Robust Principal Curvatures using Feature Adapted Integral Invariants

Yu-Kun Lai*
Tsinghua University
Beijing, China

Shi-Min Hu†
Tsinghua University
Beijing, China

Tong Fang‡
Siemens Corporate Research, Inc.
Princeton, NJ, USA

Abstract

Principal curvatures and principal directions are fundamental local geometric properties. They are well defined on smooth surfaces. However, due to the nature as higher order differential quantities, they are known to be sensitive to noise. A recent work by Yang et al. combines principal component analysis with integral invariants and computes robust principal curvatures on multiple scales. Although the freedom of choosing the radius r gives results on different scales, in practice it is not an easy task to choose the most appropriate r for an arbitrary given model. Worse still, if the model contains features of different scales, a single r does not work well at all. In this work, we propose a scheme to automatically assign appropriate radii across the surface based on local surface characteristics. The radius r is not constant and adapts to the scale of local features. An efficient, iterative algorithm is used to approach the optimal assignment and the partition of unity is incorporated to smoothly combine the results with different radii. In this way, we can achieve a better balance between robustness and accuracy of feature locations. We demonstrate the effectiveness of our approach with robust principal direction field computation and feature extraction.

1 Introduction

Principal curvatures and principal directions are fundamental concepts of local differential geometry [do Carmo 1976]. They are well defined on smooth, differentiable surfaces. However, in discrete cases like the most widely used triangle meshes, special considerations should be addressed, leading to the so-called *discrete differential geometry*, a rather intensive research topic in recent years. Discrete differential geometry extends classical theories and computational methods from smooth surfaces to discrete settings and is widely used in various geometry processing applications, including anisotropic polygonal remeshing (e.g. [Alliez et al. 2003; Lai et al. 2008]), non-photorealistic rendering (e.g. [Hertzmann and Zorin 2000]), crest lines extraction (e.g. [Ohtake et al. 2004]) etc.

Although principal curvatures and principal directions can be computed using discrete differential geometry [Desbrun et al. 2002], principal curvatures are high-order differential quantities and the computation on practical triangle meshes is not robust, due to the existence of noise or unwanted small-scale geometric details. Thus, in practice, a smoothing operator is usually applied in prior to the computation of differential quantities. However, the smoothing operator changes the underlying geometry and may be problematic by itself in certain extreme cases, like spikes on meshes etc. Moreover, such operators also have the risk of smoothing out significant features.

This work is based on the robust estimate of principal curvatures and principal directions using principal component analysis of local neighborhood [Yang et al. 2006; Pottmann et al. 2007]. Different neighborhoods could be used similarly and we use integral invariants of the ball neighborhood throughout the paper. The PCA-based integral invariants are defined at a particular radius r. When the r is approaching zero, the integral invariants are related to the traditional principal curvatures and principal directions. For a particular radius r, the method can be used to compute curvature-like quantities at the specific scale. Using a larger r, obtained results tend to be more robust to noise, while at the risk of losing relatively small-scale features. This provides the user some freedom to estimate principal curvatures at a desired scale; however, for many practical cases, users may not be able to choose the most appropriate scale. Worse still, if the model contains features of different scales (this is quite usual for practical models), it is possible that a single radius r never suits the overall model. In this work, we propose a systematic way to automatically decide appropriate neighborhood radius for a given model; treating radius as a function of surface $r(S)$ that varies to adapt to the nature of local features. To achieve this, integral invariants are computed in such a way that the local neighborhood size r is adjusted to be comparable to the minimal principal radius. This is accomplished by using an iterative process to approach the optimal distribution of $r(S)$. To significantly improve the performance, the integral invariants at a set of discrete scales are pre-computed and combined using the partition of unity to form the results, which also improves the smoothness of the output.

In Section 2, related work of principal curvature computation is briefly reviewed. Observation and motivation are discussed in Section 3. Detailed algorithm is presented in Sec. 4. Experimental results of the proposed method are given in Section 5, while conclusions and discussions for future work are given in Section 6.

2 Related Work

Principal curvatures on discrete meshes can be computed with discrete differential geometry. Original work on discrete differential geometry usually computes in one- or two-ring neighborhood [Desbrun et al. 2002]. However, certain methods derived from discrete differential geometry can actually be extended to work on a larger neighborhood of the mesh, leading to multi-scale results and improved robustness. For example, the widely used normal cycles method by Cohen-Steiner and Morvan [2003]. For a given edge, the principal direction can be naturally defined along and across the edge

*e-mail:yukun.lai@gmail.com
†e-mail:shimin@tsinghua.edu.cn
‡e-mail:tong.fang@siemens.com

vector. To compute the principal curvatures at a particular vertex, edges within a neighborhood of the vertex are integrated (summed) over. It should be noted that even if such methods can be used with larger neighborhood, producing multi-scale results are not the aim of the method and the results are still not really robust since they are apparently affected by the connectivity of the mesh.

Local fitting is another approach to estimate local curvature information on discrete meshes [Cazals and Pouget 2003; Goldfeather and Interrante 2004; Jiao and Zha 2008]. The basic idea is to first fit a local neighborhood of the mesh with some analytical (e.g. polynomial) surface. After that, principal curvatures and principal directions can be estimated on the smooth fit surface, instead of given discrete surfaces. Generally speaking, although a scale of local neighborhood can be chosen, as indicated by the comparative study in [Yang et al. 2006], the methods may not behave well when the radius is getting large or in cases the local surface patch cannot be well approximated by the analytical representation.

Some methods consider adaptive weighting scheme to improve the robustness, which is similar to our work. For example, Kalogerakis et al. [2007] uses curvature tensor fitting together with statistical Iteratively Reweighted Least Squares for adaptive estimation of principal curvatures. Our work also has some relationship to the work by Mitra et al. [2004], which considers the problem of computing appropriate radius for the estimation of normals of point clouds, however, the problems and methods are different.

The concepts of integral invariants for the curve and surface analysis have been used in different settings, including the molecular shape analysis [Connolly 1986], 2D curve matching [Manay et al. 2004] and feature extraction [Clarenz et al. 2004]. Recently, a more systematic work using principal component analysis (PCA) based integral invariants for robust curvature estimation is proposed in [Pottmann et al. 2007; Yang et al. 2006]. In the following, we will briefly review part of the method mostly related to this work. We assume that the ball neighborhood is used. A ball of radius r is placed at an arbitrary place $\mathbf{p}$ on the mesh surface Φ and denoted as $B_r(\mathbf{p})$. The interior part of the surface can be considered as a solid volume (denoted as D). Ball neighborhood is defined as $N_b^r := D \cap B_r(\mathbf{p})$. Principal component analysis of this neighborhood leads to some integral invariants related to principal curvatures. The center of this neighborhood is $\mathbf{s} := \int_{N_b^r} \mathbf{x} d\mathbf{x} / \int_{N_b^r} d\mathbf{x}$, and the covariance matrix is $\mathbf{J}(N_b^r) := \int_{N_b^r} (\mathbf{x}-\mathbf{s})(\mathbf{x}-\mathbf{s})^T d\mathbf{x}$. The three eigenvalues and eigenvectors of $\mathbf{J}$ can be computed. We denote the eigenvalues in descending order as λ_1, λ_2 and λ_3, with corresponding eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$. $\mathbf{e}_3$ corresponds to the normal direction, and $\mathbf{e}_1$ and $\mathbf{e}_2$ correspond to the two principal directions. λ_1 and λ_2 are related to principal curvatures when r tends to zero. Ignoring the higher order terms, principal curvatures can be computed with λ_1, λ_2 and the radius r [Yang et al. 2006]:

$$\kappa_1^r := \frac{6}{\pi r^6}(\lambda_2 - 3\lambda_1) + \frac{8}{5r}, \kappa_2^r := \frac{6}{\pi r^6}(\lambda_1 - 3\lambda_2) + \frac{8}{5r}.$$

For practical uses, the radius r in these equations is of significant importance. Larger r tends to produce smoother results and results more robust to noise, while at the risk of smoothing out small-scale features that may be of interest to the users. This is extremely important if the model contains both larger and smaller parts where no single r suits the need well (imagine stitching together a model and a scaled-up version and considering it as a single model). This work considers to use varied radii across the surface to adapt to the local characteristic.

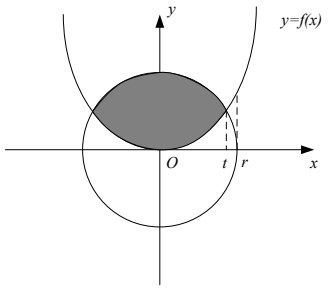


Figure 1: *Illustration of planar area integral invariant.*

3 Motivation and Observation

Differential geometry and their discrete counterpart are well defined on smooth surfaces or their sufficiently well discrete approximation. However, real geometry objects are often noisy. Integral invariants are introduced in such cases to produce robust curvature estimation. Intuitively, if the surface is free from noise, the most accurate results are obtained, with the minimum possible radius. If, on the other hand, the surface is noisy, larger radius has the effect to smooth out insignificant perturbations. However, this effect is not the same for planar regions and sharp features. Sharp features are more easily smoothed out. As an extreme case, it is clear that for noisy planar regions, maximum possible radius is most preferred.

Most practical models are neither noise-free nor completely planar, so too large or too small scale radius is usually not ideal. Computation of PCA-based integral invariants of volumes is rather involved. The computation on area integrals of 2D curves is simpler, but also gives reasonable hints to extend to the 3D case.

From a typical textbook of differential geometry [do Carmo 1976], in a canonical coordinate, if we assume that the underlying curve is sufficiently smooth and the derivative of the curvature k w.r.t. x can be safely omitted, a planar cure at a particular point can be expressed as $y = f(x) = \frac{k}{2}x^2$, where k represents the curvature at the point. As illustrated in Fig. 1, the planar area integral invariant is defined as the area of the shaded region, which can be computed as: $I^r = \frac{\pi}{2}r^2 - 2I_1^r + 2I_2^r$, where r is the scale radius, and I_1^r and I_2^r are correction terms

$$I_1^r = \int_0^r f(x)dx = \frac{k}{6}r^3, I_2^r = \int_t^r (f(x) - \sqrt{r^2 - x^2})dx. \quad (1)$$

Here, t is the x-coordinate of the intersection point. Without loss of generality, assume that k is non-negative. By Taylor expansion, t has the expansion:

$$t = \frac{\sqrt{2}}{k}\sqrt{-1 + \sqrt{1 + k^2 r^2}} = r - \frac{1}{8}k^2 r^3 + O(r^5). \quad (2)$$

To estimate I_2^r, we have to evaluate two separable integrations. For the former term, note that $t^3 = r^3 - \frac{3}{8}k^2 r^5 + O(r^6)$, $t^4 = r^4 + O(r^6)$, we have

$$\int_t^r f(x)dx = \frac{k}{6}(r^3 - t^3) = \frac{1}{16}k^3 r^5 + O(r^6). \quad (3)$$

The latter term $\int_t^r \sqrt{r^2 - x^2}dx$ can be computed similarly using Taylor expansion and induction. The major term is also of the same order $O(k^3 r^5)$. Thus, we have $I_2^r = ck^3 r^5 +$

$O(r^6)$, where c is a positive constant ≈ 0.02 (sufficient for our estimation purpose). A more accurate estimation is then obtained as

$$I^r = \frac{\pi}{2}r^2 - \frac{k}{3}r^3 + 2ck^3r^5 + O(r^6). \quad (4)$$

For the usual integral invariant, we ignore $O(r^4)$ term and estimate the curvature k as $\bar{k} = 3(\frac{\pi}{2}r^2 - I^r)/r^3$. Certainly only when r is sufficiently small, this approximation is close to the real k. By this estimation, we can now omit $O(r^6)$ term, and have a more accurate equation, from which we derive the relative systematic error for $\bar{k}$:

$$\frac{\bar{k} - k}{k} = -6ck^2r^2. \quad (5)$$

It is clear that the relative systematic error is proportional to $(|k|r)^2$ (for general k). Estimating the noise level of a given curve or surface is difficult, however, if we allow a certain amount of relative systematic error ε, the optimal scale radius against noise is

$$r = \sqrt{\frac{\varepsilon}{6c}}\frac{1}{|k|} = \alpha\frac{1}{|k|}. \quad (6)$$

In practice α may be chosen as around 1.0, which means ε is around 12%. For more noisy dataset, we should increase α, while for cleaner dataset, smaller α is reasonable. Note that this formula coincides with the previous observations. For planar regions where $k = 0$, maximum possible r is preferred. For noise-free dataset, minimum possible r helps to reduce the systematic error.

In the case of PCA-based integral invariants of 3D surfaces, the computation is more involved. We thus take the heuristic of choosing $r(\mathbf{p})$ to be comparable to the *minimal* principal radius at the local place:

$$r(\mathbf{p}) = \alpha \min\left\{\frac{1}{|\kappa_1(\mathbf{p})|}, \frac{1}{|\kappa_2(\mathbf{p})|}\right\}, \quad (7)$$

4 Algorithm

In this section, we will discuss the algorithm in detail. As noted before, we treat r as a function of position $\mathbf{p}$ over the surface Φ (denoted as $r(\mathbf{p})$). $r(\mathbf{p})$ should coincide with the nature of the local neighborhood, as given in Eqn. 7. α is a constant for the bias of using slightly larger or smaller radius globally. To our purpose, α is usually chosen around 1. Larger α increases robustness and smaller α increases the ability to capture local features and accurately locate features. For all the examples shown in the paper, a constant $\alpha = 0.9$ is used and produces reasonable results.

This definition is intuitive, however, to ensure robustness, κ_1 and κ_2 in Eqn. 7 should be estimated also with a particular radius r. To solve this, an appropriate iterative algorithm is proposed here to approach the optimal assignments. We estimate κ_1 and κ_2 with a preassigned r, and the computed values of principal curvatures may be used to update the assignment of r in turn. During this process, principal curvatures at each vertex on different scales need to be accessed frequently. In Section 4.1, an efficient approximation algorithm to estimate principal curvatures at different scales is discussed, while the iterative algorithm for appropriate radius r and related principal curvatures is detailed in Section 4.2. Post processing stage is introduced to produce more smooth distribution of scales (see Section 4.3). The overall pipeline of our method is as follows:

1. Compute the usual integral invariants at several different scales for all the vertices [Yang et al. 2006].
2. Initialize the scale radius at each vertex, see Section 4.2.
3. For each vertex v, repeat the following until convergence, see Section 4.2:
   (a) Estimate principal curvatures at v, using the current scale radius by interpolation of the precomputed values, see Section 4.1.
   (b) Update the scale radius r based on the estimated principal curvatures.
4. Globally smooth the scale radius distribution over the whole model, see Section 4.3.
5. Estimate principal curvature information using computed scale radius.

4.1 Approximation of Principal Curvatures

On discrete meshes, PCA-based integral invariants with ball neighborhood of a particular size r requires computation of barycenter $\mathbf{s}$ and covariance matrix $\mathbf{J}$ at each vertex. As noted in [Pottmann et al. 2007], the integration can be converted to 10 convolutions, which can be greatly sped up by using Fast Fourier Transforms (FFT). However, computing integral invariants at an individual vertex, or at vertices with different radii is very slow. To solve this, we propose to precompute a discrete set of principal tensors $\mathbf{J}^{r_k}(\mathbf{p}_i)$ for *all* the vertices $\{v_i\}$ at position $\{\mathbf{p}_i\}$, but with different radii r_k. Then, when we want to compute the principal tensor for a vertex v with position $\mathbf{p}$ at a specific radius r, we interpolate the covariance matrix $\mathbf{J}^r(\mathbf{p})$ with principal tensors at the same position $\mathbf{p}$ but different radii.

Assume we precompute principal tensors at n radii $r_1, r_2, \ldots, r_n$. Assume that $r_1 < r_2 < \cdots < r_n$. To ensure that such approximation leads to smooth results, we compute a set of weights $w_i(r)$, satisfying $\sum_{i=1}^n w_i(r) = 1$, using the idea of the partition of unity [Ohtake et al. 2003]. To compute w_i, we first define a set of Gaussian radial basis functions $h_i(r)$ representing the relative impact from the results obtained at radius r_i: $h_i(r) := \exp\left\{-\frac{(r-r_i)^2}{\sigma^2}\right\}$, where σ controls how the effect decreases when the distance increases. σ is usually chosen at the scale comparable to the interval of radius, i.e. $(r_{i+1} - r_i)/2$. This also ensures that approximated tensors at $r_i's$ are sufficiently close to the original values.

The weights w_i can be computed by normalizing h_i, and making it a partition of unity: $w_i(r) := \frac{h_i(r)}{\sum_{t=0}^n h_t(r)}$. Then, $\mathbf{J}^r(\mathbf{p})$ can be computed by a linear combination $\mathbf{J}^r(\mathbf{p}) := \sum_{i=1}^n w_i(r) \cdot \mathbf{J}^{r_i}(\mathbf{p})$.

The principal curvatures and principal directions can be derived from $\mathbf{J}^r$, as in the usual case. Compared with other interpolation/approximation method, this method produces smooth approximation (if we consider $\mathbf{J}$ at a particular position as a function of r), and is simple and sufficiently accurate to produce high-quality estimation of principal curvatures and principal directions, as shown later in the experiments.

Choice of r_i. r_i should be distributed in such a way that they cover the radii of interests with sufficiently high density. A simple scheme is to choose r_i as: $r_i = i|\bar{e}|$, where $|\bar{e}|$ represents the average edge length of the model, and we may set n to be 5, 7 or 9, depending on the size of the model compared with the average edge length. Choosing r_i differently will not make significant difference as long as the previously stated prerequisite is satisfied. However, the choices of r_1 and r_n do affect the results. Actually, r_1 specifies the smallest scale of interest and r_n corresponds to the largest scale. Increasing r_1 can improve the robustness of

the results. Thus for noisy models, it may be preferable to choose r_1 larger than the average edge length. Increasing r_n can improve the ability to capture large-scale features. This also improves the robustness to noise in flatter regions.

Discussions of performance. It seems that n times of computation would be involved for the process of precomputation, but in fact, it only takes slightly more than $\frac{1}{2}n$ times of computation. Assume for a single scale, the computational complexity is $t_{scan}+t_{int}$, where t_{scan} corresponds to the time for building the volume representation using scan conversion, and t_{int} corresponds to the time used for the computation of integral invariants at a single scale, which includes 10 convolutions computed using FFT. As discussed in [Yang et al. 2006], we need 11 FFTs to transform 11 functions (χ_B, χ_D, $x\chi_D$, $y\chi_D$, $z\chi_D$, $x^2\chi_D$, $xy\chi_D$, $xz\chi_D$, $y^2\chi_D$, $yz\chi_D$, $z^2\chi_D$) to the frequency domain, and need 10 inverse FFTs to compute the convolution of 10 functions other than χ_B with χ_B, where χ_B and χ_D are the characteristic functions of the ball volume (with a certain radius) and the domain. Ignoring the relatively inexpensive multipification in the frequency domain, t_{int} is equal to the time of 11 FFTs and 10 inverse FFTs, i.e. $t_{int} = 21t_{FFT}$. It's clear that scan conversion just needs to be done once, and also the conversion of the input volume to the frequency domain. Thus, the computational complexity is: $t_{scan} + 10t_{FFT} + 11nt_{FFT}$, just slightly more than $\frac{1}{2}n$ times of the original computation at a single scale.

4.2 Iterative Algorithm

The iterative algorithm tries to find an appropriate r for each vertex. The algorithm proceeds as follows: Initially, we may set $r(\mathbf{p}_i)$ to a constant radius. For example, we may simply set it to $\frac{r_1+r_n}{2}$. A better scheme is to assign $r(\mathbf{p}_i)$ to the median value of the minimal principal radius estimated at different scales. The latter approach reduces the number of iterations, and slightly improves the converged distribution of radius. Then, $r(\mathbf{p}_i)$ is updated iteratively. For each vertex, $\kappa_1^{r(\mathbf{p})}$ and $\kappa_2^{r(\mathbf{p})}$ are computed with the approximation scheme discussed in the last subsection. Using Eqn. 7, we may compute a new radius $r'(\mathbf{p}_i)$. In order to make the computation robust, we introduce a step-size control variable λ, and update $r(\mathbf{p}_i)$ to be:

$$r(\mathbf{p}_i) \leftarrow r(\mathbf{p}_i) + \lambda \left(r'(\mathbf{p}_i) - r(\mathbf{p}_i) \right), \tag{8}$$

where λ is set to a small number (e.g. 0.1). This iterative update is repeated until convergence. We then get the estimated optimal r for each vertex, and the corresponding principal curvatures and principal directions can be estimated at each vertex with the varied radius r. Although the iterative algorithm may stop at some local minimum, as shown later in Sec. 5, reasonable and robust radius distribution is usually obtained for various tested examples.

4.3 Post Processing

Note that by using radial basis functions, our approximation to the integral invariants at a particular scale is both efficient and smooth. However, the distribution of r is mainly determined by the local characteristic at different scales (though using integral invariants to estimate it has already taken the local neighborhood into account), it may not be as smooth as desired. Thus, it is reasonable to introduce an optional post processing that smoothes the distribution of r over the surface.

Generally, anisotropic smoothing that preserves significant changes of r is preferred. We use an extended bilateral filtering method [Tomasi and Manduchi 1998] to the mesh surface for our purpose. Unlike their method, we consider r as a piecewise-linear signal over the surface, and smooth r to obtain a smoother version of the radius distribution $\bar{r}$, which is used instead to compute the integral invariants.

For each vertex v_i, assume its position is $\mathbf{p}_i$ and radius is $r(\mathbf{p}_i)$. To achieve anisotropic smoothing, we first define the relative weight of any vertex v_j (within a certain neighborhood $N(v_i)$) w.r.t. v_i as $w(v_i, v_j) = s(v_i, v_j) \cdot t(v_i, v_j)$, where $s(v_i, v_j) = \exp\left\{-\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\sigma_1^2}\right\}$, and $t(v_i, v_j) = \exp\left\{-\frac{|r(\mathbf{p}_i) - r(\mathbf{p}_j)|^2}{2\sigma_2^2}\right\}$. Here, $s(v_i, v_j)$ is a weight measuring nearness in 3D positions of two vertices and $t(v_i, v_j)$ is a weight measuring closeness of radii between two vertices. σ_1 and σ_2 are two parameters used to control how rapidly the effects drop off as the difference increases. For our purpose, they can be chosen e.g. as the average of the edge length and the average of the difference of r along edges.

Then, the smoothed radius $\bar{r}$ at vertex v_i, can be computed as

$$\bar{r}(\mathbf{p}_i) = \frac{\sum_{v_j \in N(v_i)} w(v_i, v_j) r(\mathbf{p}_j)}{\sum_{v_j \in N(v_i)} w(v_i, v_j)}. \tag{9}$$

We then use $\bar{r}$ instead of r as the local neighborhood size for integral invariants computation.

5 Experimental Results

We have implemented our algorithm on an Intel Core2Duo 2GHz Laptop with 2GB RAM using C++ and tested our algorithm on a variety of models with features.

Fig. 2 shows an example of minimal principal directions computed using integral invariants. (a) and (b) show the results obtained with smaller and larger neighborhood size, respectively. Using small radius leads to noisy principal directions, while using large radius may smooth out small-scale but still significant features (emphasized by the red box). By using feature adapted integral invariants, the distribution of radius r is color-coded in (c), where blue and red correspond to small and large radii respectively. For smooth regions, we can safely use larger radius, producing more robust results. For regions contain significant features, radius is reduced to reflect the local geometric features. The obtained directions with varied r is given in (d) where the feature in the red box is successfully captured and the overall principal directions are still reasonably smooth.

Fig. 3 shows color-coded radius distribution and mean curvature before and after radius smoothing. The radius is coded using the same color scheme, and the mean curvature is color-coded so that red color corresponds to highly curved regions with positive curvature, blue color corresponds to highly curved regions with negative curvature and green color corresponds to flat regions. By using radius smoothing, both radius distribution and mean curvatures are slightly smoother, without loss of significant features.

Fig. 4 shows an example of color-coded mean curvatures of Lucy model. From left to right are results obtained with relatively small r, relatively large r and feature-adapted radius. Two close-up views are also given at the bottom of each result. Note that using smaller r tends to produce noisy results, especially in flatter regions, like the bottom of the statue. Using larger r produces results that may ignore significant small-scale features, and may not be able to accurately locate large-scale features. Using feature adapted integral invariants, we are able to produce robust results in flatter regions, more accurate results of features, and improved ability to capture features of varied scales.

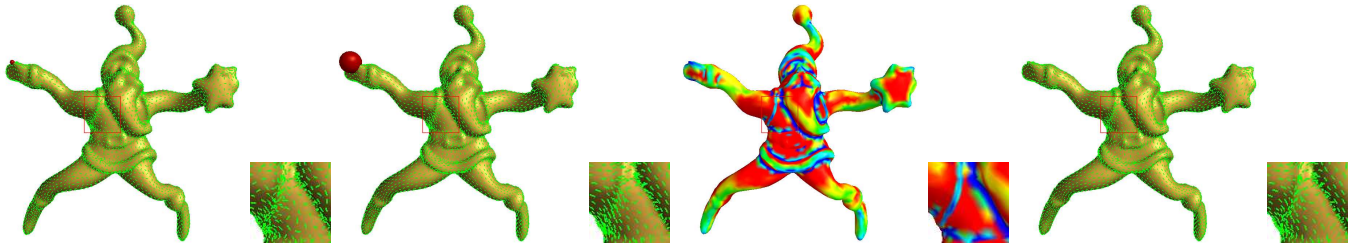


Figure 2: *Minimal principal directions computed with integral invariants of ball neighborhood. (a) the result computed with small radius; (b) the result computed with large radius; (c) color-coded radius distribution of feature adapted integral invariants; (d) the result obtained with feature adapted integral invariants.*

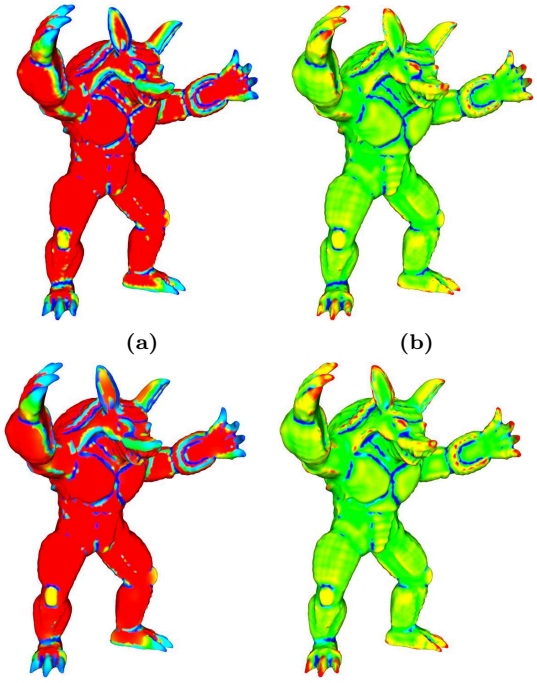


Figure 3: *Color-coded radius distribution (left) and mean curvature (right) before (top) and after (bottom) radius smoothing.*

Due to the nature of integral invariants, our approach is generally robust to noise. Moreover, we use radius adapted to local geometric features, so that we can use larger neighborhoods for flatter regions, without loss of significant features. This further improves robustness in practical applications. Fig. 5 shows an example. (a) is the Happy Buddha model with added noise. (b) gives the color-coded radius distribution. Reasonable radius distribution is obtained, even on noisy models with quite a few features. Two close-ups of minimal principal directions estimated with this method are given in (c). For flatter regions, large neighborhood is used, making it robust to otherwise dominant added noise. For feature regions, appropriate size of neighborhood is used, ensuring accurate principal directions.

We also applied our method to the problem of feature extraction. Similar to [Lai et al. 2007], features are defined as regions with at least one large principal curvature. Fig. 6 shows extracted features on Armadillo model with single-scale and feature-adapted integral invariants. For single-scale method, relatively large neighborhood size is used to produce robust results; however, large-scale features tend to have bleeding effect, and small-scale features can easily get lost. Using feature adapted integral invariants, features are extracted thinner and in more accurate locations. Combinations of ridges and valleys are better preserved. Although feature extraction may also be improved by combining extracted features from different scales [Pottmann et al. 2007], our method produces a single, consistently combined curvature estimation over the surface.

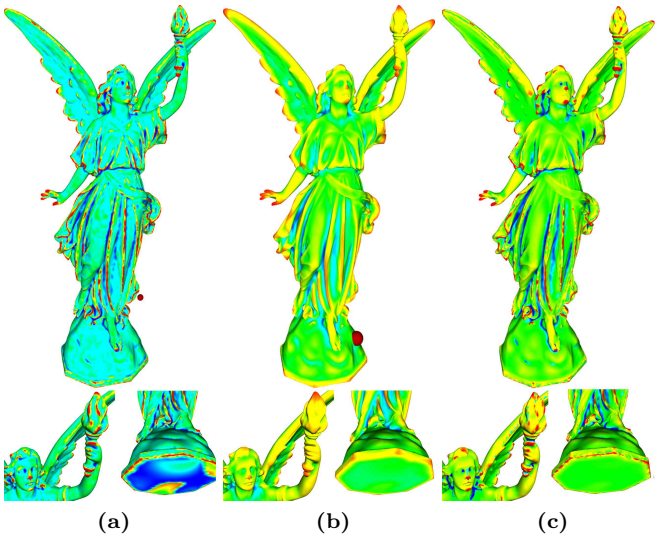


Figure 4: *Color-coded mean curvatures computed with (a) small r, (b) large r, (c) adapted r (with close-ups).*

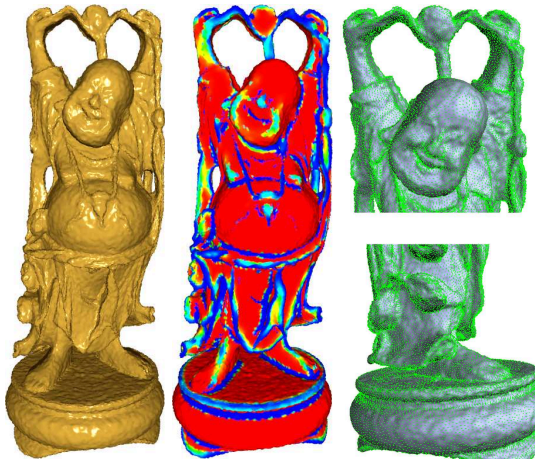
The computation time of feature adapted integral invariants is dominated by computing integral invariants at several scales. The examples in the paper used 8 scales, and the overall computation times for different models are given in Table 1.

Table 1: *Running times of different models.*

model	grid size	time (sec.)
Santa	$204 \times 172 \times 184$	87
Lucy	$194 \times 112 \times 332$	114
Armadillo	$192 \times 228 \times 174$	82
Buddha	$136 \times 332 \times 136$	113

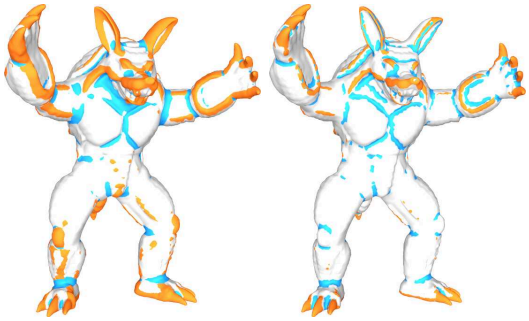
6 Conclusions

In this paper, we proposed a simple and efficient method to automatically decide radii over the surface, to achieve the ideal balance between robustness and accuracy of feature locations in the integral invariant computation. We demonstrate the effectiveness of our approach with applica-



(a) (b) (c)

Figure 5: *Minimal principal directions estimated on a noisy Happy Buddha model.*



(a) (b)

Figure 6: *Feature extraction using single-scale (left) and feature-adapted (right) integral invariants.*

tions including principal curvatures and principal direction fields as well as feature extraction. We expect that such technique may be helpful to a variety of applications that require robust and accurate principal curvature estimation. For example, line drawings in non-photorealistic rendering may benefit from this technique to produce feature adapted view-dependent curvature information. Thus, we plan to explore the applications of our method in robust line drawings.

Acknowledgements

The models in this paper are courtesy of Stanford University and AIM@SHAPE Repository. This work was supported by the National Basic Research Project of China (Project Number 2006CB303102), the Natural Science Foundation of China (Project Number 60673004, U0735001) and the National High Technology Research and Development Program of China (Project Number 2007AA01Z336).

References

Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., and Desbrun, M. 2003. Anisotropic polygonal remeshing. In *Proc. ACM SIGGRAPH*, 485–493.

Cazals, F., and Pouget, M. 2003. Estimating differential quantities using polynomial fitting of osculating jets. In *Proc. Eurographics Symp. Geometry Processing*, 177–187.

Clarenz, U., Rumpf, M., and Telea, A. 2004. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Trans. Vis. Comp. Graph. 10*, 5, 516–524.

Cohen-Steiner, D., and Morvan, J.-M. 2003. Restricted delaunay triangulations and normal cycle. In *Proc. ACM Symp. Computational Geometry*, 312–321.

Connolly, M. 1986. Measurement of protein surface shape by solid angles. *Journal of Molecular Graphics 4*, 3–6.

Desbrun, M., Meyer, M., and P. Schröder, A. B. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proc. Vis. Math.*, 35–57.

do Carmo, M. P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

Goldfeather, J., and Interrante, V. 2004. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graphics 23*, 1, 45–63.

Hertzmann, A., and Zorin, D. 2000. Illustrating smooth surfaces. In *Proc. ACM SIGGRAPH*, 517–526.

Jiao, X., and Zha, H. 2008. Consistent computation of first- and second-order differential quantities for surface meshes. In *Proc. ACM Symp. Solid and Physical Modeling*, 159–170.

Kalogerakis, E., Simari, P., Nowrouzezahrai, D., and Singh, K. 2007. Robust statistical estimation of curvature on discretized surfaces. In *Proc. Eurographics Symp. Geometry Processing*, 13–22.

Lai, Y.-K., Zhou, Q.-Y., Hu, S.-M., Wallner, J., and Pottmann, H. 2007. Robust feature classification and editing. *IEEE Trans. Vis. Comp. Graph. 13*, 1, 34–45.

Lai, Y.-K., Kobbelt, L., and Hu, S.-M. 2008. An incremental approach to feature aligned quad-dominant remeshing. In *Proc. ACM Symp. Solid and Physical Modeling*, 137–145.

Manay, S., Hong, B.-W., Yezzi, A. J., and Soatto, S. 2004. Integral invariant signatures. In *Proc. Eurographics Conf. Computer Vision*, 87–99.

Mitra, N. J., Nguyen, A., and Guibas, L. J. 2004. Estimating surface normals in noisy point cloud data. *Int'l J. Comp. Geom. App. 14*, 4–5, 261–276.

Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H.-P. 2003. Multi-level partition of unity implicits. In *Proc. ACM SIGGRAPH*, 463–470.

Ohtake, Y., Belyaev, A., and Seidel, H.-P. 2004. Ridge-valley lines on meshes via implcifit surface fitting. In *Proc. ACM SIGGRAPH*, 609–612.

Pottmann, H., Wallner, J., Yang, Y.-L., Lai, Y.-K., and Hu, S.-M. 2007. Principal curvatures from the integral invariant viewpoint. *Computer Aided Geometric Design 24*, 8–9, 428–442.

Tomasi, C., and Manduchi, R. 1998. Bilateral filtering for gray and color images. In *Proc. IEEE ICCV*, 836–846.

Yang, Y.-L., Lai, Y.-K., Hu, S.-M., and Pottmann, H. 2006. Robust principal curvatures on multiple scales. In *Proc. Eurographics Symp. Geometry Processing*, 223–226.