Journal of Computational Mathematics
Vol.34, No.1, 2016, 87–112.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1510-m4510

A VARIATIONAL APPROACH FOR DETECTING FEATURE LINES ON MESHES*

Weihua Tong
School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China
Email: tongwh@ustc.edu.cn
Xuecheng Tai
Department of Mathematics, University of Bergen, P.O. Box 7800, N-5020, Bergen, Norway
Email: tai@mi.uib.no

Abstract

Feature lines are fundamental shape descriptors and have been extensively applied to computer graphics, computer-aided design, image processing, and non-photorealistic rendering. This paper introduces a unified variational framework for detecting generic feature lines on polygonal meshes. The classic Mumford-Shah model is extended to surfaces. Using Γ-convergence method and discrete differential geometry, we discretize the proposed variational model to sequential coupled sparse linear systems. Through quadratic polynomials fitting, we develop a method for extracting valleys of functions defined on surfaces. Our approach provides flexible and intuitive control over the detecting procedure, and is easy to implement. Several measure functions are devised for different types of feature lines, and we apply our approach to various polygonal meshes ranging from synthetic to measured models. The experiments demonstrate both the effectiveness of our algorithms and the visual quality of results.

Mathematics subject classification: 65D18.
Key words: Feature lines, Variational approach, Polygonal meshes, The Mumford-Shah model, Discrete operators, Valleys of functions.

1. Introduction

In computer graphics, computer-aided design and image processing, a feature is an individual measurable heuristic property of the object being observed and is relevant for solving the computational task related to a certain application. For many applications, the feature detection is a key ingredient built on which other high-level tasks can be further performed. In this paper, the term of "feature line" is very general and may refer to sharp features (also known as creases), ridges and valleys, and contours (also known as silhouettes) and so on. These features are powerful shape descriptors which are widely used for surface reconstruction, mesh filtering, shape matching, interrogation, and non-photorealistic rendering purposes, and have gained much attention in recent years.

* Received April 13, 2014 / Revised version received June 23, 2015 / Accepted October 24, 2015 / Published online January 18, 2016 /

1.1. Previous work

To detect sharp features such as creases and corners, Hoppe et al. [1] devised an energy function that captures the number of vertices in the control mesh, their connectivity, their positions, and the number and locations of sharp features. They solved the optimization problem by decomposing it into two nested subproblems: an inner continuous optimization over the control vertex positions and an outer discrete optimization over the sharp edges. Unfortunately, their method is global, complicated and time-consuming. Contrarily, Kobbelt et al. [2] recognized the sharp features directly using some simple but effective heuristic rules based on clustering of normals. Also, Ohtake et.al [3] employed the same method so as to reconstruct the sharp features of points cloud. Their methods are straightforward to implement, but suffer from the choice of the threshold parameters and the sensitivity to noise. To overcome the latter shortcoming, robust statistic technique was introduced independently by Jones et al. [4] and Fleishman et al. [5]. They estimated smoothness of a surface using different local surface predictors, which make the main difference between theirs work. Based on robust estimation of vertex positions and smoothness, the bilateral filter [6] was applied to surface denoising while preserving features. They determined a vertex on a sharp edge by the intersection of two planes. Using the forward-search paradigm, Fleishman et al. [7] proposed a robust moving least-squares technique for reconstructing a piecewise smooth surface from a potentially noisy points cloud. In their work, sharp features were handled by treating the points across sharp features as outliers. Recently, Fan et al. [8] presented a feature-preserving mesh denoising algorithm in the same spirit. They identified piecewise smooth sub-neighborhoods using a robust density-based clustering algorithm over shared nearest neighbors, and adopted second-order bilateral filters. A common character of robust statistic-based methods is that they detect and utilize sharp features locally and implicitly. The problems of representing and recovering sharp features on various types of surfaces also have been studied by many authors, such as [9-13].

Mathematically, ridges and valleys are defined as lines on a surface where the principal curvatures attain extrema along their associated principal directions. They were first investigated by Gullstrand, 1911 Nobel Laureate in Medicine, who applied the methods of physical mathematics to the study of optical images and of the refraction of light in the eye. During the last century, ridges and valleys have been extensively studied in connection with researches on classical differential geometry and singularity theory [14], analysis of medical images [15], face recognition [16], and quality control of free-form surfaces [17] etc. In the past decade, there has been considerable effort to develop methods for extracting ridge-valley structures on polygonal meshes. Based on linear interpolation and non-maximum suppression technique, Belyaev et al. [18] presented a method for ridges and valleys detection on range images and triangular meshes. In order to achieve stable results, they employed a coupled nonlinear diffusion procedure to smooth the position and normal of vertices. Using a scheme from discrete differential geometry and a smoothing filter for higher-order surface derivatives, Hildebrandt et al. [19] provided an efficient way for feature lines detection. Along a different line, Ohtake et al. [20] fitted a multi-level implicit surface to the given polygonal mesh in advance. The curvature tensor and curvature derivatives at a mesh vertex are then estimated by those at the corresponding implicit surface point. In [21], they proposed a sparse representation SLIM for approximating a set of scattered points. The SLIM allows to estimate high order surface derivatives simply, which leads to a effective feature detection technique. For the purpose of accelerating the computation of principal curvatures and their derivatives, Kim et al. [22] and Yoshizawa et al. [23] utilized modified MLS (moving least squares) and local cubic polynomial

approximations respectively. Further, Yoshizawa et al. [24, 25] developed a fast and faithful geometry-based finite difference method, which avoids surface fitting procedure. All of these methods highly rely on a high-quality estimation of high-order surface derivatives. Recently, Lai et al. [26] presented a method for feature lines extraction, which employed the integral invariants of geodesic circles approximation in the feature-sensitive metric.

In the area of non-photorealistic rendering (NPR), contours have been investigated for many years [27, 28] due to theirs salience and view dependence. Markosian et al. [29] presented a randomized algorithm that can find the majority of visible silhouette edges at interactive rates. Their algorithm is based on the observation that only a small fraction of mesh edges are visible silhouette edges. Hertzmann et al. [30] described a method for computing silhouettes of smooth surfaces. They determined silhouette curves as zero-crossings of the dot product of the normal with the view direction. For conveying shape in an indirect way, DeCarlo et al. [31] introduced the suggestive contour which are curves on the shape that might be silhouettes in nearby views. They calculated them by finding the zero crossings of the radial curvature. In [32], they further developed two new families of lines: suggestive highlights and principal highlights. An excellent survey can be found in [33].

Variational methods are fundamental mathematical tools, and have been widely applied to various problems in areas of image processing and analysis, and computer vision. Refer to [34, 35] for details. Among them, the Mumford-Shah model has been extensively investigated for the segmentation problem. As such there is a large body of related work, and we shall only review the work that are relevant for our treatment, i.e., extending the Mumford-Shah model to 3D or surface cases. Dufour et al. [36] proposed a deformable 3D mesh model using the reduced Mumford-Shah functional, which can segment and track objects with fuzzy boundaries. In [37], Jin et al. extended the Mumford-Shah functional to functions defined on a deforming surface for reconstructing 3D shape and appearance from multi-view images. By introducing two fourth order regularization methods to the Mumford-Shah model, Droske et al. [38] applied them to aerial image segmentation and feature-preserving surface denoising. To tackle the problem of segmenting data defined on a manifold into regions of constant properties, Delaunoy et al. [39] extended the convex image labeling model to manifolds, which is the piecewise constant case of the Mumford-Shah model.

1.2. Contributions and outline

As described above, the concept of feature is very general and there are various definitions of what constitutes a specific feature. However, existing definitions typically are based on local differential properties or statistical quantities, such as ridges, valleys, contours and so on. In this paper, we bring the spirit of variational approach to the problem of generic feature lines detection on polygonal meshes. The key idea is to characterize *feature* on polygonal meshes as where the measure function changes sharply or more formally has discontinuities. Here, the measure function is a function defined on the polygon mesh, which captures a concrete property being observed, e.g. mean curvature, intensity. The motivation of this work is that human visual system is sensitive to discontinuities and often recognizes them as features. For example, the human eye exaggerates the intensity change at any edge where there is a discontinuity in magnitude or slope of intensity. This leads to Mach band effect (refer to [40]). Our approach provides another way to detect feature lines and offers the following technical features:

- A unified variational framework for feature lines detection is developed. Various feature

lines can be detected only by changing the measure function. In support of this goal, the Mumford-Shah model and its Γ-convergence approximation, and valleys of functions are extended to surfaces.

- Flexible and intuitive control over the detecting procedure is available. In our system, several parameters, such as α, β, γ, $T(C)$, can be used to tune up feature lines detection. Interactive ways to select the region of interest (ROI) and terminate the iterative process are also provided.

- Our approach is computationally effective, robust and easy to implement. Neither surface fitting to meshes nor estimation of high-order derivatives is required. Using Γ-convergence technique, our variational model is finally discretized to sequential coupled sparse linear systems .

The remainder of this paper is organized as follows. Section 2 gives some preliminaries and an overview of algorithm framework. In Section 3, we generalize the Mumford-Shah model to manifold surfaces, and present numerical algorithms for our variational model in detail. In Section 4, we propose a method for extracting feature lines from meshes according to the canyon function z. Section 5 discusses the choice of measure functions and parameters, and shows experimental results and performance of our algorithms. Section 6 concludes the paper with proposals for future work.

2. Preliminaries and Overview

In this section, we give some preliminaries and an overview of our variational method.

2.1. Polygonal meshes

In computer graphics and geometric modeling, a *polygonal mesh* is a collection of vertices, edges and faces that defines the shape of a polyhedral object. The faces usually consist of triangles, quadrilaterals or other simple polygons. For the sake of simplicity and popularity, this paper will restrict faces to triangles. A *triangle mesh* $\mathcal{M}$ can be represented by a graph structure (simplicial complex) with a set of vertices

$$\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$$

and a set of triangular faces connecting them

$$\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V},$$

where $|\cdot|$ denotes the cardinality of a set, and $\mathcal{X} \times \mathcal{Y}$ is the Cartesian product of two sets. An alternative way is to represent the connectivity by the edges of the respective graph

$$\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}, \quad e_i \in \mathcal{V} \times \mathcal{V},$$

which sometimes is more efficient. The geometric embedding of a triangle mesh into $\mathbb{R}^3$ is specified by associating a position $\mathbf{v}_i$ to each vertex v_i:

$$\mathcal{P} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{|\mathcal{V}|}\}, \quad \mathbf{v}_i = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3.$$

In addition to geometric and topological components, a triangle mesh $\mathcal{M}$ may have some constraints on the relations among its different elements, i.e., vertices, edges and faces, which impose a valid representation. In a *two-dimensional manifold* mesh, the neighborhood of every point laid on the mesh is homeomorphic to a disk (or a half-disk at boundaries). For a closed mesh, it will not contain any boundary edges. Rigorous treatments on these topics can be found in [41-43]. In this paper, we only consider orientable two-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$ of arbitrary topological type with no degenerate triangles.

2.2. The Mumford-Shah image model

In image processing and analysis, *image segmentation* is the process of partitioning the domain Ω of an image $\mathcal{I}$ into constituent parts Ω_i built upon which other high-level tasks such as object detection, recognition, and tracking can be further performed. During last decades, several important and interconnected models pertinent to image segmentation have been developed, such as active contours [44], intensity-edge mixture model [45] and Mumford-Shah's free boundary model [46]. Assume the image domain Ω is a bounded Lipschitz domain in $\mathbb{R}^2$ (whose boundary is sufficiently regular in the sense that it can be thought of as locally being the graph of a Lipschitz continuous function), and $u_0 \in L^\infty(\Omega)$ or $L^2(\Omega)$ is the initial image. A Lipschitz partition of Ω refers to a finite set partition

$$\Omega = \Omega_1 \cup \Omega_2 \cdots \cup \Omega_N \cup \Gamma,$$

where Ω_i is a connected open Lipschitz domain, and Γ is a relatively closed subset of Ω with finite 1-D Hausdorff measure. The *Mumford-Shah energy functional* can be formulated as follows:

$$E_{ms}[u, \Gamma \mid u_0] = \alpha \mathcal{H}^1(\Gamma) + \frac{\beta}{2} \int_{\Omega \backslash \Gamma} |\nabla u|^2 \mathrm{d}x_1 \wedge \mathrm{d}x_2 + \frac{\gamma}{2} \int_{\Omega} (u - u_0)^2 \mathrm{d}x_1 \wedge \mathrm{d}x_2,$$

where $\mathcal{H}^1(\Gamma) = \int_{\Gamma} ds$ (when Γ is regular) stands for the total length of the arcs making up Γ, $\mathrm{d}x_1 \wedge \mathrm{d}x_2$ is the exterior product of two 1-forms and α, β and γ are nonnegative constant weights. In the functional $E_{ms}[u, \Gamma \mid u_0]$, function u is differentiable on $\bigcup_{i=1}^{N} \Omega_i$ and is allowed to be discontinuous across Γ. As described in the seminal paper of Mumford and Shah [46], the first term asks that the discontinuity set Γ be as short as possible, and the second term imposes the condition that u be smooth in $\Omega \backslash \Gamma$, and the third term measures the fidelity to u_0.

For a given image u_0, our aim is to search a function u and a subset Γ of Ω which minimize the Mumford-Shah energy functional, i.e., trying to approximate the input image u_0 with a set of smoothly varying regions Ω that have short boundaries Γ. A variational algorithm optimizes the way in which neighboring pixels can be merged into homogeneous regions separated by qualitative discontinuities. It transforms the image segmentation problem into a particular case of what is called in physics *free boundary problems*. With proper regularity hypotheses, the existence and some basic properties of a solution for the Mumford-Shah image model have been studied by Mumford and Shah [46] and others. However, it is hard to solve it in an effective way. The main difficulty is that the Mumford-Shah image model involves two unknowns u and Γ of different natures. The lack of differentiability of the Mumford-Shah functional with a suitable norm does not allow us to use Euler-Lagrange equations. Until now, there are two prevalent methods for solving the Mumford-Shah's image model approximately. One is based on Γ-convergence approximation, the other uses level-set technique. More in-depth information can be found in Section 3.2 or [34, 35] and references therein.

2.3. Framework

A brief sketch of our generic framework is illustrated in Fig. 2.1. Begin with a given polygonal mesh, our aim is to detect feature lines according to measure function u_0 whose values are computed from mesh or loaded from data. A nonlinear smoothing procedure is executed to obtain function u. By comparing the difference between function u_0 and u, we gain function z which is strongly related to edge set Γ. With the help of visualizing function z in an interactively graphic environment, the end-user makes decision on whether to perform smoothing and detecting iterations again or not, and select the ROI. Then on each vertex of mesh within the ROI, a quadric surface is fitting to function z under certain a local coordinate system. We check some minimal conditions and locate all valley points which are then connected based on topological relations. Finally, the resulting feature lines are filtered so as to achieve pleasing shape. If the end-user does not satisfy the result, they can repeat the previous process together with tuning some parameters till the desired result is achieved. The following sections describe the details of each step in Fig. 2.1.

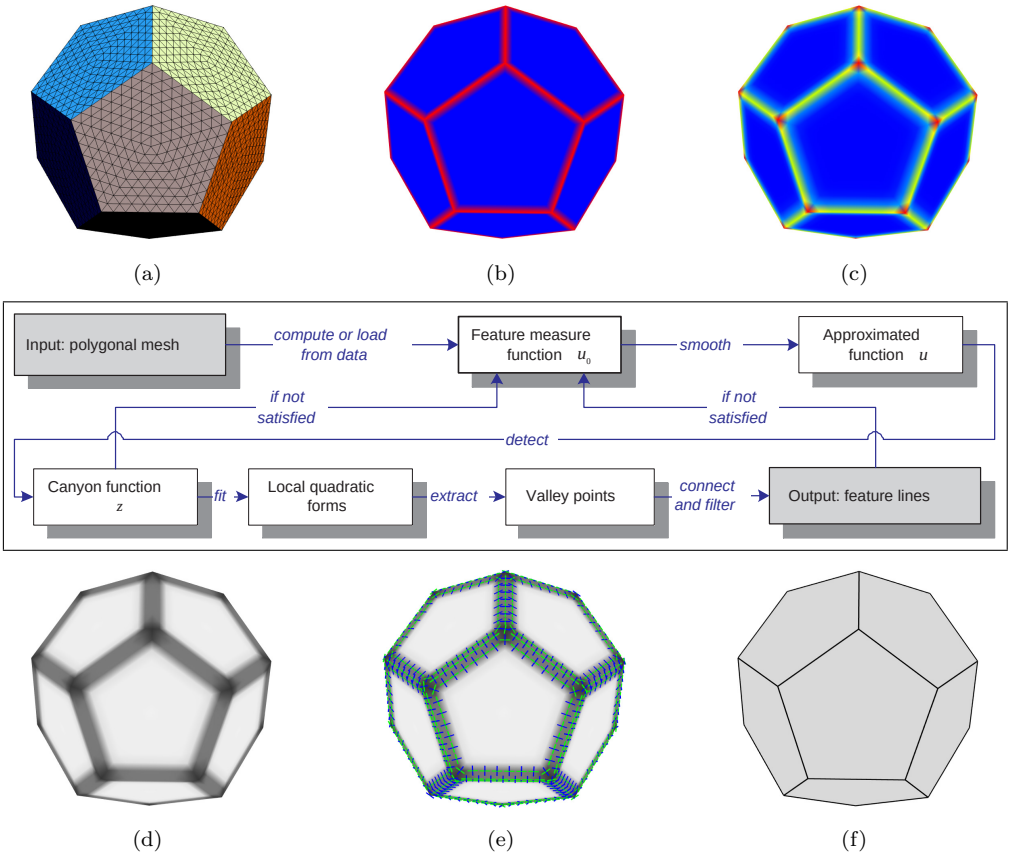


Fig. 2.1. The generic framework of our method: (a) polygonal mesh; (b) measure function u_0 (mean curvature); (c) function u; (d) canyon function z; (e) principal axes of quadric surfaces; (f) feature lines.

3. Variational Model for Feature Lines Detection

In order to detect feature on polygonal meshes, we extend the Mumford-Shah image model to surfaces. In the original model, the domain Ω of image $\mathcal{I}$ usually is a planar region in $\mathbb{R}^2$, especially a rectangle. As a result, Euclidean metric and geometry are employed. Whereas, we study functions defined on two-dimensional manifold embedded in $\mathbb{R}^3$, i.e., surface. Consequently, Riemannian metric and geometry have to be used. It distinguishes our variational model from the Mumford-Shah image model.

3.1. The Mumford-Shah model on surfaces

Assume $\mathcal{S}$ is a given surface in $\mathbb{R}^3$, and $\mathbf{x}(x_1, x_2) : U \subset \mathbb{R}^2 \to \mathcal{S}$ is a local coordinate system around $\mathbf{x} \in \mathcal{S}$. In the language of Riemannian geometry [47, 48], we denote the Riemannian metric by

$$g_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \quad i, j = 1, 2,$$

where $\mathrm{d}\mathbf{x}(x_1, x_2) = \mathbf{x}_1 \mathrm{d}x_1 + \mathbf{x}_2 \mathrm{d}x_2$. Let $u_0(\mathbf{x})$ be a measure function defined on $\mathcal{S}$, which represents geometric, topological or functional information. The choice of $u_0(\mathbf{x})$ heavily depends on the really application, and is crucial to our variational method. In this paper, mean curvature and other measures may be utilized. More details will be discussed in Section 5. Then we formulate the Mumford-Shah energy functional on surface $\mathcal{S}$ as follows:

$$E_{ms}[u, \Gamma \mid u_0] = \alpha \mathcal{H}^1(\Gamma) + \frac{\beta}{2} \int_{\mathcal{S} \setminus \Gamma} |\nabla_s u|^2 \mathrm{d}\mathcal{A} + \frac{\gamma}{2} \int_{\mathcal{S}} (u - u_0)^2 \mathrm{d}\mathcal{A}, \tag{3.1}$$

where $\mathrm{d}\mathcal{A} = \sqrt{\det(g_{ij})}\, \mathrm{d}x_1 \wedge \mathrm{d}x_2$ is the *element of surface area*, and ∇_s denotes the intrinsic gradient operator on $\mathcal{S}$ (see Section 3.3.2 for details). Likewise, we need to search for a function u and a subset Γ of $\mathcal{S}$ minimizing the energy functional. The desired feature lines locate in the discontinuity set Γ.

3.2. Γ-convergence method

As described in Section 2.2, there are two prevalent methods for solving the Mumford-Shah image model. The main idea of the level-set method is to represent the one-dimensional edge set Γ as the zero level set of a C^1 continuous (or more general Lipschitz continuous) function $\phi(\mathbf{x}) : \Omega \to \mathbb{R}$, that is

$$\Gamma = \phi^{-1}(0) = \{\mathbf{x} \in \Omega \mid \phi(\mathbf{x}) = 0\}.$$

In combination with the level-set formulation, we can reformulate the Mumford-Shah model as follows:

$$\begin{aligned} E_{ms}[u^+, u^-, \phi \mid u_0] = \alpha \int_\Omega |\delta(\phi) \nabla(\phi)| \mathrm{d}x_1 \wedge \mathrm{d}x_2 + \frac{1}{2} \int_\Omega \left[\beta |\nabla u^+|^2 + \gamma (u^+ - u_0)^2\right] H(\phi) \mathrm{d}x_1 \wedge \mathrm{d}x_2 \\ + \frac{1}{2} \int_\Omega \left[\beta |\nabla u^-|^2 + \gamma (u^- - u_0)^2\right] H(-\phi) \mathrm{d}x_1 \wedge \mathrm{d}x_2, \end{aligned}$$

where $\Omega^\pm = \{\mathbf{x} \in \Omega \mid \pm \phi(\mathbf{x}) > 0\}$, $u^\pm = u|_{\Omega^\pm}$, and $\delta(\phi)$ and $H(x)$ denote the Dirac delta function and the Heaviside step function respectively. If u restricts to being a piecewise-constant approximation of u_0, then the Mumford-Shah model reduces to the Chan and Vese's model [49]. The later can be considered as a particular case of the minimal partition problem. But, the previous models can only divide Ω into two distinct regions. Vese and Chan [50] generalized their

2-phase model to a multiphase level-set framework that can identify up to 2^k regions. For an arbitrary number of partitions, Lie et al. [51] introduced a piecewise-constant level set function method (PCLSM) which uses each constant value to represent a unique region. Recently, fast solvers for multiphase partitioning problems, such as [52-54], were developed. However, by the implicit function theorem, the zero level set of $\phi(\mathbf{x})$ is locally homeomorphic to the real line in $\mathbb{R}^2$. Therefore, $\phi^{-1}(0)$ does not contain any crack-tips and junctions.

Since feature lines on polygonal mesh commonly contain some junctions and crack-tips, we prefer to use the Γ-convergence method. In this method, the one-dimensional edge set Γ is encoded by a smooth canyon function $z : \mathcal{S} \to [0, 1]$, which ideally should behave like

$$z(\mathbf{x}) \approx \begin{cases} 0 & \text{if } \mathbf{x} \in \Gamma, \\ 1 & \text{otherwise.} \end{cases}$$

The canyon function z is an approximation of the function $(1-\chi_\Gamma)$ (χ_Γ is the indicator function of edge set Γ), and is treated as an unknown variable in our variational framework. The remarkable observation described in [55] shows that the one-dimensional Hausdorff measure of Γ can be well approximated by

$$\alpha\mathcal{H}^1(\Gamma) \simeq \frac{\alpha}{2}\int_{\mathcal{S}} \left[\varepsilon|\nabla z|^2 + \varepsilon^{-1}(1-z)^2\right] \mathrm{d}\mathcal{A},$$

where the parameter ε is used to tune the transition bandwidth of z in the vicinity of Γ. Since z almost vanishes along Γ, the Sobolev term in the Mumford-Shah model allows a natural approximation:

$$\frac{\beta}{2}\int_{\mathcal{S}\setminus\Gamma} |\nabla u|^2 \mathrm{d}\mathcal{A} \simeq \frac{\beta}{2}\int_{\mathcal{S}} z^2|\nabla u|^2 \mathrm{d}\mathcal{A}.$$

Thus, the Γ-convergence approximation of the Mumford-Shah model (3.1) can be written as follows:

$$\begin{aligned} E_{ms}[u, z \mid u_0, \varepsilon] = \frac{\alpha}{2}\int_{\mathcal{S}} \left[\varepsilon|\nabla z|^2 + \varepsilon^{-1}(1-z)^2\right] \mathrm{d}\mathcal{A} \\ + \frac{\beta}{2}\int_{\mathcal{S}} z^2|\nabla u|^2 \mathrm{d}\mathcal{A} + \frac{\gamma}{2}\int_{\mathcal{S}} (u-u_0)^2 \mathrm{d}\mathcal{A}. \end{aligned} \tag{3.2}$$

Ambrosio and Tortorelli [55] proved that

$$\lim_{\varepsilon\to 0^+} E_{ms}[u, z \mid u_0, \varepsilon] = E_{ms}[u, \Gamma \mid u_0]$$

in a sense of the Γ-convergence.

The first variation of (3.2) yields the Euler-Lagrange system of elliptic equations as follows:

$$\begin{cases} \beta \mathrm{div}_s(z^2\nabla_s u) - \gamma(u-u_0) = 0, & \text{(3.3a)} \\ \alpha[\varepsilon\Delta_s z + \varepsilon^{-1}(1-z)] - \beta|\nabla_s u|^2 z = 0, & \text{(3.3b)} \end{cases}$$

with Neumann boundary conditions, i.e., $\frac{\partial u}{\partial \mathbf{n}}\big|_{\partial\mathcal{S}} = 0$ and $\frac{\partial z}{\partial \mathbf{n}}\big|_{\partial\mathcal{S}} = 0$. The definitions of div_s, ∇_s and Δ_s are given in Section 3.3.2. The system (3.3) can be numerically solved by the alternating minimization scheme, which amounts to $z^{(n)} \to u^{(n)} \to z^{(n+1)}$:

$$\begin{cases} u^{(n)} = \mathrm{argmin}\ E_{ms}[u \mid z = z^{(n)}, u_0, \varepsilon], \\ z^{(n+1)} = \mathrm{argmin}\ E_{ms}[z \mid u = u^{(n)}, u_0, \varepsilon]. \end{cases}$$

The last scheme corresponds to the following nonlinear system of diffusion equations:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \beta \mathrm{div}_s(z^2 \nabla_s u) - \gamma(u - u_0), & (3.4a) \\ \dfrac{\partial z}{\partial t} = \alpha[\varepsilon \Delta_s z + \varepsilon^{-1}(1 - z)] - \beta |\nabla_s u|^2 z, & (3.4b) \end{cases}$$

with the same Neumann boundary conditions with (3.3). The usage of (3.4a) is to smooth out function u_0 and obtain function u, while (3.4b) is to detect edge set Γ by comparing the difference between function u_0 and u. The new model (3.4) searches for an equilibrium between two competing processes rather than a global minimum of the non-convex functional (3.2). For further information, refer to [56-59, 34].

3.3. Numerical solutions

We now present some details of our algorithms and implementation.

3.3.1. Piecewise-linear functions

Assume we are given a triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ which is a piecewise-linear approximation of surface $\mathcal{S}$ in $\mathbb{R}^3$. Let $u_0 \in L^\infty(\mathcal{S})$ or $L^2(\mathcal{S})$ be a scalar function. At each vertex $\mathbf{v}_i$ of $\mathcal{M}$, the value of u_0 is u_i, i.e.,

$$u_0(\mathbf{v}_i) = u_i \in \mathbb{R}, \quad i = 1, \ldots, m, \text{ and } m = |\mathcal{V}|.$$

Using the piecewise-linear basis functions $\{\varphi_i\}$ defined on the mesh $\mathcal{M}$, we express u_0 as follows:

$$u_0(\mathbf{v}) = \sum_{i=1}^{m} \varphi_i(\mathbf{v}) u_i, \quad \mathbf{v} \in \mathcal{S},$$

where $\varphi_i(\mathbf{v})$ takes the value 1 at vertex $\mathbf{v}_i$ and the value 0 at other vertices. Especially, if $\mathbf{v}$ locates in a triangle $f = [\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]$, then $u_0(\mathbf{v})$ can be evaluated by

$$u_0(\mathbf{v}) = r u_i + s u_j + t u_k,$$

where (r, s, t) is the barycentric coordinate of $\mathbf{v}$ with respect to the triangle f. Also, function u and z are represented in a similar way.

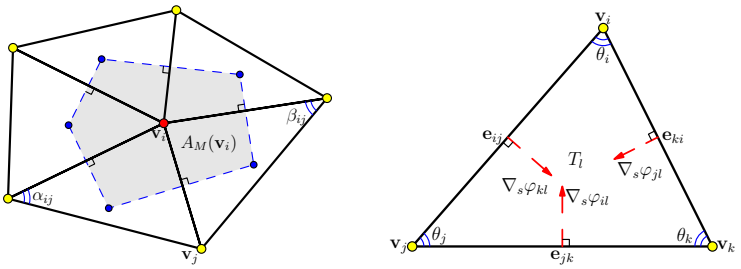
3.3.2. Discrete ∇_s and Δ_s operators

The gradient of a differentiable function $f : \mathcal{S} \to \mathbb{R}$ is a map $\nabla_s f : \mathcal{S} \to \mathbb{R}^3$, which assigns a vector $\nabla_s f(\mathbf{v}) \in T_{\mathbf{v}}(\mathcal{S})$ to each $\mathbf{v} \in \mathcal{S}$ such that

$$\langle \nabla_s f(\mathbf{v}), \mathbf{u} \rangle = \mathrm{d} f_{\mathbf{v}}(\mathbf{u}), \quad \forall\, \mathbf{u} \in T_{\mathbf{v}}(\mathcal{S}),$$

where $T_{\mathbf{v}}(\mathcal{S})$ denotes the tangent plane to $\mathcal{S}$ at $\mathbf{v}$, and $\mathrm{d} f_{\mathbf{v}}(\mathbf{u})$ is the directional derivative of f with respect to $\mathbf{u}$. If the scalar function f is represented by

$$f(\mathbf{v}) = \sum_{i=0}^{m} \varphi_i(\mathbf{v}) f_i,$$



(a) (b)

Fig. 3.1. Discrete operators: (a) the gradient operator ∇_s; (b) the Laplace-Beltrami operator Δ_s.

then we have

$$\nabla_s f(\mathbf{v}) = \sum_{i=0}^{m} f_i \nabla_s \varphi_i(\mathbf{v}).$$

Thus the $\nabla_s f(\mathbf{v})$ is given by

$$\nabla_s f(\mathbf{v}) = \begin{cases} \sum_{j\in T_l} f_j \nabla_s \varphi_{jl}, & \text{if } \mathbf{v} \in T_l^\circ, \\ \dfrac{\sum_{T_l \in NT(i)} \sum_{j\in T_l} f_j \nabla_s \varphi_{jl} |T_l|}{A(\mathbf{v}_i)}, & \text{then if } \mathbf{v} = \mathbf{v}_i, \\ 0, & \text{else,} \end{cases} \tag{3.5}$$

where $NT(i)$ is the set of all triangles immediately adjacent to the node i, $|T_l|$ and T_l° are the area and interior of triangle T_l, and $A(\mathbf{v}_i) = \sum_{T_l \in NT(i)} |T_l|$. The $\nabla_s \varphi_{jl}$ equals to the vector orthogonal to the edge $\mathbf{e}_{ki}$ opposite to $\mathbf{v}_j$ in the triangle T_l, pointing towards $\mathbf{v}_j$ and with a magnitude of $|\mathbf{e}_{ki}|/2|T_l|$ (see Fig. 3.1).

The divergence of a differentiable vector field $\mathbf{w}$ on the surface $\mathcal{S}$ is defined by

$$\mathrm{div}_s \mathbf{w} \triangleq \mathrm{tr}(\mathbf{u} \mapsto \nabla_{\mathbf{u}} \mathbf{w}),$$

where ∇ is a Levi-Civita connection and tr denotes the trace of a linear mapping [47]. The Laplace-Beltrami operator Δ_s can be written as

$$\Delta_s f(\mathbf{v}) \triangleq \mathrm{div}_s[\nabla_s f(\mathbf{v})].$$

Several discrete schemes have been developed for approximating operator Δ_s in polygonal mesh setting. For a comprehensive survey, please refer to [60]. In this paper, we use the following scheme:

$$\Delta_s f(\mathbf{v}_i) = \frac{1}{A_M(\mathbf{v}_i)} \sum_{j\in NV(i)} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2} [f(\mathbf{v}_j) - f(\mathbf{v}_i)], \tag{3.6}$$

where $NV(i)$ is the set of 1-ring vertex neighborhood of $\mathbf{v}_i$, α_{ij} and β_{ij} are the two angles opposite to the sharing edge e_{ij} in the two triangles, and $A_M(\mathbf{v}_i)$ is the area sum of $\mathbf{v}_i$'s neighboring region as depicted in Fig. 3.1(b). The div_s operator can also be discretized in a similar way, which is described in Appendix.

From the definition of mean curvature vector, we have

$$(-2\kappa_H \mathbf{n})(\mathbf{v}) = \Delta_s(\mathbf{v}).$$

By (3.6), we obtain

$$(-2\kappa_H \mathbf{n})(\mathbf{v}_i) = \frac{1}{A_M(\mathbf{v}_i)} \sum_{j \in NV(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} (\mathbf{v}_j - \mathbf{v}_i).$$

Subsequently, the mean curvature can be computed by:

$$\kappa_H(\mathbf{v}_i) = \frac{1}{A_M(\mathbf{v}_i)} \sum_{j \in NV(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{4} (\mathbf{v}_i - \mathbf{v}_j) \cdot \mathbf{n}_i, \tag{3.7}$$

where $\mathbf{n}_i$ is the unit normal of vertex $\mathbf{v}_i$ on the surface S.

3.3.3. Algorithms

We discretize the nonlinear system of elliptic equations (3.4) via semi-implicit FDM: semi-implicit time integral and spatial discretization by finite difference method. This technique is unconditionally stable and leads to systems of linear equations (see [61, 62] for details).

As a matter of convenience, we denote $u_i^n = u(\mathbf{v}_i)\big|_{t=t^n}$ and $z_i^n = z(\mathbf{v}_i)\big|_{t=t^n}$. In terms of the definitions of div_s and ∇_s, we have

$$\mathrm{div}_s(z^2 \nabla_s u) = z^2 \Delta_s u + 2z \langle \nabla_s z, \nabla_s u \rangle.$$

With a semi-implicit time integral (from t^n to t^{n+1}), (3.4a) can be discretized as:

$$\begin{aligned} \frac{u_i^{n+1} - u_i^n}{\Delta t} = \beta \Bigg[& \frac{(z_i^n)^2}{A_M(\mathbf{v}_i)} \sum_{j \in NV(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} (u_j^{n+1} - u_i^{n+1}) \\ & + \frac{2z_i^n}{A^2(\mathbf{v}_i)} \Bigg\langle \sum_{T_l \in NT(i)} \Bigg(\sum_{j \in T_l} z_j^n \nabla_s \varphi_{jl} \Bigg) |T_l|, \sum_{T_m \in NT(i)} \Bigg(\sum_{k \in T_m} u_k^{n+1} \nabla_s \varphi_{km} \Bigg) |T_m| \Bigg\rangle \Bigg] \\ & - \gamma (u_i^{n+1} - c_i), \quad i = 1, 2, \ldots, m. \end{aligned} \tag{3.8}$$

Similarly, the discrete version of (3.4b) is given by:

$$\begin{aligned} \frac{z_i^{n+1} - z_i^n}{\Delta t} = \alpha \Bigg[& \frac{\varepsilon}{A_M(\mathbf{v}_i)} \sum_{j \in NV(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} (z_j^{n+1} - z_i^{n+1}) + \frac{(1 - z_i^{n+1})}{\varepsilon} \Bigg] \\ & - \beta \Bigg| \frac{1}{A(\mathbf{v}_i)} \sum_{T_l \in NT(i)} \Bigg(\sum_{j \in T_l} u_j^{n+1} \nabla_s \varphi_{jl} \Bigg) |T_l| \Bigg|^2 z_i^{n+1}, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{3.9}$$

By rearranging coefficients of variables $\{u_i^{n+1}\}_{i=1}^N$ in (3.8) and moving constant terms into right-hand side, we can rewrite it in matrix form as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_m^{n+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \tag{3.10}$$

where a_{ij} is given by

$$a_{ij} = \begin{cases} 1 + \beta\Delta t\left[\dfrac{(z_i^n)^2}{A_M(\mathbf{v}_i)} \displaystyle\sum_{j\in NV(i)} \frac{(\cot\alpha_{ij} + \cot\beta_{ij})}{2} - \frac{2z_i^n}{A^2(\mathbf{v}_i)}\right. \\ \quad \left\langle \displaystyle\sum_{T_l\in NT(i)} \Big(\sum_{j\in T_l} z_j^n \nabla_s\varphi_{jl}\Big)|T_l|, \sum_{T_m\in NT(i)} \nabla_s\varphi_{im}|T_m| \right\rangle \Big] + \gamma\Delta t, & \text{if } j = i, \\ -\beta\Delta t\left[\dfrac{(z_i^n)^2(\cot\alpha_{ij} + \cot\beta_{ij})}{2A_M(\mathbf{v}_i)} + \frac{2z_i^n}{A^2(\mathbf{v}_i)} \left\langle \displaystyle\sum_{T_l\in NT(i)} \right.\right. \\ \quad \Big(\sum_{j\in T_l} z_j^n \nabla_s\varphi_{jl}\Big)|T_l|, \sum_{T_m\in NT(i)\bigcap NT(j)} \nabla_s\varphi_{jm}|T_m| \Big\rangle \Big], & \text{then if } j \in NV(i), \\ 0, & \text{else,} \end{cases}$$

and $b_i = u_i^n + \gamma c_i\Delta t,\ i = 1, 2, \ldots, m.$

Also (3.9) is rewritten as follows:

$$\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mm} \end{bmatrix} \begin{bmatrix} z_1^{n+1} \\ z_2^{n+1} \\ \vdots \\ z_m^{n+1} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix}, \tag{3.11}$$

where d_{ij} is given by

$$d_{ij} = \begin{cases} 1 + \alpha\Delta t\left[\dfrac{\varepsilon}{A_M(\mathbf{v}_i)} \displaystyle\sum_{j\in NV(i)} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2} + \frac{1}{\varepsilon}\right] \\ \quad + \beta\Delta t\left|\dfrac{1}{A(\mathbf{v}_i)} \displaystyle\sum_{T_l\in NT(i)} \Big(\sum_{j\in T_l} u_j^{n+1}\nabla_s\varphi_{jl}\Big)|T_l|\right|^2, & \text{if } j = i, \\ -\varepsilon\alpha\Delta t\dfrac{\cot\alpha_{ij} + \cot\beta_{ij}}{2A_M(\mathbf{v}_i)}, & \text{then if } j \in NV(i), \\ 0, & \text{else,} \end{cases}$$

and $g_i = z_i^n + \varepsilon^{-1}\alpha\Delta t,\ i = 1, 2, \ldots, m.$

Due to the unconditional stability of semi-implicit FDM, we can choose the time step $\Delta t > 0$ arbitrarily. In theory, large Δt makes the approximate solution be more close to the exact solution of (3.3) than the small one at the equivalent computational cost. But, large Δt introduces big numerical error and leads to smooth out functions u and z overly, which cause some feature lines to be missed. Thus, we should choose Δt to balance the efficiency and the accuracy. It's hard to give an optimal value of Δt. In our experiment, we choose $\Delta t = 0.001$ in all of our examples which is proved to work well.

Our iterative algorithm starts by setting canyon function $z = 1$ everywhere, which corresponds to the absence of discontinuity set Γ, and $u(\mathbf{v})\big|_{t=t^0} = u_0(\mathbf{v})$. Then just as Gauss-Seidel method, we alternating solve u^{n+1} from (3.10) with known u^n and z^n, and z^{n+1} from (3.11) with known u^{n+1} and z^n respectively. Both these systems are sparse, and can effectively be stored and solved by using numerical linear algebra packages, such as LASPACK [63] or TAUCS [64]. Until the resulting canyon function z is satisfied, we terminate this iteration process.

4. Extraction of Feature Lines

Now, we have canyon function z at hand, which represents feature lines implicitly and is enough for some applications, such as non-photorealistic rendering. However, it is fairly common to use feature lines explicitly. According to the definition of canyon function z, feature lines are loci of points where function z takes the value of zero exactly. While function z ranges from 0.0 to 1.0 and takes the form of piecewise-linear representation, we can not locate feature lines directly by finding the zero crossings of function z. More, the set of those locations at which the value of function z approaches zero (i.e., smaller than a threshold ϵ) forms a region, that is ROI. Therefore, it is natural to identify feature lines as the medial axis (or the skeleton) of ROI. In this paper, we compute the medial axis of ROI by tracing the valley lines of function z.

4.1. Valleys of functions

In image processing and computer vision, the valleys of a smooth function of two variables is a set of curves whose points are, loosely speaking, local minima in at least one dimension. Let $g(\bar{\mathbf{x}}) : \mathbb{R}^2 \to \mathbb{R}$ be a C^2 function, and denote the Jacobi matrix and the Hessian matrix of function g at $\bar{\mathbf{x}}$ as $Dg(\bar{\mathbf{x}})$ and $D^2g(\bar{\mathbf{x}})$ respectively. Assume $\mathbf{u}_1$ and $\mathbf{u}_2$ are the eigenvectors of D^2g and form a unit orthonormal set, i.e., $D^2g\,\mathbf{u}_1 = \lambda_1\mathbf{u}_1$ and $D^2g\,\mathbf{u}_2 = \lambda_2\mathbf{u}_2$ with $\lambda_1 \geqslant \lambda_2$. A local minimum point $\bar{\mathbf{x}}$ occurs when $[\mathbf{u}_1, \mathbf{u}_2]^T Dg(\bar{\mathbf{x}}) = \mathbf{0}$ and $\lambda_1 \geqslant \lambda_2 \geqslant 0$. Rather than testing for a local minimum in all 2 directions $\{\mathbf{u}_1, \mathbf{u}_2\}$, it is possible to restrict attention to only 1 direction $\mathbf{u}_1$. A point $\bar{\mathbf{x}}$ is called a 1-dimensional *valley point* or *ravine point* [15,20] of function g if $\mathbf{u}_1^T Dg(\bar{\mathbf{x}}) = \mathbf{0}$ and $\lambda_1 \geqslant 0$. Note that $\mathbf{u}_1^T Dg(\bar{\mathbf{x}}) = \mathbf{0}$ is an equation in two unknowns which, by the implicit function theorem, typically has solutions of 1-dimensional manifolds. Thus, the set of valleys points makes up valley lines.

The above definition can be generalized to function $z(\mathbf{x})$ defined on the surface $\mathcal{S}$, that is valleys in the Riemannian geometry. Let $\mathbf{x} : \mathbb{R}^2 \to \mathcal{S} \subset \mathbb{R}^3$ be a parameterization of the surface $\mathcal{S}$, and denotes as $\mathbf{x}(\bar{\mathbf{x}})$. We redefine function $z(\mathbf{x})$ as $z \circ \mathbf{x}(\bar{\mathbf{x}})$ that is a mapping from $\mathbb{R}^2$ to $\mathbb{R}$. Then, the previous definition of valley point can be applied to $z(\mathbf{x})$. Since canyon function $z(\mathbf{x})$ is an indicator of the discontinuity set Γ and behaves like distance function in the neighborhood of Γ, valley lines of $z(\mathbf{x})$ form the medical axis of ROI. For rigorous analysis of the close relationships between the medical axis, distance function singularity, and valley lines, refer to [65-67, 15].

4.2. Quadratic polynomials fitting

To obtain the expression of $z \circ \mathbf{x}(\bar{\mathbf{x}})$, we usually need a parameterization $\mathbf{x}(\bar{\mathbf{x}})$ of the polygonal mesh $\mathcal{M}$. But, mesh parameterization is not a trivial task, especially for arbitrary topological type [41]. In this paper, we directly approximate $z \circ \mathbf{x}(\bar{\mathbf{x}})$ by locally fitting with a bivariate quadratic polynomial. Our method is similar to [68–71], in which Lai, Liang, Zhao, and Wang et al. presented methods to extract local intrinsic coordinate system and approximate differential operators on the manifold discretely without parametrization or connection information. Let $\mathbf{v}_i$ be a vertex of the mesh $\mathcal{M}$, and $NV(i)$ is the set of 1-ring neighborhood vertices of $\mathbf{v}_i$. For each $\mathbf{v}_j \in NV(i)$, project it onto the tangent plane P defined by the unit normal vector $\mathbf{n}_i$ of $\mathbf{v}_i$. Here, we introduce a local orthonormal coordinate system $\{\mathbf{o}; \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ with its origin $\mathbf{o}$ at $\mathbf{v}_i$ and the axe $\mathbf{e}_3 = \mathbf{n}_i$ as shown in Fig. 4.1(a). The axes $\mathbf{e}_1$ and $\mathbf{e}_2$ can be chosen arbitrarily only

if they satisfy the requirement of unit orthonormal basis. Under the local coordinate system, the projection $\tilde{\mathbf{v}}_j$ of $\mathbf{v}_j$ onto P is computed by

$$\xi_j = (\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{e}_1, \quad \eta_j = (\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{e}_2.$$

Our desired quadratic polynomial has the following form

$$z(\xi, \eta) = a_{11}\xi^2 + 2a_{12}\xi\eta + a_{22}\eta^2 + a_1\xi + a_2\eta. \tag{4.1}$$

The unknown coefficients are determined by least-squares fitting [72, 73]

$$\min \sum_{j \in NV(i)} \left\| z(\xi, \eta) - \bar{z}_j \right\|^2,$$

where $z_j = z(\mathbf{v}_j)$ and $\bar{z}_j = z_j - z_i$.

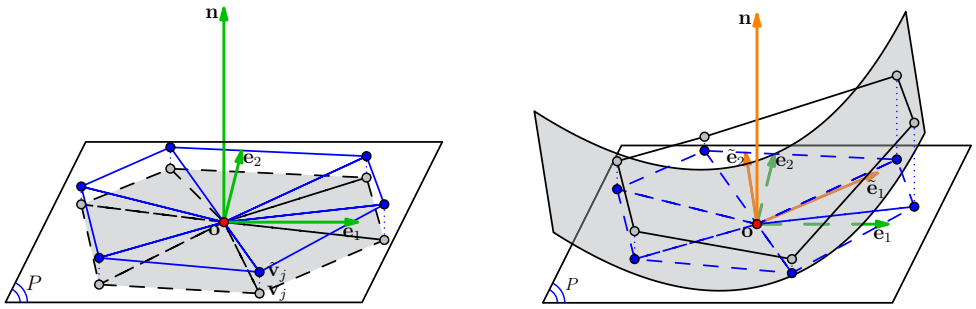


Fig. 4.1. The definition of valley points: (a) local frame; (b) quadratic polynomials fitting.

Then we rewrite the quadratic polynomial (4.1) in matrix form as

$$z(\xi, \eta) = \begin{pmatrix} \xi & \eta \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} \xi & \eta \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

Since the matrix $A = (a_{ij})_{2\times 2}$ is real-symmetric, it has real eigenvalues and eigenvectors. More, A is diagonalizable. Let λ_1 and λ_2 be eigenvalues of A with associated eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$ respectively, where $\lambda_1 \geqslant \lambda_2$ and $\{\mathbf{u}_1, \mathbf{u}_2\}$ forms a unit orthonormal set. Assume $\{\mathbf{o}; \tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, \tilde{\mathbf{e}}_3\}$ is a new local orthonormal coordinate system with its origin $\mathbf{o}$ at $\mathbf{v}_i$ and the axes $\tilde{\mathbf{e}}_1 = (\mathbf{e}_1, \mathbf{e}_2)\mathbf{u}_1$, $\tilde{\mathbf{e}}_2 = (\mathbf{e}_1, \mathbf{e}_2)\mathbf{u}_2$ and $\tilde{\mathbf{e}}_3 = \mathbf{n}_i$ as shown in Fig. 4.1(b). Now, the quadratic polynomial can be reformulated as

$$z(\bar{x}, \bar{y}) = \begin{pmatrix} \bar{x} & \bar{y} \end{pmatrix} \begin{pmatrix} \lambda_1 & \\ & \lambda_2 \end{pmatrix} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \begin{pmatrix} \bar{x} & \bar{y} \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix},$$

where $\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{pmatrix}^{-1} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$.

Because of

$$z(\bar{x}, \bar{y}) = \lambda_1 \left(\bar{x} + \frac{\mu_1}{2\lambda_1} \right)^2 + \lambda_2 \left(\bar{y} + \frac{\mu_2}{2\lambda_2} \right)^2 - \frac{\mu_1^2}{4\lambda_1} - \frac{\mu_2^2}{4\lambda_2}, \tag{4.2}$$

we should divide μ_1 by λ_1 so as to eliminate the negative impact of local parameterization:

$$\mu_1 \leftarrow \frac{\mu_1}{2\lambda_1}.$$

In fact, μ_1 is the partial derivative of $z(\bar{x},\bar{y})$ with respect to $\bar{x}$ at $\mathbf{o}$ that is close related to local parameterization; while, $\frac{-\mu_1}{2\lambda_1}$ indicates the position of $\bar{x}$ that $z(\bar{x},\bar{y})$ attains its maximum or minimum for fixed $\bar{y}$. In addition, $\tilde{\mathbf{e}}_1^i$ and $\tilde{\mathbf{e}}_1^j$ of two adjacent vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ may not be determined coherently, we need to flip $\mathbf{u}_1^j$ if the angle between them is obtuse:

$$\tilde{\mathbf{e}}_1^j \leftarrow -\tilde{\mathbf{e}}_1^j, \quad \mu_1^j \leftarrow -\mu_1^j.$$

4.3. Feature lines tracing

Due to the piecewise-linear representation of mesh $\mathcal{M}$ and function z, a tracing method proposed in [20] is adapted to find out valley lines. For each mesh edge $[\mathbf{v}_i,\mathbf{v}_j]$, we check whether it contains a valley point or not. Since we have

$$Dz(\mathbf{v}_i) = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}, \quad D^2 z(\mathbf{v}_i) = \begin{pmatrix}\lambda_1 & \\ & \lambda_2\end{pmatrix},$$

1-dimensional valley points can be characterized by

$$\mu_1 = 0, \quad \lambda_1 \geqslant 0.$$

In order to verify whether μ_1 has a zero-crossing on the edge $[\mathbf{v}_i,\mathbf{v}_j]$, we check the following condition:

$$\mu_1^i\,\mu_1^j \leqslant 0. \tag{4.3}$$

It follows from (4.2) that function $z(\bar{x},\bar{y})$ attains a minimum (restricting to $\bar{x}$-direction) at

$$\mathbf{v}_i - \frac{\mu_1}{2\lambda_1}\tilde{\mathbf{e}}_1,$$

if $\lambda_1 \geqslant 0$. So, we apply the following tests:

$$\mu_1^i\big[(\mathbf{v}_j-\mathbf{v}_i)\cdot\tilde{\mathbf{e}}_1^i\big] \leqslant 0, \quad \mu_1^j\big[(\mathbf{v}_i-\mathbf{v}_j)\cdot\tilde{\mathbf{e}}_1^j\big] \leqslant 0, \tag{4.4}$$

to determine whether the edge $[\mathbf{v}_i,\mathbf{v}_j]$ contains a point satisfying the minimal condition. If both (4.3) and (4.4) are satisfied, we use linear interpolation

$$\mathbf{v} = \frac{|\mu_1^j|\mathbf{v}_i + |\mu_1^i|\mathbf{v}_j}{|\mu_1^j| + |\mu_1^i|}$$

or nearest-neighbor interpolation

$$\mathbf{v} = \begin{cases} \mathbf{v}_i, & |\mu_1^i| < |\mu_1^j| \\ \mathbf{v}_j, & |\mu_1^i| \geqslant |\mu_1^j| \end{cases}$$

to compute a valley point $\mathbf{v}$ on the edge $[\mathbf{v}_i,\mathbf{v}_j]$.

Once all valleys points are determined, we are ready to construct valley lines. If two valley points are detected on edges of a triangle in the polygonal mesh, they are connected by a straight segment. If all three edges of a triangle contain valley points, valley points from neighboring triangles are connected with the centroid of this triangle.

As described in [20, 25], valley lines detected on complex shapes with many small wrinkles usually have poor connectivity. To reduce the small fragmentation of feature lines, we use a similar measurement of saliency

$$T(C) = \int_0^L \frac{1}{1+z}\mathrm{d}s \approx \sum_i \frac{1}{1+\frac{z(\mathbf{v}_i)+z(\mathbf{v}_{i+1})}{2}} \parallel \mathbf{v}_{i+1}-\mathbf{v}_i \parallel$$

as a threshold for a filtering procedure.

4.4. ROI selection

Feature lines are located in the ROI, where the values of function z are smaller than a threshold ϵ. Under a graphic environment, the end-user may interactively select the ROI by setting ϵ. Then we only perform quadratic polynomials fitting and feature lines tracing on the mesh vertices pertaining to the ROI. This simple strategy saves lots of computation, and provides an intuitive way to control the procedure of feature lines detection.

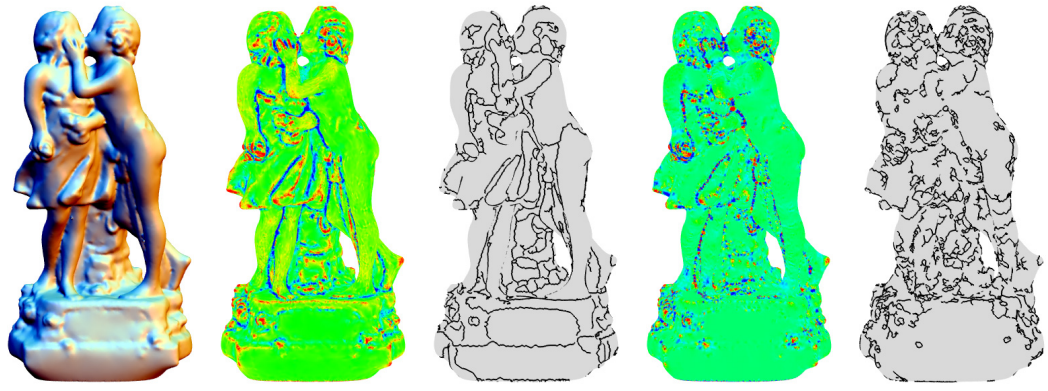
5. Results and Discussion

In this section, we discuss the choice of measure functions and parameters, and present results of experiments with the algorithms described in the preceding sections.

5.1. Measure functions

In our framework, there is no special limit of the choice of measure function u_0. The value of u_0 can be computed from polygonal meshes or loaded from data. It highly depends on the specific application at hand.

As described in [74], normal curvature is the length magnification of a small arc under Gaussian map. It is very like the dihedral angle at a polyhedral edge. If we average the magnification over all possible directions, mean curvature is obtained, just as the Euler's formula stated. Since creases are curves on a surface along which the surface normal varies sharply, mean curvature can be used to detect them. Although Gaussian curvature can also be employed, we observed that the results of using mean curvature normally are better as shown in Fig. 5.1. In this paper, we choose mean curvature to serve as the measurement of feature lines by default if no additional declaration is made.



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

Fig. 5.1. Comparison between using mean curvature (b)-(c) and Gaussian curvature (d)-(e).

A silhouette is the image of an object or scene consisting of the outline and a featureless interior, and is view-dependent feature. Consider a view of a smooth and closed surface $\mathcal{S}$ from a perspective camera centered at $\mathbf{c}$. Mathematically, the contour generator is defined as the set of points

$$\{\mathbf{p} \in \mathcal{S} \mid \mathbf{n}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{p}) = 0\},$$

where $\mathbf{n}(\mathbf{p})$ is the unit surface normal at $\mathbf{p}$, and $\mathbf{v}(\mathbf{p}) = \mathbf{c} - \mathbf{p}$. From a common viewpoint, the contour generator is a set of disconnected loops on the surface $\mathcal{S}$. The contour consists of the visible portions of these curves, projected onto the image plane. A simple technique for rendering contours is to shade each vertex $\mathbf{v}_i$ using the intensity

$$c_i = \begin{cases} 1.0, & \text{if } \left|\mathbf{n}(\mathbf{v}_i) \cdot \mathbf{v}(\mathbf{v}_i)\right| < \delta, \\ 0.0, & \text{otherwise,} \end{cases} \tag{5.1}$$

where $\delta = \cos(\frac{3\pi}{8})$. Fig. 5.2(b) gives an example. When we set the intensity function c_i as the measure function u_0, the contour can be detected by our variational method as demonstrated in Fig. 5.2(d). If the preceding intensity function c_i is replaced with

$$c_i = \mathbf{n}(\mathbf{v}_i) \cdot \mathbf{v}(\mathbf{v}_i), \tag{5.2}$$

we will obtain feature lines which look like suggestive contours [31]. For an illustration, see Fig. 5.2(e) and 5.2(h).

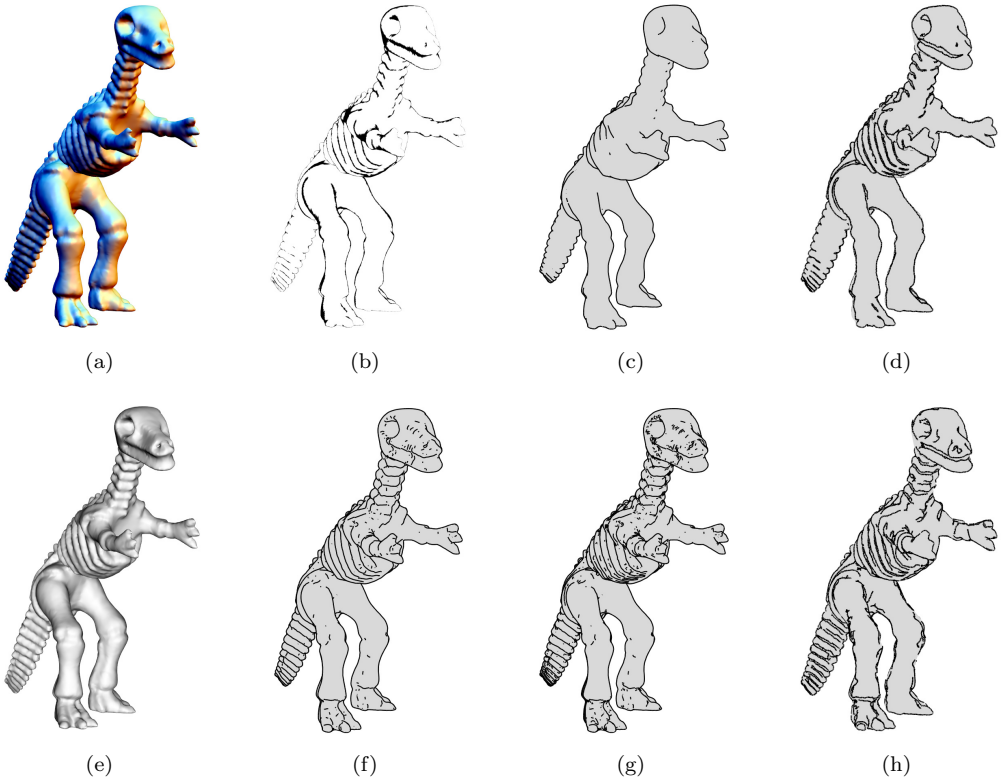
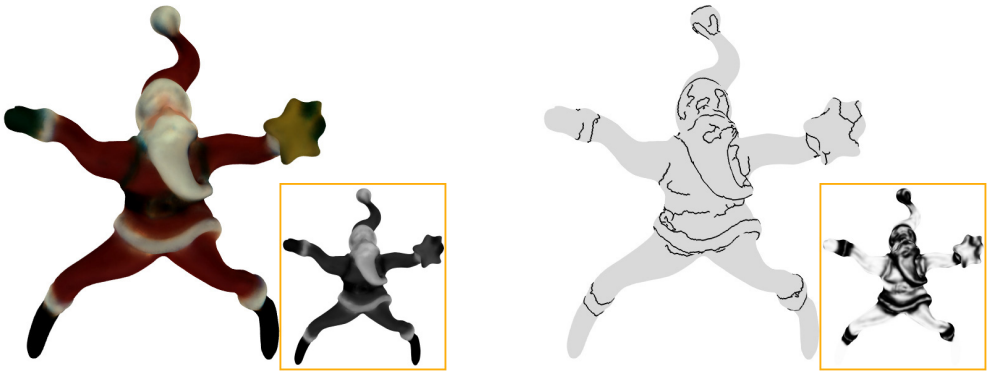


Fig. 5.2. Comparison between our variational approach and state-of-the-art suggestive contours method [75] (RTSC-1.5): (a) dinosaur dataset; (b) intensity function (5.1); (c) contours of RTSC-1.5; (d) contours of ours; (e) intensity function (5.2); (f) contours + suggestive contours of RTSC-1.5; (g) contours + suggestive contours + suggestive highlights of RTSC-1.5; (h) suggestive-like contours of ours.

Generic functions defined on the surface $\mathcal{S}$ can also be utilized, such as lightness function, texture function and so on. Fig. 5.3(a) shows a polygonal mesh acquired by color 3D scanner. On each vertex, we have the RGB values from which lightness functions are obtained as

illustrated in the framed image of Fig. 5.3(a). Feature lines and canyon function are shown in 5.3(b).



(a) (b)

Fig. 5.3. Feature lines detection based on generic functions:
(a) polygonal mesh and lightness function; (b) feature lines and canyon function.

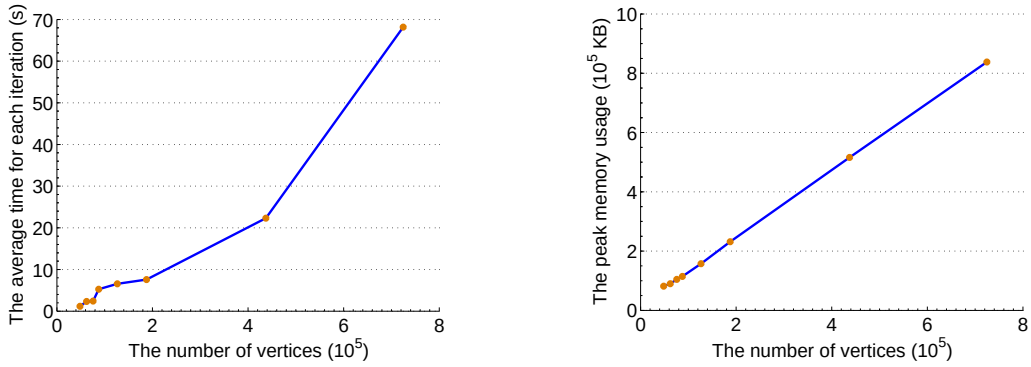
5.2. Performance

In order to provide a roughly estimate of performance, we have tested the presented method with our minimally optimized code on various polygonal meshes ranging from synthetic to measured models. Table 5.1 shows the mesh statistics, parameters, and detailed timings of our experiments measured on a 2.6 GHZ Intel Dual-Core CPU with 2 GB of main memory.

Table 5.1: Performance results on a variety of models. PM denotes the peak memory usage, NI is the number of iterations, and AT is the average time for each iteration. All running times are measured in seconds.

Model	$\|V\|$	$\|F\|$	α	β	γ	ε	ϵ	$T(C)$	PM	NI	AT
Fig. 5.5(c)	48,318	96,632	0.2	0.18	0.5	0.5	0.15	15	81,600K	20	1.18
Fig. 5.5(f)	62,088	124,244	0.2	0.10	0.5	0.5	0.15	10	90,080K	4	2.32
Fig. 5.3(b)	75,781	151,558	0.2	0.20	0.5	0.5	0.42	25	104,696K	6	2.42
Fig. 5.1(c)	87,494	174,996	0.2	12.0	0.5	0.5	0.55	15	114,648K	4	5.27
Fig. 5.2(d)	126,614	253,224	0.2	0.30	0.5	0.5	0.05	15	157,348K	1	6.56
Fig. 5.6(c)	187,644	375,284	0.2	1.00	0.5	0.5	0.15	15	231,680K	5	7.58
Fig. 5.7(b)	437,372	874,740	0.2	2.00	0.5	0.5	0.45	15	516,088K	5	22.32
Fig. 5.7(d)	724,569	1,449,162	0.2	3.00	0.5	0.5	0.25	15	837,916K	5	68.17

From these timings, we can see that the average time for each iteration and the peak memory usage are nearly linear functions of the number of vertices with constant coefficients as depicted in Fig. 5.4. (Notice that the average time for each iteration in Table 5.1 depends on the spectral properties of matrices in (3.10) and (3.11) and the residual norm, because we solve them using iterative solvers.) So, our method can be served as a practical tool for feature lines detection on complex polygonal meshes.



(a) (b)

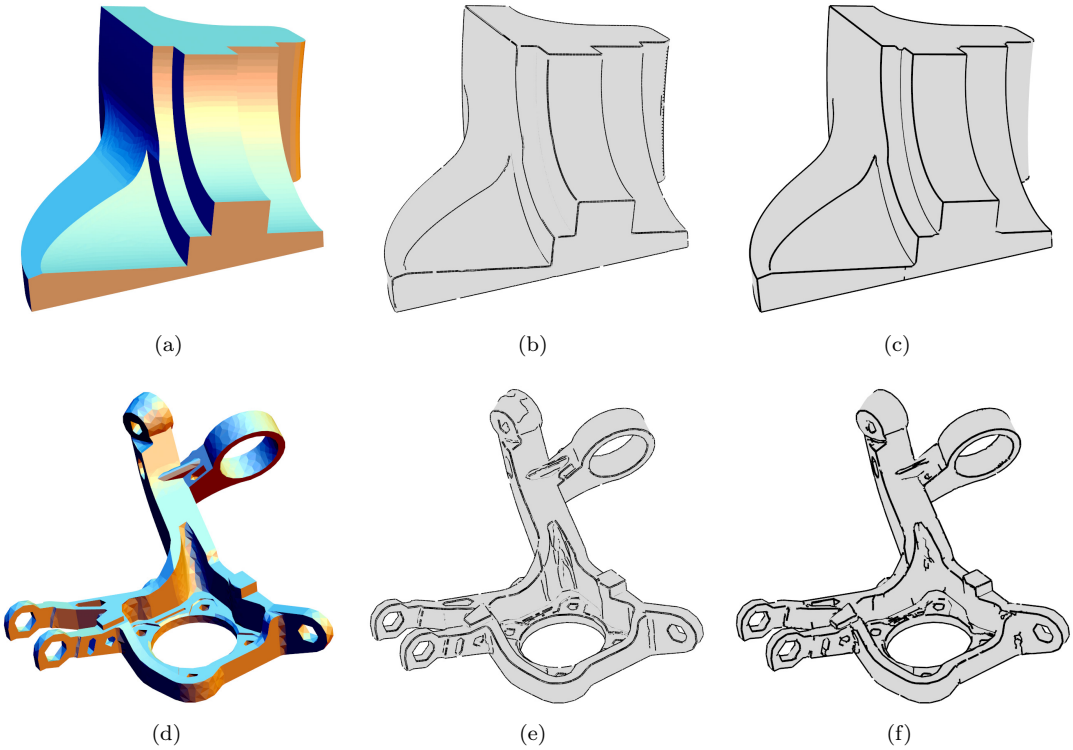
Fig. 5.4. Graphs of performance data: (a) the average time for each iteration and (b) the peak memory usage as functions of the number of vertices.

5.3. Discussion of examples

In order to demonstrate the viability of our variational approach, we provide a comparison with two state-of-the-art feature detection methods: suggestive contours method [31], [32] (the latest version RTSC-1.5 is available at http://gfx.cs.princeton.edu/proj/sugcon/index.html [75]), and crest lines method [23] (the source code Crest is available at http://www.riken.jp/brict/Yoshizawa/Research/Crest.html [76]). For fairness, we run their implementations with default parameters. Fig. 5.2 shows the results of RTSC-1.5 and our variational approach. The outputs of ours are comparable in quality to RTSC-1.5's. Fig. 5.5 indicates that the presented method is well suited for sharp features detection. Since we directly evaluates mean curvature by (3.7) instead of using surface fitting based methods (e.g. [23, 20]) and do not use curvature derivatives, the sharp features detected by our method generally are more accurate, as shown in Fig. 5.5. To be honesty, our current variational approach is slower than RTSC-1.5 and Crest. But, it can provide a unified framework for detecting generic feature lines, such as those in Fig. 5.3, which is not capable for existing approaches.

In order to demonstrate the impact of parameter β on feature lines detection, we provide an example on Isis model as shown in Fig. 5.6. From (3.4a), we know that β indicates the scale level at which smoothing step is being carried out. From (3.4b), we see that β is also the scale parameter for determining the edge set Γ. Thus, β can be used to tune the scale of feature lines detection. To get fine features, we set β a small value; otherwise, a large value should be used. Fig. 5.6 provides an example confirmed it. Other parameters, such as α, γ, ε and so on, may affect the final result too. Although the problem of choosing parameters in an automatically way is worthy of further exploration, most of them can be set as default values for generic models as reported in Table 5.1.

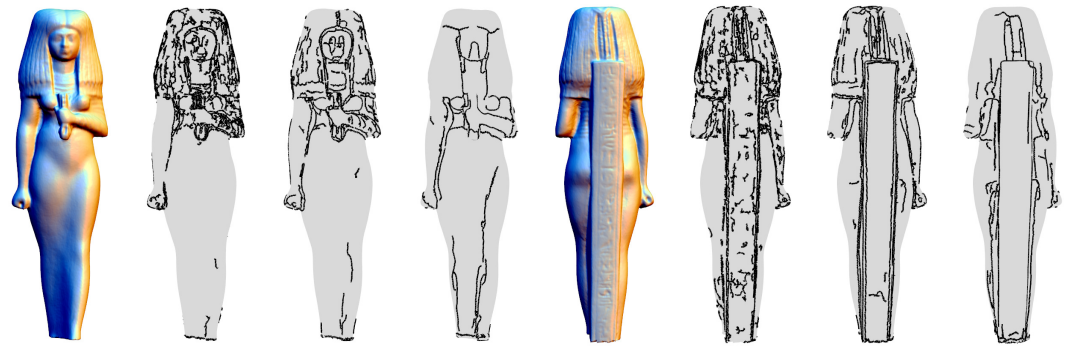
Two complex 3D models of real objects were tested for the purposes of verifying the efficiency and robustness of our algorithms. Fig. 5.7(a) shows the Pierrot model, which has 437,372 vertices and 874,740 triangles. Fig. 5.7(c) shows the dancing children model, which consists of 724,569 vertices and 1,449,162 triangles. The genus of them are 0 and 7 respectively. The detected feature lines are illustrated in Fig. 5.7(b) and Fig. 5.7(d). These examples give us confidence in the ability of the presented method to detect feature lines on general polygonal meshes.



(a) (b) (c)

(d) (e) (f)

Fig. 5.5. Comparison between our variational approach and state-of-the-art crest lines method [23](Crest): (a) fandisk dataset; (b) crest lines of Crest with $T = 1.8$ and $k = 2$, running times = 3.48s; (c) sharp feature lines of ours (d) fusee dataset. (e) crest lines of Crest with $T = 1.2$ and $k = 2$, running times = 4.66s; (f) sharp feature lines of ours.



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

Fig. 5.6. Comparison of using various scale parameter β: (b) and (f) $\beta = 0.1$; (c) and (g) $\beta = 1.0$; (d) and (h) $\beta = 18.0$.

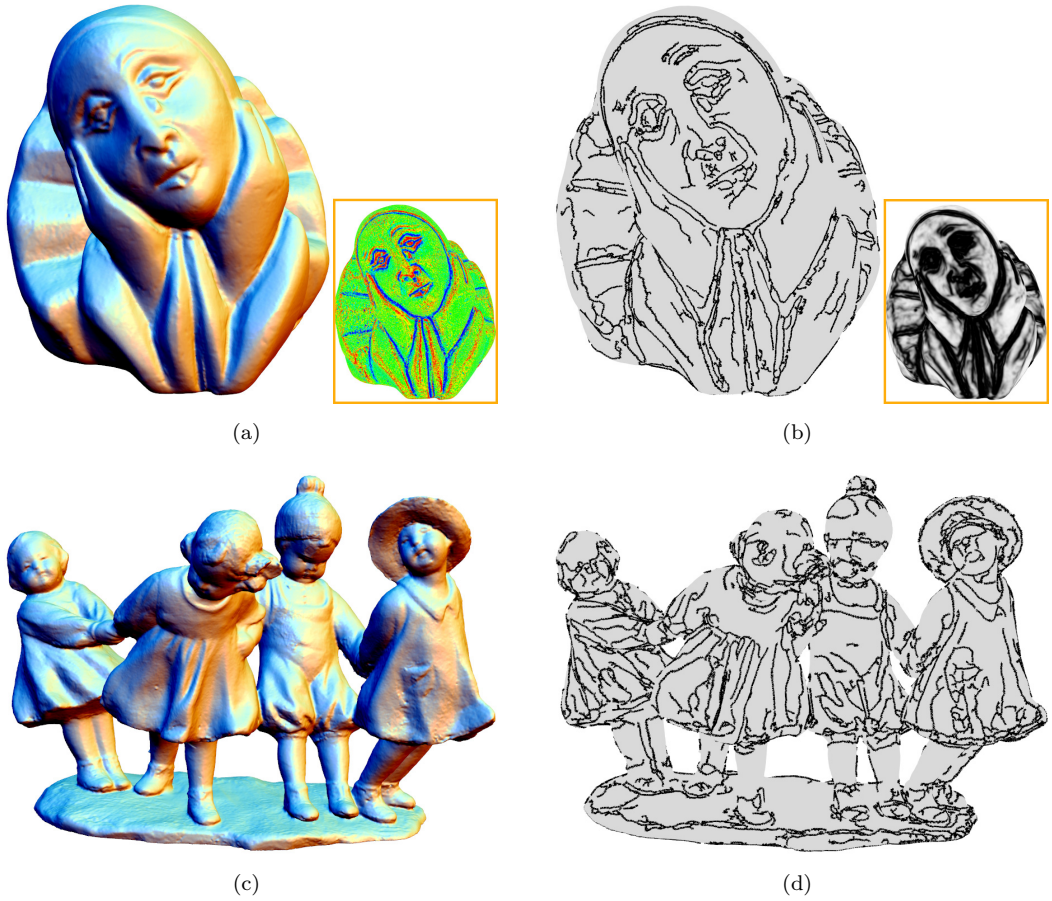


Fig. 5.7. Feature lines detection on complex polygonal meshes: (a)-(b) Pierrot dataset of genus 0 with 437K vertices and 874K triangles; (c)-(d) dancing children dataset of genus 7 with 724K vertices and 1,449K triangles.

6. Conclusions and Future Work

In this paper, we have presented a unified variational framework for detecting generic feature lines on polygonal meshes, along with a practical implementation. To attain this goal, we extend the Mumford-Shah model and valleys of functions to surfaces. Using Γ-convergence method and discrete differential geometry, we discretize the presented variational model to sequential coupled sparse linear systems. Our approach offers several desirable properties: various types of feature lines can be detected only by changing the measure function; flexible and intuitive control over the detecting procedure is available; the presented algorithms are straightforward to implement. The effectiveness, robustness and visual quality have been evidenced by numerical examples.

In the future we plan to investigate 3D mesh segmentation using a similar variational framework, which is an important and challenging problem. We will consider the invention of new measure functions to detect other types of feature lines. Another interesting direction for future research is to redesign our algorithms so that computations can be performed on the GPUs for

accelerating. Farther reaching future work includes theoretic analysis on the Mumford-Shah surface model. Although the correctness of the Γ-convergence approximation of the Mumford-Shah model on surface $\mathcal{S}$ seems to be obvious and has been validated by our experiments, a rigorous proof still be absent at present. It is worth to be investigated in the future.

Appendix

According to the divergence theorem, we have

$$\int_{\Omega} \mathrm{div}_s(\mathbf{u})\ \mathrm{d}\mathcal{A} = \int_{\partial\Omega} \langle \mathbf{u}, \mathbf{n}_c \rangle\ \mathrm{d}s,$$

where $\mathbf{n}_c$ is the outward unit normal vector field along $\partial\Omega$, and s is the arc length parameter of $\partial\Omega$. If Ω equals to $A_M(\mathbf{v}_i)$, then we get

$$\mathbf{n}_c\ \mathrm{d}s = -\nabla\varphi_{il}|T_l|,$$

where T_l is a triangle in the 1-ring neighborhood faces of $\mathbf{v}_i$. Thus the integral can be rewritten as

$$\int_{A_M(\mathbf{v}_i)} \mathrm{div}_s(\mathbf{u})\ \mathrm{d}\mathcal{A} = -\sum_{T_l \in NT(i)} \mathbf{u}\Big|_{T_l} \cdot \nabla\varphi_{il}|T_l|.$$

Accordingly, the discrete div_s operator can be defined as

$$\mathrm{div}_s(\mathbf{u})\Big|_{\mathbf{v}_i} = \frac{-1}{A_M(\mathbf{v}_i)} \sum_{T_l \in NT(i)} \mathbf{u}\Big|_{T_l} \cdot \nabla_s\varphi_{il}|T_l|, \tag{.1}$$

which is compatible with (3.6). In fact, we know

$$\nabla_s f(\mathbf{v})\Big|_{T_l} = f(\mathbf{v}_i)\nabla_s\varphi_{il} + f(\mathbf{v}_j)\nabla_s\varphi_{jl} + f(\mathbf{v}_k)\nabla_s\varphi_{kl}$$

and

$$\begin{aligned}
\nabla_s\varphi_{il} \cdot \nabla_s\varphi_{il} &= \frac{\cot\theta_k + \cot\theta_j}{2|T_l|},\\
\nabla_s\varphi_{il} \cdot \nabla_s\varphi_{kl} &= -\frac{\cot\theta_j}{2|T_l|},\\
\nabla_s\varphi_{il} \cdot \nabla_s\varphi_{kj} &= -\frac{\cot\theta_k}{2|T_l|},
\end{aligned}$$

where θ_k and θ_j are the incident angles of vertices $\mathbf{v}_k$ and $\mathbf{v}_j$ respectively as shown in Fig. 3.1(a). If we let $\mathbf{u}$ be equal to $\nabla_s f(\mathbf{v}_i)$ in (.1), we obtain (3.6) immediately.

Acknowledgment. We would like to thank Szymon Rusinkiewicz and Doug DeCarlo et al. for making suggestive contours RTSC-1.5 [75] available; Shin Yoshizawa for making crest lines Crest [76] available. The models used in this paper are courtesy of the AIM@SHAPE shape repository, Cyberware and FastSCAN. We are very grateful to the anonymous reviewers for their detailed constructive comments. This work was supported by NRF-2007 IDM-IDM002-010 under Nangyang Technological university, Singapore. The work of W.H. Tong was partially supported by the NSF of China (Nos. 10901149, 11171322 and 11571338) and by the Fundamental Research Funds for the Central Universities.

References

[1] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer and W. Stuetzle, Piecewise smooth surface reconstruction, SIGGRAPH '94: ACM SIGGRAPH 1994 Papers, 295–302, 1994.

[2] L.P. Kobbelt, M. Botsch, U. Schwanecke and H.P. Seidel, Feature sensitive surface extraction from volume data, SIGGRAPH '01: ACM SIGGRAPH 2001 Papers, 57–66, 2001.

[3] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.P. Seidel, Multi-level partition of unity implicits, SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, 463–470, 2003.

[4] T.R. Jones, F. Durand and M. Desbrun, Non-iterative, feature-preserving mesh smoothing, SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, 943–949, 2003.

[5] S. Fleishman, I. Drori and D. Cohen-Or, Bilateral mesh denoising, SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, 950–953, 2003.

[6] C. Tomasi and R. Manduchi, Bilateral filtering for gray and color images, ICCV '98: Proceedings of the Sixth International Conference on Computer Vision, 839–846, 1998.

[7] S. Fleishman, D. Cohen-Or and C.T. Silva, Robust moving least-squares fitting with sharp features, SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, 544–552, 2005.

[8] H. Fan, Y. Yu and Q. Peng, Robust feature-preserving mesh denoising based on consistent subneighborhoods, *IEEE T. Vis. Comput. Gr.*, **16**:2 (2010), 312–324.

[9] T. Ju, F. Losasso, S. Schaefer and J. Warren, Dual contouring of hermite data, SIGGRAPH '02: ACM SIGGRAPH 2002 Papers, 339–346, 2002.

[10] M. Attene, B. Falcidieno, J. Rossignac and M. Spagnuolo, Sharpen&bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling, *IEEE T. Vis. Comput. Gr.*, **11**:2 (2005), 181–192.

[11] C.C.L. Wang, Bilateral recovering of sharp edges on feature-insensitive sampled meshes, *IEEE T. Vis. Comput. Gr.*, **12**:4 (2006), 629–639.

[12] C.Y. Chen and K.Y. Cheng, A sharpness-dependent filter for recovering sharp features in repaired 3D mesh models, *IEEE T. Vis. Comput. Gr.*, **14**:1 (2008), 200–212.

[13] D. Bommes, H. Zimmer and L. Kobbelt, Mixed-integer quadrangulation, SIGGRAPH '09: ACM SIGGRAPH 2009 papers, 1–10, 2009.

[14] I.R. Porteous, Geometric Differentiation for the Intelligence of Curves and Surfaces, Cambridge University Press, 2nd ed., 2001.

[15] D. Eberly, Ridges in Image and Data Analysis, Kluwer Academic Publishers, 1996.

[16] G.G. Gordon, Face recognition based on depth maps and surface curvature, Geometric Method in Computer Vision, Proc. SPIE Vol. 1570, 234–247, 1991.

[17] M. Hosaka, Modeling of Curves and Surfaces in CAD/CAM, Springer-Verlag Press, 1992.

[18] A.G. Belyaev, Y. Ohtake and K. Abe, Detection of ridges and ravines on range images and triangular meshes, Vision Geometry IX, Proc. SPIE Vol. 4117, 146–154, 2000.

[19] K. Hildebrandt, K. Polthier and M. Wardetzky, Smooth feature lines on surface meshes, SGP '05: Proceedings of the third Eurographics symposium on Geometry processing, 85–90, 2005.

[20] Y. Ohtake, A. Belyaev and H.P. Seidel, Ridge-valley lines on meshes via implicit surface fitting, SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, 609–612, 2004.

[21] Y. Ohtake, A. Belyaev and M. Alexa, Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing, Proceedings of the third Eurographics symposium on Geometry processing, 149–158, 2005.

[22] S.K. Kim and C.H. Kim, Finding ridges and valleys in a discrete surface using a modified mls approximation, *Comput.-Aided Des.*, **38**:2 (2006), 173–180.

[23] S. Yoshizawa, A. Belyaev and H.P. Seidel, Fast and robust detection of crest lines on meshes, SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling, 227–232, ACM, 2005.

[24] S. Yoshizawa, A. Belyaev, H. Yokota and H.P. Seidel, Fast and faithful geometric algorithm for detecting crest lines on meshes, PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, 231–237, 2007.
[25] S. Yoshizawa, A. Belyaev, H. Yokota and H.P. Seidel, Fast, robust, and faithful methods for detecting crest lines on meshes, *Comput. Aided Geom. Design*, **18**:8 (2008), 545–560.
[26] Y.K. Lai, Q.Y. Zhou, S.M. Hu, J. Wallner and H. Pottmann, Robust feature classification and editing, *IEEE T. Vis. Comput. Gr.*, **13**:1 (2007), 34–45.
[27] B. Gooch and A.A. Gooch, Non-Photorealistic Rendering, A K Peters Press, 2001.
[28] T. Strothotte and S. Schlechtweg, Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation, Morgan Kaufmann, 2002.
[29] L. Markosian, M.A. Kowalski, D. Goldstein, S.J. Trychin, J.F. Hughes and L.D. Bourdev, Real-time nonphotorealistic rendering, SIGGRAPH '97: ACM SIGGRAPH 1997 papers, 415–420, 1997.
[30] A. Hertzmann and D. Zorin, Illustrating smooth surfaces, SIGGRAPH '00: ACM SIGGRAPH 2000 papers, 517–526, 2000.
[31] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz and A. Santella, Suggestive contours for conveying shape, SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, 848–855, 2003.
[32] D. DeCarlo and S. Rusinkiewicz, Highlight lines for conveying shape, NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering, 63–70, 2007.
[33] S. Rusinkiewicz, F. Cole, D. DeCarlo and A. Finkelstein, Line drawings from 3d models, SIGGRAPH '08: ACM SIGGRAPH 2008 classes, 1–356, 2008.
[34] T.F. Chan and J. Shen, Image Processing and Analysis: Varational, PDE, Wavelet and Stochastic Mehtods, SIAM, Philadephia, 2005.
[35] G. Aubert and P. Kornprobst, Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, Springer-Verlag, New York, 2nd edition, 2006.
[36] A. Dufour, N. Vincent and A. Genovesio, 3D Mumford-Shah based active mesh, Progress in Pattern Recognition, Image Analysis and Applications, 208–217, Springer-Verlag Press, 2006.
[37] H. Jin, A.J. Yezzi and S. Soatto, Mumford-Shah on the move: region-based segmentation on deforming manifolds with application to 3-d reconstruction of shape and appearance from multi-view images, *J. Math. Imaging Vis.*, **29**:2-3 (2007), 219–234.
[38] M. Droske and A. Bertozzi, Higher-order feature-preserving geometric regularization, *SIAM J. Imaging Sci.*, **3**:1 (2010), 21–51.
[39] A. Delaunoy, K. Fundana, E. Prados and A. Heyden, Convex multi-region segmentation on manifolds, ICCV'09: the 12th IEEE International Conference on Computer Vision, 662–669, 2009.
[40] J.D. Foley, A.V. Dam, S.K. Feiner and J.F. Hughes, Computer Graphics: Principles and Practice in C, Addison-Wesley, 2nd edition, 1995.
[41] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff and C. Rossl, Geometric modeling based on polygonal meshes, ACM SIGGRAPH Course Notes, 2007.
[42] A. Hatcher, Algebraic Topology, Cambridge University Press, 2002.
[43] J.R. Munkres, Elements of Algebraic Topology, Westview Press, 1996.
[44] M. Kass, A. Witkin and D. Terzopoulos, Snakes: active contour models, *Int. J. Comput. Vis.*, **1**:4 (1988), 321–331.
[45] S. Geman and D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, Readings in computer vision: issues, problems, principles, and paradigms, 564–584, Morgan Kaufmann, 1987.
[46] D. Mumford and J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, *Comm. Pure Appl. Math.*, **42**:5 (1989), 577–685.
[47] I. Chavel, Riemannian Geometry: A Modern Introduction, Cambridge University Press, 2nd edition, 2006.

[48] M.P.do Carmo, Riemannian Geometry, Birkhüser Boston, 1992.
[49] T.F. Chan and L.A. Vese, Active contours without edges, *IEEE Trans. Image Process.*,**10**:2(2001), 266–277.
[50] L.A. Vese and T.F. Chan, A multiphase level set framework for image segmentation using the Mumford and Shah model, *Int. J. Comput. Vis.*, **50**:3 (2002), 271–293.
[51] J. Lie, O.M. Lysaker and X.C. Tai, A variant of the level set method and applications to image segmentation, *Math. Comp.*, **75**:255 (2006), 1155–1174.
[52] E. Bae, J. Yuan and X.C. Tai, Global minimization for continuous multiphase partitioning problems using a dual approach, *Int. J. Comput. Vis.*, **92**:1 (2011), 112–129.
[53] J. Yuan, E. Bae, X.C. Tai and Y. Boykov, A study on continuous max-flow and min-cut approaches, 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2217–2224, 2010.
[54] E. Bae, J. Yuan, X.C. Tai and Y. Boykov, A fast continuous max-flow approach to non-convex multilabeling problems, International Dagstuhl Seminar 2011: Efficient Algorithms for Global Optimization Methods in Computer Vision, 134–154, 2014.
[55] L. Ambrosio and V.M. Tortorelli, Approximation of functional depending on jumps by elliptic functional via Γ-convergence, *Comm. Pure Appl. Math.*, **43**:8 (1990), 999–1036.
[56] J. Shah, Segmentation by nonlinear diffusion, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Computer Graphics Proceedings, Annual Conference Series, 202–207, 1991.
[57] J. Shah, Segmentation by nonlinear diffusion, II, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Computer Graphics Proceedings, Annual Conference Series, 202–207, 1992.
[58] R. March, Visual reconstruction with discontinuities using variational methods, *Image Vision Comput.*, **10**:1 (1992), 30–38.
[59] R. March, A variational method for the recovery of smooth boundaries, *Image Vision Comput.*, **15**:9 (1997), 705–712.
[60] M. Meyer, M. Desbrun, P. Schroder and A.H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, Proceedings of VisMath, Berlin, Germany, 2002.
[61] J.W. Thomas, Numerical Partial Differential Equations, Springer-Verlag Press, 1995.
[62] M. Desbrun, M. Meyer, P. Schröder and A.H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, SIGGRAPH '99: ACM SIGGRAPH 1999 papers, 317–324, 1999.
[63] T. Skalicky, Laspack reference manual, http://www.mgnet.org/mgnet/Codes/laspack/html/ laspack.html, 1995.
[64] S. Toledo, D. Chen and V. Rotkin, Tacus: A library of sparse linear solvers, http://www.tau.ac.il/ stoledo/taucs, 2003.
[65] A.G. Belyaev, E.V. Anoshkina and T.L. Kunii, Ridges, ravines, and singularities, A.T. Fomenko and T.L. Kunii, editors, Topological Modeling for Visualization, 375–383, Springer-Verlag Press, 1997.
[66] E.V. Anoshkina, A.G. Belyaev and T.L. Kunii, Detection of ridges and ravines based on caustic singularities, *Int. J. Shape Model.*, **1**:1 (1994), 13–22.
[67] K. Watanabe and A.G. Belyaev, Detection of salient curvature features on polygonal surfaces, *Comput. Graphics Forum*, **20**:3 (2001), 385–392.
[68] J. Liang, R. Lai, T.W. Wong and H. Zhao, Geometric understanding of point clouds using laplace-beltrami operator, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 214–221, 2012.
[69] J. Liang and H. Zhao, Solving partial differential equations on point clouds, *SIAM J. Sci. Comput.*, **35**:3 (2013), 1461–1486.
[70] R. Lai, J. Liang and H. Zhao, A local mesh method for solving pdes on point clouds, *Inverse Probl. Imaging*, **7**:3 (2013), 737–755.

[71] R. Wang, Z. Yang, L. Liu and Q. Chen, Discretizing laplacecbeltrami operator from differential quantities, *Commun. Math. Stat.*, **1**:3 (2013), 331–350.
[72] D. Kincaid and W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, American Mathematical Society, 3rd ed., 2002.
[73] G.H. Golub and C.F.V. Loan, Matrix Computation, The Johns Hopkins University Press, 4th ed., 2013.
[74] J.J. Koenderink, Solid Shape, The MIT Press, 1990.
[75] A. Finkelstein, S. Rusinkiewicz, M. Burns, J. Klawe, D. DeCarlo and A. Santella, Suggestive contours, http://gfx.cs.princeton.edu/proj/sugcon/index.html, 2007.
[76] S. Yoshizawa, Fast and robust detection of crest lines on meshes, http://www.riken.jp/brict/Yoshizawa/Research/Crest.html, 2011.