Consistent Computation of First- and Second-Order Differential Quantities for Surface Meshes

Xiangmin Jiao*
Dept. of Applied Mathematics & Statistics
Stony Brook University

Hongyuan Zha†
College of Computing
Georgia Institute of Technology

Abstract

Differential quantities, including normals, curvatures, principal directions, and associated matrices, play a fundamental role in geometric processing and physics-based modeling. Computing these differential quantities *consistently* on surface meshes is important and challenging, and some existing methods often produce inconsistent results and require *ad hoc* fixes. In this paper, we show that the computation of the gradient and Hessian of a height function provides the foundation for consistently computing the differential quantities. We derive simple, *explicit* formulas for the transformations between the first- and second-order differential quantities (i.e., normal vector and principal curvature tensor) of a smooth surface and the first- and second-order derivatives (i.e., gradient and Hessian) of its corresponding height function. We then investigate a general, flexible numerical framework to estimate the derivatives of the height function based on local polynomial fittings formulated as weighted least squares approximations. We also propose an iterative fitting scheme to improve accuracy. This framework generalizes polynomial fitting and addresses some of its accuracy and stability issues, as demonstrated by our theoretical analysis as well as experimental results.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Algorithms, Design, Experimentation

Keywords: normals, curvatures, principal directions, shape operators, height function, stabilities

1 Introduction

Computing normals and curvatures is a fundamental problem for many geometric and numerical computations, including feature detection, shape retrieval, shape registration or matching, surface fairing, surface mesh adaptation or remeshing, front tracking and moving meshes. In recent years, a number of methods have been introduced for the computation of the differential quantities (see e.g., [Taubin 1995; Meek and Walton 2000; Meyer et al. 2002; Cazals and Pouget 2005]). However, some of the existing methods may produce inconsistent results. For example, when estimating the mean curvature using the cotangent formula and estimating the Gaussian curvature using the angle deficit [Meyer et al. 2002], the principal curvatures obtained from these mean and Gaussian curvatures are not guaranteed to be real numbers. Such inconsistencies often require *ad hoc* fixes to avoid crashing of the code, and their effects on the accuracy of the applications are difficult to analyze.

The ultimate goal of this work is to investigate a mathematically sound framework that can compute the differential quantities *consistently* (i.e., satisfying the intrinsic constraints) with provable *convergence* on general surface meshes, while being flexible and easy to implement. This is undoubtly an ambitious goal. Although we may have not fully achieved the goal, we make some contributions toward it. First, using the singular value decomposition [Golub and Van Loan 1996] of the Jacobian matrix, we derive explicit formulas for the transformations between the first- and second-order differential quantities of a smooth surface (i.e., normal vector and principal curvature tensor) and the first- and second-order derivatives of its corresponding height function (i.e., gradient and Hessian). We also give the explicit formulas for the transformations of the gradient and Hessian under a rotation of the coordinate system. These transformations can be obtained without forming the shape operator and the associated computation of its eigenvalues or eigenvectors. We then reduce the problem, in a consistent manner, to the computation of the gradient and Hessian of a function in two dimensions, which can then be computed more readily by customizing classical numerical techniques.

Our second contribution is to develop a general and flexible method for differentiating a height function, which can be viewed as a generalization of the polynomial fitting of osculating jets [Cazals and Pouget 2005]. Our method is based on the Taylor series expansions of a function and its derivatives, then solved by a weighted least squares formulation. We also propose an iterative-fitting method that improves the accuracy of fittings. Furthermore, we address the numerical stability issues by a systematic point-selection strategy and a numerical solver with safeguard. We present experimental results to demonstrate the accuracy and stability of our approach for fittings up to degree six.

The remainder of this paper is organized as follows. Section 2 reviews some related work on the computation of differential quantities of discrete surfaces. Section 3 analyzes the stability and consistency of classical formulas for computing differential quantities for continuous surfaces and establishes the theoretical foundation for consistent computations using a height function. Section 4 presents a general framework and a unified analysis for computing the gradient and Hessian of a height function based on polynomial fittings, including an iterative-fitting scheme. Section 5 presents some experimental results to demonstrate the accuracy and stability of our approach. Section 6 concludes the paper with a discussion on future research directions.

2 Related Work

Many methods have been proposed for computing or estimating the first- and second-order differential quantities of a surface. In recent years, there have been significant interests in the convergence and consistency of these methods. We do not attempt to give a comprehensive review of these methods but consider only a few of them that are more relevant to our proposed approach; readers are referred to [Petitjean 2002] and [Gatzke and Grimm 2006] for com-

*e-mail: jiao@ams.sunysb.edu
†e-mail: zha@cc.gatech.edu

prehensive surveys. Among the existing methods, many of them estimate the different quantities separately of each other. For the estimation of the normals, a common practice is to estimate vertex normals using a weighted average of face normals, such as area weighting or angle weighting. These methods in general are only first-order accurate, although they are the most efficient.

For curvature estimation, Meyer et al. [2002] proposed some discrete operators for computing normals and curvatures, among which the more popular one is the cotangent formula for mean curvatures, which is closely related to the formula for Dirichlet energy of Pinkall and Polthier [1993]. Xu [2004] studied the convergence of a number of methods for estimating the mean curvatures (and more generally, the Laplace-Beltrami operators). It was concluded that the cotangent formula does not converge except for some special cases, as also noted in [Hildebrandt et al. 2006; Wardetzky 2007]. Langer et al. [2005a; 2005b] proposed a tangent-weighted formula for estimating mean-curvature vectors, whose convergence also relies on special symmetric patterns of a mesh. Agam and Tang [2005] proposed a framework to estimate curvatures of discrete surfaces based on local directional curve sampling. Cohen-Steiner and Morvan [2003] used normal cycle to analyze convergence of curvature measures for densely sampled surfaces.

Vertex-based quadratic or higher-order polynomial fittings can produce convergent normal and curvature estimations. Meek and Walton [2000] studied the convergence properties of a number of estimators for normals and curvatures. It was further generalized to higher-degree polynomial fittings by Cazals and Pouget [2005]. These methods are most closely related to our approach. It is well-known that these methods may encounter numerical difficulties at low-valence vertices or special arrangements of vertices [Lancaster and Salkauskas 1986], which we address in this paper. Razdan and Bae [2005] proposed a scheme to estimate curvatures using biquadratic Bézier patches. Some methods were also proposed to improve robustness of curvature estimation under noise. For example, Ohtake et al. [2004] estimated curvatures by fitting the surface implicitly with multi-level meshes. Recently, Yang et al. [2006] proposed to improve robustness by adapting the neighborhood sizes. These methods in general only provide curvature estimations that are meaningful in some average sense but do not necessarily guarantee convergence of pointwise estimates.

Some of the methods estimate the second-order differential quantities from the surface normals. Goldfeather and Interrante [2004] proposed a cubic-order formula by fitting the positions and normals of the surface simultaneously to estimate curvatures and principal directions. Theisel et al. [2004] proposed a face-based approach for computing shape operators using linear interpolation of normals. Rusinkiewicz [2004] proposed a similar face-based curvature estimator from vertex normals. These methods rely on good normal estimations for reliable results. Zorin and coworkers [Zorin 2005; Grinspun et al. 2006] proposed to compute a shape operator using mid-edge normals, which resembles and "corrects" the formula of [Oñate and Cervera 1993]. Good results were obtained in practice but there was no theoretical guarantee of its order of convergence.

3 Consistent Formulas of Curvatures for Continuous Surfaces

In this section, we first consider the computation of differential quantities of continuous surfaces (as opposed to discrete surfaces). This is a classical subject of differential geometry (see e.g., [do Carmo 1976]), but the differential quantities are traditionally expressed in terms of the first and second fundamental forms, which are geometrically intrinsic but sometimes difficult to understand when a change of coordinate system is involved. Furthermore, as we will show shortly, it is not straightforward to evaluate the classical formulas consistently (i.e., in a backward-stable fashion) in the presence of round-off errors due to the potential loss of symmetry and orthogonality. Starting from these classical formulas, we derive some simple formulas by a novel use of the singular value decomposition of the Jacobian of the height functions. The formulas are easier to understand and to evaluate. To the best of our knowledge, some of our formulas (including those for the symmetric shape operator and principal curvature tensor) have not previously appeared in the literature.

3.1 Classical Formulas for Height Functions

We first review some well-known concepts and formulas in differential geometry and geometric modeling. Given a smooth surface defined in the global xyz coordinate system, it can be transformed into a local uvw coordinate system by translation and rotation. Assume both coordinate frames are orthonormal right-hand systems. Let the origin of the local frame be $[x_0, y_0, z_0]^T$ (note that for convenience we treat points as column vectors). Let $\hat{\boldsymbol{t}}_1$ and $\hat{\boldsymbol{t}}_2$ be the unit vectors in the global coordinate system along the positive directions of the u and v axes, respectively. Then, $\hat{\boldsymbol{m}} = \hat{\boldsymbol{t}}_1 \times \hat{\boldsymbol{t}}_2$ is the unit vector along the positive w direction. Let $\boldsymbol{Q}$ denote the orthogonal matrix composed of column vectors $\hat{\boldsymbol{t}}_1$, $\hat{\boldsymbol{t}}_2$, and $\hat{\boldsymbol{m}}$, i.e.,

$$\boldsymbol{Q} \equiv \left[\begin{array}{c|c|c} \hat{\boldsymbol{t}}_1 & \hat{\boldsymbol{t}}_2 & \boldsymbol{m} \end{array}\right], \tag{1}$$

where '|' denotes concatenation. Any point $\boldsymbol{x}$ on the surface is then transformed to a point $[u, v, f(\boldsymbol{u})]^T \equiv \boldsymbol{Q}^T (\boldsymbol{x} - \boldsymbol{x}_0)$, where $\boldsymbol{u} \equiv (u, v)$. In general, f is not a one-to-one mapping over the whole surface, but if the uv plane is close to the tangent plane at a point $\boldsymbol{x}$, then f would be one-to-one in a neighborhood of $\boldsymbol{x}$. We refer to this function $f(\boldsymbol{u}) : \mathbb{R}^2 \to \mathbb{R}$ (more precisely, from a subset of $\mathbb{R}^2$ into $\mathbb{R}$) as a *height function* at x in the uvw coordinate frame. In the following, all the formulas are given in the uvw coordinate frame unless otherwise stated.

If the surface is smooth, the height function f defines a smooth surface composed of points $\boldsymbol{p}(\boldsymbol{u}) = [u, v, f(\boldsymbol{u})]^T \in \mathbb{R}^3$. Let $\nabla f \equiv \begin{bmatrix} f_u \\ f_v \end{bmatrix}$ denote the gradient of f with respect to $\boldsymbol{u}$ and $\boldsymbol{H} \equiv \begin{bmatrix} f_{uu} & f_{uv} \\ f_{vu} & f_{vv} \end{bmatrix}$ the Hessian of f, where $f_{uv} = f_{vu}$. The Jacobian of $\boldsymbol{p}(\boldsymbol{u})$ with respect to $\boldsymbol{u}$, denoted by $\boldsymbol{J}$, is then

$$\boldsymbol{J} \equiv \left[\begin{array}{c|c} \boldsymbol{p}_u & \boldsymbol{p}_v \end{array}\right] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ f_u & f_v \end{bmatrix}. \tag{2}$$

The vectors $\boldsymbol{p}_u$ and $\boldsymbol{p}_v$ form a basis of the tangent space of the surface at $\boldsymbol{p}$, though they may not be unit vectors and may not be orthogonal to each other. Let $d\boldsymbol{u}$ denote $\begin{bmatrix} du \\ dv \end{bmatrix}$. The *first fundamental form* of the surface is given by the quadratic form

$$I(d\boldsymbol{u}) = d\boldsymbol{u}^T \boldsymbol{G} d\boldsymbol{u}, \text{ where } \boldsymbol{G} \equiv \boldsymbol{J}^T \boldsymbol{J} = \begin{bmatrix} 1 + f_u^2 & f_u f_v \\ f_u f_v & 1 + f_v^2 \end{bmatrix}.$$

$\boldsymbol{G}$ is called the *first fundamental matrix*. Its determinant is $g \equiv \det(\boldsymbol{G}) = 1 + f_u^2 + f_v^2$. We introduce a symbol $\ell \equiv \sqrt{g}$, which will reoccur many times throughout this section. Geometrically, $\ell = \|\boldsymbol{p}_u \times \boldsymbol{p}_v\|$, so it is the "area element" and is equal to the area

of the parallelogram spanned by $\boldsymbol{p}_u$ and $\boldsymbol{p}_v$. The unit normal to the surface is then

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{p}_u \times \boldsymbol{p}_v}{\ell} = \frac{1}{\ell} \begin{bmatrix} -f_u \\ -f_v \\ 1 \end{bmatrix}. \quad (3)$$

The normal in the global xyz frame is then

$$\hat{\boldsymbol{n}}_g = \frac{1}{\ell} \boldsymbol{Q} \begin{bmatrix} -f_u \\ -f_v \\ 1 \end{bmatrix} = \frac{1}{\ell} (\hat{\boldsymbol{m}} - f_u \hat{\boldsymbol{t}}_1 - f_v \hat{\boldsymbol{t}}_2). \quad (4)$$

The *second fundamental form* in the basis $\{\boldsymbol{p}_u, \boldsymbol{p}_v\}$ is given by the quadratic form

$$II(d\boldsymbol{u}) = d\boldsymbol{u}^T \boldsymbol{B} d\boldsymbol{u}$$

where

$$\boldsymbol{B} = -\begin{bmatrix} \hat{\boldsymbol{n}}_u^T \boldsymbol{p}_u & \hat{\boldsymbol{n}}_u^T \boldsymbol{p}_v \\ \hat{\boldsymbol{n}}_v^T \boldsymbol{p}_u & \hat{\boldsymbol{n}}_v^T \boldsymbol{p}_v \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{n}}^T \boldsymbol{p}_{uu} & \hat{\boldsymbol{n}}^T \boldsymbol{p}_{uv} \\ \hat{\boldsymbol{n}}^T \boldsymbol{p}_{uv} & \hat{\boldsymbol{n}}^T \boldsymbol{p}_{vv} \end{bmatrix}. \quad (5)$$

$\boldsymbol{B}$ is called the *second fundamental matrix*, and it measures the change of surface normals. It is easy to verify that

$$\boldsymbol{B} = \boldsymbol{H}/\ell \quad (6)$$

by plugging (2) and (3) into (5).

In differential geometry, the well-known *Weingarten equations* read $[\hat{\boldsymbol{n}}_u \mid \hat{\boldsymbol{n}}_v] = -[\boldsymbol{p}_u \mid \boldsymbol{p}_v]\boldsymbol{W}$, where $\boldsymbol{W}$ is the *Weingarten matrix* (a 2×2 matrix a.k.a. the *shape operator*) at point $\boldsymbol{p}$ with basis $\{\boldsymbol{p}_u, \boldsymbol{p}_v\}$. By left-multiplying $\boldsymbol{J}^T$ on both sides, we have $\boldsymbol{B} = \boldsymbol{G}\boldsymbol{W}$, and therefore,

$$\boldsymbol{W} = \boldsymbol{G}^{-1}\boldsymbol{B} = \frac{1}{\ell}(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{H}. \quad (7)$$

The eigenvalues κ_1 and κ_2 of the Weingarten matrix $\boldsymbol{W}$ are the *principal curvatures* at $\boldsymbol{p}$. The *mean curvature* is $\kappa_H = (\kappa_1 + \kappa_2)/2$ and is equal to half of the trace of $\boldsymbol{W}$. The *Gaussian curvature* at $\boldsymbol{p}$ is $\kappa_G = \kappa_1\kappa_2$ and is equal to the determinant of $\boldsymbol{W}$. Note that $\kappa_H^2 \geq \kappa_G$, because $4(\kappa_H^2 - \kappa_G) = (\kappa_1 - \kappa_2)^2$. Let $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ denote the eigenvectors of $\boldsymbol{W}$. Then $\hat{\boldsymbol{e}}_1 = \boldsymbol{J}\hat{\boldsymbol{d}}_1/\|\boldsymbol{J}\hat{\boldsymbol{d}}_1\|$ and $\hat{\boldsymbol{e}}_2 = \boldsymbol{J}\hat{\boldsymbol{d}}_2/\|\boldsymbol{J}\hat{\boldsymbol{d}}_2\|$ are the *principal directions*.

The principal curvatures and principal directions are often used to construct the 3×3 matrix

$$\boldsymbol{C} \equiv \kappa_1 \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \kappa_2 \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T. \quad (8)$$

We refer to $\boldsymbol{C}$ as the *principal curvature tensor* in the local coordinate system. Note that $\boldsymbol{C}$ is sometimes simply called the *curvature tensor*, which is an overloaded term (see e.g., [do Carmo 1992]), but we nevertheless will use it for conciseness in later discussions of this paper. Note that $\hat{\boldsymbol{n}}$ must be in the null space of $\boldsymbol{C}$, i.e., $\boldsymbol{C}\hat{\boldsymbol{n}} = \boldsymbol{0}$. The curvature tensor is particularly useful because it can facilitate coordinate transformation. In particular, the curvature tensor in the global coordinate system is $\boldsymbol{C}_g = \boldsymbol{Q}\boldsymbol{C}\boldsymbol{Q}^T$.

3.2 Consistency of Classical Formulas

The Weingarten equation is a standard way for defining and computing the principal curvatures and principal directions in differential geometry [do Carmo 1976], so it is important to understand its stability and consistency. We notice that the singular-value decomposition (SVD) of the Jacobian matrix $\boldsymbol{J}$ is given by

$$\boldsymbol{J} = \underbrace{\begin{bmatrix} c/\ell & -s \\ s/\ell & c \\ \|\nabla f\|/\ell & 0 \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \ell & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{\Sigma}} \underbrace{\begin{bmatrix} c & -s \\ s & c \end{bmatrix}^T}_{\boldsymbol{V}^T}, \quad (9)$$

with

$$c = \cos\theta = \frac{f_u}{\|\nabla f\|}, \text{ and } s = \sin\theta = \frac{f_v}{\|\nabla f\|}, \quad (10)$$

where $\theta = \arctan(f_v/f_u)$. In the special case of $f_u = f_v = 0$, we have $\ell = 1$, and any θ leads to a valid SVD of $\boldsymbol{J}$. To resolve this singularity, we define $\theta = 0$ when $f_u = f_v = 0$.

In (9), the column vectors of $\boldsymbol{U}$ are the left singular vectors of $\boldsymbol{J}$, which form an orthonormal basis of the tangent space, the diagonal entries of $\boldsymbol{\Sigma}$ are the singular values, and $\boldsymbol{V}$ is an orthogonal matrix composed of the right singular vectors of $\boldsymbol{J}$. The condition number of a matrix in 2-norm is defined as the ratio between its largest and the smallest singular values, so we obtain the following.

Lemma 1 *The condition number of $\boldsymbol{J}$ in 2-norm is ℓ and the condition number of the first fundamental matrix $\boldsymbol{G}$ is ℓ^2.*

Note that the condition number of $\boldsymbol{G}$ is equal to its determinant. This is a coincidence because one of the singular values of $\boldsymbol{J}$ and in turn $\boldsymbol{G}$ happens to be equal to 1. Suppose $\boldsymbol{H}$ has an absolute error $\delta\boldsymbol{H}$ of $O(\epsilon\|\boldsymbol{H}\|_2)$. From the standard perturbation analysis, we know that the relative error in $\boldsymbol{W}$ computed from (7) can be as large as $O(\ell^2\epsilon)$. The mean and Gaussian curvatures, if computed from the trace and determinant of $\boldsymbol{W}$, would then have an absolute error of $O(\ell^2\epsilon\|\boldsymbol{W}\|_2)$. Therefore, the Weingarten equation must be avoided if $\ell^2 \gg 1$, and it is desirable to keep ℓ as close to 1 as possible. Note that $\boldsymbol{W}$ is nonsymmetric, and more precisely it is nonnormal, i.e., $\boldsymbol{W}^T\boldsymbol{W} \neq \boldsymbol{W}\boldsymbol{W}^T$. The eigenvalues of nonsymmetric matrices are not necessarily real, and the eigenvectors of a nonnormal matrix are not orthogonal to each other [Golub and Van Loan 1996]. Therefore, in the presence of round-off or discretization errors, the principal curvatures computed from the Weingarten matrix are not guaranteed to be real, and the principal directions are not guaranteed to be orthogonal, causing potential inconsistencies.

Unlike the Weingarten matrix $\boldsymbol{W}$, a curvature tensor is always symmetric. The curvature tensors are important concepts and are widely used nowadays (see e.g., [Gatzke and Grimm 2006; Theisel et al. 2004]). In principle, the two larger eigenvalues (in terms of magnitude) of a curvature tensors corresponding to the principal curvatures and their corresponding eigenvectors give a set of orthogonal principal directions. As long as the estimated curvatures are real, one can use (8) to construct a curvature tensor $\boldsymbol{C}$, except that $\boldsymbol{C}$ would be polluted by the errors associated with the nonorthogonal eigenvectors of $\boldsymbol{W}$. This pollution can in turn lead to large errors in both the eigenvalues and eigenvalues of $\boldsymbol{C}$. For example, for a spherical patch $z = \sqrt{4 - (x - 0.5)^2 - (y - 0.5)^2}$ over the interval $[0, 1] \times [0, 1]$, we observed a relative error of up to 8.5% in the eigenvalues of the curvature tensors constructed from a nonsymmetric Weingarten matrix. Another serious problem associated with curvature tensors is that if the minimum curvature is close to zero, then the eigenvectors of a curvature tensor are extremely ill-conditioned. Therefore, one must avoid computing the principal directions as the eigenvectors of the curvature tensor if the minimum curvature is close to 0.

3.3 Simple, Consistent Formulas

To overcome the potential inconsistencies of the classical formulas of principal curvatures and principal directions in the presence of large ℓ or round-off errors, we derive a different set of explicit formulas. Our main idea is to enforce orthogonality and symmetry explicitly and to make the error propagation independent of ℓ as much as possible, starting from the following simple observation.

Lemma 2 *Let $\boldsymbol{U} = [\boldsymbol{u}_1 \mid \boldsymbol{u}_2]$ be a 3×2 matrix whose column vectors form an orthonormal basis of the Jacobian, and $\boldsymbol{C}$ be the curvature tensor. $\boldsymbol{U}^T\boldsymbol{C}\boldsymbol{U}$ is the shape operator in the basis $\{\boldsymbol{u}_1, \boldsymbol{u}_2\}$,*

and it is a symmetric matrix with an orthogonal diagonalization $\boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1}$, whose eigenvalues (i.e., the diagonal entries of $\boldsymbol{\Lambda}$) are real and whose eigenvectors are orthonormal (i.e., $\boldsymbol{X}^T = \boldsymbol{X}^{-1}$). The column vectors of $\boldsymbol{U}\boldsymbol{X}$ are the principal directions.

PROOF. Let κ_i and $\hat{\boldsymbol{d}}_i$ be the eigenvalues and eigenvectors of the Weingarten matrix $\boldsymbol{W}$ in (7). Therefore,

$$\boldsymbol{B}\hat{\boldsymbol{d}}_i = \boldsymbol{G}\boldsymbol{W}\hat{\boldsymbol{d}}_i = \kappa_i \boldsymbol{G}\hat{\boldsymbol{d}}_i. \tag{11}$$

Let $\boldsymbol{S} = \boldsymbol{U}^T\boldsymbol{J}$. By left-multiplying $\boldsymbol{S}^{-T}$ on both sides of (11), we have

$$\boldsymbol{S}^{-T}\boldsymbol{B}\boldsymbol{S}^{-1}(\boldsymbol{S}\hat{\boldsymbol{d}}_i) = \kappa_i(\boldsymbol{S}^{-T}\boldsymbol{G}\hat{\boldsymbol{d}}_i) = \kappa_i(\boldsymbol{S}\hat{\boldsymbol{d}}_i).$$

Therefore, the eigenvalues of $\boldsymbol{S}^{-T}\boldsymbol{B}\boldsymbol{S}^{-1}$ are the principal curvatures κ_i and its eigenvectors are the principal directions $\tilde{\boldsymbol{d}}_1 \equiv \boldsymbol{S}\hat{\boldsymbol{d}}_1$ and $\tilde{\boldsymbol{d}}_2 \equiv \boldsymbol{S}\hat{\boldsymbol{d}}_2$ in the basis $\{\boldsymbol{u}_1, \boldsymbol{u}_2\}$, so $\boldsymbol{S}^{-T}\boldsymbol{B}\boldsymbol{S}^{-1}$ is the shape operator in this basis. This shape operator is symmetric, so its eigenvalues are real and its eigenvectors are orthonormal. By definition of the curvature tensor,

$$\begin{aligned}\boldsymbol{C} &= \kappa_1 \boldsymbol{U}\tilde{\boldsymbol{d}}_1\tilde{\boldsymbol{d}}_1^T\boldsymbol{U}^T + \kappa_2 \boldsymbol{U}\tilde{\boldsymbol{d}}_2\tilde{\boldsymbol{d}}_2^T\boldsymbol{U}^T \\ &= \boldsymbol{U}(\kappa_1\tilde{\boldsymbol{d}}_1\tilde{\boldsymbol{d}}_1^T + \kappa_2\tilde{\boldsymbol{d}}_2\tilde{\boldsymbol{d}}_2^T)\boldsymbol{U}^T \\ &= \boldsymbol{U}\boldsymbol{S}^{-T}\boldsymbol{B}\boldsymbol{S}^{-1}\boldsymbol{U}^T,\end{aligned}$$

so $\boldsymbol{U}^T\boldsymbol{C}\boldsymbol{U}$ is equal to the shape operator $\boldsymbol{S}^{-T}\boldsymbol{B}\boldsymbol{S}^{-1}$. ∎

Lemma 2 holds for any orthonormal basis of the Jacobian. There is an infinite number of such bases. Algorithmically, such a basis can be obtained from either the QR factorization or SVD of $\boldsymbol{J}$. We choose the latter for its simplicity and elegance, and it turns out to be particularly revealing.

Theorem 1 *Let $\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be the reduced SVD of the Jacobian of a height function f as given by (9). Let $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ denote the column vectors of $\boldsymbol{U}$ and let $\boldsymbol{S} = \boldsymbol{\Sigma}\boldsymbol{V}^T$. The shape operator in the orthonormal basis $\{\boldsymbol{u}_1, \boldsymbol{u}_2\}$ is the symmetric matrix*

$$\tilde{\boldsymbol{W}} = \boldsymbol{S}^{-T}\boldsymbol{B}\boldsymbol{S}^{-1} = \frac{1}{\ell}\begin{bmatrix} c/\ell & s/\ell \\ -s & c \end{bmatrix}\boldsymbol{H}\begin{bmatrix} c/\ell & -s \\ s/\ell & c \end{bmatrix}, \tag{12}$$

where c and s are defined in (10).

PROOF. Note that

$$\boldsymbol{S}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} 1/\ell & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c/\ell & -s \\ s/\ell & c \end{bmatrix}.$$

Following the same argument as for Lemma 2, Eq. (12) holds. ∎

To the best of our knowledge, the symmetric shape operator $\tilde{\boldsymbol{W}}$ defined in (12) has not previously appeared in the literature. Because $\tilde{\boldsymbol{W}}$ is symmetric, its eigenvalues (i.e., principal curvatures) are guaranteed to be real, and its eigenvectors (i.e., principal directions) are guaranteed to be orthonormal. In addition, because $\tilde{\boldsymbol{W}}$ is given explicitly by (12), the relative error in $\boldsymbol{H}$ is no longer amplified by a factor of ℓ^2 (or even ℓ) in the computation of $\tilde{\boldsymbol{W}}$. Therefore, this symmetric shape operator provides a consistent way for computing the principal curvatures or principal directions. Furthermore, using this result we can now compute the curvature tensor even without forming the shape operator.

Theorem 2 *Let the transformation from the local to global coordinate system be $\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{u} + \boldsymbol{x}_0$. The curvature tensors in the local and global coordinate systems are*

$$\boldsymbol{C} = \boldsymbol{J}^{+T}\boldsymbol{B}\boldsymbol{J}^{+} = \frac{1}{\ell}\boldsymbol{J}^{+T}\boldsymbol{H}\boldsymbol{J}^{+} \tag{13}$$

$$\boldsymbol{C}_g = \boldsymbol{J}_g^{+T}\boldsymbol{B}\boldsymbol{J}_g^{+} = \frac{1}{\ell}\boldsymbol{J}_g^{+T}\boldsymbol{H}\boldsymbol{J}_g^{+}, \tag{14}$$

respectively, where

$$\boldsymbol{J}^{+} = \frac{1}{\ell^2}\begin{bmatrix} 1+f_v^2 & -f_u f_v & f_u \\ -f_u f_v & 1+f_u^2 & f_v \end{bmatrix} \tag{15}$$

is the pseudo-inverse of $\boldsymbol{J}$, and $\boldsymbol{J}_g^{+} = \boldsymbol{J}^{+}\boldsymbol{Q}^T$ is the pseudo-inverse of $\boldsymbol{J}_g = \boldsymbol{Q}\boldsymbol{J}$. In addition, $\boldsymbol{H} = \ell\boldsymbol{J}^T\boldsymbol{C}\boldsymbol{J} = \ell\boldsymbol{J}_g^T\boldsymbol{C}_g\boldsymbol{J}_g$.

PROOF. The principal directions are $\hat{\boldsymbol{e}}_1 = \boldsymbol{U}\tilde{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{e}}_2 = \boldsymbol{U}\tilde{\boldsymbol{d}}_2$ in the local coordinate system, where $\tilde{\boldsymbol{d}}_i$ are the eigenvectors of $\tilde{\boldsymbol{W}}$. Therefore,

$$\begin{aligned}\boldsymbol{C} &= \kappa_1\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T + \kappa_2\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T \\ &= \boldsymbol{U}\tilde{\boldsymbol{W}}\boldsymbol{U}^T \\ &= (\boldsymbol{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^T)\boldsymbol{B}(\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T) \\ &= \boldsymbol{J}^{+T}\boldsymbol{B}\boldsymbol{J}^{+}.\end{aligned}$$

By left multiplying $\ell\boldsymbol{J}^T$ and right multiplying $\boldsymbol{J}$ of both sides, we then have

$$\ell\boldsymbol{J}^T\boldsymbol{C}\boldsymbol{J} = \ell\boldsymbol{J}^T\boldsymbol{J}^{+T}\boldsymbol{B}\boldsymbol{J}^{+}\boldsymbol{J} = \ell\boldsymbol{B} = \boldsymbol{H}.$$

The curvature tensor in the global coordinate system is $\boldsymbol{C}_g = \boldsymbol{Q}\boldsymbol{C}\boldsymbol{Q}^T$, which results in (14) and $\boldsymbol{H} = \ell\boldsymbol{J}_g^T\boldsymbol{C}_g\boldsymbol{J}_g$. To obtain the explicit formula for $\boldsymbol{J}^{+}$, simply plug in c, s, and ℓ into

$$\boldsymbol{J}^{+} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} 1/\ell & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} c/\ell & s/\ell & \|\nabla f\|/\ell \\ -s & c & 0 \end{bmatrix}.$$

Alternatively, we may obtain $\boldsymbol{J}^{+}$ by expanding and simplifying $(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T$. ∎

Theorem 2 allows us to transform between the curvature tensor and the Hessian of a height function, both of which are symmetric matrices. Furthermore, we can easily compute the gradient and Hessian of one height function from their corresponding values of another height function in a different coordinate system corresponding to the same point of a smooth surface.

Corollary 1 *Given the gradient $\nabla f = \begin{bmatrix} f_u \\ f_v \end{bmatrix}$ and Hessian $\boldsymbol{H}$ of a height function f in an orthonormal coordinate system, let ℓ denote $\sqrt{1+f_u^2+f_v^2}$, $\hat{\boldsymbol{n}}$ the unit normal $[-f_u, -f_v, 1]^T/\ell$, $\boldsymbol{C}$ the curvature tensor $\boldsymbol{J}^{+T}\boldsymbol{H}\boldsymbol{J}^{+}/\ell$, where $\boldsymbol{J} = [\boldsymbol{I}_{2\times 2} \mid \nabla f]^T$. Let $\hat{\boldsymbol{Q}} \equiv [\hat{\boldsymbol{q}}_1 \mid \hat{\boldsymbol{q}}_2 \mid \hat{\boldsymbol{q}}_3]$ be the orthogonal matrix whose column vectors form the axes of another coordinate system, where $\hat{\boldsymbol{q}}_3^T\hat{\boldsymbol{n}} > 0$. Let $[\alpha, \beta, \gamma]^T$ denote $\hat{\boldsymbol{Q}}^T\hat{\boldsymbol{n}}$, and let $\tilde{f}$ denote the height function in the latter coordinate frame corresponding to the same smooth surface. Then the gradient of $\tilde{f}$ is $\nabla\tilde{f} = \begin{bmatrix} -\alpha/\gamma \\ -\beta/\gamma \end{bmatrix}$, and the Hessian of $\tilde{f}$ is $\tilde{\boldsymbol{J}}^T\boldsymbol{C}\tilde{\boldsymbol{J}}/\gamma$, where $\tilde{\boldsymbol{J}} = \hat{\boldsymbol{Q}}\left[\boldsymbol{I}_{2\times 2} \mid \nabla\tilde{f}\right]^T$.*

A direct implication of this corollary is that knowing the *consistent* surface normal and curvature tensor of a smooth surface in the global coordinate frame is equivalent to knowing the *consistent* gradient and Hessian of its corresponding height function in any local coordinate frame in which the Jacobian is nondegenerate (i.e., $\ell > 0$). Here, the consistency refers to the fact that the normal $\hat{\boldsymbol{n}}$ is in the null space of the curvature tensor $\boldsymbol{C}$ and that both $\boldsymbol{C}$ and the Hessian $\boldsymbol{H}$ are symmetric (i.e., $\boldsymbol{C}\hat{\boldsymbol{n}} = \boldsymbol{0}$, $\boldsymbol{C}^T = \boldsymbol{C}$, and $\boldsymbol{H} = \boldsymbol{H}^T$).

The preceding formulas for the symmetric shape operator and curvature tensor appear to be new and are particularly useful when

computing the principal curvatures and principal directions. Because many applications require the mean curvature and Gaussian curvature, we also give the following simple formulas, which are equivalent to those in classical differential geometry [do Carmo 1976, p. 163] but are given here in a more concise form.

Theorem 3 *The mean and Gaussian curvature of the height function $f(\boldsymbol{u}) : \mathbb{R}^2 \to \mathbb{R}$ are*

$$\kappa_H = \frac{tr(\boldsymbol{H})}{2\ell} - \frac{(\nabla f)^T \boldsymbol{H} (\nabla f)}{2\ell^3}, \text{ and } \kappa_G = \frac{det(\boldsymbol{H})}{\ell^4}. \quad (16)$$

PROOF. Let $\boldsymbol{v} = [c, s]^T$ and $\boldsymbol{v}^\perp = [-s, c]^T$, where c and s are defined in (10). The trace of $\tilde{\boldsymbol{W}}$ is

$$\begin{aligned}
\text{tr}(\tilde{\boldsymbol{W}}) &= \text{tr}\left(\frac{1}{\ell}\begin{bmatrix} c/\ell & s/\ell \\ -s & c \end{bmatrix} \boldsymbol{H} \begin{bmatrix} c/\ell & -s \\ s/\ell & c \end{bmatrix}\right) \\
&= \frac{1}{\ell}\left(\frac{\boldsymbol{v}^T}{\ell}\boldsymbol{H}\frac{\boldsymbol{v}}{\ell} + \boldsymbol{v}^{\perp T}\boldsymbol{H}\boldsymbol{v}^\perp\right) \\
&= \frac{1}{\ell^3}\left(\boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v} + \ell^2\boldsymbol{v}^{\perp T}\boldsymbol{H}\boldsymbol{v}^\perp\right) \\
&= \frac{1}{\ell^3}\left(\ell^2\text{tr}(\boldsymbol{H}) - (\ell^2-1)\boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v}\right),
\end{aligned}$$

where the last step uses $\boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v} + \boldsymbol{v}^{\perp T}\boldsymbol{H}\boldsymbol{v}^\perp = \text{tr}(\boldsymbol{H})$. Since $\sqrt{\ell^2-1}\boldsymbol{v} = \nabla f$, therefore,

$$\kappa_H = \frac{1}{2}\text{tr}(\tilde{\boldsymbol{W}}) = \frac{\text{tr}(\boldsymbol{H})}{2\ell} - \frac{(\nabla f)^T\boldsymbol{H}(\nabla f)}{2\ell^3}.$$

With regard to κ_G, we have

$$\kappa_G = \det(\tilde{\boldsymbol{W}}) = \det(\boldsymbol{S}^{-1})^2\det(\boldsymbol{H}/\ell) = \frac{\det(\boldsymbol{H})}{\ell^4}.$$

■

This completes our description of the explicit formulas for the differential quantities. Since their derivations, especially for the principal curvatures, principal directions, and curvature tensor, are based on the shape operator associated with the left singular-vectors of the Jacobian, many intermediate terms cancel out due to symmetry and orthogonality, and the final formulas are remarkably simple. It is important to note that even if the gradient or Hessian contain input errors, as long as the Hessian is symmetric and the Jacobian is nondegenerate, our formulas are consistent in the following sense: The principal curvatures are real, the principal directions are orthonormal, and the surface normal is orthogonal to the column space of $\boldsymbol{J}$, the row spaces of $\boldsymbol{J}^+$, and the column space of the curvature tensor.

The consistency of our formulas is enabled by a simple fact: A consistent set of first- and second-order differential quantities have five degrees of freedom, which is a direct result of Corollary 1. In the Weingarten equations, numerically there are six degrees of freedom (three in $\boldsymbol{G}$ and three in $\boldsymbol{B}$), so there is an implicit constraint (which should have enforced the orthogonality of the principal directions) not present in the equations. In general, if a system of equations has more degrees of freedom than the intrinsic dimension of the problem (i.e., with implicit constraints), its numerical solution would likely lead to inconsistencies in the presence of round-off errors. The same argument may be applied to other methods that assume more than five independent parameters per point for the first- and second-order differential quantities. For example, if the differential quantities are evaluated from a parameterization of a surface where x, y, and z coordinates are viewed as independent functions of u and v, one must compute more than five parameters, so their numerical solutions may not be consistent. The gradient and the Hessian of the height function contain exactly five degrees of freedom when the symmetry of the Hessian is enforced, so we have reduced the problem consistently into the computation of the gradient and Hessian of the height function. We therefore build our method based on the estimation of the gradient and Hessian of the height functions.

4 Computing Gradient and Hessian of Height Function

To apply the formulas for continuous surfaces to discrete surfaces, we must select a region of interest, define a local uvw coordinate frame, and approximate the gradient and Hessian of the resulting height function. We build our method based on a local polynomial fitting. Polynomial fitting is not a new idea and has been studied intensively (e.g., [Lancaster and Salkauskas 1986; de Boor and Ron 1992; Meek and Walton 2000; Cazals and Pouget 2005]). However, it is well-known that polynomial fitting may suffer from numerical instabilities, which in turn can undermine convergence and lead to large errors. We propose some techniques to overcome instabilities and to improve the accuracy of fittings. These techniques are based on classical concepts in numerical linear algebra but are customized here for derivative computations. In addition, we propose a new iterative-fitting scheme and also present a unified error analysis of our approach.

4.1 Local Polynomial Fitting

The local polynomial fitting can be derived from the Taylor series expansion. Let $f(\boldsymbol{u})$ denote a bivariate function, where $\boldsymbol{u} = (u, v)$, and let c_{jk} be a shorthand for $\frac{\partial^{j+k}}{\partial u^j \partial v^k} f(\boldsymbol{0})$. The Taylor series expansion of f about the origin $\boldsymbol{u}_0 = (0, 0)$ is

$$f(\boldsymbol{u}) = \sum_{p=0}^{\infty} \sum_{j,k\geq 0}^{j+k=p} c_{jk} \frac{u^j v^k}{j!k!}. \quad (17)$$

Given a positive integer d, a function $f(\boldsymbol{u})$ can be approximated to $(d+1)$st order accuracy about the origin $\boldsymbol{u}_0$ as

$$f(\boldsymbol{u}) = \underbrace{\sum_{p=0}^{d} \sum_{j,k\geq 0}^{j+k=p} c_{jk} \frac{u^j v^k}{j!k!}}_{\text{Taylor polynomial}} + \underbrace{O(\|\boldsymbol{u}\|^{d+1})}_{\text{remainder}}, \quad (18)$$

assuming f has $d+1$ continuous derivatives. The derivatives of the Taylor polynomial are the same as f at $\boldsymbol{u}_0$ up to degree d.

Given a set of points sampling a small patch of a smooth surface, the method of local polynomial fitting approximates the Taylor polynomial by estimating c_{jk} from the given points. The degree of the polynomial, denoted by d, is called the *degree of fitting*. We will limit ourselves to relatively low degree fittings (say $d \leq 6$), because high-degree fittings tend to be more oscillatory and less stable. To estimate c_{jk} at a vertex of a surface mesh, we choose the vertex to be the origin of a local uvw coordinate frame, where the w component of the coordinates in this frame would be the height function f. We use an approximate vertex normal (e.g., obtained by averaging the face normals) as the w direction, so that the condition number of the Jacobian of f would be close to 1. Plugging in each given point $[u_i, v_i, f_i]^T$ into (18), we obtain an approximate equation

$$\sum_{p=0}^{d} \sum_{j,k\geq 0}^{j+k=p} c_{jk} \frac{u_i^j v_i^k}{j!k!} \approx f_i, \quad (19)$$

which has $n \equiv (d+1)(d+2)/2$ unknowns (i.e., c_{jk} for $0 \leq j+k \leq d$, $j \geq 0$ and $k \geq 0$). As a concrete example, for cubic fitting the equation is

$$c_{00} + c_{10}u_i + c_{01}v_i + c_{20}\frac{u_i^2}{2} + c_{11}u_iv_i + c_{02}\frac{v_i^2}{2} + c_{30}\frac{u_i^3}{6} + c_{21}\frac{u_i^2v_i}{2} + c_{12}\frac{u_iv_i^2}{2} + c_{03}\frac{v_i^3}{6} \approx f_i. \quad (20)$$

Let m denote the number of these given points (including the vertex $\boldsymbol{u}_0$ itself), we then obtain an $m \times n$ rectangular linear system. If we want to enforce the fit to pass through the vertex itself, we can simply set $c_{00} = 0$ and remove the equation corresponding to $\boldsymbol{u}_0$, leading to an $(m-1) \times (n-1)$ rectangular linear system. In our experiments, it seems to have little benefit to do this, so we do not enforce $c_{00} = 0$ in the following discussions. However, our framework can be easily adapted to enforce $c_{00} = 0$ if desired.

Suppose we have solved this rectangular linear system and obtained the approximation of c_{jk}. At $\boldsymbol{u}_0$, the gradient of f is $\begin{bmatrix} c_{10} \\ c_{01} \end{bmatrix}$ and the Hessian is $\begin{bmatrix} c_{20} & c_{11} \\ c_{11} & c_{02} \end{bmatrix}$. Plugging their approximations into the formulas in Section 3 would give us the normal and curvature approximations at $\boldsymbol{u}_0$. This approach is similar to the fitting methods in [Cazals and Pouget 2005; Meek and Walton 2000]. Note that the Taylor series expansions of f_u and f_v about $\boldsymbol{u}_0$ are

$$f_u(\boldsymbol{u}) \approx \sum_{p=0}^{d-1} \sum_{j,k\geq 0}^{j+k=p} c_{(j+1)k} \frac{u^j v^k}{j!k!}, \quad (21)$$

$$f_v(\boldsymbol{u}) \approx \sum_{p=0}^{d-1} \sum_{j,k\geq 0}^{j+k=p} c_{j(k+1)} \frac{u^j v^k}{j!k!}, \quad (22)$$

where the residual is $O(\|\boldsymbol{u}\|^d)$. The expansions for the second derivatives have similar patterns. Plugging in the approximations of c_{jk} into these series, we can also obtain the estimations of the gradient, Hessian, and in turn the curvatures at a point $\boldsymbol{u}$ near $\boldsymbol{u}_0$. The remaining questions for this approach are the theoretical issue of solving the rectangular system as well as the practical issues of the selection of points and robust implementation.

4.2 Weighted Least Squares Formulation

Let us denote the rectangular linear system obtained from (19) for $i = 1, \ldots, m$ as

$$\boldsymbol{V}\boldsymbol{c} \approx \boldsymbol{f}, \quad (23)$$

where $\boldsymbol{c}$ is an n-vector composed of c_{jk}, $\boldsymbol{f}$ is an m-vector composed of f_i, and $\boldsymbol{V}$ is a *generalized Vandermonde matrix*. For now, let us assume that $m \geq n$ and $\boldsymbol{V}$ has full rank (i.e., its column vectors are linearly independent). Such a system is said to be *over-determined*. We will address the robust numerical solutions for more general cases in the next subsection.

The simplest solution to (23) is to minimize the 2-norm of the residual $\boldsymbol{V}\boldsymbol{c} - \boldsymbol{f}$, i.e., $\min_{\boldsymbol{c}} \|\boldsymbol{V}\boldsymbol{c} - \boldsymbol{f}\|_2$. A more general formulation is to minimize a weighted norm (or semi-norm), i.e.,

$$\min_{\boldsymbol{c}} \|\boldsymbol{V}\boldsymbol{c} - \boldsymbol{f}\|_{\boldsymbol{W}} \equiv \min_{\boldsymbol{c}} \|\boldsymbol{W}(\boldsymbol{V}\boldsymbol{c} - \boldsymbol{f})\|_2, \quad (24)$$

where $\boldsymbol{W}$ is an $m \times m$ diagonal matrix with nonnegative entries. We refer to $\boldsymbol{W}$ as a *weighting matrix*. Such a formulation is called *weighted least squares* [Golub and Van Loan 1996, p. 265], and it has a unique solution if $\boldsymbol{W}\boldsymbol{V}$ has full rank. It is equivalent to a linear least squares problem

$$\boldsymbol{A}\boldsymbol{c} \approx \boldsymbol{b}, \text{ where } \boldsymbol{A} \equiv \boldsymbol{W}\boldsymbol{V} \text{ and } \boldsymbol{b} \equiv \boldsymbol{W}\boldsymbol{f}. \quad (25)$$

Let ω_i denote the ith diagonal entry of $\boldsymbol{W}$. Algebraically, ω_i assigns a weight to each row of the linear system. Geometrically, it assigns a priority to each point (the larger ω_i, the higher the priority). If $\boldsymbol{f}$ is in the column space of $\boldsymbol{V}$, then a nonsingular weighting matrix has no effect on the solution of the linear system. Otherwise, different weighting matrices may lead to different solutions. Furthermore, by setting ω_i to be zero or close to zero, the weighting matrix can be used as a mechanism for filtering out outliers in the given points.

The weighted least squares formulation is a general technique, but the choice of $\boldsymbol{W}$ is problem dependent. For local polynomial fitting, it is natural to assign lower priorities to points that are farther away from the origin or whose normals differ substantially from the w direction of the local coordinate frame. Furthermore, the choice of $\boldsymbol{W}$ may affect the residual and also the condition number of $\boldsymbol{A}$. Let $\hat{\boldsymbol{m}}_i$ denote an initial approximation of the unit normal at the ith vertex (e.g., obtained by averaging face normals). Based on the above considerations, we choose the weight of the ith vertex as

$$\omega_i = \gamma_i^+ \Big/ \left(\sqrt{\|\boldsymbol{u}_i\|^2 + \epsilon}\right)^{d/2}, \quad (26)$$

where $\gamma_i^+ \equiv \max(0, \hat{\boldsymbol{m}}_i^T \hat{\boldsymbol{m}}_0)$ and $\epsilon \equiv \frac{1}{100m}\sum_{i=1}^m \|\boldsymbol{u}_i\|^2$. In general, this weight is approximately equal to $\|\boldsymbol{u}_i\|^{-d/2}$, where the exponent $d/2$ tries to balance between accuracy and stability. The factor γ_i^+ approaches 1 for fine meshes at smooth areas, but it serves as a safeguard against drastically changing normals for coarse meshes or nonsmooth areas. The term ϵ prevents the weights from becoming too large at points that are too close to $\boldsymbol{u}_0$.

The weighting matrix scales the rows of $\boldsymbol{V}$. However, if u_i or v_i are close to zero, the columns of $\boldsymbol{V}$ can be poorly scaled, so that the ith row of $\boldsymbol{A}$ would be close to $[\omega_i, 0, \ldots, 0]$. Such a matrix would have a very large condition number. A general approach to alleviate this problem is to introduce a *column scaling matrix* $\boldsymbol{S}$. The least squares problem then becomes

$$\min_{\boldsymbol{d}} \|(\boldsymbol{A}\boldsymbol{S})\boldsymbol{d} - \boldsymbol{b}\|_2, \text{ where } \boldsymbol{d} \equiv \boldsymbol{S}^{-1}\boldsymbol{c}. \quad (27)$$

Here, $\boldsymbol{S}$ is a nonsingular $n \times n$ diagonal matrix. Unlike the weighting matrix $\boldsymbol{W}$, the scaling matrix $\boldsymbol{S}$ does not change the solution of $\boldsymbol{c}$ under exact arithmetic. However, it can significantly improve the condition number and in turn improve the accuracy in the presence of round-off errors. In general, let $\boldsymbol{a}_j$ denote the jth column vector of $\boldsymbol{A}$. We choose $\boldsymbol{S} = \text{diag}(1/\|\boldsymbol{a}_1\|_2, \ldots, 1/\|\boldsymbol{a}_n\|_2)$, as it approximately minimizes the 2-norm condition number of $\boldsymbol{A}\boldsymbol{S}$ (see [Golub and Van Loan 1996, p. 265] and [van der Sluis 1969]).

4.3 Robust Implementation

Our preceding discussions considered only over-determined systems. However, even if $m \geq n$, the matrix $\boldsymbol{V}$ (and in turn $\boldsymbol{A}\boldsymbol{S}$) may not have full rank for certain arrangements of points, so the weighted least squares would be practically under-determined with an infinite number of solutions. Numerically, even if $\boldsymbol{V}$ has full rank, the condition number of $\boldsymbol{A}\boldsymbol{S}$ may still be arbitrarily large, which in turn may lead to arbitrarily large errors in the estimated derivatives. We address these issues by proposing a systematic way for selecting the points and a safeguard for QR factorization for solving the least squares problem.

4.3.1 Section of Points

As pointed out in [Lancaster and Salkauskas 1986], the condition number of polynomial fitting can depend on the arrangements of

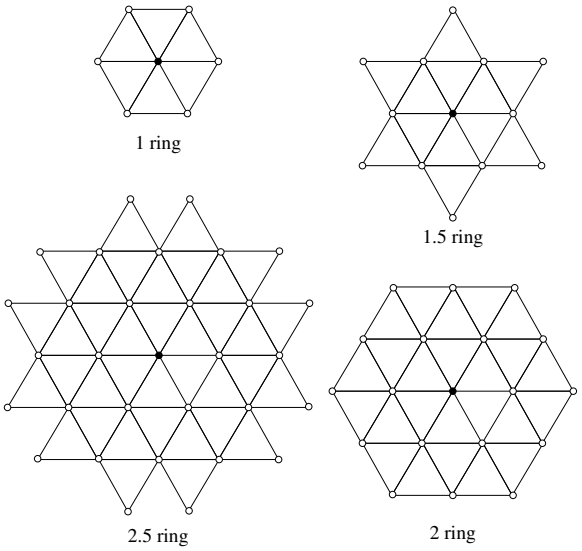


Figure 1: *Schematics of 1-ring, 1.5-ring, 2-ring, and 2.5-ring neighbor vertices of center vertex (black) in each illustration.*

points. The situation may appear to be hopeless, because we in general have little control (if any at all) about the locations of the points. However, the problem can be much alleviated when the number of points m is substantially larger than the number of unknowns n. To the best of our knowledge, no precise relationship between the condition number and m/n has been established. However, it suffices here to have an intuitive understanding from the geometric interpretation of condition numbers. Observe that an $m \times n$ matrix $\boldsymbol{M}$ (with $m \geq n$) is rank deficient if its row vectors are co-planar (i.e, contained in a hyperplane of dimension less than n), and its condition number is very large if its row vectors are nearly co-planar (i.e., if they lie in the proximity of a hyperplane). Suppose the row vectors of $\boldsymbol{M}$ are random with a nearly uniform distribution, then the probability of the vectors being nearly co-planar decreases exponentially as m increases. Therefore, having more points than the unknowns can be highly beneficial in decreasing the probability of ill-conditioning for well-scaled matrices. In addition, it is well-known that having more points also improves noise resistance of the fitting, which however is beyond the scope of this paper.

Based on the above observation, we require m to be larger than n, but a too large m would compromise efficiency and may also undermine accuracy. As a rule of thumb, it appears to be ideal for m to be between $1.5n$ and $2n$. For triangular meshes, we define the following neighborhoods that meet this criterion. First, let us define the *1-ring neighbor faces* of a vertex to be the triangles incident on it. We define the *1-ring neighborhood* of a vertex to be the vertices of its 1-ring neighbor faces, and define the *1.5-ring neighborhood* as the vertices of all the faces that share an edge with a 1-ring neighbor face. This definition of 1-ring neighborhood is conventional, but our definition of 1.5-ring neighborhood appears to be new. Furthermore, for $k \geq 1$, we define the $(k+1)$*-ring neighborhood* of a vertex to be the vertices of the k-ring neighborhood along with their 1-ring neighbors, and define the $(k+1.5)$*-ring neighborhood* to be the vertices of the k-ring neighborhood along with their 1.5-ring neighbors. Figure 1 illustrates these neighborhood definitions up to 2.5 rings.

Observe that a typical $\frac{d+1}{2}$-ring neighborhood has about the ideal number of points for the dth degree fittings at least up to degree six, which is evident from Table 1. Therefore, we use this as the general guideline for selecting the points. Note that occasionally

Table 1: *Numbers of coefficients in dth degree fittings versus numbers of points in typical $\frac{d+1}{2}$ rings.*

degree (d)	1	2	3	4	5	6
#coeffs.	3	6	10	15	21	28
#points in $\frac{d+1}{2}$ ring	7	13	19	31	37	55

(such as for the points near boundary) the $\frac{d+1}{2}$-ring neighborhood may not have enough points. If the number of points is less than $1.5n$, we increase the ring level by 0.5 up to 3.5. Note that we do not attempt to filter out the points that are far away from the origin at this stage, because such a filtering is done more systematically through the weighting matrix in (24).

4.3.2 QR Factorization with Safeguard

Our discussions about the point selection was based on a probability argument. Therefore, ill-conditioning may still occur especially near the boundary of a surface mesh. The standard approaches for addressing ill-conditioned rectangular linear systems include SVD or QR factorization with column pivoting (see [Golub and Van Loan 1996, p. 270]. When applied to (27), the former approach would seek a solution that minimizes the 2-norm of the solution vector $\|\boldsymbol{d}\|_2$ among all the feasible solutions, and the latter provides an efficient but less robust approximation to the former. Neither of these approaches seems appropriate in this context, because they do not give higher priorities to the lower derivatives, which are the solutions of interest.

Instead, we propose a safeguard for QR factorization for the local fittings. Let the columns of $\boldsymbol{V}$ (and in turn of $\boldsymbol{A}$) be sorted in increasing order of the derivatives (i.e., in increasing order of $j+k$). Let the reduced QR factorization of $\boldsymbol{AS}$ be

$$\boldsymbol{AS} = \boldsymbol{QR},$$

where $\boldsymbol{Q}$ is $m \times n$ with orthonormal column vectors and $\boldsymbol{R}$ is an $n \times n$ upper-triangular matrix. The 2-norm condition number of $\boldsymbol{AS}$ is the same as that of $\boldsymbol{R}$. To determine whether $\boldsymbol{AS}$ is nearly rank deficient, we estimate the condition number of $\boldsymbol{R}$ in 1-norm (instead of 2-norm, for better efficiency), which can be done efficiently for triangular matrices and is readily available in linear algebra libraries (such as DGECON in LAPACK [Anderson et al. 1999]). If the condition number of $\boldsymbol{R}$ is too large (e.g., $\geq 10^3$), we decrease the degree of the fitting by removing the last few columns of $\boldsymbol{AS}$ that correspond to the highest derivatives. Note that QR factorization need not be recomputed after decreasing the degree of fitting, because it can be obtained by removing the corresponding columns in $\boldsymbol{Q}$ and removing the corresponding rows and columns in $\boldsymbol{R}$. If the condition number is still large, we would further reduce the degree of fitting until the condition number is small or the fitting becomes linear. Let $\tilde{\boldsymbol{Q}}$ and $\tilde{\boldsymbol{R}}$ denote the reduced matrices of $\boldsymbol{Q}$ and $\boldsymbol{R}$, and the final solution of $\boldsymbol{c}$ is given by

$$\boldsymbol{c} = \boldsymbol{S}\tilde{\boldsymbol{R}}^{-1}\tilde{\boldsymbol{Q}}^T\boldsymbol{b}, \tag{28}$$

where $\tilde{\boldsymbol{R}}^{-1}$ denotes a back substitution step. Compared to SVD or QR with partial pivoting, the above procedure is more accurate for derivative estimation, as it gives higher priorities to lower derivatives, and at the same time it is more efficient than SVD.

4.4 Iterative Fitting of Derivatives

The polynomial fitting above uses only the coordinates of the given points. If accurate normals are known, it can be beneficial to take

advantage of them. If the normals are not given a priori, we may estimate them first by using the polynomial fitting described earlier. We refer to this approach as *iterative fitting*.

First, we convert the vertex normals into the gradients of the height function within the local uvw frame at a point. Let $\hat{\boldsymbol{n}}_i = [\alpha_i, \beta_i, \gamma_i]^T$ denote the unit normal at the ith point in the uvw coordinate system, and then the gradient of the height function is $\begin{bmatrix} -\alpha_i/\gamma_i \\ -\beta_i/\gamma_i \end{bmatrix}$. Let $a_{jk} \equiv c_{(j+1)k}$ and $b_{jk} \equiv c_{j(k+1)}$. By plugging u_i, v_i, and $f_u(u_i, v_i) \approx -\alpha_i/\gamma_i$ into (21), we obtain an equation for the coefficients as; similarly for f_v and bs. For example, for cubic fittings we obtain the equations

$$a_{00} + a_{10}u_i + a_{01}v_i + a_{20}\frac{u_i^2}{2} + a_{11}u_iv_i + a_{02}\frac{v_i^2}{2} + a_{30}\frac{u_i^3}{6} + a_{21}\frac{u_i^2v_i}{2} + a_{12}\frac{u_iv_i^2}{2} + a_{03}\frac{v_i^3}{6} \approx -\frac{\alpha_i}{\gamma_i}, \quad (29)$$

$$b_{00} + b_{10}u_i + b_{01}v_i + b_{20}\frac{u_i^2}{2} + b_{11}u_iv_i + b_{02}\frac{v_i^2}{2} + b_{30}\frac{u_i^3}{6} + b_{21}\frac{u_i^2v_i}{2} + b_{12}\frac{u_iv_i^2}{2} + b_{03}\frac{v_i^3}{6} \approx -\frac{\beta_i}{\gamma_i}. \quad (30)$$

If we enforce $a_{j,k+1} = b_{j+1,k}$ explicitly, we would obtain a single linear system for all the as and bs with a reduced number of unknowns. Alternatively, a_{jk} and b_{jk} can be solved separately using the same coefficient matrix as (23) and the same weighted least squares formulation, but two different right-hand side vectors, and then $a_{j,k+1}$ and $b_{j+1,k}$ must be averaged. The latter approach compromises the optimality of the solution without compromising the symmetry and the order of convergence. We choose the latter approach for simplicity.

After obtaining the coefficients as and bs, we then obtain the Hessian of the height function at $\boldsymbol{u}_0$ as

$$\boldsymbol{H}_0 = \begin{bmatrix} a_{10} & (a_{01}+b_{10})/2 \\ (a_{01}+b_{10})/2 & b_{01} \end{bmatrix}.$$

From the gradient and Hessian, the second-order differential quantities are then obtained using the formulas in Section 3. We summarize the overall iterative fitting algorithm as follows:
1) Obtain initial estimation of vertex normals by averaging face normals for the construction of local coordinate systems at vertices;
2) For each vertex, determine $\frac{d+1}{2}$-ring neighbor vertices and upgrade neighborhood if necessary;
3) For each vertex, transform its neighbor points into its local coordinate frame, solve for the coefficients using QR factorization with safeguard, and convert gradients into vertex normals;
4) If iterative fitting is desired, for each vertex, transform normals of its neighbor vertices to the gradient of the height function in its local coordinate system to solve for Hessian;
5) For each vertex, convert Hessian into symmetric shape operator to compute curvatures, principal directions, or curvature tensor.

In the algorithm, some steps (such as the collection of neighbor points) may be merged into the loops in later steps, with a tradeoff between memory and computation. Note that iterative fitting is optional, because we found it to be beneficial only for odd-degree fittings, as discussed in more detail in the next subsection. Without iterative fitting, the Hessian of the height function should be obtained at step 3. Finally, note that we can also apply the idea of iterative fitting to convert the curvature tensor to the Hessian of the height function and then construct a fitting of the Hessian to obtain higher derivatives. However, since the focus of this paper is on first- and second-order differential quantities, we do not pursue this idea further.

4.5 Error Analysis

To complete the discussion of our framework, we must address the fundamental question of whether the computed differential quantities converge as the mesh gets refined, and if so, what is the convergence rate. While it has been previously shown that polynomial fittings produce accurate normal and curvature estimations in noise-free contexts [Cazals and Pouget 2005; Meek and Walton 2000], our framework is more general and uses different formulas for computing curvatures, so it requires a more general analysis. Let h denote the average edge length of the mesh. We consider the errors in terms of h. Our theoretical results have two parts, as summarized by the following two theorems.

Theorem 4 *Given a set of points in uvw coordinate frame that interpolate a smooth height function f or approximate f with an error of $O(h^{d+1})$ along the w direction. Assume the point distribution and the weighing matrix are independent of the mesh resolution, and the scaled matrix $\boldsymbol{AS}$ in (27) has a bounded condition number. The degree-d weighted least squares fitting approximates c_{jk} to $O(h^{d-j-k+1})$.*

PROOF. Let $\boldsymbol{c}$ denote the vector composed of c_{jk}, the exact partial derivatives of f. Let $\tilde{\boldsymbol{b}}$ denote $\boldsymbol{Ac}$. Let $\boldsymbol{r} \equiv \boldsymbol{b} - \tilde{\boldsymbol{b}}$, which we consider as a perturbation in the right-hand side of

$$\boldsymbol{Ac} \approx \tilde{\boldsymbol{b}} + \boldsymbol{r}.$$

Because the Taylor polynomial approximates f to $O(h^{d+1})$, and f is approximated to $O(h^{d+1})$ by the given points, each component of $\boldsymbol{f} - \boldsymbol{Vc}$ in (23) is $O(h^{d+1})$. Since $\boldsymbol{r} = \boldsymbol{W}(\boldsymbol{f} - \boldsymbol{Vc})$ and the entries in $\boldsymbol{W}$ are $\Theta(1)$, each component of $\boldsymbol{r}$ is $O(h^{d+1})$.

The error in c_{jk} and in $\boldsymbol{r}$ are connected by the scaled matrix $\boldsymbol{AS}$. Since the point distribution are independent of the mesh resolution, $u_i = \Theta(h)$ and $v_i = \Theta(h)$. The entries in the column of $\boldsymbol{A}$ corresponding to c_{jk} are then $\Theta(h^{j+k})$, so are the 2-norm of the column. After column scaling, the entries in $\boldsymbol{AS}$ are then $\Theta(1)$, so are the entries in its pseudo-inverse $(\boldsymbol{AS})^+$. The error of $\boldsymbol{d}$ in (27) is then $\delta\boldsymbol{d} = (\boldsymbol{AS})^+\boldsymbol{r}$. Because each component of $\boldsymbol{r}$ is $O(h^{d+1})$, each component of $\delta\boldsymbol{d}$ is $O(\kappa h^{d+1})$, where κ is the condition number of $\boldsymbol{AS}$ and is bounded by a constant by assumption. The error in $\boldsymbol{c}$ is then $\boldsymbol{S}\delta\boldsymbol{d}$, where the scaling factor associated with c_{jk} is $\Theta(1/h^{j+k})$. Therefore, the coefficient c_{jk} is approximated to $O(h^{d-j-k+1})$. ■

Note that our weights given in (26) appear to depend on the mesh resolution. However, we can rescale it by multiplying $h^{d/2}$ to make it $\Theta(1)$ without changing the solution, so Theorem 4 applies to our weighting scheme. Under the assumptions of Theorem 4, degree-d fitting approximates the gradient to $O(h^d)$ and the Hessian to $O(h^{d-1})$ at the origin of the local frame, respectively. Using the gradient and Hessian estimated from our polynomial fitting, the estimated differential quantities have the following property.

Theorem 5 *Given the position, gradient, and Hessian of a height function that are approximated to $O(h^{d+1})$, $O(h^d)$ and $O(h^{d-1})$, respectively. a) The angle between the computed and exact normals is $O(h^d)$; b) the components of the shape operator and curvature tensor are approximated $O(h^{d-1})$ by (12) and (13); c) the Gaussian and mean curvatures are approximated to $O(h^{d-1})$ by (16).*

PROOF. a) Let $\tilde{f}_u$ and $\tilde{f}_v$ denote the estimated derivatives of f, which approximate the true derivatives f_u and f_v to $O(h^d)$. Let $\tilde{\ell}$ and ℓ denote $\|[-\tilde{f}_u, -\tilde{f}_v, 1]\|$ and $\|[-f_u, -f_v, 1]\|$, respectively. Let $\tilde{\boldsymbol{n}}$ denote the computed unit normal $[-\tilde{f}_u, -\tilde{f}_v, 1]^T/\tilde{\ell}$ and $\hat{\boldsymbol{n}}$

the exact unit normal $[-f_u, -f_v, 1]^T/\ell$. Therefore,

$$\tilde{\boldsymbol{n}} - \hat{\boldsymbol{n}} = \frac{\ell[-\tilde{f}_u, -\tilde{f}_v, 1]^T - \tilde{\ell}[-f_u, -f_v, 1]^T}{\ell\tilde{\ell}},$$

where the numerator is $O(h^d)$ and the denominator is $\Theta(1)$. Let θ denote $\arccos \tilde{\boldsymbol{n}}^T\hat{\boldsymbol{n}}$. Because $\|\tilde{\boldsymbol{n}} - \hat{\boldsymbol{n}}\|_2^2 = 2 - 2\tilde{\boldsymbol{n}}^T\hat{\boldsymbol{n}} = \theta^2 + O(\theta^4)$, it then follows that θ is $O(h^d)$.

b) In (12), if $f_u = f_v = 0$, then $\tilde{\boldsymbol{W}} = \boldsymbol{H}$, which is approximated to $O(h^{d-1})$. Otherwise, $\boldsymbol{H}$ is approximated to $O(h^{d-1})$ while ℓ, c, and s are approximated to $O(h^d)$, so the error in $\tilde{\boldsymbol{W}}$ is $O(h^{d-1})$. Similarly, in equations (13) and (14), the dominating error term is the $O(h^{d-1})$ error in $\boldsymbol{H}$, while $\boldsymbol{J}^+$ and $\tilde{\boldsymbol{J}}^+$ are approximated to $O(h^d)$ by their explicit formulas, so $\boldsymbol{C}$ and $\boldsymbol{C}_g$ are approximated to $O(h^{d-1})$.

c) In (16), ℓ and ∇f are approximated to $O(h^d)$ and $\boldsymbol{H}$ is approximated to $O(h^{d-1})$. Therefore, κ_G and κ_H are both approximated to $O(h^{d-1})$. ■

The above analysis did not consider iterative fitting. Following the same argument, if the vertex positions and normals are both approximated to $O(h^{d+1)})$ and the scaled coefficient matrix has full rank, then the coefficients a_{jk} and b_{jk} are approximated to order $O(h^{d-j-k+1)})$ by our iterative fitting. The error in the Hessian would then be $O(h^d)$, so are the estimated curvatures. Therefore, iterative fitting is potentially advantageous, given accurate normals. Note that none of our analyses requires any symmetry of the input data points to achieve convergence. For even-degree fittings, the leading term in the remainder of the Taylor series is odd degree. If the input points are perfectly symmetric, then the residual would also exhibit some degree of symmetry, and the leading-order error term may cancel out as in the centered-finite-difference scheme. Therefore, superconvergence may be expected for even-degree polynomial fittings, and iterative fitting may not be able to further improve their accuracies. Regarding to the principal directions, they are inherently unstable at the points where the maximum and minimum curvatures have similar magnitude. However, if the magnitudes of the principal curvatures are well separated, then the principal directions would also have similar convergence rates as the curvatures.

5 Experimental Results

In this section, we present some experimental results of our framework. We focus on the demonstration of accuracy and stability as well as the advantages of iterative fitting, the weighting scheme, and the safeguarded numerical solver. We do not attempt a thorough comparison with other methods; readers are referred to [Gatzke and Grimm 2006] for such a comparison of earlier methods. We primarily compare our method against the baseline fitting methods in [Cazals and Pouget 2005; Meek and Walton 2000], and assess them for both closed and open surfaces.

5.1 Experiments with Closed Surfaces

We first consider two simple closed surfaces: a sphere with unit radius, and a torus with inner radius 0.7 and outer radius 1.3. We generated the meshes using GAMBIT, a commercial software from Fluent Inc. Our focuses here are the convergence rates with and without iterative fitting as well as the effects of the weighting scheme. For convergence test, we generated four meshes for each surface independently of each other by setting the desired edge lengths to 0.1, 0.05, 0.025, and 0.0125, respectively. Figure 2 shows two meshes that are coarser but have similar unstructured connectivities as our test meshes.

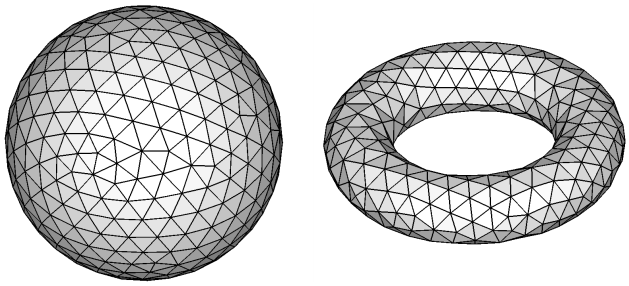


Figure 2: *Sample unstructured meshes of sphere and torus.*

We first assess the computations of normals using fittings of degrees between one and six. Figure 3 shows the errors in the computed normals versus the "mesh refinement level." We label the plots by the degrees of fittings. Let v denote the total number of vertices, and let $\hat{\boldsymbol{n}}_i$ and $\tilde{\boldsymbol{n}}_i$ denote the exact and computed unit vertex normals at the ith vertex. We measure the relative L_2 errors in normals as

$$\sqrt{\frac{1}{v}\sum_1^v \|\tilde{\boldsymbol{n}}_i - \hat{\boldsymbol{n}}_i\|_2^2}.$$

We compute the convergence rates as

$$\text{convergence rate} = \frac{1}{3}\log_2\left(\frac{\text{error of level 1}}{\text{error of level 4}}\right),$$

and show them at the right ends of the curves. In our tests, the convergence rates for normals were equal to or higher than the degrees of fittings. For spheres, the convergence rates of even-degree fittings were about one order higher than predicted, likely due to nearly perfect symmetry and error cancellation.

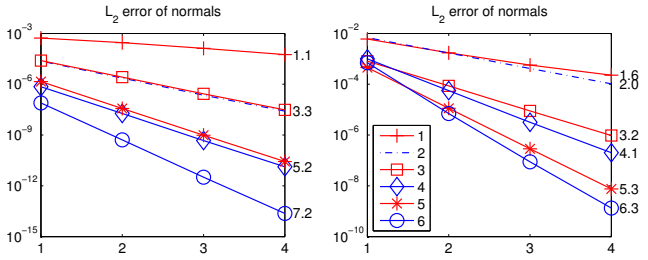


Figure 3: *L_2 errors in computed normals for sphere (left) and torus (right).*

Second, we consider the computations of curvatures. It would be excessive to show all the combinations, so we only show some representative results. Figure 4 shows the errors in the minimum and maximum curvatures for the sphere. Figure 5 shows the errors in the mean and Gaussian curvatures for the torus. In the labels, '+' indicates the use of iterative fitting. Let k_i and $\tilde{k}_i$ denote the exact and computed quantities at the ith vertex, we measure the relative errors in L_2 norm as

$$\frac{\|\tilde{\kappa} - \kappa\|_2}{\|\kappa\|_2} \equiv \sqrt{\sum_{i=1}^v (\tilde{\kappa}_i - \kappa_i)^2} \Big/ \sqrt{\sum_{i=1}^v \kappa_i^2}. \qquad (31)$$

Let d denote the degree of a fitting. In these tests, the convergence rates were $d-1$ or higher as predicted by theory. In addition, even-degree fittings converged up to one order faster due to error cancellation. The converge rates for odd-degree polynomials were about $d-1$ but were also boosted to approximately d when iterative fitting is used. Therefore, iterative fitting is effective in improving odd-degree fittings. In our experiments, iterative fitting did not improve even-degree fittings.

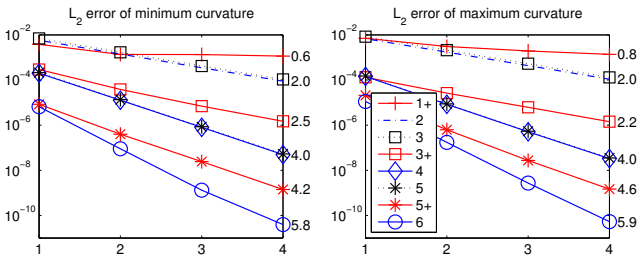


Figure 4: *L_2 errors in computed minimum and maximum curvatures for sphere.*

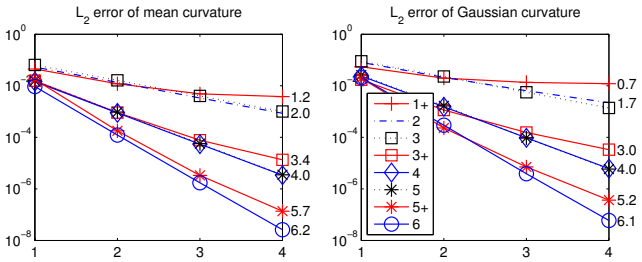


Figure 5: *L_2 errors in computed mean and Gaussian curvatures for torus.*

The preceding computations used the weighting scheme described in Section 4.1, which tries to balance conditioning and error cancellation. This weighting scheme improved the results in virtually all of our tests. Figure 6 shows a representative comparison with and without weighting (as labeled by "·nw" and "·w", respectively) for the maximum curvatures of the sphere and torus.

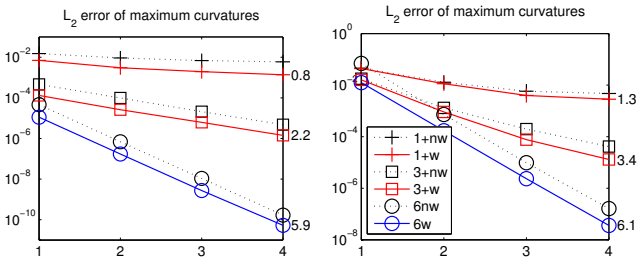


Figure 6: *Comparisons of curvature computations with and without weighting for sphere (left) and torus (right).*

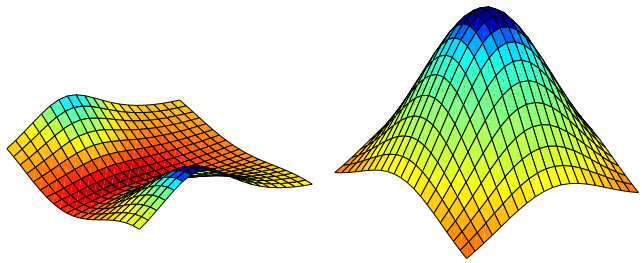
5.2 Experiments with Open Surfaces

We now consider open surfaces, i.e. surfaces with boundary. We focus on the study of stabilities and the effects of boundary and irregular connectivities. We use two surfaces defined by the following functions adopted from [Xu 2004]:

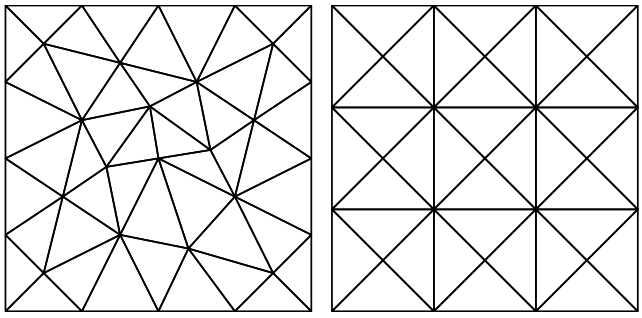
$$z = F_1(x,y) = \frac{1.25+\cos(5.4y)}{6+6(3x-1)^2}, \tag{32}$$

$$z = F_2(x,y) = \exp\left(-\frac{81}{16}\left((x-0.5)^2+(y-0.5)^2\right)\right), \tag{33}$$

where $(x,y) \in [0,1]\times[0,1]$. Figure 7(a-b) shows these surfaces, color-coded by the mean curvatures. We use two types of meshes, including irregular and semi-regular meshes (see Figure 7(c-d)). For convergence study, we refine the irregular meshes using the standard one-to-four subdivision [Gallier 2000, p. 283] and refine the semi-regular meshes by replicating the pattern. We computed the "exact" differential quantities using the formulas in Section 3 in the global coordinate system, but performed all other computations in local coordinate systems. For rigorousness of the tests, we consider both L_2 and L_∞ errors. In addition, border vertices are included in all the error measures, posting additional challenges to the tests. Note that the results for vertices far away from the boundary would be qualitatively similar to those of closed surfaces. We primarily consider fittings of up to degree four, since higher convergence rates may require larger neighborhoods for border vertices (more than 3.5-ring neighbors). To limit the length of presentation, we report only some representative results to cover the aforementioned different aspects.



(a) F_1. (b) F_2.



(c) Irregular mesh. (d) Semi-regular mesh.

Figure 7: *Test surfaces (a-b) and meshes (c-d). Surfaces are color-coded by mean curvatures.*

We first assess the errors in the computed normals for open surfaces. Figure 8 shows the results for F_1 over irregular meshes. We label all the plots and convergence rates in the same way as for closed surfaces. Let d denote the degree of a fitting. All these fittings delivered convergence rate of d or higher in L_2 errors. In L_∞ errors, computed as $\max_i \|\tilde{\boldsymbol{n}}_i - \hat{\boldsymbol{n}}_i\|_2$, the convergence rates were $d-0.5$ or higher, close to theoretical predictions.

Second, we assess the errors in the curvatures. Figure 9 shows the errors in mean curvatures for F_1 over irregular meshes, and Fig-

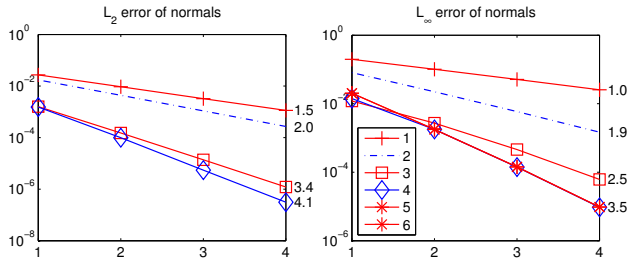


Figure 8: *L_2 (left) and L_∞ (right) errors in computed normals for F_1 over irregular meshes.*

ure 10 shows the errors in Gaussian curvatures for F_2 over semi-regular meshes. Let κ and $\tilde{\kappa}$ denote the exact and computed quantities. We computed the L_2 error using (31) and computed the L_∞ errors as

$$\max_i |\tilde{\kappa}_i - \kappa_i| \,/\, \max\{|\kappa_i|, \epsilon\}, \tag{34}$$

where $\epsilon = 0.01 \max_i |\kappa_i|$ was introduced to avoid division by too small numbers. The convergence rates for curvatures were approximately equal to $d - 1$ or higher for even-degree fittings and odd-degree iterative fittings.

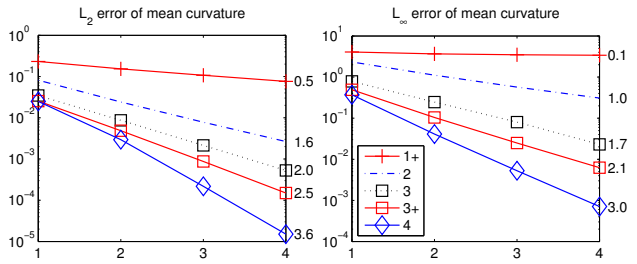


Figure 9: *L_2 (left) and L_∞ (right) in computed mean curvatures for F_1 over irregular meshes.*

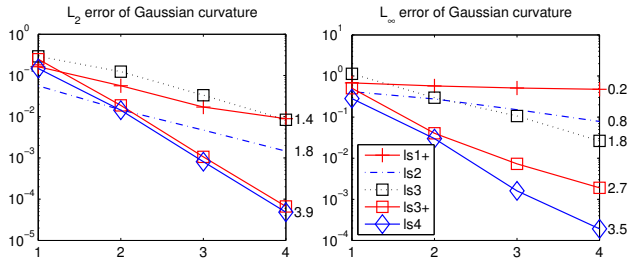


Figure 10: *L_2 (left) and L_∞ (right) in computed Gaussian curvatures for F_2 over semi-regular meshes.*

As noted earlier, the principal directions are inherently unstable when the principal curvatures are roughly equal to each other (such as at umbilic points). In Figure 11, we show the L_2 and L_∞ errors of principal directions for F_1 over semi-regular meshes, where the errors are measured similarly as for normals. This surface is free of umbilic points, and the principal directions converged at comparable rates as curvatures for iterative cubic fitting and quartic fitting.

Finally, to demonstrate the importance and effectiveness of our conditioning procedure, Figure 12 shows comparison of computed normals and mean curvatures with and without conditioning (as labeled by "·nc" and "·c", respectively) for fittings of degrees three, five, and six. Here, the conditioning refers to requiring number of

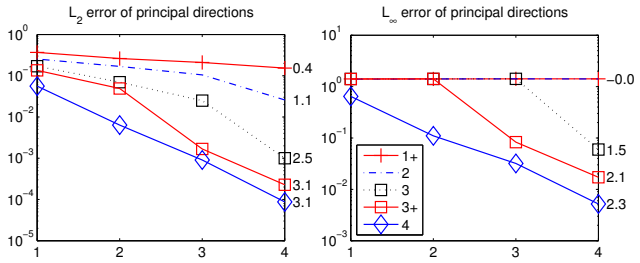


Figure 11: *L_2 (left) and L_∞ (right) in computed principal directions for F_1 over semi-regular meshes.*

points to be 1.5 or more times of the number of unknowns as well as the checking of condition numbers. Without conditioning, the results exhibited large errors for normals and catastrophic failures for curvatures, due to numerical instabilities. With conditioning, our framework is stable for all the tests, although it did not achieve the optimal convergence rate for the sixth-degree fittings due to too small neighborhoods for the vertices near boundary.

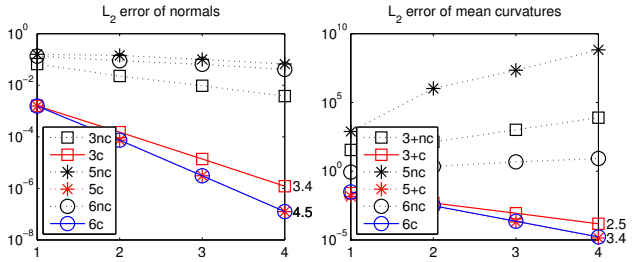


Figure 12: *Comparisons of errors in computed normals (left) and mean curvatures (right) with and without conditioning for F_1 over irregular meshes.*

6 Discussions

In this paper, we presented a computational framework for computing the first- and second-order differential quantities of a surface mesh. This framework is based on the computation of the first- and second-order derivatives of a height function, which are then transformed into differential quantities of a surface in a simple and consistent manner. We proposed an iterative fitting method to compute the derivatives of the height function starting from the points with or without the surface normals, solved by weighted least squares approximations. We improve the numerical stability by a systematic point-selection strategy and QR factorization with safeguard. By achieving both accuracy and stability, our method delivered converging estimations of the derivatives of the height function and in turn the differential quantities of the surfaces.

The main focus of this paper has been on the consistent and converging computations of differential quantities. We did not address the robustness issues for input surface meshes with large noise and singularities (such as sharp ridges and corners). We have conducted some preliminary comparisons with other methods, which we will report elsewhere. One of the major motivating application of this work is provably accurate and stable solutions of geometric flows for geometric modeling and physics-based simulations. We are currently investigating the stability of our proposed methods for such problems. Another future direction is to generalize our method to compute higher-order differential quantities.

Acknowledgements

This work was supported by the National Science Foundation under award number DMS-0713736 and in part by a subcontract from the Center for Simulation of Advanced Rockets of the University of Illinois at Urbana-Champaign funded by the U.S. Department of Energy through the University of California under subcontract B523819. We thank anonymous referees for their helpful comments.

References

AGAM, G., AND TANG, X. 2005. A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE Trans. Vis. Comput. Graph. 11*, 573–583.

ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. 1999. *LAPACK User's Guide*, 3rd ed. SIAM.

CAZALS, F., AND POUGET, M. 2005. Estimating differential quantities using polynomial fitting of osculating jets. *Comput. Aid. Geom. Des. 22*, 121–146.

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted Delaunay triangulations and normal cycle. In *Proc. of 19th Annual Symposium on Computational Geometry*, 312–321.

DE BOOR, C., AND RON, A. 1992. Computational aspects of polynomial interpolation in several variables. *Math. Comp. 58*, 705–727.

DO CARMO, M. P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

DO CARMO, M. P. 1992. *Riemannian Geometry*. Birkhäuser Boston.

GALLIER, J. 2000. *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*. Morgan Kaufmann.

GATZKE, T., AND GRIMM, C. 2006. Estimating curvature on triangular meshes. *Int. J. Shape Modeling 12*, 1–29.

GOLDFEATHER, J., AND INTERRANTE, V. 2004. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph. 23*, 45–63.

GOLUB, G. H., AND VAN LOAN, C. F. 1996. *Matrix Computation*, 3rd ed. Johns Hopkins.

GRINSPUN, E., GINGOLD, Y., REISMAN, J., AND ZORIN, D. 2006. Computing discrete shape operators on general meshes. *Eurographics (Computer Graphics Forum) 25*, 547–556.

HILDEBRANDT, K., POLTHIER, K., AND WARDETZKY, M. 2006. On the convergence of metric and geometric properties of polyhedral surfaces. *Geometria Dedicata 123*, 89–112.

LANCASTER, P., AND SALKAUSKAS, K. 1986. *Curve and Surface Fitting: An Introduction*. Academic Press.

LANGER, T., BELYAEV, A. G., AND SEIDEL, H.-P. 2005. Analysis and design of discrete normals and curvatures. Tech. rep., Max-Planck-Institut Für Informatik.

LANGER, T., BELYAEV, A. G., AND SEIDEL, H.-P. 2005. Exact and approximate quadratures for curvature tensor estimation. In *Poster Proc. of Symposium on Geometry Processing*.

MEEK, D. S., AND WALTON, D. J. 2000. On surface normal and Gaussian curvature approximations given data sampled from a smooth surface. *Comput. Aid. Geom. Des. 17*, 521–543.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. 2002. Discrete differential geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics*, H.-C. Hege and K. Poltheir, Eds., vol. 3. Springer, 34–57.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H.-P. 2004. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 609–612.

OŃATE, E., AND CERVERA, M. 1993. Derivation of thin plate bending elements with one degree of freedom per node. *Engineering Computations 10*, 543.

PETITJEAN, S. 2002. A survey of methods for recovering quadrics in triangle meshes. *ACM Comput. Surveys 34*, 211–262.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Exper. Math. 2*, 15–36.

RAZDAN, A., AND BAE, M. 2005. Curvature estimation scheme for triangle meshes using biquadratic Bézier patches. *Comput. Aid. Des. 37*, 1481–1491.

RUSINKIEWICZ, S. 2004. Estimating curvatures and their derivatives on triangle meshes. In *Proc. of 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, 486–493.

TAUBIN, G. 1995. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. of Int. Conf. on Computer Vision*, 902–907.

THEISEL, H., ROSSL, C., ZAYER, R., AND SEIDEL, H.-P. 2004. Normal based estimation of the curvature tensor for triangular meshes. In *12th Pacific Conf. on Computer Graphics and Applications*, 288–297.

VAN DER SLUIS, A. 1969. Condition numbers and equilibration of matrices. *Numer. Math. 14*, 14–23.

WARDETZKY, M. 2007. Convergence of the cotan formula - an overview. In *Discrete Differential Geometry*, A. I. Bobenko, J. M. Sullivan, P. Schröder, and G. Ziegler, Eds. Birkhäuser Basel.

XU, G. 2004. Convergence of discrete Laplace-Beltrami operators over surfaces. *Comput. Math. Appl. 48*, 347–360.

YANG, Y.-L., LAI, Y.-K., HU, S.-M., AND POTTMANN, H. 2006. Robust principal curvatures on multiple scales. In *Eurographics Symposium on Geometry Processing*.

ZORIN, D. 2005. Curvature-based energy for simulation and variational modeling. In *Int. Conf. on Shape Modeling and Applications*, 196–204.