer Graphics Forum*, 23(3):321–330, 2004. Eurographics 2004 issue.

[FH02] M. S. Floater and K. Hormann. Parameterization of triangulations and unorganized points. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, Mathematics and Visualization, pages 287–316. Springer, Berlin, Heidelberg, 2002.

[FH04] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Multiresolution in Geometric Modelling*, pages 157–186, 2004.

[Flo97] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[Flo03] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[FSD99] B. Fischl, M. I. Sereno, and A. M. Dale. Cortical surface-based analysis. ii: Inflation, flattening, and a surface-based coordinate system. *Neuroimage*, 9(2):195–207, 1999.

[FSD05] I. Friedel, P. Schröder, and M. Desbrun. Unconstrained spherical parameterization. In *ACM SIGGRAPH Technical Sketch*, 2005.

[GCO05] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. In *Under revision for ACM TOG*, 2005.

[GGGZ05] T. Gatzke, C. Grimm, M. Garland, and S. Zelinka. Curvature maps for local shape comparison. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 244–256, 2005.

[GGH02] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *Proceedings of ACM SIGGRAPH*, pages 355–361, 2002.

[GGS03] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3D meshes. *ACM Transactions on Graphics*, 22(3):358–363, 2003. Proceedings of ACM SIGGRAPH.

[GGT06] S. J. Gortler, C. Gotsman, and D. Thurston. Discrete one-forms on meshes and applications to 3d mesh parameterization. *Computer Aided Geometric Design*, 33(2):83–112, 2006.

[GH96] M. Giaquinta and S. Hildebrandt. *Calculus of Variations I*. Springer-Verlag Berlin Heidelberg, 1996.

[GH97] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH*, pages 209–216, 1997.

[GI04] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1):45–63, 2004.

[GM98] U. Grenader and M. I. Miller. Computational anatomy: An emerging discipline. *Quarterly of Applied Mathematics*, 56(4):617–694, 1998.

[GPA97] A. P. Guéziec, X. Pennec, and N. Ayache. Medical image registration using geometric hashing. *IEEE Comput. Sci. Eng.*, 4(4):29–41, 1997.

[Gre94] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13:143–154, 1994.

[GSZ05] G. Gilboa, N. Sochen, and Y. Y. Zeevi. Estimation of the optimal variational parameter via SNR analysis. In *Scale-Space 2005, LNCS 3459*, pages 230–241, 2005.

[Gué93] A. P. Guéziec. Large deformable splines, crest lines and matching. In *Proceedings of IEEE Fourth Int'l Conf. Computer Vision*, pages 650–657, 1993.

[Gul04] A. Gullstrand. Zur kenntnis der kreispunkte. *Acta Mathematica*, 29:59–100, 1904.

[Gul11] A. Gullstrand. How i found the mechanism of intracapsular accommodation. *Nobel Lecture, http://nobelprize.org/medicine/laureates/1911/gullstrand-lecture.pdf*, pages 414–431, December 1911.

[Gus02] I. Guskov. An anisotropic mesh parameterization scheme. In *Proceedings of 11th International Meshing Roundtable*, pages 325–332, 2002.

[GWH01] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of ACM Symp. on Interactive 3D Graphics*, pages 49–58, 2001.

[GWY03] X. Gu, Y. Wang, and S.-T. Yau. Geometric compression using riemann surface structure. *Communications in Information and Systems*, 3(3):171–182, 2003.

[GY02] X. Gu and S.-T. Yau. Computing conformal structures of surfaces. *Communications in Information and Systems*, 2:121–146, 2002.

[GY03] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of Eurographics Symposium on Geometry Processing 2003*, pages 135–146, 2003.

[HAT+00] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

[HBK02] M. Hisada, A. Belyaev, and T. L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.

[HDW98] J. Hoschek, U. Dietz, and W. Wilke. A geometric concept of reverse engineering of shape: Approximation and feature lines. In M. Dæhlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*, pages 253–262. Vanderbilt Univ. Press, 1998.

[HG99] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 1999.

[HGC99] K. Hormann, G. Greiner, and S. Campagna. Hierarchical parametrization of triangulated surfaces. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *Proceedings of Vision, Modeling, and Visualization 1999*, pages 219–226, Erlangen, Germany, nov. 1999. infix.

[HGY+99] P. L. Hallinan, G. G. Gordon, A. L. Yuille, P. Giblin, and D. Mumford. *Two- and Three-Dimensional Patterns of the Face*. A K Peters, 1999.

[HH92] H. Hagen and S. Hahmann. Surface interrogation algorithms. In *Proceedings of IEEE Visualization*, pages 70–76, 1992.

[HH03] J. Hermansson and P. Hansbo. A variable diffusion method for mesh smoothing. *Communications in Numerical Methods in Engineering*, 19:897–908, 2003.

[HHK92] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Proceedings of ACM SIGGRAPH*, pages 177–184, New York, NY, USA, 1992. ACM Press.

[HHS+92] H. Hagen, S. Hahmann, T. Schreiber, Y. Nakajima, B. Wördenweber, and H. Hollemann-Grundstedt. Surface interrogation algorithms. *IEEE Computer Graphics and Applications*, 12(5):53–60, 1992.

[HHS95] H. Hagen, S. Hahmann, and T. Schreiber. Visualization and computation of curvature behaviour of free-form curves and surfaces. *Computer-Aided Design*, 27(7):545–552, 1995.

[Hir90] A. E. Hirst. Blending cones and planes by using cyclides of dupin. *Bull. Inst. Math. Appl.*, 26(3):41–46, 1990.

[HKS92] L. Hsu, R. Kusner, and J. Sullivan. Minimizing the Squared Mean Curvature Integral for Surfaces in Space Forms. *Experimental Mathematics*, 1(3):191–207, 1992.

[Hos92] M. Hosaka. *Modeling of curves and surfaces in CAD/CAM*. Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[HP04] K. Hildebrandt and K. Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23(3):391–400, 2004. Eurographics 2004 issue.

[HPW05] K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 85–90, 2005.

[HR85] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985.

[HSL+06] J. Huang, X. Shi, X. Liu, K. Zhou, L. Wei, S. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics*, 25(3):1126–1134, 2006. Proceedings of ACM SIGGRAPH.

[Hua01] W. Huang. Variational mesh adaptation: Isotropy and equidistribution. *J. Comput. Phys.*, 174:903–924, 2001.

[IFP95] V. Interrante, H. Fuchs, and S. Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 52, Washington, DC, USA, 1995. IEEE Computer Society.

[JDD03] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3):943–949, 2003. Proceedings of ACM SIGGRAPH.

[JKS05] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum*, 24(3):981–990, 2005. Eurographics 2005 issue.

[JSW05] T. J., S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, 2005. Proceedings of ACM SIGGRAPH.

[Kan04] T. Kanai. Hierarchical computation of conformal spherical embeddings. In *6th International Conference on Mathematical Methods for Curves and Surfaces (Tromso, Norway, 1-6 July)*, 2004.

[KBS00] L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3):249–260, 2000.

[KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 105–114, 1998.

[KG03] Y. Kho and M. Garland. User-guided simplification. In *ACM Symposium on Interactive 3D Graphics*, pages 123–126, Washington, D.C., USA, 2003.

[KGG05] Z. Karni, C. Gotsman, and S. J. Gortler. Free-boundary linear parameterization of 3d meshes in the presence of constraints. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 266–275, 2005.

[KK05] S.-K. Kim and C.-H. Kim. Finding ridges and valleys in a discrete surface using a modified mls approximation. *Computer-Aided Design*, 37(14):1533–1542, 2005.

[KLML96] J. T. Kent, D. Lee, K. V. Mardia, and A. D. Linney. Using curvature information in shape analysis. In K. V. Mardia, G. A. Gill, and I. L. Dryden, editors, *Proceedings of Image Fusion and Shape Variability Techniques*, pages 88–99. Leeds University Press, 1996.

[KMW96] J. T. Kent, K. V. Mardia, and J. M. West. Ridge curves and shape analysis. In *The British Machine Vision Conference 1996*, pages 43–52, 1996.

[KO03] K. G. Kobayashi and K. Ootsubo. t-ffd: Free-form deformation by using triangular mesh. In *Proceedings of 8th ACM Symposium on Solid Modeling and Applications*, pages 226–234, 2003.

[Koe90] J. J. Koenderink. *Solid Shape*. MIT Press, 1990.

[KOJ05] S. Kindermann, S. Osher, and P. W. Jones. Deblurring and denoising of images by non-local functionals. *Multiscale Modeling & Simulation (SIAM interdisciplinary journal)*, 4(4):1091–1115, 2005.

[KSS06] L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics*, 25(2):412–438, 2006.

[Lat91] J.-C. Latombe. *Robot motion planning*. Kluwer Academic, 1991.

[LCF00] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH*, pages 165–172, 2000.

[LCJ94] F. Lazarus, S. Coquillart, and P. Jancéne. Axial deformations: an intuitive deformation technique. *Computer-Aided Design*, 26(8):607–613, 1994.

[Ley87] M. Leyton. Symmetry-curvature duality. *Comput. Vision Graph. Image Process.*, 38(3):327–341, 1987.

[LFM96] R. Lengagne, P. Fua, and O. Monga. Using crest lines to guide surface reconstruction from stereo. In *ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I*, page 9, Washington, DC, USA, 1996. IEEE Computer Society.

[Lie03] P. Liepa. Filling holes in meshes. In *Proceedings of Eurographics Symposium on Geometry Processing 2003*, pages 200–205, 2003.

[Lis04] V. D. Liseikin. *A Computational Differential Geometry Approach to Grid Generation*. Springer, 2004.

[LKL02] Y. Lee, H. S. Kim, and S. Lee. Mesh parameterization with a virtual boundary. *Computers and Graphics*, 26(5):677–686, 2002.

[LMH00] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proceedings of ACM SIGGRAPH*, pages 85–94, 2000.

[LP05] C. Lange and K. Polthier. Anisotropic fairing of point sets. *Special Issue of Computer Aided Geometric Design.*, 22(7):680–692, 2005.

[LPRM02] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, July 2002. Proceedings of ACM SIGGRAPH.

[LS01] J. J. Little and P. Shi. Structural lines, TINs and DEMs. *Algorithmica*, 30(2):243–263, 2001.

[LSCO+04] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 181–190, 2004.

[LSCOL05] Y. Lipman, O. Sorkine, D. Cohen-Or, and D. Levin. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics*, 24(3):479–487, 2005. Proceedings of ACM SIGGRAPH.

[LTZ01] R. Li, T. Tang, and P. Zhang. Moving mesh methods in multiple dimensions based on harmonic maps. *J. Comput. Phys.*, 170(2):562–588, 2001.

[MA06] D. M. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching. In *www.cs.umd.edu/~mount/ANN*, 2006.

[MAM97] O. Monga, N. Armande, and P. Montesinos. Thin nets and crest lines: Application to satellite data and medical images. *Computer Vision and Image Understanding: CVIU*, 67(3):285–295, 1997.

[MAVdF05] B. Mederos, N. Amenta, L. Vehlo, and L. H. de Figueiredo. Surface reconstruction from noisy point clouds. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 53–62, 2005.

[Max99] N. Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.

[MB95] O. Monga and S. Benayoun. Using partial derivatives of 3D images to extract typical surface features. *Computer Vision and Image Understanding: CVIU*, 61:171–195, 1995.

[MBF92] O. Monga, S. Benayoun, and O. D. Faugeras. From partial derivatives of 3-d density images to ridge lines. In *Computer Vision and Pattern Recognition CVPR'92*, pages 354 – 359, June 1992.

[MDSB02] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *VisMath,*, 2002.

[MJ96] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of ACM SIGGRAPH*, pages 181–188, New York, NY, USA, 1996. ACM Press.

[MN03] P. Mrázek and M. Navara. Selection of optimal stopping time for nonlinear diffusion filtering. *International Journal of Computer Vision*, 52(2/3):189–203, 2003.

[Mor96] R. Morris. The sub-parabolic lines of a surface. In G. Mullineux, editor, *Mathematics of Surfaces VI*, IMA new series 58, pages 253–262. Clarendon Press, 1996.

[MS92] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 167–176, August 1992.

[MS05] M. Mahmoudi and G. Sapiro. Fast image and video denoising via nonlocal means of similar neighborhoods. *Signal Processing Letters*, 12(12):839–842, 2005.

[MTLT88] N. Magnenat-Thalmann, R. Laperriére, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings of Graphics Interface*, pages 26–23, 1988.

[MYV93] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of ACM SIGGRAPH*, pages 27–34, 1993.

[Neu83] J. W. Neuberger. *Steepest Descent for General Systems of Linear Differential Equation s in Hilbert Space*. Springer, 1983.

[Neu97] J. W. Neuberger. *Sobolev Gradients and Differential Equations*. Springer-Verlag, 1997.

[NNS06] J. Novatnack, K. Nishino, and A. Shokoufandeh. Extracting 3d shape features in discrete scale-space. In *Proceedings of International Symposium on 3DPVT (3D Data Processing, Visualization, and Transmission)*, 2006.

[OBA05] Y. Ohtake, A. Belyaev, and M. Alexa. Sparse low-degree implicits with applications to high quality rendering, feature extraction, and smoothing. In *Third Eurographics Symposium on Geometry Processing*, pages 149–158, 2005.

[OBB00] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski. Polyhedral Surface Smoothing with Simultaneous Mesh Regularization. In *Proceedings of Geometric Modeling and Processing 2000*, pages 229–237, 2000.

[OBS04] Y. Ohtake, A. Belyaev, and H.-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics*, 23(3):609–612, August 2004. Proceedings of ACM SIGGRAPH.

[Owe98] S. Owen. A survey of unstructured mesh generation technology. In *Proceedings of the 7th International Meshing Roundtable*, pages 239–267, 1998.

[PH03] E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, 2003. Proceedings of ACM SIGGRAPH.

[Pin86] U. Pinkall. Cyclides of dupin. In Gerd Fischer, editor, *Mathematical Models from the Collections of Universities and Museums*, volume 2, pages 28–30. Vieweg, 1986.

[PKS+02] D. L. Page, A. Koschan, Y. Sun, J. K. Paik, and M. A. Abidi. Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Journal of Graphical Models*, 64:1–31, 2002.

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990.

[Por87] I. R. Porteous. Ridges and umbilics of surfaces. In R. R. Martin, editor, *The Mathematics of Surfaces II*, pages 447–458, Oxford, 1987. Clarendon Press.

[Por01] I. R. Porteous. *Geometric Differentiation for the Intelligence of Curves and Surfaces*. Cambridge University Press, Cambridge, 1994, 2nd Edition, 2001.

[PP93] U. Pinkall and K Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[Pto91] C. Ptolemy. *The Geography*. Dover, 1991. Translated by E. L. Stevenson.

[PTVF88] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C*. Cambridge University Press, 1988.

[Ram94] L. Ramshaw. Béziers and b-splines as multiaffine maps. *Theoretical Foundations of Computer Graphics and CAD*, 40:757–776, 1994. NATO ASI Series F: Computer and Systems Sciences.

[Res68] Y. G. Reshetnyak. Mappings with bounded deformation as extremals of Dirichlet type integrals. *Siberian Mathematical Journal*, 9:487–498, 1968.

[RL03] N. Ray and B. Lévy. Hierarchical least squares conformal map. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 263–270, Washington, DC, USA, 2003. IEEE Computer Society.

[RLL+06] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *To appear at ACM Transactions on Graphics*, 2006.

[RNJ00] R. Raffin, M. Neveu, and F. Jaar. Curvilinear displacement of free-form-based deformation. *The Visual Computer*, 16(1):38–46, 2000.

[SACO04] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics*, 23(3):878–887, 2004. Proceedings of ACM SIGGRAPH.

[SB97] S. M. Smith and J. M. Brady. SUSAN - A new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, 1997.

[SCOGL02] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization*, pages 355–362, 2002.

[SCOL+04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 179–188, 2004.

[SDS02] A. Sheffer and E. De Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics*, 21(4):874–890, 2002.

[Set96] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1996.

[SF98] K. Singh and E. Fiume. Wires: A geometric deformation technique. In *Proceedings of ACM SIGGRAPH*, pages 405–414, 1998.

[SF03] G. Stylianou and G. Farin. Crest lines extraction from 3D triangulated meshes. In G. Farin, B. Hamann, and H. Hagen, editors, *Hierarchical and Geometrical Methods in Scientific Visualization*, pages 269–281. Springer, 2003.

[SF04] G. Stylianou and G. Farin. Crest lines for surface segmentation and flattening. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):536–544, September/October 2004.

[SGSH02] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Proceedings of Eurographics Workshop on Rendering*, pages 87–98, 2002.

[SK00a] R. Schneider and L. Kobbelt. Generating Fair Meshes with G^1 Boundary Conditions. In *Proceedings of Geometric Modeling and Processing*, pages 251–260, 2000.

[SK00b] K. Singh and E. Kokkevis. Skinning characters using surface-oriented free-form deformations. In *Proceedings of Graphics Interface*, pages 35–42, 2000.

[SK01] R. Schneider and L. Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design*, 18(4):359–379, 2001.

[SK04] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *Proceedings of International Symposium on 3DPVT (3D Data Processing, Visualization, and Transmission)*, pages 68–75, 2004.

[SLMB05] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov. ABF++: fast and robust angle based flattening. *ACM Transactions on Graphics*, 24(2):311–330, 2005.

[SN96] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley, 1996.

[Sor05] O. Sorkine. Laplacian mesh processing. In *Eurographics: State-of-the-art report*, 2005.

[SP86] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Proceedings of ACM SIGGRAPH*, pages 151–160, 1986.

[SP04] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, 2004. Proceedings of ACM SIGGRAPH.

[SPW96] E. C. Sherbrooke, N. M. Patrikalakis, and F.-E. Wolter. Differential and topological properties of medial axis transforms. *CVGIP: Graphical Model and Image Processing*, 58(6):574–592, 1996.

[SS01] W. Sweldens and P. Schröder. *Digital Geometry Processing*. ACM, 2001. SIGGRAPH Course Note.

[SSGH01] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of ACM SIGGRAPH*, pages 409–416, 2001.

[Str88] D. J. Struik. *Lectures on Classical Differential Geometry: Second Edition*. Dover Publications, Inc. New York, 1988.

[SU01] A. Sheffer and A. Ungor. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17(3):326–337, 2001.

[Sur03] K. Suresh. Automating the cad/cae dimensional reduction process. In *8th ACM Symposium on Solid Modeling and Applications*, pages 76–85, 2003.

[SWG+03] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 146–155, 2003.

[TACSD06] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Surface tiling with discrete harmonic forms. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 201–210, 2006.

[Tau95] G. Taubin. A Signal Processing Approach to Fair Surface Design. In *Proceedings of ACM SIGGRAPH*, pages 315–358, 1995.

[Tau01] G. Taubin. Linear anisotropic mesh filtering. In *Tech. Rep. IBM Research Report RC2213*, 2001.

[TCR03] S. Toledo, D. Chen, and V. Rotkin. Taucs: A library of sparse linear solvers. In *www.tau.ac.il/ stoledo/taucs*, 2003.

[TG96] J.-P. Thirion and A. Gourdon. The 3D marching lines algorithm and its application to crest lines extraction. *Graphical Models and Image Processing 1996, Rapport de recherche de l'INRIA, RR-1672, Sophia Antipolis 1992*, 58(6):503–509, 1992, 1996.

[Thi96] J.-P. Thirion. The extremal mesh and the understanding of 3D surfaces. *International Journal of Computer Vision*, 19(2):115–128, 1996.

[TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, pages 839–846, 1998.

[TSG+97] G. M. Turkiyyah, D. W. Storti, M. Ganter, H. Chen, and M. Vimawala. An accelerated triangulation method for computing the skeletons of free-form solid models. *Computer-Aided Design*, 29(1):5–19, 1997.

[TSS+04] G. Tewari, J. Snyder, S. Sander, S. S. Gortler, and H. Hoppe. Signal-specialized parameterization for piecewise linear reconstruction. In *Proceedings of Eurographics Symposium on Geometry Processing 2004*, pages 57–66, 2004.

[Tut63] W. T. Tutte. How to draw a graph. In *Proceedings of the London Mathematical Society*, pages 13:743–768, 1963.

[TWBO03] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. *ACM Transactions on Graphics*, 22(4):1012–1033, 2003.

[TWM85] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation*. North-Holland, Amsterdam, 1985.

[vFTS06] W. von Funck, H. Theisel, and H.-P. Seidel. Vector-field-based shape deformations. *ACM Transactions on Graphics*, 25(3):1118–1125, 2006. Proceedings of ACM SIGGRAPH.

[Wan05] C. C. L. Wang. Length-preserved natural boundary for intrinsic parameterization. In *Proceedings of International Conference on Computer Aided Design and Computer Graphics*, pages 295–300, 2005.

[WB01] K. Watanabe and A. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3):385–392, 2001. Eurographics 2001 issue.

[WDSB00] Z. J. Wood, M. Desbrun, P. Schröder, and D. Breen. Semi-regular mesh extraction from volumes. In *Proceedings of IEEE Visualization*, pages 275–282, 2000.

[Wei98] J. Weickert. *Anisotropic Diffusion in Image Processing*. ECMI Series, Teubner-Verlag, Stuttgart, Germany, 1998.

[Weu94] J. Weule. Iteration nichtlinearer gauß-filter in der bildverarbeitung. In *PhD thesis, Heinrich-Heine-Universität Düsseldorf*, Germany, 1994.

[WF00] F.-E. Wolter and K.-I. Friese. Local & global geometric methods for analysis interrogation, reconstruction, modification & design of shape. In *Proceedings of Computer Graphics International*, pages 137–151, 2000.

[WGMY05] L. Wang, X. Gu, K. Mueller, and S.-T. Yau. Uniform texture synthesis and texture mapping using global parameterization. *Visual Computer*, 21(8-10):801–810, 2005. Special Issues of Pacific Graphics 2005.

[Win67] A. M. Winslow. Numerical solution of the quasilinear poisson equation. *Journal of Computational Physics*, 2:149–172, 1967.

[Win81] A. M. Winslow. Adaptive mesh zoning by equipotential method. Technical Report UCID-19062, Lawrence Livermore Laboratory, 1981.

[Wol92] F.-E. Wolter. Cut locus and medial axis in global shape interrogation and representation. Technical Report memorandum 92-2, MIT, Department of Ocean Engineering, January 1992.

[WTY05] C. C. L. Wang, K. Tang, and B. M. L. Yeung. Freeform surface flattening based on fitting a woven mesh model. *Computer-Aided Design*, 37(8):799–814, 2005.

[WW92] W. Welch and A. Witkin. Variational Surface Modeling. In *Proceedings of ACM SIGGRAPH*, pages 157–166, 1992.

[WW94] W. Welch and A. Witkin. Free-Form Shape Design Using Triangulated Surfaces. In *Proceedings of ACM SIGGRAPH*, pages 247–256, 1994.

[XPB06] G. Xu, Q. Pan, and C. L. Bajaj. Discrete surface modelling using partial differential equations. *Comput. Aided Geom. Des.*, 23(2):125–145, 2006.

[Yar85] L. P. Yaroslavsky. *Digital Picture Processing - An Introduction*. Springer, 1985.

[YB02] S. Yoshizawa and A. Belyaev. Fair triangle mesh generation with discrete elastica. In *Geometric Modeling and Processing*, pages 119–123, 2002.

[YBS02] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. A simple approach to interactive free-form shape deformations. In *Proceedings of Pacific Graphics*, pages 471–474, 2002. Poster.
A Java3D Toolkit for Interactive Free-Form Shape Deformations is available from
`www.mpi-inf.mpg.de/~shin/Research/DeformMesh/DeformMesh.html`.

[YBS03] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Free-form skeleton-driven mesh deformations. In *8th ACM Symposium on Solid Modeling and Applications*, pages 247–253, 2003.

[YBS04] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 200–208, 2004. C++ code is available from `www.mpi-inf.mpg.de/~shin/Research/Param1/Param1.html`.

[YBS05a] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Fast and robust detection of crest lines on meshes. In *ACM Symposium on Solid and Physical Modeling and Applications*, pages 227–232, New York, NY, USA, 2005. ACM Press. Technical Sketch.
C++ code and Java3D viewer are available from `www.mpi-inf.mpg.de/~shin/Research/Crests/Crests.html`.

[YBS05b] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. A moving mesh approach to stretch-minimizing mesh parameterization. *Int. J. of Shape Modeling*, 11(1):25–42, 2005.

[YBS06a] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Skeleton-driven laplacian mesh deformations. Technical Report MPI-I-2006-4-005, MPI-Informatik, 2006.

[YBS06b] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Smoothing by example: Mesh denoising by averaging with similarity-based weights. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 38–44, 2006. C++ Code and Java3D viewer are available from `www.mpi-inf.mpg.de/~shin/Research/NL/NonLocal.html`.

[YBS06c] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Thickness-preserving shape deformations. In *Proceedings of 1st Int. Workshop on Shapes and Semantics*, pages 7–12, 2006.

[YGZS05] H. Yamauchi, S. Gumhold, R. Zayer, and H.-P. Seidel. Mesh segmentation driven by gaussian curvature. *Visual Computer*, 21(8-10):649–658, 2005. Special Issues of Pacific Graphics 2005.

[YL90] A. L. Yuille and M. Leyton. 3D symmetry-curvature duality theorems. *Comput. Vision Graph. Image Process.*, 52(1):124–140, 1990.

[Yos01] S. Yoshizawa. Shape modeling with dynamic meshes. In *Master Thesis, University of Aizu*, Aizu-Wakamatsu, Fukushima, Japan, 2001.

[Yui89] A. L. Yuille. Zero crossings on lines of curvature. *Graphical Models and Image Processing*, 45(1):68–87, 1989.

[YYSZ06] J. Yan, X. Yang, P. Shi, and D. Zhang. Mesh parameterization by minimizing the synthesized distortion metric with the coefficient-optimizing algorithm. *IEEE Trans. Vis. Comput. Graph.*, 12(1):83–192, 2006.

[YZX+04] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):644–651, 2004. Proceedings of ACM SIGGRAPH.

[ZG04a] S. Zelinka and M. Garland. Similarity-based surface modelling using geodesic fans. In *Second Eurographics Symposium on Geometry Processing*, pages 209–218, 2004.

[ZG04b] M. Zwicker and C. Gotsman. Meshing point clouds using spherical parameterization. In *Proceedings of Eurographics Symposium on Point-Based Graphics*, 2004.

[ZHS+05] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Transactions on Graphics*, 24(3):496–503, 2005. Proceedings of ACM SIGGRAPH.

[ZMT05] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 2005.

[ZRKS05] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. *Computer Graphics Forum*, 24(3):601–609, 2005.

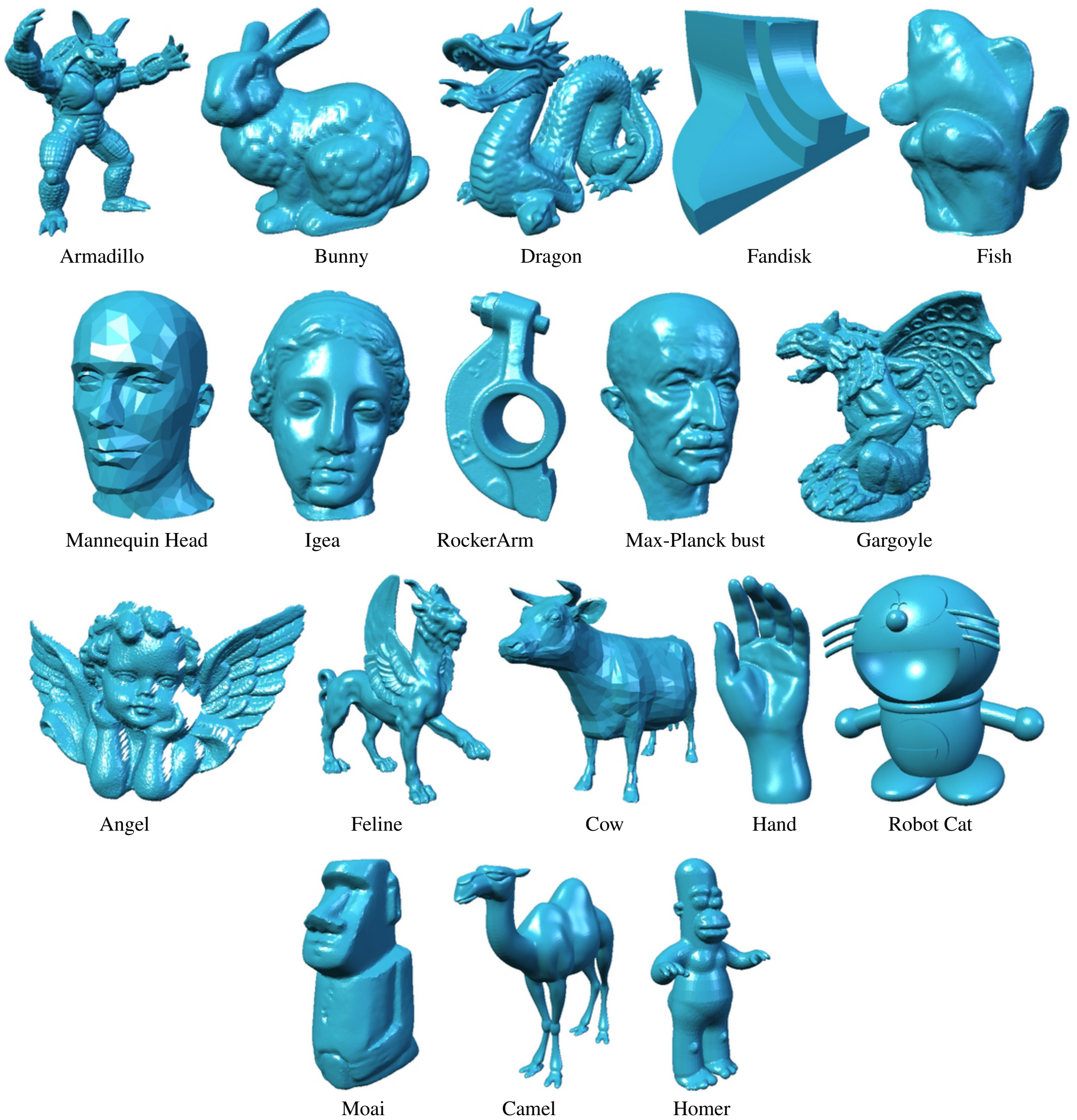
[ZRS05a] R. Zayer, C. Rössl, and H.-P. Seidel. Discrete tensorial quasi-harmonic maps. In *Proceedings of IEEE International Conference on Shape Modeling and Applications*, pages 276–285, 2005.

[ZRS05b] R. Zayer, C. Rössl, and H.-P. Seidel. Setting the boundary free: A composite approach to surface parameterization. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 91–100, 2005.

[ZSGS04] K. Zhou, J. Synder, B. Guo, and H.-Y. Shum. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of Eurographics Symposium on Geometry processing*, pages 45–54, New York, NY, USA, 2004. ACM Press.

Mesh Sources

3D models used in this thesis are courtesy of Stanford University (Armadillo, Bunny, and Dragon), University of Washington (Fandisk, Fish, and Mannequin Head), Cyberware Inc. (Igea and RockerArm), MPI Informatik (Max-Planck bust), CNR-IMATI (Gargoyle), Caltech Vision Group (Angel), Caltech Multi-Res Modeling Group (Feline), Kitware Inc. (Cow), FarField Technology Ltd (Hand), Dr. Alexander Pasko (Robot Cat), Dr. Thouis Jones (Noisy Dragon Head, original from the Stanford University), Dr. Yutaka Ohtake (Moai), and AIM@SHAPE shape repository (Camel and Homer).



Summary

In this thesis, new approaches for surface interrogating, fairing, and designing are developed. The approaches are based on computational differential geometry. They are first designed for processing smooth continuous surfaces and then adapted for dealing with triangulated polygonal surfaces. This our strategy to start from differential geometry concepts and then develop and use their proper discrete analogs turned out to be quite successful and led us to the following contributions in the geometric modeling area.

A new and powerful mesh/soup denoising technique is described in Chapter 2. Our technique is based on similarity-weighted averaging, therefore, high quality denoising results are achieved by preserving features (local shape patterns). The basic idea of our technique is inspired by the recent NL-means image filtering approach proposed by Buades et al. [BCM05a, BCM05b, BCM06]. We have extended the NL-means concept to the 3D meshes and triangle soups approximating piecewise smooth surfaces. The extension is far from being straightforward, since the original NL-means approach relies heavily on the image structure regularity. We think we have found a simple and elegant solution to the problem by employing local RBF approximations in order to estimate similarities for irregular data. A new scheme for comparing different mesh/soup denoising methods is also suggested.

In Chapter 3, a novel mesh fairing and restoration scheme is presented. Our scheme is build upon a discrete approximation of Willmore flow. A tangent speed component is introduced to the discrete Willmore flow in order to improve the quality of the evolving mesh and to increase computational stability. Contributions of our work include combining the mesh evolution approach with mesh refinement.

A new technique for fast and robust detection of salient curvature extrema on meshes is proposed in Chapter 4. Our technique consists of a novel local polynomial fitting procedure, a new curvature derivatives formula, and a smart thresholding scheme. The results of our crest line detection procedure depend only slightly on the quality of the mesh. Our method is fast and capable of achieving high quality results in detecting salient curvature extrema to compare with schemes based on global fitting procedures. Our thresholding scheme for removing unessential crest lines is based on interesting relationships between Dupin cyclides, focal sets, curvature extrema, and variational functionals. We use cyclideness as the main ingredient of our filtering scheme and measure the strength of crest lines by a scale-independent quantity. Applications to feature-sensitive mesh simplification and partition problems are also demonstrated.

In Chapter 5, a powerful moving mesh approach is described to a mesh parameterization problem. Given a triangle mesh, we first construct an initial mesh parameterization as mapping and then improve the parameterization gradually: at each improvement step we optimize the parameterization generated at the previous step. The optimization is achieved by minimiz-

ing a weighted quadratic energy with positive weights chosen to minimize the parameterization stretch. Our approach equalizes local stretches over a mesh by solving a few sparse systems of linear equations. Consequently, our method is significantly faster than the conventional method [SSGH01, SGSH02] and capable of achieving high quality mesh parameterizations. Moreover the generated mesh parameterizations do not have both high anisotropic distortions and triangle flips. Application to a remeshing problem is also considered by using a new double parameterization scheme.

A powerful approach for feature-preserving free-form shape deformations is proposed in Chapter 6. First a skeletal mesh, a Voronoi-based approximation of the medial axis, is extracted from a given mesh. Next the skeletal mesh is modified by free-form deformations. Then a desired global shape deformation is obtained by reconstructing the shape corresponding to the deformed skeletal mesh. The use of the medial axis prevents the so-called collapsing joint defects which are thickness changing effects where a large bending or twisting deformation is applied via conventional space deformations. Thus, our approach can generate natural-looking large-scale deformations by preserving original shape thickness. Mesh fairing procedures for removing possible global and local self-intersections are also developed. Finally, we combine our skeleton-driven deformation method with the variational approach and multiresolution representation. Combining our approach with the multiresolution representation reduces excessive complexity of the skeletal mesh and accelerates the deformation process. Also the use of a variational technique improve stability and quality for the deformation process.

Zusammenfassung

In dieser Dissertation werden neue Ansätze für Abfrage, Glätten und Konstruktion von Flächen entwickelt. Die Ansätze basieren auf rechnergestützter Differentialgeometrie. Sie werden zunächst für die Bearbeitung glatter, stetiger Flächen entwickelt und dann angepaßt, um auf triangulierte, polygonale Flächen anwendbar zu sein. Diese unsere Strategie, von differentialgeometrischen Konzepten ausgehend geeignete diskrete Entsprechungen zu entwickeln, stellte sich als sehr erfolgreich heraus und führte uns zu folgenden Beiträgen im Bereich der geometrischen Modellierung.

Eine neue, leistungsfähige Technik zum Entfernen von Rauschen in Dreiecksnetzen wird in Kapitel 2 beschrieben. Unsere Technik basiert auf einer nach Ähnlichkeit gewichteten Mittelung, folglich werden hochwertige Resultate beim Entfernen von Rauschen durch das Bewahren von Flächencharakteristika (lokale Form-Muster) erzielt. Die Grundidee unserer Technik ist durch den neuen "NL-means"-Ansatz zum Filtern von Bildern inspiriert, die von Buades et al. [BCM05a, BCM05b, BCM06] vorgeschlagen wurde. Wir haben das "NL-means"-Konzept auf 3D-Dreiecksnetze mit und ohne Konnektivität erweitert, die stückweise glatte Oberflächen approximieren. Die Erweiterung ist alles andere als einfach, da der ursprüngliche "NL-means"-Ansatz stark auf der Regularität der Bildstruktur beruht. Wir denken, eine einfache und elegante Lösung für das Problem gefunden zu haben, indem wir lokale RBF-Näherungen einsetzen, um Ähnlichkeiten für unregelmäßige Daten abzuschätzen. Ein neues Schema für das Vergleichen unterschiedlicher Dreiecksnetz-Glättungsmethoden wird vorgeschlagen.

In Kapitel 3 wird ein neues Schema zur Erzeugung und Rekonstruktion von ästhetischen Dreiecksnetzen vorgestellt. Unser Methode basiert auf einer diskreten Näherung des Willmore-Flusses. Eine Tangentialgeschwindigkeitskomponente wird im diskreten Willmore-Fluss eingeführt, um die Qualität des entstehenden Dreiecksnetzes zu verbessern und die Berechnungsstabilität zu erhöhen. Einer der Beiträge unserer Arbeit ist das Verknüpfen des diskreten Willmore-Flusses mit der Verfeinerung von Dreiecksnetzen.

Eine neue Technik für schnelles und robustes Erkennen von auffälligen Krümmungsextrema und Kammlinien auf Dreiecksnetzen wird in Kapitel 4 vorgeschlagen. Unsere Technik besteht aus einem neuen Verfahren zur lokalen Anpassung von Polynomen, einem eleganten Schwellwert-Schema und einer einfachen, neuen Formel für das Berechnen von Richtungsableitungen von Krümmung. Die Resultate unseres Verfahrens hängen nur geringfügig von der Qualität des Dreiecksnetzes ab. Unsere Methode ist schnell und kann qualitativ hochwertige Ergebnisse beim Erkennen auffälliger Krümmungsextrema erzielen, verglichen mit globalen Methoden. Unser Schwellwert-Schema für das Entfernen unwesentlicher Kammlinien basiert auf interessanten Beziehungen zwischen "Dupin's cyclides", Brennpunktmengen, Krümmungsextrema und Variationsfunktionalen. Wir verwenden "cyclideness" als den

Hauptbestandteil unseres Schwellwert-Schemas und messen die Stärke der Kammlinien durch größenunabhängige Maße. Anwendungen auf Dreiecksnetzvereinfachung und -segmentierung unter Berücksichtigung von Flächencharakteristika werden auch demonstriert.

In Kapitel 5, wird eine leistungsfähige "moving mesh"-Methode zu einem Parametrisierungsproblem von Dreiecksnetzen beschrieben. Gegeben ein Dreiecksnetz, konstruieren wir zuerst eine Ausgangs-Parametrisierung als Abbildung eines Dreiecksnetzes. Dann verbessern wir die Parametrisierung schrittweise. In jedem Verbesserungsschritt optimieren wir die Parametrisierung, die im vorhergehenden Schritt erzeugt wurde. Die Optimierung wird erzielt, indem man ein gewichtetes, quadratisches Energiefunktional minimiert. Dabei werden positive Gewichte so gewählt, dass die Streckung der Parametrisierung minimiert wird. Unsere Methode verteilt lokale Streckung gleichmäßig über ein Dreiecksnetz, indem wenige dünnbesetzte lineare Gleichungssysteme gelöst werden. Infolgedessen ist unsere Methode erheblich schneller als herkömmliche Methoden [SSGH01, SGSH02] und kann qualitativ hochwertige Dreiecksnetz-Parametrisierungen erzielen. Weiterhin haben die erzeugten Dreiecksnetz-Parametrisierungen weder hohe anisotrope Verzerrungen noch Überschneidungen von Dreiecken. Die Anwendung auf ein Neuvernetzungsproblem wird auch betrachtet, indem man eine neue Technik benutzt, die auf dem Verwenden von zwei Parametrisierungen basiert.

Eine leistungsfäehiger Ansatz für Freiform-Deformationen unter Beibehaltung von Flächencharakteristika wird in Kapitel 6 vorgeschlagen. Zuerst wird ein Skelett-Dreiecksnetz, eine Voronoi-basierte Annäherung der medialen Achse, von einem gegebenen Dreiecksnetz extrahiert. Als nächstes wird das Skelett-Dreiecksnetz durch Freiform-Deformationen verändert. Dann wird eine gewünschte globale Deformation erreicht, indem die Form, die dem verformten Skelett-Dreiecksnetz entspricht, rekonstruiert wird. Der Gebrauch der medialen Achse verhindert sogenannte kollabierende Gelenk-Defekte, welches Effekte durch Änderungen der ursprünglichen Dicke des Körpers sind, die auftreten, wenn eine große Biegung oder Drehung mit herkömmliche Raumdeformationen durchgeführt wird. Somit kann unser Ansatz natürlich wirkende, großmaßstäbliche Deformationen von Dreiecksnetzen erzeugen, indem die ursprüngliche Dicke des Körpers bewahrt wird. Außerdem werden neue Techniken zum Glätten von möglichen globalen und lokalen Selbst-Überschneidungen entwickelt. Schließlich kombinieren wir unsere Skelett-kontrollierte Deformationsmethode mit Auflösungs-Hierarchien und Variationsverfahren. Das Kombinieren unseres Ansatzes mit den Auflösungs-Hierarchien verringert die Komplexität des Skelett-Dreiecksnetzes außerordentlich und beschleunigt den Deformationsprozeß. Ferner verbessert der Gebrauch einer Variationstechnik Stabilität und Qualität des Deformationsprozeßes.