Efficient Multi-scale Curvature and Crease Estimation

D. Panozzo*
DISI - Università di Genova

E. Puppo†
DISI - Università di Genova

L. Rocca‡
DISI - Università di Genova

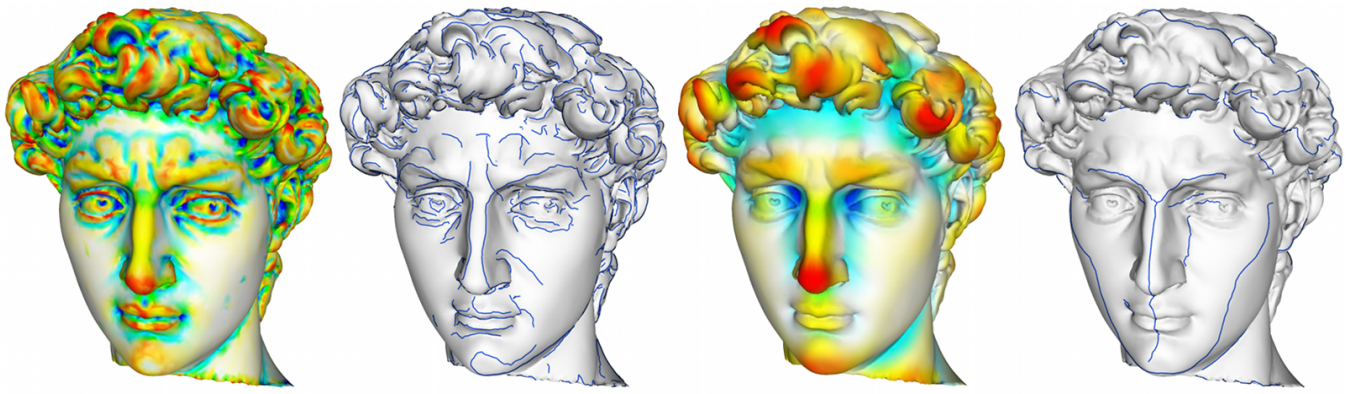

Figure 1: *Principal curvatures and creases computed at scales* $2e$ *and* $10e$*, with* e *the average length of mesh edges. Color map combines principal curvatures with a non-linear mapping designed to enhance small variations: red convex; yellow flat-convex; green saddle; cyan flat-concave; blue concave; white flat. At the large scale, the two ridges along the nose merge into one ridge that extends downwards to the chin; transverse creases along lips and eyelids disappear; and a new crease along the convex ridge through cheekbones-cheeks-chin appears.*

Abstract

We consider the problem of multi-scale estimation of principal curvatures and crease lines on a surface represented with a mesh of triangles and affected by noise. We show that curvature at different scales can be efficiently and accurately estimated by modifying a fitting technique and applying it to neighborhoods of various size, depending on scale: we discard bending portions of surfaces during fitting, and we apply Monte-Carlo sampling to speed up computation. Next we propose a new technique for extracting crease lines and we show how such lines can summarize shape features at the various scales. This is a first step towards building a scale-space of surface features.

Keywords: geometric mesh, curvature estimation, crease estimation, multi-scale surface analysis

1 Introduction

Several problems in computer graphics, geometric modeling and engineering involve the computation of differential properties of surfaces that are smooth in principle, while most often they are approximated with polygonal meshes. This subject has been treated by many authors in different contexts during the last twenty years, and several algorithms have been proposed for computing differential properties on geometric meshes or clouds of points.

Many real datasets, e.g., those generated from range scanning, are affected by noise. Processing noisy meshes to estimate the differential properties of surfaces they represent can be a very challenging task, since the effect of noise is highly amplified by differentiation. This fact drastically restricts the range of applicability of most methods for differential surface analysis.

The problem of noise can be addressed in a more general way by multi-scale analysis, which has been successfully and extensively applied in image processing and computer vision [Lindberg 2009; Koenderink 1994]. Scale is a parameter to be used while analyzing the surface, which implicitly introduces a smoothing effect: multi-scale differential analysis of a surface is in fact analogous to differential analysis of different versions of the surface smoothed with filters at various scales.

In this framework, both noise and true shape features that are too small for the purpose of a given application can be filtered out. The finest relevant scale for a given representation cannot be finer than the scale σ_n of noise. Therefore, if the surface is analyzed at a scale smaller than σ_n, estimates shall be unreliable; if it is analyzed at a scale $\sigma > \sigma_n$, estimates should be robust to noise, while also disregarding surface features at scales smaller than σ.

Multi-scale surface analysis may eventually lead to the construction of a *scale-space* of surfaces [Pauly et al. 2006]. Careful scale-space analysis may provide scale-persistent features to be used for diverse purposes such as surface simplification and remeshing, surface description and retrieval, non-photorealistic rendering, etc.

In [Yang et al. 2006; Pottmann et al. 2007], Pottmann et al. proposed a method for evaluating the curvature of a mesh at different scales without prior smoothing the mesh, but simply considering point neighborhoods of various sizes to perform integral computations. In line with their work, we present here a different approach to multi-scale estimation of curvatures, and we extend it to multi-scale extraction of creases (i.e., third order differential properties).

The contribution of our work is threefold. First, we show how a fitting technique for curvature estimation can be made efficient, accurate and robust to noise, and we show its application to multi-scale surface analysis. Second, we propose a new discrete and efficient method for estimating multi-scale creases, which exploits the results obtained during curvature estimation. Finally, we show the behavior of creases extracted at the various scales, thus paving the way for the construction of a scale-space of surface features. Extensive experiments are presented to test our methods on real data and to compare them with existing methods.

*email:panozzo@disi.unige.it
†email:puppo@disi.unige.it
‡email:rocca@disi.unige.it

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we give the necessary background. In Section 4 we describe the method for curvature estimation, while in Section 5 we describe the method for crease extraction. In Section 6 we present experimental results. Finally, in Section 7 we make some concluding remarks.

2 Related work

The literature on computing curvatures and other differential quantities on surfaces and meshes is huge and many different methods have been developed in engineering, geometric modeling and computer graphics. Some recent accounts on the subject can be found, e.g., in [Costa Batagelo and Wu 2007; Gatzke and Grimm 2006; Grinspun et al. 2006]. In the following we briefly discuss just issues directly related to multi-scale estimation of differential quantities on surfaces potentially affected by noise.

Methods for estimating surface curvature can be broadly divided in two categories: *fitting methods* that fit analytic surfaces to data and derive differential properties analytically; and *discrete methods* that are all based on concepts of discrete differential geometry [Desbrun et al. 2005].

Discrete methods are usually fast and accurate in capturing the local surface properties, but most of them are very sensitive to noise. Moreover, most such methods are based on very local information (e.g., from the 1-ring of each vertex) and they can be hardly extended to a multi-scale approach. Notable exceptions are: the *Integral invariant* method proposed by Pottmann et al. [Pottmann et al. 2007; Yang et al. 2006]; the *Normal cycles* method proposed by Cohen-Steiner and Morvan [Cohen-Steiner and Morvan 2003]; and the method based on adaptive curve sampling proposed by Agam and Tang [Agam and Tang 2005]. The former method has been designed specifically for multi-scale computation, while the latter two can be easily extended to the purpose.

Fitting methods are overall more robust to noise and they naturally extend to multi-scale computation by considering larger neighborhoods for fitting. On the other hand, they are usually slower than discrete methods, and they introduce smoothing effects even at small scales. Moreover, they can become very inaccurate if fitting assumptions (e.g., on the projectability of the surface to a reference plane) are violated.

Different fitting methods are characterized by the type and degree of functions that they fit and by the information they use. Many methods assume that the surface normal is either available, or reliably estimated during pre-processing, and they fit a polynomial defined with respect to the tangent plane at the surface in a given point. One notable example is the method proposed by Goldfeather and Interrante [Goldfeather and Interrante 2004], which use cubic polynomials to fit both the position and the surface normal of data from the 1-ring of each vertex. As reported in [Gatzke and Grimm 2006], this method is very accurate if the surface normal is computed analitically, but it is very sensitive to error in estimating the normals. Cazals and Pouget [Cazals and Pouget 2005a] show that in fact accuracy of the estimated normal may have a high influence on the computation of higher order differential quantities. Thus, they propose the *Osculating Jets* method [Cazals and Pouget 2005a; Cazals and Pouget 2008], which fits polynomials of arbitrary degree, defined on a reference plane that goes through the given point, but is not necessarily tangent to the surface. They show the convergence of the method for analytic surfaces. Our method for estimating the normal direction and curvature tensor is a variation of the Osculating Jets.

Douros and Buxton [Douros and Buxton 2002] fit implicit functions locally to data and derive differential properties analytically from them. This approach could be extended to a multi-scale by using neighborhoods of various size, but it is computationally more involved - thus less efficient - than the Osculating Jets. Ohtake et al. [Ohtake et al. 2004] fit an implicit function to the whole dataset and derive differential properties analytically from it. Since the implicit function is generated from a sequence of approximations of the original data at different scales, this method can also be extended to a multi-scale one. On the other hand, generating the implicit function is equivalent to resolving a problem of surface reconstruction from a point cloud, which is far more complicated than the problem of estimating differential quantities.

Less works have been proposed to evaluate higher order differential quantities, such as creases. Ohtake et al. [Ohtake et al. 2004] evaluate extremalities (i.e., curvature derivatives, which are third order differential quantities - see Section 3) analytically from the implicit function they fit to data, then they extract the intersections of ridges with mesh edges by linear interpolation. They also propose a simple technique for filtering spurious ridges on the basis of the integral of curvature along each ridge. We adopt their filter in our work. Yoshizawa et al. [Yoshizawa et al. 2005] use the cubic fit technique of [Goldfeather and Interrante 2004] to estimate extremalities analytically, then they use variations of the method in [Ohtake et al. 2004] to extract and filter ridges. Hildebrandt et al. [Hildebrandt et al. 2005] use discrete and very local methods to evaluate all differential quantities. Once curvatures and curvature directions have been computed, they estimate extremalities on each triangle of the mesh by linear interpolation, and they extend such extremalities to each vertex by averaging values obtained at its incident triangles. Laplacian smoothing is performed on the resulting piecewise-linear scalar field, and creases are extracted triangle by triangle by linear interpolation.

Cazals and Pouget [Cazals and Pouget 2008] compute third and fourth order differential quantities necessary to compute extremalities analytically from a polynomial of order four (at least) fitted to data via the Osculating Jets. Once such coefficients have been computed, ridges are extracted with an algorithm described in [Cazals and Pouget 2005b], which also detects umbilical points and correctly manages ridges in their vicinity.

We are not aware of any work on extracting creases and higher order differential quantities at multiple scales.

3 Background

We summarize basic notions of differential geometry, which can be found in detail in any textbook, e.g., [Porteous 2001].

Let $\mathcal{S}$ be a smooth surface and let $N_{\mathcal{S}} : \mathcal{S} \rightarrow \mathbb{R}^3$ be its normal field (aka *Gauss map*), i.e., the field associating to each point $P \in \mathcal{S}$ its surface normal $N_{\mathcal{S}}(P)$. The *shape operator* is the negative differential of the Gauss map, i.e.:

$$S = -dN_{\mathcal{S}}$$

that associates to each point $P \in \mathcal{S}$ a linear operator describing how the normal vector changes along any direction on the tangent plane of $\mathcal{S}$ at P. The shape operator is a tensor which can be described at each point P by a 2×2 matrix S_P, referred to a local orthonormal frame $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ having its origin at P and $\mathbf{n} = N_{\mathcal{S}}(P)$. Vectors $\mathbf{u}$ and $\mathbf{v}$ are a basis of the tangent space $T(P)$ at P. The eigenvalues k_1 and k_2 and eigenvectors $\mathbf{t_1}$ and $\mathbf{t_2}$ of matrix S_P (in the local frame) define the values and directions of the *principal curvatures* of $\mathcal{S}$ at P, respectively. The principal directions of curvature are mutually orthogonal and lie on the tangent plane $T(P)$. Note that principal curvatures define line fields, rather than vector fields, on

the surfaces of $\mathcal{S}$, therefore the orientations of $\mathbf{t_1}$ and $\mathbf{t_2}$ are defined arbitrarily. Hereafter we will assume $k_1 \geq k_2$ and we will select an orientation for curvature directions such that $(\mathbf{t_1}, \mathbf{t_2}, \mathbf{n})$ form a right-handed coordinate system, called the *Monge frame*.

Given $P \in \mathcal{S}$, the surface in a neighborhood of P can be expressed in parametric form as $\mathbf{X}(u, v)$ with $(u, v) \in \Omega \subseteq \mathbb{R}^2$. In this case, the shape operator at P can be described in terms of the first and second fundamental forms of $\mathbf{X}$.

The *first fundamental form* is the inner product on the tangent space $T(P)$: let $P = \mathbf{X}(u, v)$ and let $\mathbf{v}$ and $\mathbf{w}$ be two vectors in $T(P)$,

$$I(\mathbf{v}, \mathbf{w}) = \mathbf{v}^T \begin{bmatrix} E & F \\ F & G \end{bmatrix} \mathbf{w},$$

with $E =< \mathbf{X}_u, \mathbf{X}_u >$, $F =< \mathbf{X}_u, \mathbf{X}_v >$ and $G =< \mathbf{X}_v, \mathbf{X}_v >$, where $\mathbf{X}_u$ and $\mathbf{X}_v$ denote the first derivatives of the parametric function $\mathbf{X}$ computed at (u, v) and $< \cdot, \cdot >$ denotes the inner product in $\mathbb{R}^3$. The first derivatives of $\mathbf{X}$ at (u, v) span the tangent space $T(P)$, so the surface normal at P can be defined in terms of their cross product:

$$\mathbf{n} = N_{\mathcal{S}}(P) = \frac{\mathbf{X}_u(u, v) \times \mathbf{X}_v(u, v)}{|\mathbf{X}_u(u, v) \times \mathbf{X}_v(u, v)|}.$$

The *second fundamental form* is a tensor defined by projecting the second partial derivatives of $\mathbf{X}$ on the normal direction $\mathbf{n}$, which is represented by the matrix:

$$II = \begin{bmatrix} L & M \\ M & N \end{bmatrix}$$

where $L =< \mathbf{X}_{uu}, \mathbf{n} >$, $M =< \mathbf{X}_{uv}, \mathbf{n} >$, $N =< \mathbf{X}_{vv}, \mathbf{n} >$ and $\mathbf{X}_{uu}$, $\mathbf{X}_{uv}$, and $\mathbf{X}_{vv}$ denote the second partial derivatives of $\mathbf{X}$ computed at (u, v).

The partial derivatives of $\mathbf{n}$ with respect to u and v are defined in terms of the coefficients of the first and second fundamental forms by the Weingarten equations

$$\mathbf{n}_u = \frac{FM - GL}{EG - F^2}\mathbf{X}_u + \frac{FL - EM}{EG - F^2}\mathbf{X}_v$$

$$\mathbf{n}_u = \frac{FN - GM}{EG - F^2}\mathbf{X}_u + \frac{FM - EN}{EG - F^2}\mathbf{X}_v$$

from which we get the following expression of the shape operator at P:

$$S_P = (EG - F^2)^{-1} \begin{pmatrix} LG - MF & ME - LF \\ ME - LF & NE - MF \end{pmatrix}.$$

On a sufficiently small neighborhood of P, the surface can be expressed in terms of an explicit function defined on a frame $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ with origin at P and $\mathbf{w}$ axis not parallel to the tangent plane $T(P)$. Note that $\mathbf{w}$ needs not be aligned with the surface normal $\mathbf{n}$. In this case, the parametric function $\mathbf{X}$ has the form

$$\mathbf{X}(u, v) = (u, v, f(u, v)),$$

where $f(u, v)$ is a bivariate scalar function. The coefficients of the first and the second fundamental form, hence the shape operator, are easily derived from the first and second partial derivatives of f.

A point P is said to be *regular* if the two principal curvatures at P are different. Each principal curvature forms a *line field* that is defined at all regular points and rules the surface. The points where the two principal curvatures are equal correspond to singularities of the curvature line fields and they are called *umbilical points*. Principal directions are undefined at umbilical points.

Consider the principal curvatures k_1 and k_2 as scalar fields defined on $\mathcal{S}$. Then, the gradient ∇k_i of curvature k_i is a vector field on the tangent bundle of $\mathcal{S}$. The *extremality* e_i at a regular point P is defined as the inner product between the gradient of curvature k_i and its related direction, i.e.,

$$e_i =< \nabla k_i, \mathbf{t_i} >= \frac{\partial k_i}{\partial \mathbf{t_i}} \tag{1}$$

where all quantities are evaluated at P. Note that the sign of e_i is not well defined, since it depends on an arbitrary orientation of $\mathbf{t}_i$. *Creases* are defined as those regular points where extremalities vanish, with the following additional constraints:

$$e_1 = 0 \quad \wedge \quad \frac{\partial e_1}{\partial \mathbf{t_1}} < 0 \quad \wedge \quad k_1 > |k_2| \tag{2}$$

$$e_2 = 0 \quad \wedge \quad \frac{\partial e_2}{\partial \mathbf{t_2}} > 0 \quad \wedge \quad k_2 > -|k_1|. \tag{3}$$

Creases defined by e_1 are also called *ridges*, while creases defined by e_2 are also called *valleys.*

4 Curvature estimation

We take in input a mesh of triangles M and we evaluate the shape operator, hence the principal curvatures and curvature directions, at each vertex P of M by using a surface fitting method, which is a variation of the *Osculating Jets* proposed in [Cazals and Pouget 2005a; Cazals and Pouget 2008].

The original method of [Cazals and Pouget 2005a] can be briefly summarized as follows:

1. gather the vertices of M in a neighborhood of P (usually just a few rings around P, depending on the selected degree of the polynomial to be fitted);
2. set a local frame $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ - hereafter called the *fitting frame* - centered at P with its $\mathbf{w}$ axis not parallel to the tangent plane at P (an early implementation used a coordinate axis, while a later implementation selects a better fitting frame by performing principal component analysis (PCA) of the neighborhood);
3. express gathered data in the local frame and fit a polynomial $f(u, v)$ of given degree m, by resolving a least square problem (usually with Singular Value Decomposition);
4. evaluate the shape operator, hence the principal curvatures and directions, at $(0, 0, f(0, 0))$ in the local frame and set those values to estimate the curvatures at P.

In our approach, we wish to estimate the local shape at various scales by fitting a polynomial on a more or less extended neighborhood of P. Similarly to [Yang et al. 2006], we use the radius r of a neighborhood as a scale parameter. A fundamental assumption of the Osculating Jets is that the input surface can be expressed in explicit form in the neighborhood B of a point P. If this assumption is violated, then the method may become highly inaccurate. This hardly happens in small rings, but it is likely to occur, even with slightly large neighborhoods, in the proximity of regions with a high curvature. Since we wish to evaluate the curvature at quite large scales, i.e., in the order of a few tenths of the diameter of the bounding box of an object, we cannot be careless in gathering data from a neighborhood. Thus, given a vertex P and a scale parameter r, we proceed as follows:

- We perform a breadth-first traversal of the mesh, starting at P, until we get vertexes that lie in the ball of radius r centered at P, and we gather all such vertices in a set V_P;
- Next we set the $\mathbf{w}$ axis of the fitting frame to be equal to the average $\frac{1}{|V_P|}\sum_i \mathbf{n}_i$, where summation is on all elements of V_P and $\mathbf{n}_i$ is a pseudo-normal of the surface estimated at v_i with a standard method (e.g., a weighted average of surface normals of triangles in its 1-ring). Pseudo-normals are estimated once and for all during pre-processing.
- For all i, we compute the scalar product $< \mathbf{n}_i, \mathbf{w} >$ and we discard from V_P each vertex v_i giving a negative value (i.e., vertices corresponding to a flipping portion of surface).

Note that pseudo-normals $\mathbf{n}_i$ are *not* used to set the fitting frame at their respective vertexes. They are just used to either accept or discard vertexes in the neighborhood B. Now we are left with a relatively large set of vertexes to be plugged into a least square problem. We improve efficiency and scalability by acting on the degree of the polynomial and on the size of the data set.

Since we are wish to extract creases, a logical choice would be to fit polynomials of degree at least three (for instance, polynomials of order four are used in [Cazals and Pouget 2008]). However, polynomials of degree three [four] would give us a least square problem with ten [fifteen] unknowns. We can speed-up computation an order of magnitude by fitting a polynomial of degree two, which is sufficient to evaluate second order differential quantities, while relying on a discrete approach for extracting higher order differential quantities. Moreover, we constrain such a polynomial to go through point P, thus forcing the coefficient of order zero to vanish. This reduces the unknowns to just five coefficients, i.e.,

$$f(u,v) = au^2 + bv^2 + cuv + du + ev. \tag{4}$$

The loss of accuracy with respect to fitting a complete polynomial is compensated by the fact that our polynomial goes exactly through P, thus it is not necessary to approximate the differential quantities at P with those computed at its footprint on the graph of f.

Second, we perform Monte-Carlo sampling on the set of data V_P. In fact, the amount of data necessary to obtain e reliable fit is not a function of the size of the neighborhood, but rather a function of the number of unknowns, which is fixed to five, provided that data are sampled uniformly in the neighborhood. Therefore, we can afford sampling a relatively small set of points. In Section 6, we analyze the variation in estimates of curvatures and curvature directions as a function of the number of sampled points, and we show that sampling about 50 vertexes gives excellent results.

With these modifications, the Osculating Jets becomes robust and fast enough to be used for multi-scale curvature estimation even at large scales and with very noisy data, as we show in Section 6.

5 Extraction of creases

Once principal curvatures and curvature directions have been extracted, information we need for extracting creases already come at the proper scale. This means that we do not need to extend our computations to a large neighborhood. However, we must be careful to make small use of mesh geometry, because this has *not* been smoothed, while it still refers to the finest scale. Therefore, we develop a discrete method that, at each vertex, makes use only of information from its 1-ring, and relies on geometry of the quadric surfaces fit to data during curvature computation, rather than on the original mesh.

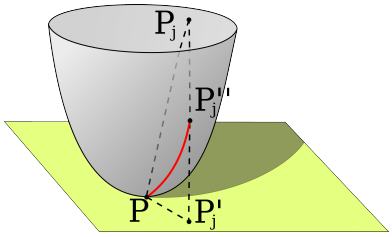

Figure 2: *The quadric surface approximating the (smoothed) surface in the neighborhood of a vertex P, obtained during curvature estimation. The arc length of the curve joining P to P_j'' is used to compute an approximation of the true distance between P and P_j on the smoothed surface.*

Let $k_i : \mathcal{S} \to \mathbb{R}$, for $i = 1, 2$ be the fields of principal curvature, estimated at all vertexes of mesh M. We first estimate their gradients $\nabla k_i : \mathcal{S} \to T_{\mathcal{S}}$, where $T_{\mathcal{S}}$ denotes the tangent bundle of $\mathcal{S}$.

Let P be a vertex of M and let P_j, for $j = 1 \ldots h$, be its neighbors. Let P_j' be the projection of P_j on the tangent plane $T(P)$ and let $\mathbf{t}_j$ be the direction of $P_j' - P$. Note that this is the only datum from the geometry of M that we use in our computation. Since mesh smoothing displaces vertexes essentially in the normal direction, this projection is not likely to change much through the scales. We estimate the derivative of k_i along $\mathbf{t}_j$ with a finite difference, thus obtaining the following equation:

$$< \nabla k_i(P), \mathbf{t}_j >= \frac{k_i(P_j) - k_i(P)}{d(P, P_j)}, \tag{5}$$

where $d(P, P_j)$ denotes a distance between P and P_j. We collect such equations for all $j = 1 \ldots h$ and we obtain the components of ∇k_i by resolving the corresponding least square problem with two unknowns and h equations.

Note that distance $d(P, P_j)$ in Equation 5 should be measured on the (unknown) smoothed version of $\mathcal{S}$ at scale r. The best approximations that we have of that surface in the proximity of P and P_j are provided by the quadric functions that we used at P and P_j, respectively, during curvature estimation, i.e.,

$$w = \frac{k_1}{2}u^2 + \frac{k_2}{2}v^2 \tag{6}$$

where the equation is expressed in the Monge frame at either P or P_j, respectively, and the values of k_1 and k_2 are taken at the corresponding point. Given one of these two surfaces - say the one centered at P - we measure the arc length on this quadric surface between P and the vertical projection of P_j on the same surface (see Figure 2). This value is computed analytically by resolving a line integral on the conic line obtained by sectioning the surface with the vertical plane through direction $\mathbf{t}_j$. The formula can be derived easily through a solver, like *Maxima*, and it involves Equation 6 as well as the coordinates of P_j'. We do not report it here for brevity. We repeat the same computation by considering the surface centered at P_j and we compute the average between the two arc lengths.

Once the gradients of the principal curvatures have been computed at each vertex, our method proceeds similarly to that of [Hildebrandt et al. 2005]:

1. Compute extremalities through Equation 1;
2. Extract creases triangle by triangle, by setting the orientation of principal axes at the vertexes, so that corresponding axes form acute angles (see also [Cazals and Pouget 2005b] about

the "acute rule") and recomputing the signs of extremalities accordingly. The sign of the derivative of extremalities, which is required in Equations 2 and 3, is computed by finite differences along two edges of t, by applying the same method as in Equation 5, where k_i is replaced with e_i. Each triangle t can contain at most one crease segment, whose endpoints are computed by linear interpolation on the edges of t;

3. Trace creases to form polylines.

Two optional steps can be performed to improve the shape of creases: Laplacian smoothing on the field of extremalities can be performed before Step 2; and creases can be filtered after Step 3, as suggested in [Ohtake et al. 2004]. Creases are filtered by compting the line integral of curvature magnitudes along each polyline, and discarding polylines with a value below a given threshold.

6 Experimental results

We implemented our methods in C++ by using the VCG Library [VCG] for geometric data structures and the Eigen library [Eig] for numerical computations. Experiments were run on a PC with a 2.67Ghz Core i5 processor equipped with 4Gb of memory, using a single core. We tested our algorithms against other methods at the state-of-the-art on some of the data sets available in the public domain for benchmarks.

6.1 Curvature

We first consider smooth datasets to extract curvatures at various scales. In each mesh, we use the average length of edges e as a reference to set the scale r for fitting. An example of curvature extracted at various scales from a large mesh is reported in Figure 3. The color map combines principal curvatures with a non-linear mapping, designed to enhance variations also at small curvatures. Color codes are as follows: red convex; yellow flat-convex; green saddle; cyan flat-concave; blue concave; white flat. At the finest scale the curvature map enhances the artifacts of object reconstruction on the blade, as well as the fine detail of the rugged bottom part of the object. Curvature of such details, as well as small details on the edges of the blade, and bas-relief letters on the bottom part, progressively disappear at the larger scales, while the curvature of larger details is correctly characterized throughout all scales.

We compare our method for curvature extraction with the classical Osculating Jets and with the integral invariant method of Pottmann et al. Curvatures are estimated at various scales, ranging from twice to 16 times the average edge length in the mesh. Our method is applied by performing Monte-Carlo sampling with a threshold of 50 vertexes.

An implementation of the classical Osculating Jets with degree two is derived from the implementation of our method, the most important difference being that the whole neighborhood is always used for fitting. We do not report running times for this method, because they are quite similar to ours.

For the integral invariant method, we use the implementation provided by the authors, which is based on efficient computation of PCA on ball neighborhoods via FFT. The method performs a space discretization, which has high memory requirements. The program is an executable for Windows that cannot work beyond the allowed threshold of 2Gb of memory. Therefore, we could not run it on any dataset with a discretization step smaller than 0.005 times the size of the bounding box. This fact puts a severe lower bound on the meaningful scales that can be used: we have run experiments with this method only when the scale was not finer than 5 times the size of the voxel, i.e., 0.025 times the size of the bounding box.

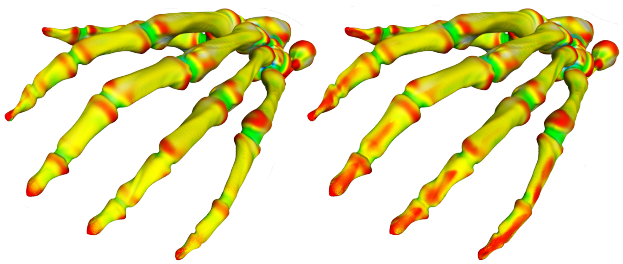

Figure 4: *Principal curvatures computed with our method (left) and with Osculating Jets (right). Scale is* $16e$*, which is about twice the diameter of the fingers. The Osculating Jets incorrectly classifies as convex (red) some cylindrical parts (yellow).*

Numerical results are reported in Table 1, while visual results for some examples are shown in Figure 4 and 5.

Dataset	$\lvert M \rvert$	2e		4e		8e		16e	
Gargoyle	25k	0.4	-	1.1	12.9	4.7	13.1	22.9	13.7
RockerArm	35k	0.5	-	1.8	14.0	3.8	14.1	28.1	14.6
David head	50k	0.9	-	2.6	30.7	9.7	31.7	37.6	30.5
Angel	237k	4.7	-	13.1	-	50.0	-	220.2	-
Turbin Blade	883k	12.9	-	33.7	-	128.0	22.9	809.6	23.5

Table 1: *Running times (in seconds) for extracting curvatures on various datasets at different scales with our method (left columns) and with the integral invariant method (right columns). Sizes of the datasets are given by the number of vertexes (in thousands). Scales are expressed in terms of the average edge length* e*. The Angel dataset could not be loaded with the integral invariant program.*

The standard Osculating jets behaves similarly to our algorithm at small enough scales, but it produces artifacts near small details at higher scales, where a large enough neighborhood captures also portions of surface that are flipped with respect to the fitting plane. Some such artifacts can be seen in Figure 4, where Osculating Jets marks as convex large cylindric parts of the mesh near the fingertips.

As shown in Figure 5, the discretization scale of the integral invariant method is too coarse to produce meaningful results at a fine scale $2e$, while our method correctly detects the curvature of small features (note for example the bas-relief letters). At scale $4e$ discretization artifacts still appear (diagonal stripes on the flat part of the object). At larger scales, both methods produce very smooth results, which are comparable and compatible with object features at that scale. In terms of running times, the integral invariant method is competitive on large objects at large scales, while our method is faster at small scales and for objects of moderate size.

Next we show the effect of applying our method with Monte-Carlo sampling against using the whole neighborhood. Since we do not have any ground truth for curvatures on these data sets, we take the computation with the whole neighborhood as a reference, and we report the deviation from those results at different rates of Monte-Carlo sampling. Numerical results are reported in Table 2, while visual results are shown in Figure 6. Note that the loss of accuracy is quite small, even at large scales, by using 100 samples, while results with 200 samples are almost identical to those obtained with the whole neighborhood, which fits to about 1200 data points on average.

As expected, Monte-Carlo sampling drastically reduces the time required for fitting at large scale. Unfortunately, computation times are dominated from the breadth-first traversal of the mesh, hence the global speed-up is just moderate. We believe that a careful im-

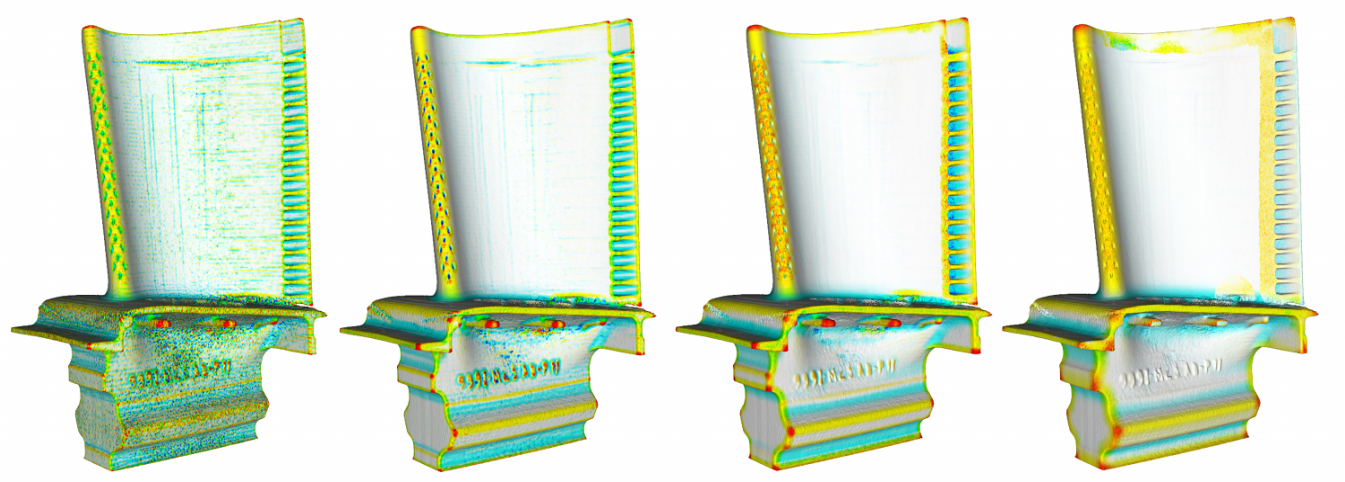

Figure 3: *Principal curvatures computed on the turbin blade dataset at scales* $2e$*,* $4e$*,* $8e$ *and* $16e$*.*

		Time w/o M-C		Number of Monte-Carlo samples											
				50				100				200			
Dataset	$\|M\|$	Fit	Total	$\bar{x}$	σ	Fit	Total	$\bar{x}$	σ	Fit	Total	$\bar{x}$	σ	Fit	Total
Gargoyle	25k	3.4	23.0	0.95	0.13	0.2	20.0	0.97	0.09	0.3	20.3	0.98	0.06	0.5	20.5
RockerArm	35k	3.7	28.3	0.97	0.09	0.3	25.2	0.99	0.06	0.4	25.4	0.99	0.04	0.7	25.7
David head	50k	7.1	37.3	0.96	0.12	0.4	31.3	0.98	0.08	0.6	31.6	0.99	0.06	1.0	32.0
Angel	237k	29.5	220.0	0.98	0.08	1.9	195.3	0.99	0.06	2.9	196.7	0.99	0.04	4.9	198.9
Turbin Blade	883k	108.6	809.5	0.96	0.12	6.7	719.0	0.98	0.09	10.4	724.3	0.99	0.07	18.0	733.1

Table 2: *Average deviation, variance and computational time using different numbers of Monte-Carlo samples at scale* $32e$*. Deviation is measured as the scalar product between the principal directions of curvature: 1.0 means perfect match. Time is in seconds.*

plementation of the search with ad hoc data structures could drastically reduce computation times, thus achieving a better scalability.

Next we test robustness by adding noise in the normal direction to vertexes. Noise is generated by extracting random values in a range $[-\sigma, \sigma]$, where σ is defined as a fraction of the average edge length in the mesh. We show results with $\sigma = 0.25e, 0.5e, 1.0e$ at scales $8e$ and $16e$. Our method is compared just to the integral invariant method in this case. Each method takes as reference values the estimates it obtains on the noiseless version of each dataset at the same scale. Numerical results, reported in Table 3, show that the two methods produce similar results, the integral invariant performing slightly better for large error.

6.2 Creases

Finally, we test the performance of our method for extracting creases against (our implementation of) the method proposed in [Hildebrandt et al. 2005]. While the two methods produce comparable results on clean data and at small scales, our method performs much better than the other one when data are noisy, or scale is large, or both. Creases extracted with the method of [Hildebrandt et al. 2005] from noisy data may be highly fragmented and they are generally more irregular. Moreover, they do not always merge correctly to follow large scale features. A visual comparison at scale $8e$is reported in Figure 7. In the clean dataset (top row), note how our method correctly merges the parallel ridges around the rim of the hole, and along the left rib, while such ridges remain separated with the other method. Note that creases are drawn on the original mesh, thus they necessarily follow the jagged surface in the noisy case.

An example of the behavior of creases through different scales is reported in Figure 8. Note how creases slide through the surface, merge and disappear by increasing scale. For instance: the nose ridge is marked by two creases at fine scale, which soon merge into one crease that persists throughout the following scales. At the highest scale, this crease extends to mark the bilateral symmetry of the object. Creases that outline the arms of the Moai statue are detected at the fine scale, then they disappear. At fine scales, the body of the statue is outlined longitudinally by several creases, which progressively merge, ending up to the central longitudinal crease. Transverse creases at the eyebrows and at the top of the forehead first merge, then disappear when scale gets bigger; the same happens with other transverse creases, such as those marking the chin, the nostrils, and below the nose.

7 Concluding remarks

We have shown that fitting methods can be accurate in curvature estimation at different scales, provided that the fitting frame is chosen carefully and portions of surface bending away are discarded. We have also shown that Monte-Carlo sampling can reduce the time needed for fitting, without much loss of accuracy. However, at large scales, computation is still dominated by the time needed to extract vertex neighborhoods. This problem can be probably resolved by developing suitable data structures for mesh traversal. We will address this specific problem in our future work.

We have presented a new discrete method for extracting creases that is accurate and robust, and combines nicely with our multiscale curvature estimation. We have demonstrated on some examples how creases subsume shape features at the various scales: they may either disappear, or slide over the surface as scale increases,

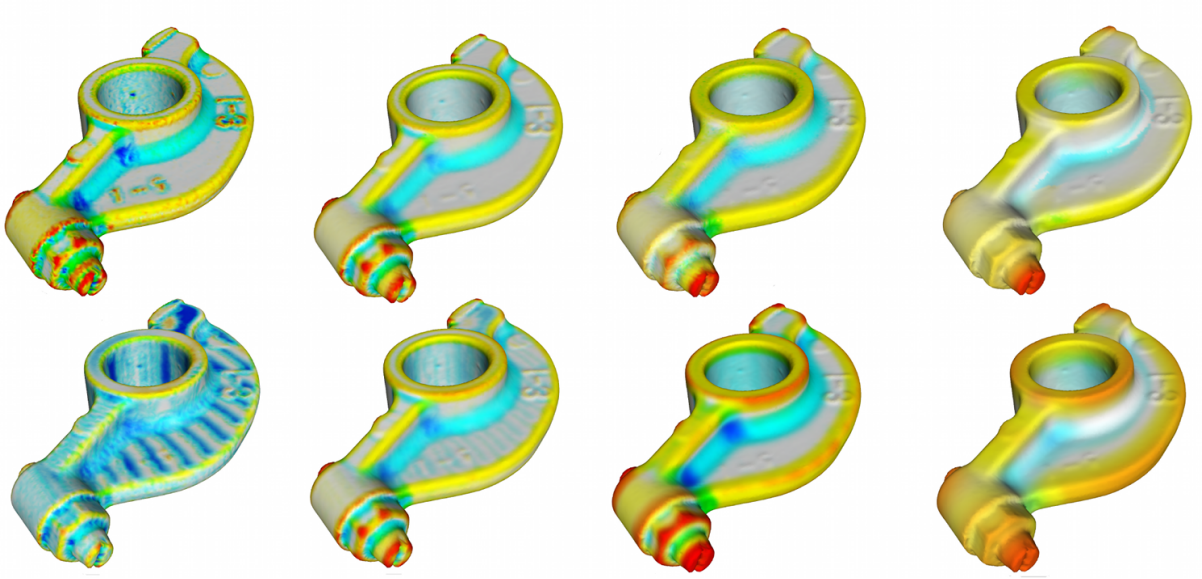

Figure 5: *Curvature extracted from the rockerarm dataset at scales* $2e$*,* $4e$*,* $8e$*,* $16e$*. Top row: our method; bottom row: integral invariant method.*

σ	0.25e				0.50e				1e			
Scales	8e		16e		8e		16e		8e		16e	
RockerArm	0.99	0.99	0.99	0.99	0.99	0.99	0.96	0.99	0.97	0.98	0.81	0.99
David head	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.93	0.97	0.94	0.98

Table 3: *Average deviation of curvature directions at different scales and different noise* σ*. The error made by our method (left columns), and by the integral invariant method (right columns) is measured as the scalar product between the principal directions of curvature.*

and they may eventually merge.

It would be interesting to compare our results with respect to creases extracted from progressively smoothed versions of the same surface, e.g., with Gaussian smoothing applied at different scales. More generally, it is an open issue to establish a formal mathematical relation between the radius r of the neighborhood used to evaluate curvature and the scale of a corresponding smoothing filter.

Our goal in the near future is to build a scale-space for surface features, which can provide a flexible tool for shape analysis and processing. In the scale space, creases are characterized with both position and persistence through scales. Since creases evolve in a quite complicated way, tracing them through the various scales is likely to be a challenging task.

Acknowledgements

We wish to thank the authors of [Yang et al. 2006] for providing the software to compute multi-scale curvatures with the integral invariant method.

References

AGAM, G., AND TANG, X. 2005. A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE Transactions on Visualization and Computer Graphics 11*, 5, 573–583.

CAZALS, F., AND POUGET, M. 2005. Estimating Differential Quantities using Polynomial fitting of Osculating Jets. *Computer Aided Geometric Design 22*, 121–146.

CAZALS, F., AND POUGET, M. 2005. Topology driven algorithms for ridge exctraction on meshes. Tech. Rep. 5526, INRIA.

CAZALS, F., AND POUGET, M. 2008. Algorithm 889: Jet_fitting_3: A generic c++ package for estimating the differential properties on sampled surfaces via polynomial fitting. *ACM Trans. Math. Softw. 35*, 3, 1–20.

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted delaunay triangulations and normal cycle. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, ACM, New York, NY, USA, 312–321.

COSTA BATAGELO, H., AND WU, S.-T. 2007. Estimating curvatures and their derivatives on meshes of arbitrary topology from sampling directions. *The Visual Computer 23*, 9, 803–812.

DESBRUN, M., GRINSPUN, E., AND SCHRÖDER, 2005. Discrete differential geometry: an applied introduction. ACM SIGGRAPH 2005 Course Notes.

DOUROS, I., AND BUXTON, B. 2002. Three-dimensional surface curvature estimation using qudric surface patches. In *Proceedings Scanning 2002*.

Eigen. `http://eigen.tuxfamily.org/`.

GATZKE, T. D., AND GRIMM, C. M. 2006. Estimating curvature on triangular meshes. *International Journal on 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*, 1, 45–63.

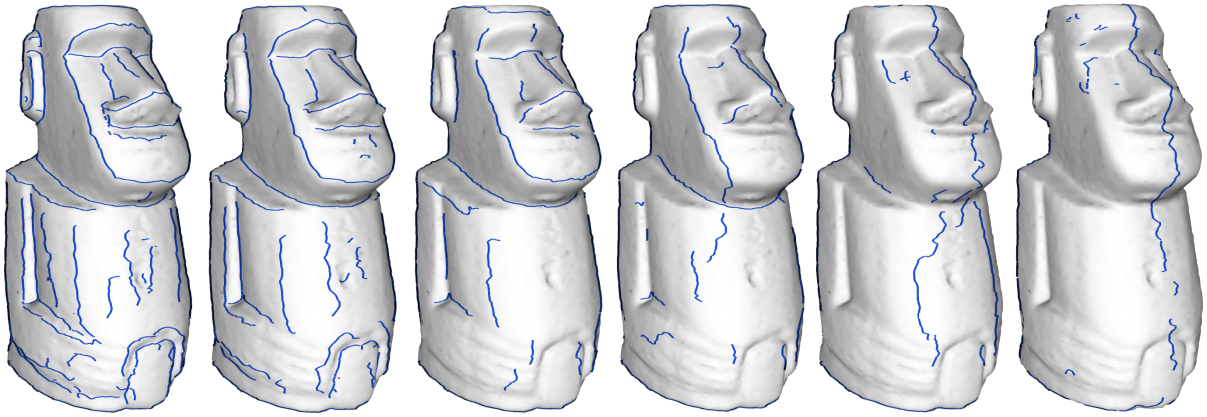

Figure 8: *Crease slide through the surface, merge and disappear as the scale increases.*

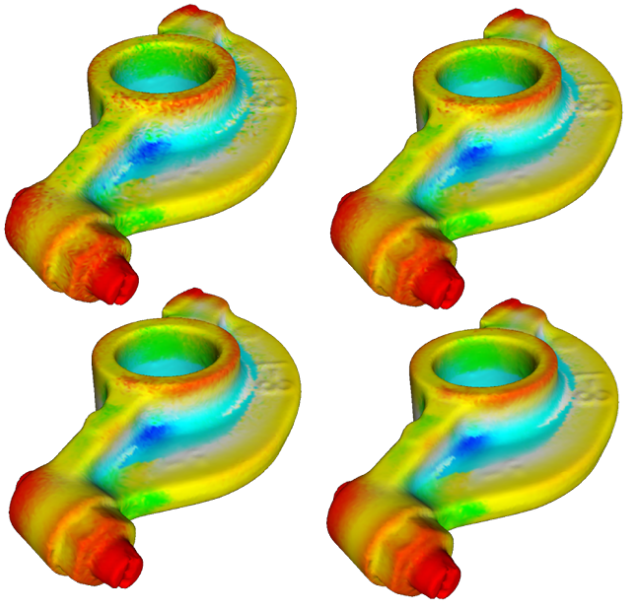

Figure 6: *Curvature extracted at scale* $16e$*. Top: Monte-Carlo with 50 samples (left) and 100 samples (right); bottom: Monte-Carlo with 200 samples (left) and full neighborhood using about 1200 data points on average (right).*

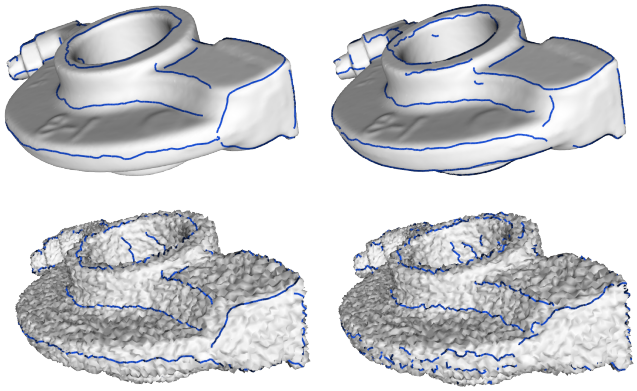

Figure 7: *Creases computed at scale* $8e$ *with our method (left) and with the method of [Hildebrandt et al. 2005] (right). Clean data (top) and noisy data with* $\sigma = 1.0e$ *(bottom).*

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

HILDEBRANDT, K., POLTHIER, K., AND WARDETZKY, M. 2005. Smooth feature lines on surface meshes. In *Proc. 3rd Eurographics Symp. on Geom. Proc.*, 85.

KOENDERINK, J. 1994. *Scale-space theory in computer vision*. Kluwer Academic.

LINDBERG, T. 2009. Scale-space. In *Encyclopedia of Computer Science ad Engineering*, B. Wah, Ed. John Wiley and Sons, 2495–2504.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H.-P. 2004. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 609–612.

PAULY, M., KOBBELT, L. P., AND GROSS, M. 2006. Point-based multiscale surface representation. *ACM Trans. Graph. 25*, 2, 177–193.

PORTEOUS, I. 2001. *Geometric Differentiation (2nd edition)*. Cambridge University Press.

POTTMANN, H., WALLNER, J., YANG, Y.-L., LAI, Y.-K., AND HU, S.-M. 2007. Principal curvatures from the integral invariant viewpoint. *Comput. Aided Geom. Des. 24*, 8-9, 428–442.

Vcglib. `http://vcg.sourceforge.net/`.

YANG, Y.-L., LAI, Y.-K., HU, S.-M., AND POTTMANN, H. 2006. Robust principal curvatures on multiple scales. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 223–226.

YOSHIZAWA, S., BELYAEV, A., AND SEIDEL, H.-P. 2005. Fast and robust detection of crest lines on meshes. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, ACM, New York, NY, USA, 227–232.