Eurographics Symposium on Point-Based Graphics (2007)
M. Botsch, R. Pajarola (Editors)

Direct Computing of Surface Curvatures for Point-Set Surfaces

Pinghai Yang and Xiaoping Qian†

Department of Mechanical, Materials and Aerospace Engineering
Illinois Institute of Technology, Chicago, IL, 60616

Abstract

Accurate computing of the curvatures of a surface from its discrete form is of fundamental importance for many graphics and engineering applications. The moving least-squares (MLS) surface from Levin [Lev2003] and its variants have been successfully used to define point-set surfaces in a variety of point cloud data based modeling and rendering applications.

*This paper presents a set of analytical equations for direct computing of surface curvatures from point-set surfaces based on the explicit definition from [AK04a, AK04b]. Besides the Gaussian parameter involved in the MLS definition, these analytical equations allow us to conduct **direct** and **exact** differential geometric analysis on the point-set surfaces without specifying any subjective parameters.*

Our experimental validation on both synthetic and real point cloud data demonstrates that such direct computing from analytical equations provides a viable approach for surface curvature evaluation for unorganized point cloud data.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations

1. Introduction

Rapid advancement of various 3D sensing technologies has led to dense and accurate point cloud data acquired from 3D objects to be readily available. Growing use of such point cloud data in various shape modeling, graphical rendering and engineering design and manufacturing applications necessitates research on direct geometric processing and analysis of point cloud data. This paper concerns the accurate computing of the curvatures of a surface from its discrete point form, a task that is of fundamental importance for many point cloud data based applications.

The moving least-squares (MLS) surface from Levin [Lev03] and its variants have been successfully used to define point-set surfaces in a variety of point cloud data based modeling and rendering applications [AK04a, AK04b, DGS05, DS05].

This paper presents a set of analytical equations for direct computing of surface curvatures from point-set surfaces based on the explicit definition from [AK04a, AK04b]. Besides the Gaussian parameter involved in the MLS definition, these analytical equations allow us to conduct *direct* and *exact* differential geometric analysis on the point-set surfaces without specifying any subjective parameters. As the MLS is being increasingly adopted as a standard definition of the point-set surface, such direct computing of surface curvatures will become growingly important for analyzing point cloud data.

The rest of the paper is organized as follows. In Section 2, we provide a review of relevant work. In Section 3, we introduce MLS surfaces as our underlying representation of the point set surface. In Section 4, we derive closed formulas for curvature computing based on the implicit definition of MLS surfaces. In Section 5, we present experimental examples to illustrate the efficacy of the direct computing of surface curvatures in point-set surfaces. After discussing the effect of Gaussian factor *h* in Section 6, we conclude this paper in Section 7.

2. Literature Review

The curvature information, due to its invariance with respect to rigid transformations, serves as an important description of intrinsic surface characteristics. Hence, the computing of surface curvatures in point-set surfaces has become a fundamental task in many applications such as feature recognition, segmentation and rendering.

The previous work on curvature estimation can be roughly divided into three main categories: curved based, polygonal mesh based and high-order surface based.

† qian@iit.edu

Curve based methods estimate the surface curvatures through the directional curve curvatures, which are estimated through curve fitting [DMSB00, TA05].

Polygonal mesh based methods estimate the curvatures from polygonal meshes [CS92, TAU95]. A comparison of curvature estimation methods based on triangular meshes is available in [SMS03].

High-order surface based methods estimates the curvatures by locally fitting a surface [DB02, YL99], e.g. by least square fitting of a local parametric quadric surface [YL99].

Due to the plurality of prior work in curvature estimation, the above references are by no means a full reflection of all the work done in this area, but a brief analysis of different categories of curvature estimation methods. The proposed approach falls into the categories of surface-based method, but differs from other surface-based methods in that it is based on point-set surfaces.

Our method and the method in [OBS04] are similar in that both use an implicit surface form to compute derivatives. In this paper, we employ an implicit surface defined by [AK04a, AK04b], even though, other variants, e.g. the implicit function in [AA03], can also be used.

Recently, in computer graphics, a number of point based representations, such as surfel [PKKG03, KB04] and MLS [AK04a, AK04b, DGS05, DS05, Lev98, Lev03], have been proposed and proven to be successful in 3D modeling and rendering. Moreover, as a smooth surface defined by a projection process [Lev98, Lev03], MLS surfaces can also be used for point set de-noising, up-sampling, down-sampling, offsetting and so on. Based on a more general definition of this projection MLS in [AK04a, AK04b], a mathematical proof of the convergence of the projection procedure is presented in [DGS05, DS05]. Meanwhile, the resulting MLS surface is proven to be isotopic to the original sampled surface.

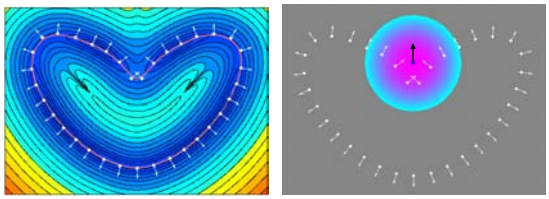
3. Introduction on MLS Surfaces

This section gives a brief introduction on an MLS surface as described in [Lev98, Lev03, AK04a, AK04b, DGS05, DS05], which forms the basis of the subsequent differential geometric analysis of point-set surfaces.

Levin [Lev98,Lev03] initially defined the MLS surface S as the stationary set of a projection operator ψ_P, i.e.,

$$S = \{\mathbf{x} \in R^3 \mid \psi_P(\mathbf{x}) = \mathbf{x}\} \tag{1}$$

Such projection based MLS surfaces are referred to projection MLS surfaces. Amenta and Kil [AK04a, AK04b] gave an explicit definition for projection MLS surfaces as the local minima of an energy function $e(\mathbf{y},\mathbf{a})$ ($\mathbf{y}$ is a position vector and $\mathbf{a}$ is a direction vector) along the directions given by a vector field $\mathbf{n}(\mathbf{x})$, as shown in Figure 1. Based on this definition, they derived a projection procedure for taking a point onto the MLS surface S implied by $\mathbf{n}$ and e, which can be summarized and intuitively illustrated in Figure 2.



(a) (b)

Figure 1: *Illustration of the energy field and vector field for MLS. (a) An energy field $e(\mathbf{y},\mathbf{a})$. (b) The construction of a vector field $\mathbf{n}(\mathbf{x})$.*

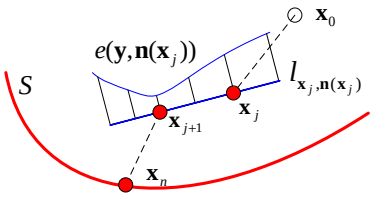


Figure 2: *Illustration of the projection process of the projection-based MLS.*

For details of this projection procedure, please refer to [AK04a, AK04b]. Here we just briefly present two key points in this procedure: Evaluating the normal direction through a vector field **n(x)**; Searching for the local minimum of an energy function $e(\mathbf{y},\mathbf{n}(\mathbf{x}))$.

When evaluating the normal vector, we assume that the normal information at each input point data is available. If the normal information is not readily available as in some applications, we can easily compute this normal information, which we will discuss in the next section. Then we can compute a normal vector for any point with the normals of the nearby sample points, i.e., define a normal vector field as the normalized weighted average of the normals at the sample points. Suppose a normal vector $\mathbf{v}_i$ is assigned to each point $\mathbf{q}_i \in R^3$ of an input point set $\mathbf{Q}$, we have:

$$\mathbf{n}(\mathbf{x}) = \frac{\sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{v}_i \theta(\mathbf{x},\mathbf{q}_i)}{\left\| \sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{v}_i \theta(\mathbf{x},\mathbf{q}_i) \right\|} \tag{2}$$

where

$$\theta(\mathbf{x},\mathbf{q}_i) = e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \tag{3}$$

is a Gaussian weighting function, in which h is a Gaussian scale parameter that determines the width of the Gaussian kernel, which has been discussed extensively in [DGS05,DS05,Pau03]. In these papers, various schemes have been developed in determining h, e.g., defining h as a fraction of the local feature size [DGS05] or varying h based on the local point density [Pau03]. For the sake of simplicity, in this paper, we keep h constant, even though adaptive choice of h, e.g.

through the scheme in [Pau03] would likely lead to more accurate surface estimates.

In the *j*-th iteration of the overall projection process, we need to search the local minimum $\mathbf{x}_{j+1}$ of an energy function along a line $l_{\mathbf{x}_j,\mathbf{n}(\mathbf{x}_j)}$ given by $\mathbf{x}_j$ and $\mathbf{n}(\mathbf{x}_j)$, as shown in Figure 2. Such an energy function $e: R^3 \times R^3 \to R$ can be defined as

$$e(\mathbf{y}, \mathbf{n}(\mathbf{x}_j)) = \sum_{\mathbf{q}_i \in \mathbf{Q}} \left((\mathbf{y} - \mathbf{q}_i)^T \mathbf{n}(\mathbf{x}_j) \right)^2 \theta(\mathbf{y}, \mathbf{q}_i) \qquad (4)$$

To facilitate the search of the local minimum, we can substitute $\mathbf{y} = \mathbf{x}_j + t \cdot \mathbf{n}(\mathbf{x}_j)$ into Eqn. (4) and restate it as a function of variable *t*:

$$e(t) = \sum_{\mathbf{q}_i \in \mathbf{Q}} \left((\mathbf{x}_j + t \cdot \mathbf{n}(\mathbf{x}_j) - \mathbf{q}_i)^T \mathbf{n}(\mathbf{x}_j) \right)^2 \cdot \theta(\mathbf{x}_j + t \cdot \mathbf{n}(\mathbf{x}_j), \mathbf{q}_i)$$

With a vector field **n(x)** and an energy function e, we now have an elegant scheme to project a point onto a MLS surface. Throughout the rest of this paper, this projection based MLS scheme will be used for locally approximating an underlying surface from a set of sample points.

4. Direct Computing of MLS Surface Curvatures

4.1 Prerequisite: Normal Estimation of Measured Data

As we introduced in the previous section, the computation of the normal vector field **n(x)** requires a pre-assigned normal at each point of the input point set **Q**. When this normal information is missing, a statistical analysis of the neighboring samples can be applied to estimate the normal vectors, e.g., an eigenanalysis of the covariance matrix of the point positions.

Let **c** be the weighted centroid of the neighborhood of **q**, i.e.,

$$\mathbf{c} = \sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{q}_i \cdot \theta(\mathbf{q}, \mathbf{q}_i)$$

The 3×3 covariance matrix **C** for the sample point **q** is then given by

$$\mathbf{C} = \sum_{\mathbf{q}_i \in \mathbf{Q}} (\mathbf{q}_i - \mathbf{c}) \cdot (\mathbf{q}_i - \mathbf{c})^T \cdot \theta(\mathbf{q}, \mathbf{q}_i)$$

Since matrix **C** is symmetric and positive semi-definite, all its three eigenvalues λ_0, λ_1 and λ_2 are real-valued. Assuming $\lambda_0 \le \lambda_1 \le \lambda_2$, we can use the eigenvector $\mathbf{v}_0$ of the smallest eigenvalue λ_0 to approximate the surface normal $\mathbf{n}_\mathbf{q}$ at **q** [Pau03].

4.2 Closed Formulas for Curvature Calculation in MLS Surfaces

To calculate the principal curvatures of an MLS surface, we first convert the native form of MLS into an implicit form. It has been proved in [AK04a,DGS05] that the MLS surface is actually the implicit surface given by the zero-level set of the implicit function

$$g(\mathbf{x}) = \mathbf{n}(\mathbf{x})^T \left(\frac{\partial e(\mathbf{y}, \mathbf{n}(\mathbf{x}))}{\partial \mathbf{y}} \Big|_{\mathbf{y}=\mathbf{x}} \right) \qquad (5)$$

where $\mathbf{n}: R^3 \to R^3$ is the vector field defined by Eqn. (2) and $e: R^3 \times R^3 \to R$ is the energy function defined by Eqn. (4). Applying the curvature formulas for implicit surfaces given in [Gol05], we have the Gaussian and mean curvatures of this implicitly defined MLS surface as

$$\begin{cases} k_{Gaussian} = -\dfrac{Det\begin{pmatrix} H(g(\mathbf{x})) & \nabla^T g(\mathbf{x}) \\ \nabla g(\mathbf{x}) & 0 \end{pmatrix}}{\|\nabla g(\mathbf{x})\|^4} \\ k_{Mean} = -\left(\dfrac{\nabla g(\mathbf{x}) \cdot H(g(\mathbf{x})) \cdot \nabla^T g(\mathbf{x})}{\|\nabla g(\mathbf{x})\|^3} - \dfrac{\|\nabla g(\mathbf{x})\|^2 \cdot Trace(H)}{\|\nabla g(\mathbf{x})\|^3} \right) \end{cases} \qquad (6)$$

where

$$\nabla g(\mathbf{x}) = \left(\frac{\partial g(\mathbf{x})}{\partial x} \quad \frac{\partial g(\mathbf{x})}{\partial y} \quad \frac{\partial g(\mathbf{x})}{\partial z} \right)^T$$

is the gradient of $g(\mathbf{x})$ and

$$H(g(\mathbf{x})) = \nabla(\nabla(g(\mathbf{x}))) = \begin{pmatrix} \frac{\partial g(\mathbf{x})}{\partial x \partial x} & \frac{\partial g(\mathbf{x})}{\partial x \partial y} & \frac{\partial g(\mathbf{x})}{\partial x \partial z} \\ \frac{\partial g(\mathbf{x})}{\partial x \partial y} & \frac{\partial g(\mathbf{x})}{\partial y \partial y} & \frac{\partial g(\mathbf{x})}{\partial y \partial z} \\ \frac{\partial g(\mathbf{x})}{\partial x \partial z} & \frac{\partial g(\mathbf{x})}{\partial y \partial z} & \frac{\partial g(\mathbf{x})}{\partial z \partial z} \end{pmatrix}$$

is the Hessian matrix of $g(\mathbf{x})$. Notice that *Det*(**A**) and *Trace*(**A**) denote the determinant and the trace of the matrix **A** correspondingly.

Note, the principal curvatures can be derived from the Gaussian curvature and mean curvature. To further expand the formula of Eqn. (6), we first take the derivative of Eqn. (4) with respect to **y** and setting **y** equal to **x**, which gives

$$\frac{\partial e(\mathbf{y}, \mathbf{n}(\mathbf{x}))}{\partial \mathbf{y}} \Big|_{\mathbf{y}=\mathbf{x}} = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \left(\left((\mathbf{x} - \mathbf{q}_i)^T \mathbf{n}(\mathbf{x}) \right) \cdot \mathbf{n}(\mathbf{x}) - \frac{1}{h^2} \left((\mathbf{x} - \mathbf{q}_i)^T \mathbf{n}(\mathbf{x}) \right)^2 \cdot (\mathbf{x} - \mathbf{q}_i) \right)$$

Substituting it into Eqn. (5), and notice that $(\mathbf{n}(\mathbf{x}))^T \cdot \mathbf{n}(\mathbf{x}) = 1$, we have

$$g(\mathbf{x}) = \mathbf{n}(\mathbf{x})^T \left(\frac{\partial e(\mathbf{y}, \mathbf{n}(\mathbf{x}))}{\partial \mathbf{y}} \Big|_{\mathbf{y}=\mathbf{x}} \right)$$
$$= \sum_{\mathbf{q}_i \in \mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \cdot \left(1 - \frac{1}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x})\right)^2\right) \cdot (\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x}) \quad (7)$$

Suppose that $A = e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}$ and $B = (\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x})$, then Eqn. (7) can be written as

$$g(\mathbf{x}) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2A \cdot \left(B - \frac{1}{h^2} B^3 \right)$$

Hence, the gradient of $g(\mathbf{x})$ can be expressed as

$$\nabla(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2 \cdot \left(\left(\frac{\partial A}{\partial \mathbf{x}} \right) \cdot \left(B - \frac{1}{h^2} B^3 \right) + A \cdot \left(1 - \frac{3}{h^2} B^2 \right) \cdot \left(\frac{\partial B}{\partial \mathbf{x}} \right) \right)$$

Notice that

$$\begin{cases} \frac{\partial A}{\partial \mathbf{x}} = -\frac{2}{h^2} e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \cdot (\mathbf{x}-\mathbf{q}_i) \\ \frac{\partial B}{\partial \mathbf{x}} = \mathbf{n}(\mathbf{x}) + \nabla^T(\mathbf{n}(\mathbf{x})) \cdot (\mathbf{x}-\mathbf{q}_i) \end{cases}$$

We finally have

$$\nabla(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2} \left(\frac{2}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x})\right) \cdot \left(\frac{1}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x})\right)^2 - 1 \right) \cdot (\mathbf{x}-\mathbf{q}_i) + \left(1 - \frac{3}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T \mathbf{n}(\mathbf{x})\right)^2\right) \cdot \left(\mathbf{n}(\mathbf{x}) + \nabla^T(\mathbf{n}(\mathbf{x})) \cdot (\mathbf{x}-\mathbf{q}_i)\right) \right)$$

Suppose that $\mathbf{C} = (\mathbf{x}-\mathbf{q}_i)$ and $\mathbf{D} = \mathbf{n}(\mathbf{x}) + \nabla^T(\mathbf{n}(\mathbf{x})) \cdot (\mathbf{x}-\mathbf{q}_i)$, then the above equation can be written as

$$\nabla(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2A \cdot \left(\frac{2}{h^2} B \cdot \left(\frac{1}{h^2} B^2 - 1 \right) \cdot \mathbf{C} + \left(1 - \frac{3}{h^2} B^2 \right) \cdot \mathbf{D} \right)$$

Hence, the Hessian of $g(\mathbf{x})$ can be expressed as

$$H(g(\mathbf{x})) = \sum_{\mathbf{q}_i \in \mathbf{Q}} 2\left(\frac{2}{h^2} B \cdot \left(\frac{1}{h^2} B^2 - 1 \right) \cdot \mathbf{C} + \left(1 - \frac{3}{h^2} B^2 \right) \cdot \mathbf{D} \right) \cdot \left(\frac{\partial A}{\partial \mathbf{x}} \right)^T + 2A \cdot \left(\frac{6}{h^4} B^2 - \frac{2}{h^2} \right) \cdot \mathbf{C} \cdot \left(\frac{\partial B}{\partial \mathbf{x}} \right)^T + 2A \cdot \frac{2}{h^2} B \cdot \left(\frac{1}{h^2} B^2 - 1 \right) \cdot \left(\frac{\partial \mathbf{C}}{\partial \mathbf{x}} \right) - 2A \cdot \frac{6}{h^2} B \cdot \mathbf{D} \cdot \left(\frac{\partial B}{\partial \mathbf{x}} \right)^T + 2A \cdot \left(1 - \frac{3}{h^2} B^2 \right) \cdot \left(\frac{\partial \mathbf{D}}{\partial \mathbf{x}} \right) \quad (8)$$

For further expansion of this formula, please refer to the Appendix. Substituting the above expressions of gradient and Hessian of $g(\mathbf{x})$ into Eqn. (6), we finally get closed formulas for direct computing of surface curvatures for MLS surfaces.

5. Examples

Based on the above closed formulas, we implemented the direct computing method for surface curvatures of point-set surfaces. We evaluate below the performance of the direct curvature computing quantitatively on synthetic data and qualitatively on real data.

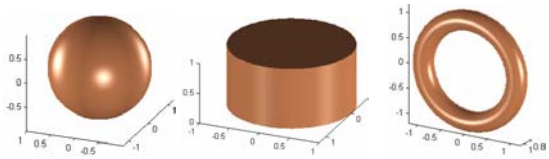
5.1 Synthetic Data

The first two examples, as shown in Figure 3, are based on sampled data from a synthetic sphere of radius 1 and a synthetic cylinder of radius 1 and height 1. The spherical and cylindrical surfaces, due to constant values of principal curvatures k_1 and k_2, are widely accepted as testing examples. By adopting these two examples, a good estimation of the performance can be obtained by examining the means (μ) and the standard deviations (σ) of the difference between the computed principal curvatures and the nominal principal curvatures of the original surface.

The synthetic data of the sphere is composed of 50,000 points, which is generated by near-uniform sampling the sphere. With $h = 0.025$, we got an accurate result in terms of both the mean and the standard deviation of the two principal curvatures, as shown in Table 1.

Table 1. Results of k1 and k2 errors on synthetic data

	k_1 error		*k_2 error*	
	μ	σ	μ	σ
Sphere	0.0000	4.024e-11	0.0000	4.335e-11
Cylinder	0.0000	3.324e-7	0.0000	1.495e-5
Torus	0.0018	0.0041	-8.176e-5	4.141e-4
NURBS surface 1	-0.0006	0.0075	0.0034	0.0074
NURBS surface 2	-0.0016	0.0092	0.0010	0.0111



(a) (b) (c)

Figure 3: *Nominal geometry for the three primitive surfaces. (a) Sphere. (b) Cylinder. (c) Torus.*

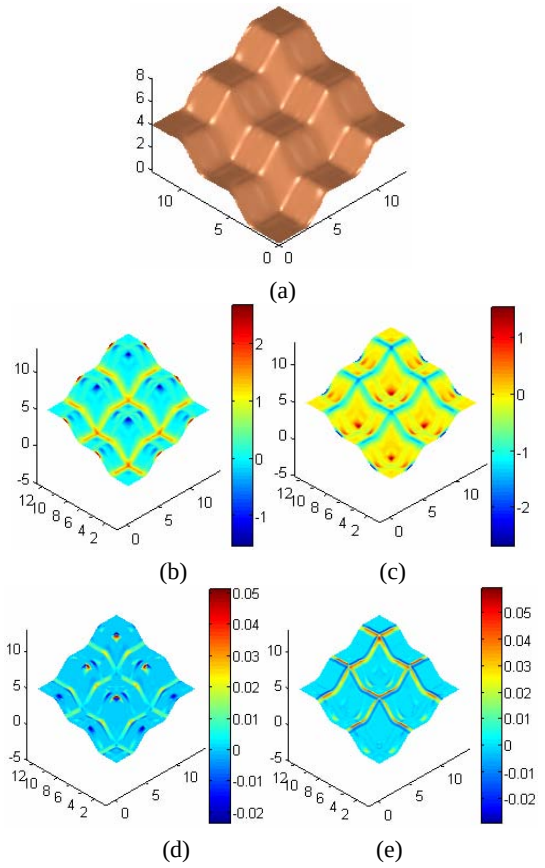


Figure 4: *NURBS Surface 1. (a) Nominal geometry. (b) Computed maximum principal curvatures. (c) Computed minimal principal curvatures. (d) Error in maximum principal curvatures. (e) Error in minimal principal curvatures.*

The synthetic data of the cylinder is composed of uniformly sampled 628×100 points. The result is also accurate with $h = 0.02$ where the μ and σ are computed from 60×60 points on the surface.

These numerical results from our direct computing algorithm compare favorably to some of the existing curvature estimating algorithms [CS92 , TAU95], even though exact experimental conditions in those algorithms are not available.

Besides these two examples, we examine the statistics of the error in principal curvatures with three more complicated surfaces, namely, a synthetic torus, NURBS surface 1 and NURBS surface 2.

The torus surface has an $R = 1$ and $r = 0.2$, which are the distance from the center of the tube to the center of the torus and the radius of the tube correspondingly. The synthetic torus data is composed by 500×100 points sampled uniformly in the parametric domain. With $h = 0.023$, we can get the statistics of errors in principal curvatures in Table 1.

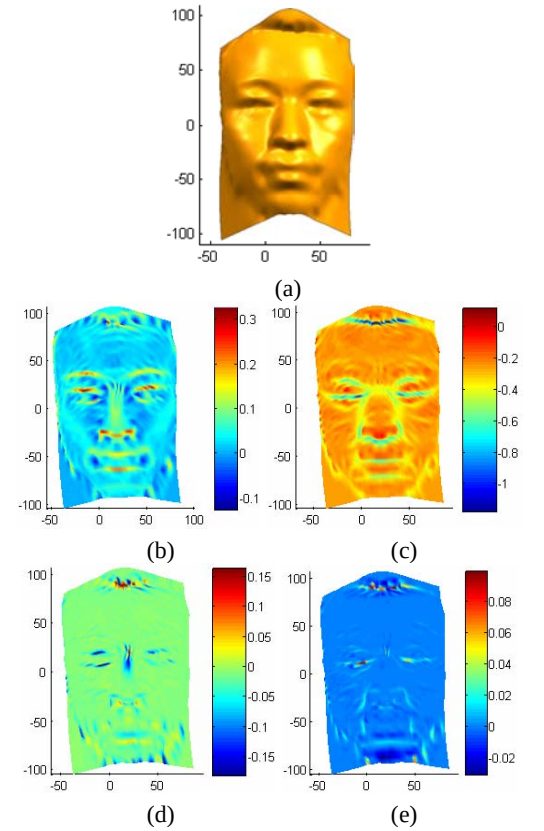


Figure 5: *NURBS Surface 2 (a) Nominal geometry. (b) Computed maximum principal curvatures. (c) Computed minimal principal curvatures. (d) Error in maximum principal curvatures. (e) Error in minimal principal curvatures.*

NURBS surface 1 is a bi-cubic NURBS surface with a 33×33 control lattice, which approximates a point cloud data from Piegl's NURBS book [PT97]. The synthetic sampled data, which contains 1000×1000 points, is generated by uniformly sampling in the parametric domain. The results are shown in Table 1 with $h = 0.06$. As expected, in Figure 4, the errors of computed principal curvatures from the nominal principal curvatures are larger at locations of higher curvature variation. A further discussion on improving the accuracy is available in Section 6.

NURBS surface 2 is chosen to demonstrate how the direct computing works for complex shape and it is a bi-fifth degree NURBS surface with a 117×84 control lattice, which approximates a human face scanned by the Minolta VIVID910. The synthetic data contains 900×900 uniformly sampled in the parametric domain. The results at $h = 2$ are shown in Table 1 and Figure 5. Similar to the NURBS surface 1, larger errors are observed at locations of high curvature variation.

5.2 Synthetic Data with Gaussian Noise

To validate the robustness of our algorithm in handling noisy point sets, we generate two sets of noisy data by adding random noise with different amplitudes of standard deviations (i.e., 0.0001 and 0.001) to the synthetic torus data. The resulting mean errors and the standard deviations for the two principal curvatures are shown in Table 2 and Figure 6. These results suggest our algorithm is stable against noise.

Table 2. Results of k_1 and k_2 errors of torus data with different level of noise

noise	*k1 error*		*k2 error*	
	μ	*σ*	*μ*	*σ*
0.0000	*0.0018*	*0.0041*	*-8.176e-5*	*4.141e-4*
0.0001	*0.0666*	*0.0648*	*0.0134*	*0.0178*
0.0010	*0.0846*	*0.1060*	*0.0517*	*0.0889*

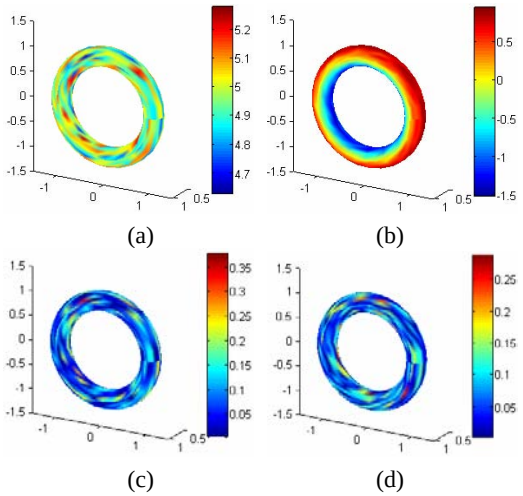


Figure 6: *Synthetic data of a torus with Gaussian noise (standard deviation is 0.001). (a) Maximum principal curvatures. (b) Minimal principal curvatures. (c) Error in maximum principal curvatures. (d) Error in minimal principal curvatures.*

5.3 Real Data

A sculptured artifact scanned by the Minolta VIVID910 is used to further examine how the direct computing of the surface curvatures perform on realistic scanned point cloud data. After the scanning, the angel's head model contains 91,284 points (Figure 7(a)) and has a size of 15×6×1.5 (inches).

From the graphical illustration of the principal curvature distribution (Figure 7(b) and (c)), we can qualitatively see that the curvature distribution matches well with the surface shape.

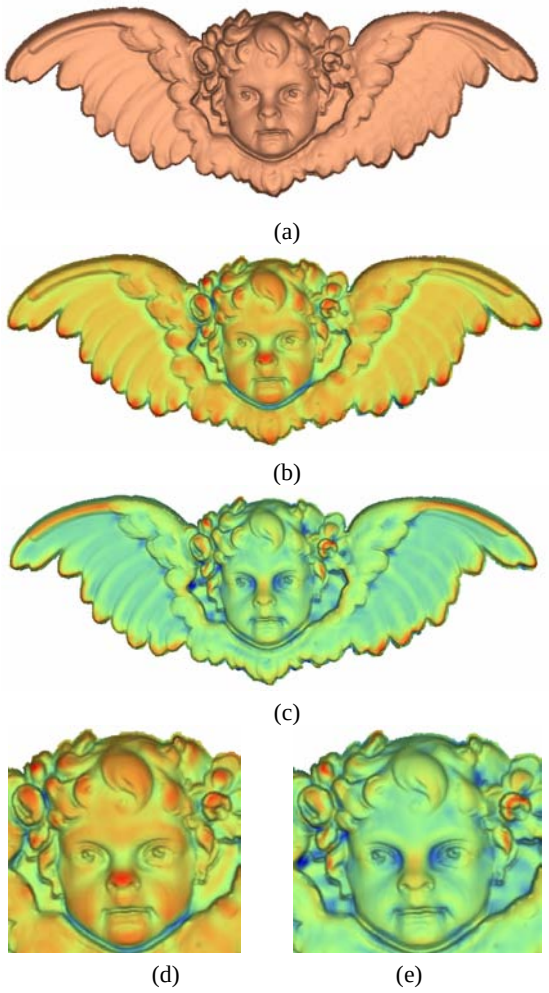


Figure 7: *Sculpture of an Angel's Head. (a) Scanned point data. (b) Maximum principal curvatures. (c) Minimal principal curvatures. (d) Zoom in of (b) at the face region. (e) Zoom in of (c) at the face region.*

6. Discussion

In the above experiments, for the sake of simplicity, we have chosen a constant *h* for each example. However at locations of large curvature variation, there are large errors in the computed curvatures, as demonstrated in the examples of NURBS surface 1 and NURBS surface 2. In fact, both the sampling density and Gaussian parameter affect the resulting accuracy of the computed curvatures. For example, using the synthetic data sampled from the NURBS surface 1 at different sampling densities, we can get curvatures of different accuracy, as characterized by the mean error and the standard deviation from the nominal curvatures shown in Figure 8. We can see that the denser the sampling is the smaller the mean errors and standard deviations are.

Notice that in this experiment, each scale factor h_i is linear to the local sampling density:

$$h_i = \frac{60}{n_i}$$

where n_i is the number of sampling points in u and v directions, which is shown in Figure 8 as the x coordinates.

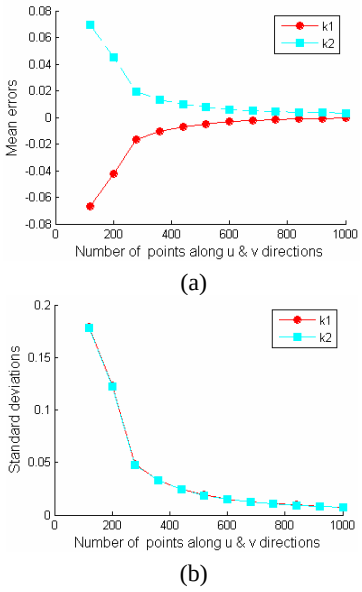


Figure 8: *Error statistics of NURBS surface 1 with different sampling densities. (a) Mean error. (b) Standard deviation.*

However, the increase of sampling density alone would not be able to completely eliminate the uneven distribution of error, as shown in Figure 4(d) and (e).

One method to resolve this problem is to set the Gaussian factor h such that it is adapted to the local sampling density and the local curvature. Unlike the scale factor h defined only by the local sampling density, this adaptive scheme for h would make the resulting MLS surface sensitive to features of the original surface.

We finish this section with an experiment about another synthetic data, which contains 250×250 points uniformly sampled from the NURBS surface 1. By setting h = 0.24, we can get the nominal principal curvature maps and the corresponding error maps as shown in Figure 9(a), (b), (c) and (d). Again, we observe the eight-line error pattern, where the curatures errors and the curvature variation are both higher than the neighboring regions. By doubling the sampling density and halving the Gaussian factor h at four of these eight lines, we can see a dramatic relief of these patterns, as shown in Figure 9(e) and (f), which indicates that we can significantly reduce the number of sampling points to achieve the same accuracy in curvature computing by adopting the adaptive sampling density and Gaussian factor h.

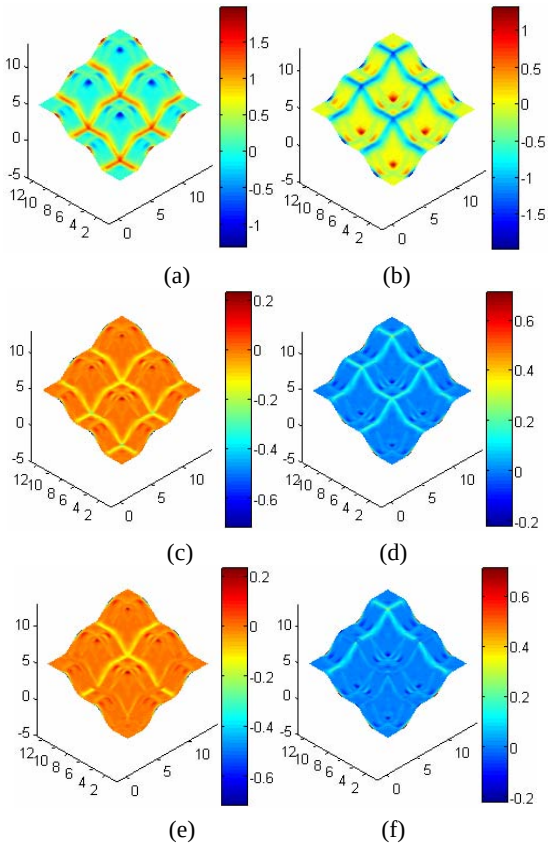


Figure 9: *Adaptive NURBS Surface 1. (a) Computed maximum principal curvatures. (b) Computed minimal principal curvatures. (c) Error in maximum principal curvatures. (d) Error in minimal principal curvatures. (e) Error in maximum principal curvatures with adaptive h. (f) Error in minimal principal curvatures with adaptive h.*

7. Conclusion and Future work

We have presented a set of analytical equations for differential geometric analysis of point-set surfaces. These equations are derived from the explicit definition of moving least-squares surfaces for point cloud data. Our experiments on both nominal and noisy data validate the correctness of these equations. They also demonstrate that direct computing of principal curvatures in point-set surfaces from these analytical equations is viable for a variety of both nominal and noisy data.

As the MLS is being increasingly adopted as a standard definition for the point-set surface, such direct computing of surface curvatures will become growingly important for analyzing point cloud data.

It is important to note that the key to accurate computing of principal curvatures in the direct computing approach lies in the definition of MLS itself. The accuracy of the computed curvatures depends on the local density of point cloud data and the Gaussian parameter used in defining the MLS surface, not on the

curvature computing procedure or the analytical equations. As such, the direction for future research will include a rigorous understanding on how sample density and Gaussian parameter affect the resulting MLS surface and the corresponding surface curvature properties.

References

[AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. *In Proceedings of EG Symposium on Geometry Processing* (2003), pp. 245-254.

[AK04a] AMENTA N., KIL Y. J.: Defining point-set surfaces. *ACM Trans. Graph.* (2004), vol. 23(3), pp. 264-270.

[AK04b] AMENTA N., KIL Y. J.: The domain of a point set surface. *Eurographics Workshop on Point-based Graphics* (2004), pp. 139-147.

[CS92] CHEN X., SCHMITT F.: Intrinsic surface properties from surface triangulation. *Proc. European Conf. Computer Vision (ECCV '92)* (1992), pp. 739-743.

[DB02] DOUROS I., BUXTON B.F.: Three-dimensional surface curvature estimation using quadric surface patches. *Scanning 2002 Proceedings* (2002), Paris.

[DGS05] DEY T. K., GOSWAMI S., SUN, J.: Extremal surface based projections converge and reconstruct with isotopy. *Technical Report OSU-CISRC-4-05-TR25* (April 2005).

[DMSB00] DESBRUN M., MEYER M., SCHRODER P., BARR A.: Discrete differential-geometry operators in nd. *CIT Technical report* (2000).

[DS05] DEY T. K., SUN J.: Adaptive mls surfaces for reconstruction with guarantees. *Proc. Eurographics Symposium on Geometry Processing* (2005), pp. 43-52.

[Gol05] GOLDMAN R.: Curvature formulas for implicit curves and surfaces," *Computer Aided Geometric Design* (October 2005), vol. 22(7), pp. 632-658.

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers & Graphics* (2004), vol. 28(6), pp. 801-814.

[Lev03] LEVIN D.: Mesh-independent surface interpolation. *In*: Brunnett, G., Hamann, B., Muller, H. and Linsen, L. (Eds.), *Geometric Modelling for Scientific Visualization* (2003), Springer-Verlag, pp. 37-49.

[Lev98] LEVIN D.: The approximation power of moving least-squares. *Mathematics of Computation 67* (1998), pp. 1517-1531.

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.* (2004), vol. 23(3), pp. 609-612.

[Pau03] PAULY M.: *Point Primitives for Interactive Modeling and Processing of 3d Geometry*, Ph.D. thesis. ETH Zurich, 2003.

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modelling with point-sampled geometry. *ACM Trans. Graph. (SIGGRAPH'03)* (2003), Vol. 22(3), pp. 641-650.

[PT97] PIEGL L., TILLER W.: *The NURBS Book*. Springer-Verlag, New York, 1997.

[SMS03] SURAZHSKY T., MAGID E., SOLDEA O., ELBER G., RIVLIN E.: A comparison of gaussian and mean curvatures estimation methods on triangular meshes. *Proceedings of IEEE International Conf. on Robotics and Automation* (2003), pp. 1021-1026.

[TA05] TANG X., AGAM G.: A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE Transactions on Visualization and Computer Graphics* (2005), vol. 11(5), pp. 573-583.

[TAU95] TAUBIN G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. *Proceedings of the Fifth International Conference on Computer Vision* (June 1995), pp. 902.

[YL99] YANG M., LEE E.: Segmentation of measured point data using a parametric quadric surface approximation. *Computer-Aided Design* (1999), vol. 31(7), pp. 449-457.

Appendix: Expansion of the Hessian matrix

For variables C and D defined in Section 4, we have

$$\begin{cases} \dfrac{\partial \mathbf{C}}{\partial \mathbf{x}} = \mathbf{I} \\ \dfrac{\partial \mathbf{D}}{\partial \mathbf{x}} = \nabla(\mathbf{n}(\mathbf{x})) + \nabla^T(\mathbf{n}(\mathbf{x})) + \nabla^T\left(\nabla(\mathbf{n}(\mathbf{x}))\right)\cdot(\mathbf{x}-\mathbf{q}_i) \end{cases}$$

Substituting them into Eqn.(8), we finally have

$$\begin{aligned} H(g(\mathbf{x})) = \nabla\left(\nabla\left(g(\mathbf{x})\right)\right) = \sum_{\mathbf{q}_i\in\mathbf{Q}} & -\frac{4}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{2}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)\right. \\ & \cdot\left(\frac{1}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)^2-1\right)\cdot(\mathbf{x}-\mathbf{q}_i)+\left(1-\frac{3}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)^2\right) \\ & \left.\cdot\left(\mathbf{n}(\mathbf{x})+\nabla^T(\mathbf{n}(\mathbf{x}))\cdot(\mathbf{x}-\mathbf{q}_i)\right)\right)\cdot(\mathbf{x}-\mathbf{q}_i)^T \\ & +2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(\frac{6}{h^4}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)^2-\frac{2}{h^2}\right)\cdot(\mathbf{x}-\mathbf{q}_i) \\ & \cdot\left(\mathbf{n}^T(\mathbf{x})+(\mathbf{x}-\mathbf{q}_i)^T\cdot\nabla(\mathbf{n}(\mathbf{x}))\right) \\ & +\frac{4}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)\cdot\left(\frac{1}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)^2-1\right)\cdot\mathbf{I} \\ & -\frac{12}{h^2}e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)\cdot\left(\mathbf{n}(\mathbf{x})+\nabla^T(\mathbf{n}(\mathbf{x}))\cdot(\mathbf{x}-\mathbf{q}_i)\right) \\ & \cdot\left(\mathbf{n}^T(\mathbf{x})+(\mathbf{x}-\mathbf{q}_i)^T\cdot\nabla(\mathbf{n}(\mathbf{x}))\right) \\ & +2e^{-\|\mathbf{x}-\mathbf{q}_i\|^2/h^2}\left(1-\frac{3}{h^2}\left((\mathbf{x}-\mathbf{q}_i)^T\mathbf{n}(\mathbf{x})\right)^2\right) \\ & \cdot\left(\nabla(\mathbf{n}(\mathbf{x}))+\nabla^T(\mathbf{n}(\mathbf{x}))+\nabla^T\left(\nabla(\mathbf{n}(\mathbf{x}))\right)\cdot(\mathbf{x}-\mathbf{q}_i)\right) \end{aligned}$$

where $\mathbf{I}$ is the identity matrix and $\nabla(\mathbf{n}(\mathbf{x}))$ is the 3×3 Jacobian matrix of $\mathbf{n}(\mathbf{x})$ defined as:

$$\nabla(\mathbf{n}(\mathbf{x})) = \left(\mathbf{I}-\frac{\left(\sum_{\mathbf{q}_i\in\mathbf{Q}}\mathbf{v}_i\theta(\mathbf{x},\mathbf{q}_i)\right)\cdot\left(\sum_{\mathbf{q}_i\in\mathbf{Q}}\mathbf{v}_i\theta(\mathbf{x},\mathbf{q}_i)\right)^T}{\left\|\sum_{\mathbf{q}_i\in\mathbf{Q}}\mathbf{v}_i\theta(\mathbf{x},\mathbf{q}_i)\right\|^2}\right) \cdot\frac{\sum_{\mathbf{q}_i\in\mathbf{Q}}\mathbf{v}_i\dfrac{\partial\theta(\mathbf{x},\mathbf{q}_i)}{\partial\mathbf{x}}}{\left\|\sum_{\mathbf{q}_i\in\mathbf{Q}}\mathbf{v}_i\theta(\mathbf{x},\mathbf{q}_i)\right\|}$$