STAG: Smart Tools and Applications in Graphics (2022)
D. Cabiddu, T. Schneider, G. Cherchi, and R. Scateni (Editors)

Nearly smooth differential operators on surface meshes

C. Mancinelli[1] and E. Puppo[1]

[1]DIBRIS - University of Genoa, Italy

Abstract

Estimating the differential properties of a signal sampled on a surface is of paramount importance in many fields of applied sciences. In the common practice, the surface is discretized with a polygonal mesh, the signal is sampled at its vertices and extended linearly over the triangles. This means that the polyhedral metric is assumed over the surface; the first derivatives of the signal become discontinuous across edges; and the second derivatives vanish. We present a new method based on surface fitting, which efficiently estimates the metric tensor, and the first and second order Riemannian differential operators at any point on the surface. All our differential operators are smooth within each triangle and continuous across the edges, providing a much better estimate of differential quantities on the - yet unknown - underlying smooth manifold.

1. Introduction

Roughly speaking, a surface S can be defined as a set of points that, locally, "looks like" a plane. Such local characterization can be formalized by some function $\mathbf{x}(u,v)$, called a *local parametrization*, which maps an open set $V \subset \mathbb{R}^2$ to a subset U of S. This correspondence plays a crucial role in the study of S, since it encodes the deformation to which an infinitesimal portion of the real plane undergoes when mapped to S. In order to set a *differentiable structure* on S, we need $\mathbf{x}$ to be smooth: the more information we ask about S, the more smoothness we require to $\mathbf{x}$. For instance: C^1 continuity is sufficient to define a *metric* on S, and to estimate its surface normal; while other quantities such as curvature or the Laplacian of a scalar function on S require C^2 continuity.

When considering the discretization of a surface, e.g., a triangle (manifold) mesh M, we implicitly assume a piecewise linear parametrization $\mathbf{x}$, which has a trivial structure inside faces and is just C^0 across edges. In this case, it is not possible to endow M with a differentiable structure in the sense described above.

Traditionally, to overcome this limitation, discrete counterparts of the concepts of differential geometry have been proposed [MDSB03]. For example, given a scalar field f defined at the vertices of M, linear interpolation inside faces is usually assumed; hence a piecewise constant gradient within each triangle, which will be discontinuous across edges; and vanishing second derivatives everywhere. Note that, in this way, the Euclidean (flat) metric is implicitly assumed within triangles. The resulting *polyhedral metric* over the mesh has singularities at all (non-flat) vertices, causing behaviors that contradict the smooth setting, e.g., in the computation of geodesic fields.

In this paper, we follow a different approach. We look at M as a discrete representation of an unknown smooth surface S, and we estimate the differential structure of S *pointwise*, through an approximation of a smooth local parametrization $\mathbf{x}_p(u,v)$ in the neighborhood of each point p. We achieve this goal by fitting smooth surfaces to the neighborhoods of vertices of M, and blending such surfaces inside triangles. Although approaches based on surface fitting have been used extensively in the literature, they are usually bound to computations at a discrete set of points, typically vertices. Subdivision schemes may be used to obtain a closed form surface that either interpolates or approximates the initial data, but, as explained below, there are drawbacks and/or limitations that make them unsuited for our purposes. In this paper, we propose an efficient way to estimate the differential structure at any point, warranting continuous values over the whole surface for all operators up to second order.

Our method allows us to define a non-flat metric which smoothly varies within the triangles of M, and with continuity across the edges. This, in turn, leads to the computation of first and second order *covariant* derivatives at any point of M, which have a closed form in terms of the local parametrization. We apply the same approach to extend a discrete scalar field defined at the vertices to the whole surface. Focusing our attention to geodesic distance fields, we demonstrate that our differential operators provide much better results than the commonly used linear interpolation. In particular, we show how computation on coarse meshes can approximate well the behavior of fields defined over finely tessellated surfaces, with obvious smoothing effects due to the loss of detail.

We believe that our approach may lead to a change of paradigm in many problems in geometry processing: from shortest paths tracing to convex optimization.

DOI: 10.2312/stag.20221255

2. Related work

Linear regression using a polynomial function approximating a given finite set of points is a well known technique in computer graphics since decades. It has been used either to estimate differential quantities [JS02, LZ13, SK15, WYLC13, SK15], or geometric features [MW00, PPR10] or both [CP05, Xu13]. The idea is to obtain a closed form from the best fitting polynomial (usually quadratic) function in the least squares sense, which approximates the geometry of a given point set and then compute the quantities of interest analytically. Our starting point is the algorithm proposed by Xu [Xu13], which is described in Section 4.

Pointwise evaluation of differential quantities may also be obtained through discrete exterior calculus (DEC) [DKT08, CdGDS13], or finite element methods (FEM) [PP93, War06, WMKG07]. These methods are quite efficient, but, as discussed by Wardetzky and colleagues [WMKG07] and by Xu [Xu13], they are less robust than methods based on fitting, and pointwise convergence is not always guaranteed.

Bézier and Hermite patches can be used to have a closed form of an approximation of the geometry represented by a triangular mesh [VPBM04, JK13]. At the best of our knowledge, the only approach in the literature that uses these techniques to improve the pointwise estimation of a given field on a triangular are the PN-triangles introduced in [VPBM04]. Nevertheless, the quadratic varying normal field defined on these patches is only C^0 across edges, while by construction ensure C^1 at the vertices of the mesh.

Subdivision schemes may also be used to define a smooth surface that approximates a given geometry [Doo78, CC78, Loo87, ZSS96]. They can be either approximating or interpolating. The latter are known to produce lower quality shape with respect to the former [ZS99]. When dealing with triangular meshes, one of the most used is the Loop subdivision scheme [Loo87]. Jos Stam studied such a scheme in [Sta98], in which he showed how the limit surface of such scheme can be expressed in parametric form at every point of the mesh. However, the Loop scheme is not interpolatory, meaning that the limit surface will not pass through the initial vertices. Although it is possible to apply an iterative process that converge to an auxiliary mesh $\hat{M}$ such that the limit surface of the Loop subdivision scheme applied to $\hat{M}$ will interpolate the vertices of our initial mesh M [DM12, CFL*09], computing $\hat{M}$ in a straightforward way starting from the vertices of M is expensive. Moreover, within a triangle incident to an extraordinary vertex (i.e. vertices with valence $\neq 6$), the limit surface of the Loop scheme is defined on a parametric domain that keeps shrinking when approaching such a vertex, eventually collapsing into one point when reaching it. Therefore, at these vertices, the only information we can have is their position on the smooth surface, but nothing else.

3. Preliminaries

We briefly review the basic concepts related to the differentiable structure of a smooth surface. Further details can be found in any introductory book of differential geometry [dC92, Sak97].

Let $S \subset \mathbb{R}^3$ be an embedded surface. For $p \in S$, we consider a local smooth parametrization $\mathbf{x}_p : V \to S$ mapping an open set $V \subset \mathbb{R}^2$ to a neighborhood U of p. We will conventionally use the same small letter with and without a bar to denote corresponding points through the parametrization, i.e., $p = \mathbf{x}_p(\bar{p})$ for $\bar{p} \in V$. For brevity, we will omit the argument $\bar{p}$ of all operators, whenever this does not cause ambiguity, e.g., we will write $\frac{\partial \mathbf{x}_p}{\partial u^i}$ for $\frac{\partial \mathbf{x}_p}{\partial u^i}(\bar{p})$.

Let (u^0, u^1) be the coordinates of a frame in $\mathbb{R}^2$. The Jacobian

$$J(\bar{p}) = \left[\frac{\partial \mathbf{x}_p}{\partial u^0}, \frac{\partial \mathbf{x}_p}{\partial u^1}\right]$$

of $\mathbf{x}_p$ at $\bar{p}$ provides a basis of the tangent plane T_pS of S at p. Since $\mathbf{x}_p$ maps a portion of the plane into a curved domain, lengths and angles are not preserved in general. The *metric tensor* (a.k.a. *first fundamental form*) keeps track of how such quantities are transformed in a neighborhood of p. The metric tensor at p is a bilinear operator, which can be represented as a 2×2 symmetric matrix G_p in the given local reference system, defined component-wise as

$$g_{ij} = \langle \frac{\partial \mathbf{x}_p}{\partial u^i}, \frac{\partial \mathbf{x}_p}{\partial u^j} \rangle.$$

The metric tensor defines an inner product on T_pS, giving a rule to multiply its vectors. As a bilinear operator, it is independent of the specific parametrization and depends just on the intrinsic properties of S, controlling how distances, angles and areas are measured in the neighborhood of p. By defining the surface normal

$$\mathbf{n} = \frac{\frac{\partial \mathbf{x}_p}{\partial u^0} \times \frac{\partial \mathbf{x}_p}{\partial u^1}}{\| \frac{\partial \mathbf{x}_p}{\partial u^0} \times \frac{\partial \mathbf{x}_p}{\partial u^1} \|},$$

then the *second fundamental form* is defined as follows

$$\mathbf{II} = \begin{pmatrix} \frac{\partial^2 \mathbf{x}_p}{\partial^2 u^0} \cdot \mathbf{n} & \frac{\partial^2 \mathbf{x}_p}{\partial u^1 \partial u^0} \cdot \mathbf{n} \\ \frac{\partial^2 \mathbf{x}_p}{\partial u^0 \partial u^1} \cdot \mathbf{n} & \frac{\partial^2 \mathbf{x}_p}{\partial^2 u^1} \cdot \mathbf{n} \end{pmatrix}.$$

The first and the second fundamental forms together provide full information on relevant differential properties of surface S at p, up to the second order. For instance, the *Gaussian curvature* K can be computed as the ratio between their determinants.

Now let $f : S \to \mathbb{R}$ be a scalar function defined on S. Then the *Riemannian gradient* ∇f is a vector field on S that associates to every point $p \in S$ a vector in T_pS. Second order derivatives are more involved to define in the Riemannian setting, since the they involve combining vectors belonging to different tangent spaces. This entails the definition of an *affine connection* on S, which we will not describe for brevity. It is sufficient to mention here that, given a proper affine connection, the definitions of the usual Laplacian and Hessian operators can be extended to the Riemannian setting, and they are defined just in terms of the first and second derivatives of the parametric function $\mathbf{x}$ and of the metric G. This, in turn, implies that if a local parametrization for S is given, then all the quantities mentioned above can be computed in closed form.

In the following, whenever no ambiguity arises, we will omit the dependence on point p in all our notations. Likewise, we will replace the notation (u^0, u^1) of coordinates in the parametric space with the more common notation (s,t), possibly adding subscripts when dealing with more than one parametric domain.

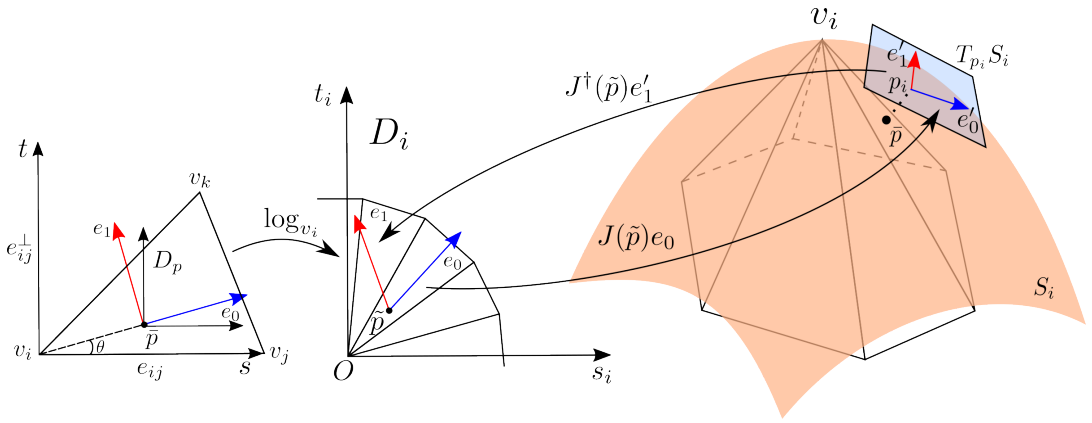

Figure 1: *The discrete logarithmic map* $\log_{v_i}$ *at vertex* v_i *maps the star of* v_i *to its tangent plane; triangle* t_{ijk} *is mapped to a warped version of it, bringing point* $\bar{p}$ *to* $\tilde{p}$ *(left to center). The quadric* $\mathbf{x}_i$ *maps* $\tilde{p}$ *to* p_i *on surface* S_i *computed at* v_i*; its Jacobian at* $\tilde{p}$ *maps vectors in the (discrete) tangent space at* v_i *of* M *to vectors in the tangent space of* S_i *at* p_i*: vector* $\tilde{p}-O$ *is mapped to* e'_0 *(center to right). The pseudo-inverse of the Jacobian* $J^\dagger(\tilde{p})$ *maps the direction* e'_1 *orthogonal to* e'_0 *to direction* e_1*. Transformation* T_0 *is the affine mapping between frames* (e'_0, e'_1, p_i) *and* $(e_0, e_1, \tilde{p})$*. Transformation* T_1 *is just a rotation for an angle* $-\theta$ *to align* $(e_0, e_1, \tilde{p})$ *with the reference system of* t_{ijk}*.*

4. Method

In a discrete setting, we are given a mesh M, which is a piecewise-linear surface. Hence no smooth parametrization $\mathbf{x}$ could be defined in the neighborhood of any point lying on an edge, or at a vertex of M. As outlined before, we will assume M as an approximation of an unknown smooth surface S, interpolating M at its vertices, and we will estimate the differential operators on S.

Our approach consists of two steps:

1. **Quadratic Fitting at Vertices.** We approximate the surface S around each vertex with a quadric, using the method proposed in [Xu13]. The result provides a smooth local parametrization $\mathbf{x}_i$ of surface S around vertex v_i. This is computed once for all vertices and the coefficients of all quadrics are stored in our data structure. We use the same approach to fit a local smooth function to a discrete scalar field f defined at the vertices of M.
2. **Blending of Quadrics.** For any other point $\bar{p} \in M$, belonging to a triangle t, we blend on-the-fly the quadrics at the vertices of t to obtain a new quadric, which provides a local smooth parametrization $\mathbf{x}_p$ of S around point p corresponding to $\bar{p}$, and likewise for the field f. The blended quadrics provide a pointwise estimate of the surface S and the field f, and their differentiable structure at p. This step is our main contribution.

Once a local smooth parametrization is available at a generic point p, which is expressed in closed form with polynomial functions, the differential operators defined in the previous sections can be trivially computed from the derivatives of such functions.

The two steps of our method are described in detail in the following subsections.

4.1. Quadratic Fitting at Vertices

We briefly describe how the quadric $\mathbf{x}_i : D_i \to \mathbb{R}^3$ approximating the geometry around a vertex $v_i \in M$ is computed, following the approach proposed in [Xu13].

Let $\{v_{i_1}, \ldots, v_{i_h}\}$ be the vertices in the 1-ring of v_i. We map such vertices to D_i, coincident with the tangent plane at v_i, by rescaling the angles formed by the edges incident at v_i (see e.g. [ZMT07]), so that every one of them has its coordinate in the parametric domain.

By denoting with

$$\{Q_\ell(s,t)\}_{\ell=1}^5 := \{s, t, s^2, st, t^2\},$$

the basis functions of a quadric polynomial through the origin, then we define as

$$\mathbf{x}_i(s,t) = \sum_{\ell=1}^{5} c_\ell Q_\ell(s,t) + v_i,$$

where the coefficients $c_\ell \in \mathbb{R}^3$ are chosen so that $\mathbf{x}_i$ is the best fitting quadric of the neighborhood of v_i in the least square sense. For computational reasons, we store the inverse of the matrix associated to the linear system (if such matrix is singular, we compute its pseudo-inverse [GVL96]). We denote such a matrix with C_i. Note that, since v_i is identified with the origin of D_i, by the choice of the basis function we have that $\mathbf{x}_i$ interpolates v_i.

Likewise, given a discrete scalar field $\mathbf{f} = (f_1, \ldots, f_n)$, sampled at the n vertices of M, we extend $\mathbf{f}$ to the neighborhood of v_i with the scalar function

$$\hat{f}_i(s,t) = \sum_{\ell=1}^{5} d_\ell Q_\ell(s,t) + \mathbf{f}_i,$$

where $d_\ell \in \mathbb{R}$.

Both the parametrization $\mathbf{x}_i$ and the corresponding matrix C_i are computed once at all vertices of M and stored in a data structure. Once a discrete field $\mathbf{f}$ is given, the evaluation of function $\hat{f}_i$ entails a simple matrix-vector computation. Note that this formulation is best for local evaluations of $\hat{f}_i$, while it can be optimized for global computations over the whole mesh, by gathering the coefficients of all the C_i's in a unique sparse $5n \times n$ matrix.

It is worth pointing out that the choice of quadratic polynomials is justified by the fact that we are interested in estimating second order differential quantities, so 2 is the least degree we can work with. Besides the fact that higher order polynomials would require a larger stencil, we think that increasing the degree of the fitting polynomial may introduce shape artifacts and bumpy regions. In the future, we plan to demonstrate this claim as well as relaxing the interpolatory constraint on the fitting polynomials.

4.2. Blending of Quadrics

We now extend the differential operators, which were computed at a vertex v_i in the previous section, to the rest of mesh M. We will use the notation $\bar{p}$ to denote a generic point of M, assuming an implicit one-to-one mapping between $\bar{p}$ and a corresponding point p on the (unknown) smooth surface S. Differential quantities will refer to the points of S, while the triangles of M will be taken as local parametric domains to define such implicit mapping.

Let t_{ijk} be a triangle of M with vertices v_i, v_j, v_k. A point $\bar{p}$ of t_{ijk} can be represented with barycentric coordinates (α, β) w.r.t. t_{ijk} as

$$\bar{p} = (1-\alpha-\beta)v_i + \alpha v_j + \beta v_k.$$

Let $\mathbf{x}_i(s_i,t_i)$, $\mathbf{x}_j(s_j,t_j)$ and $\mathbf{x}_k(s_k,t_k)$ be the quadric surfaces parametrized at the vertices of t_{ijk}, as described before. We will define a parametrization $\mathbf{x}_p(s,t)$ about $\bar{p}$ by blending such quadrics, with weights corresponding to the barycentric coordinates of $\bar{p}$. This will provide a family of quadrics that vary smoothly across t_{ijk}, and continuously through its edges and vertices.

To this aim, we need to reparametrize the quadrics $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$ to a common domain. We set a 2D reference system in the plane of t_{ijk} having its axes aligned with edge e_{ij}, and with its orthogonal direction $e_{ij}^{\perp}$, respectively, and the origin at vertex v_i. The local parametrization $\mathbf{x}_p(s,t)$ will be defined with respect to a parametric domain D_p, which is a copy of the same reference system with origin translated in $\bar{p}$. See Fig. 1 left.

In the following, we describe how to reparametrize $\mathbf{x}_i$ in such domain; the descriptions for $\mathbf{x}_j$ and $\mathbf{x}_k$ are analogous. We define a correspondence between the parametric domain D_i of $\mathbf{x}_i$ and D_p, which in our case is a linear mapping $T_i : D_p \to D_i$ such that quadric $\mathbf{x}_i(s_i,t_i)$ is reparametrized as $\mathbf{x}_i \circ T_i(s,t)$. The mapping T_i is obtained as a composition of two transformations T_0 and T_1 (dependence of T_0 and T_1 from index i is omitted to keep a light notation).

Let $\widetilde{p}$ be the image of $\bar{p}$ into the tangent plane at v_i, which is obtained with the logarithmic map centered at v_i, as in the previous section. The parametric domain in this tangent plane is denoted D_i with reference frame (s_i,t_i,O) (see Fig. 1 left to center). Let S_i be the trace of $\mathbf{x}_i$. We set $p_i = \mathbf{x}_i(\widetilde{p})$ the image of such point through the quadric at v_i. Transformation T_0 maps the tangent plane $T_{p_i}S_i$ to domain D_i through the inverse of the Jacobian J of $\mathbf{x}_i$ (see Fig. 1 right to center). More precisely, the tangent plane at p_i is spanned by the images through J of the reference axes s_i and t_i. We set $e_0 = \widetilde{p} - O$ and e_0' its normalized image in $T_{p_i}S_i$ through $J(\widetilde{p})$; and we choose e_1' as the orthogonal direction to e_0'. We set an orthonormal reference frame in $T_{p_i}S_i$ defined as (e_0', e_1', p_i) (see Fig. 1 center). Now we consider the (non orthonormal) reference frame in D_i defined with $(e_0, e_1, \widetilde{p})$, where axes are computed through the inverse of the Jacobian, i.e., $(e_0, e_1) = J^{\dagger}(p_i)(e_0', e_1')$. Finally, transformation T_0 is defined as the affine mapping from frame (e_0', e_1', p_i) to frame $(e_0, e_1, \widetilde{p})$.

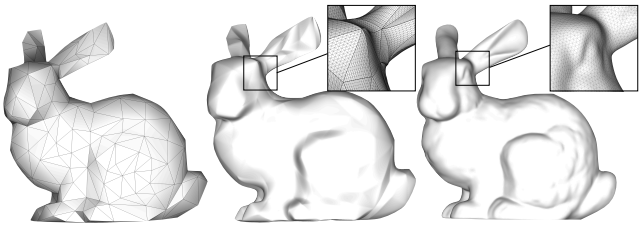

Figure 2: *The coarse mesh in input consisting of 500 triangles (left), is obtained from simplification of a reference mesh consisting of $\sim$ 70k triangles (right). Our nearly smooth reconstruction is obtained from the coarse mesh and has about the same number of triangles of the reference mesh (center).*

At this point, we have three independent reparametrizations of $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$ with respect to three orthonormal frames with origin at p_i, p_j, p_k, respectively. Note that p_i, p_j, p_k are the estimates of our point p on S according to $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$, respectively. In order to blend such estimates, we need to reconcile the three reference frames. To this aim, we first rigidly identify domain $T_{p_i}S_i$ with the plane of t_{ijk}, mapping p_i to $\bar{p}$ and direction e_0' to $\bar{p} - v_i$, then we apply a rotation to aligning the two reference frames. Transformation T_1 is defined as a rotation for an angle $-\theta$, where θ is the angle between $\bar{p} - v_i$ and the s axis in the reference frame of t_{ijk} (see Fig. 1 left).

With obvious notation, we now have that $\mathbf{x}_i(T_i(s,t)), \mathbf{x}_j(T_j(s,t))$ and $\mathbf{x}_k(T_k(s,t))$ are three quadrics defined in the same system of coordinates, thus we can define $\mathbf{x}_p(s,t)$ as

$$\mathbf{x}_p(s,t) = (1-\alpha-\beta)\mathbf{x}_i(T_i(s,t)) + \alpha\mathbf{x}_j(T_j(s,t)) + \beta\mathbf{x}_k(T_k(s,t)).$$

Note that, since the quadrics defined along an edge e_{ij} depend just on $\mathbf{x}_i$, $\mathbf{x}_j$ and are reparametrizad in equivalent ways (up to rotation) in the two triangles incident at e_{ij}, the family of blended quadrics vary with continuity across e_{ij}.

We use exactly the same mechanism to blend scalar fields $\hat{f}_i$, $\hat{f}_j$, $\hat{f}_k$, estimated about the vertices, to obtain an estimate of scalar field $\hat{f}_p$ about p. At this point, $\mathbf{x}_p$ and $\hat{f}_p$ provide local estimates of the surface and the field, respectively, in the neighborhood of p, which can be used to compute all differential operators at p in closed form.

5. Experimental results

In the following, we present some results obtained with the method described above to pointwise estimate the quantities introduced in Section 3. We report experiments on just one object; extensive experiments to assess robustness and performance are deferred to future work. We take a watertight version of the original Stanford bunny, consisting of about 70k triangles, and a drastically simplified version of the same object, consisting of just 500 triangles. We compare results obtained on points sampled from the coarse mesh with our method, against results on the same mesh and the same

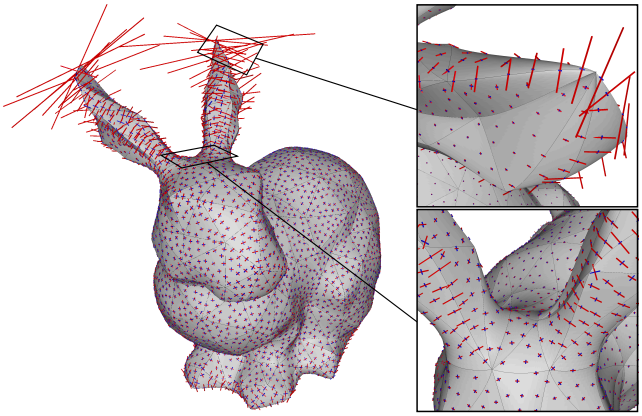

Figure 3: *The metric tensor estimated from the coarse mesh is visualized on top of the reconstructed mesh. Each glyph shows the eigen directions e_i of the tensor, rescaled with respect to their corresponding eingenvalues λ_i. The scale factor applied to eigenvector e_i is $1/\sqrt{\lambda_i}$, $i = 0, 1$.*

points obtained with linear interpolation, and against results on the high-resolution mesh.

We will use two different sampling techniques: a regular sampling (RS), to tessellate every triangle with a regular triangular grid, and a Poisson disk sampling (PS) to obtain a more uniform distribution of random points over the mesh. Both mesh simplification and the Poisson disk sampling have been computed with Meshlab [CCC*08].

5.1. Shape properties

We first present results on properties that depend just on the underlying shape, namely: surface reconstruction, metric tensor, and curvature.

Figure 2 shows the coarse mesh, the reference mesh, and our reconstruction, which is obtained by tessellating every triangle of the coarse mesh into 150 triangles with the RS method (thus obtaining a resolution similar to the reference mesh) and estimating the position of the vertices with our functions $\mathbf{x}_p$, as described in the previous section. Although the fine details have been lost with simplification, our reconstruction provides a relatively smooth version, which approximates quite well the reference shape. As expected, the surface is not fully smooth across the edges of the coarse mesh, while it varies smoothly inside its triangles.

In Figure 3, we show the metric tensor computed on points from the PS method. We visualize glyphs representing the eingenvectors of the metric tensor, rescaled with respect to their corresponding eingenvalues. The relative lengths of the exas of each cross reflect the anisotropy of the metric, which is related to the local curvature. It can be clearly seen that the tensor varies smoothly inside triangles and with continuity across edges.

In Figure 4 we show the Gaussian curvature K computed as described in Section 3.

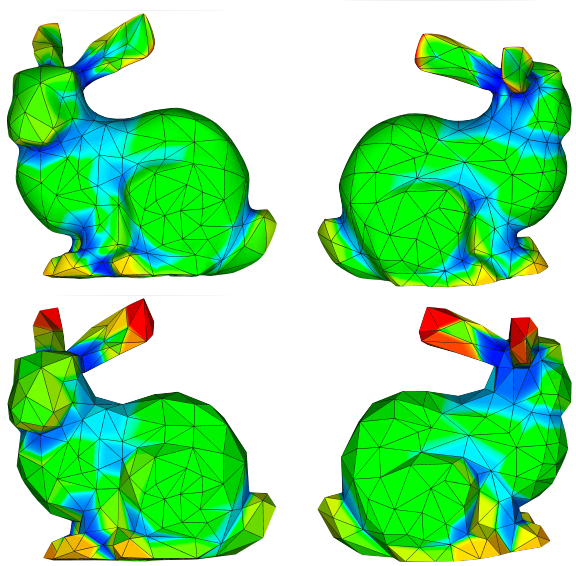

Figure 4: *Gaussian curvature computed with our method (top) and using the approach of [Xu13] to estimate it at the vertices and using linear interpolation inside triangles (bottom).*

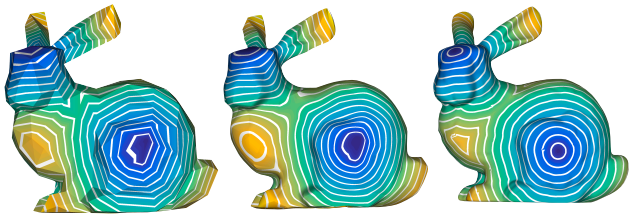

Figure 5: *Heatmap and isolines of a geodesic distance field from three sources. Our method (center) provides a much smoother result than the traditional linear interpolation (left) and closely approximates the result obtained on the reference mesh for a distance field from similarly-placed sources (right).*

Results are shown with heatmaps by using linear interpolation over triangles of the coarse mesh versus our RS reconstructed mesh. It is evident how linear interpolation causes abrupt changes, while our method provides a much smoother result without evident artifacts. The differences are particularly visible in regions where the curvature varies the most (e.g., at ears and feet): linear interpolation poorly catches such variations, resulting in quick changes from one triangle to another. On the other hand, since our method estimates the curvature at any given point within a triangle by blending smooth surfaces, we obtain a much smoother transition even across the edges.

5.2. Field dependent differential operators

We take in input a geodesic distance field from multiple sources manually picked on the shape. The distance field is computed with

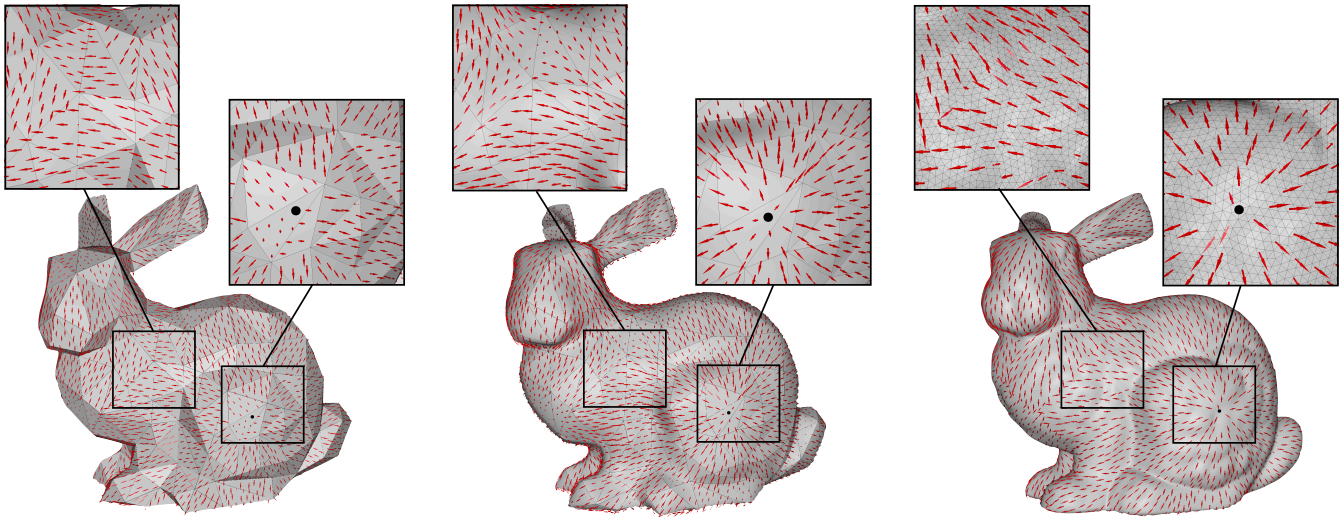

Figure 6: *Gradient of the distance field shown in Figure 5: linear interpolation on coarse mesh (left), our method (center) and linear interpolation on the reference mesh (right). On the coarse mesh, the variation of the gradient field is abrupt with linear interpolation and quite smooth with our method, which has a behavior consistent with what happens on the reference mesh. The major differences can be noticed in the proximity of singularities such as sources (bottom-right inset), and saddles (top-left inset): on the coarse mesh, they are completely lost, on the reference mesh, they can be easily seen since the gradient is discontinuous there, while with our method the gradient, they are critical points of the field.*

the VTP algorithm [XW07] and stored just at the vertices of the coarse mesh.

We estimate the field and its differential structure at all vertices of either our RS reconstruction, or from the PS set. Figure 5 shows a comparison between the field linearly interpolated on the coarse mesh, our result on the reconstructed shape, and on the reference mesh. Again, it can be easily seen, both from the heatmap and from the isolines, that our result is smoother and overall consistent with the one from similarly-placed sources on the reference mesh.

Figure 6 shows the gradient of the same distance field of Fig. 5 at the points of the PS set. We compare our method (center) with the traditional linear interpolation over triangles both on the coarse mesh (left) and on the reference mesh (right).

While the gradient is constant inside each face by using linear interpolation, our results vary smoothly inside each face and nicely across edges, capturing quite well the overall flow of the field. This is particularly evident if we compare the result obtained on the coarse mesh with ours: singularities such as sources and saddles are completely lost with the linear interpolation, while they are detected clearly with our method, even if they lie in the middle of triangles. Moreover, even if they are overall consistent, the comparison between the gradient field on our reconstruction and the one on the reference mesh indicates that the field we are estimating is smoother than the geodesic distance field. In fact, our gradient smoothly varies when approaching the cut locus, while the gradient on the reference mesh "breaks", as it should, since the geodesic distance is not differentiable there.

When solving optimization problems, it is important to assess whether the energy E we are trying to minimize is convex or not.

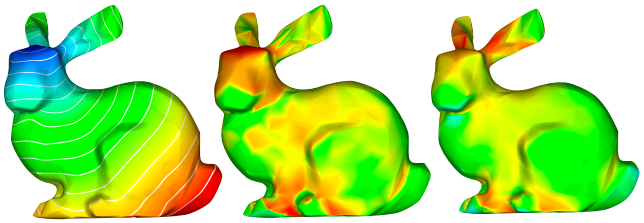

Figure 7: *Heatmap of the minimum and maximum eigenvalue (center and right, respectively) of the second covariant derivative of the distance field from a point placed on the head of the bunny (left).*

Usually, this is done by checking the positive-definiteness of its Hessian. In Figure 7, we choose $E(q) = d_{\bar{p}}(q)$, i.e. a distance field from a point $\bar{p}$ on the head of the bunny(left), and we show the heatmap of the minimum and maximum eigenvalue of $\nabla^2 E$ (center and right, respectively).

We then consider the field $\sin(10E(q))$ (Figure 8, left), and we compute the Laplacian of the such field using our method. We compare our result with the one obtained on the reference mesh, where the Laplacian in this case has been computed using the cotangent Laplacian. The right portion of the Figure 8 shows that, our result (bottom) is overall consistent with the one computed on the reference mesh (top). It is interesting to note how the geometric details that we have lost during the decimation process have not a clearly visible effect on the distance field (left), but visibly influence the behavior of the Laplacian (right).

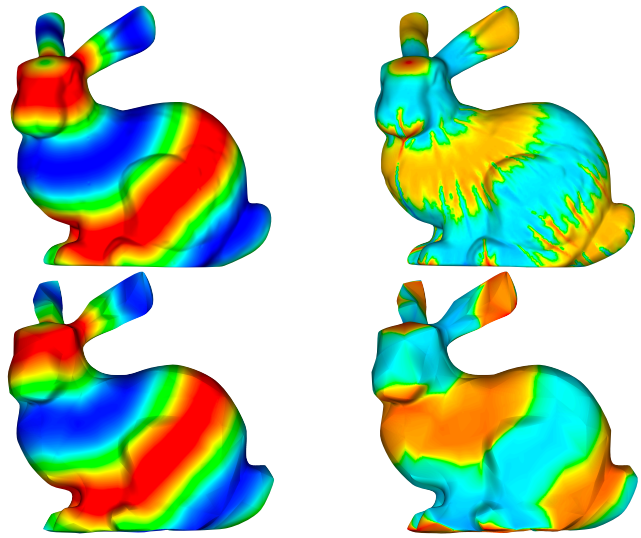

Figure 8: *The results obtained computing the Laplacian of* $\sin(10 d_{\bar{p}}(q))$ *with our method on the reconstructed mesh (bottom-right) and on the reference mesh (top-right). The heatmap of signal is depicted on the left of the image.*

6. Concluding remarks

We have presented a method based on surface fitting to estimate pointwise differential operators, up to second order, over surface meshes and scalar fields defined on them. All our operators are smooth on triangles and continuous across edges. Most computations to estimate these operators are performed just once on the input mesh, while pointwise evaluation entail simple and efficient matrix-vector computations.

This is just preliminary work, which we plan to extend in several directions in the near future. First of all, we plan to make extensive comparisons about both accuracy and performance of the method with respect to other methods at the state of the art, and possibly widening the range of operators supported by our method (e.g. divergence, mean curvature, etc.).

Next, we will address applications, especially in the context of the many problems that require to use gradient descent on functions (usually energies) defined over the surface. Since we can efficiently compute both the gradient and the Hessian, we expect to obtain relevant improvements in terms of both accuracy and speed.

Our method may be also improved by addressing a better way to compute the quadrics at vertices, since the method of Xu [Xu13] is not always stable on vertices with low degree, non-regular neighborhood, and meshes with elongated triangles.

While our approach already improves over the state of the art, is remains a pointwise estimator, which does not guarantee smoothness across edges and vertices. An important open problem is to devise a globally smooth reconstruction of the surface and estimate of its differential properties. Moreover, such reconstruction should take into account sharp features that the user might want to preserve, which is not possible in our current implementation.

References

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355. 2

[CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G.: MeshLab: an Open-Source Mesh Processing Tool. In *EG Ital. Chap. Conf.* (2008), pp. 129–136. 5

[CdGDS13] CRANE K., DE GOES F., DESBRUN M., SCHRÖDER P.: Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (New York, NY, USA, 2013), SIGGRAPH '13, Association for Computing Machinery. 2

[CFL*09] CHENG F. H., FAN F. T., LAI S. H., HUANG C. L., WANG J. X., YONG J. H.: Loop subdivision surface based progressive interpolation. *Journal of Computer Science and Technology 24* (1 2009), 39–46. 2

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of oscu-lating jets, 2005. 2

[dC92] DO CARMO M.: *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992. 2

[DKT08] DESBRUN M., KANSO E., TONG Y.: Discrete differential forms for computational modeling. *Discrete Differential Geometry* (2008), 287–324. 2

[DM12] DENG C., MA W.: Weighted progressive interpolation of loop subdivision surfaces. *CAD Computer Aided Design 44* (5 2012), 424–431. 2

[Doo78] DOO D.: A subdivision algorithm for smoothing down irregular shaped polyhedrons. pp. 157–165. 2

[GVL96] GOLUB G. H., VAN LOAN C. F.: *Matrix Computations*, third ed. The Johns Hopkins University Press, 1996. 3

[JK13] JAKLIČ G., KANDUČ T.: Hermite interpolation by triangular cubic patches with small willmore energy. *International Journal of Computer Mathematics 90* (9 2013), 1881–1898. 2

[JS02] JIRKA T., SKALA V.: Gradient vector estimation using quadratic regression function. Wojciechowski K., (Ed.), Silesian University of Technology, pp. 380–386. 2

[Loo87] LOOP C.: *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, January 1987. 2

[LZ13] LIANG J., ZHAO H.: Solving partial differential equations on point clouds. *SIAM Journal on Scientific Computing 35*, 3 (2013), A1461–A1486. 2

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and Mathematics III*. Springer, Berlin, Heidelberg, Berlin, Heidelberg, 2003, pp. 35–57. 1

[MW00] MEEK D., WALTON D.: On surface normal and gaussian curvature approximations given data sampled from a smooth surface. *Computer Aided Geometric Design 17*, 6 (2000), 521–543. 2

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Math. 2*, 1 (1993), 15–36. 2

[PPR10] PANOZZO D., PUPPO E., ROCCA L.: Efficient multi-scale curvature and crease estimation. In *2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa 2010 - Workshop Proceedings* (2010), pp. 9–16. 2

[Sak97] SAKAI T.: *Riemannian Geometry*, vol. 149. American Mathematical Society, 1997. 2

[SK15] SZIRMAY-KALOS L.: Filtering and gradient estimation for distance fields by quadratic regression. *Periodica polytechnica Electrical engineering and computer science 59* (2015), 175–180. 2

[Sta98] STAM J.: Evaluation of loop subdivision surfaces. In *SIGGRAPH 1998 CDROM Proceedings* (1998), Citeseer. 2

[VPBM04] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. 159–166. 2

[War06] WARDETZKY M.: *Discrete Differential Operators on Polyhedral Surfaces: Convergence and Approximation*. PhD thesis, Freie University at Berlin, 2006. 2

[WMKG07] WARDETZKY M., MATHUR S., KÄLBERER F., GRINSPUN E.: Discrete laplace operators: No free lunch. *Eurographics Symposium on Geometry Processing* (2007). 2

[WYLC13] WANG R., YANG Z., LIU L., CHEN Q.: Discretizing Laplace-Beltrami Operator from Differential Quantities. *Comm. Math. Stati. 1*, 3 (2013), 331–350. 2

[Xu13] XU G.: Consistent approximations of several geometric differential operators and their convergence. *Applied Numerical Mathematics 69* (2013), 1–12. 2, 3, 5, 7

[XW07] XIN S. Q., WANG G. J.: Efficiently determining a locally exact shortest path on polyhedral surfaces. *CAD Computer Aided Design 39* (2007), 1081–1090. 6

[ZMT07] ZHANG E., MISCHAIKOW K., TURK G.: Vector field design on surfaces. *ACM Transactions on Graphics 25* (2007), 1294–1326. 3

[ZS99] ZORIN D., SCHRÖDER P.: *Subdivision for Modeling and Animation*. SIGGRAPH '99 Course Notes. 1999. 2

[ZSS96] ZORIN D., SCHRÖDER P., SWELDENS W.: Interpolating subdivision for meshes with arbitrary topology. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996). 2