Discrete Differential-Geometry Operators in *n*D

Mathieu Desbrun[†‡] Mark Meyer[†] Peter Schröder[†] Alan H. Barr[†]

†Caltech ‡USC

July 22, 2000

Abstract

This paper provides a consistent set of flexible tools to approximate important geometric attributes, including normal vectors and curvatures, on arbitrary 2D and 3D meshes embedded in *n* dimensions. We present a consistent derivation of these first and second order differential properties using Voronoi cells and the mixed Finite-Element/Finite-Volume method. The new operators are closely related to the continuous case, guaranteeing an appropriate extension from the continuous to the discrete setting: they respect the intrinsic properties of the continuous differential operators. We show that these estimates are optimal in accuracy under mild smoothness conditions, and demonstrate their numerical quality.

We finally present applications of these operators, such as mesh smoothing and enhancement, quality checking, and denoising of arbitrary vectors fields, such as flow fields or tensor images.

1 Introduction

Despite extensive use of triangle meshes in Computer Graphics, there is no consensus on the most appropriate way to estimate simple geometric attributes such as normal vectors and curvatures on discrete surfaces. Many surface-oriented applications require an approximation of the first and second order properties with as much accuracy as possible. This could be done by polynomial reconstruction and analytical evaluation, but this often introduces overshooting or unexpected surface behavior between sample points. The triangle mesh is thus often the only "reliable" approximation of the continuous surface at hand. Unfortunately, since meshes are piecewise linear surfaces, the notion of continuous normal vectors or curvatures is non trivial.

It is fundamental to guarantee accuracy in the treatment of discrete surfaces in many applications. For example, robust curvature estimates are important in the context of mesh

simplification to guarantee optimal triangulations [HG99]. Even if the quadric error defined in [GH97] measures the Gaussian curvature on an infinitely subdivided mesh, the approximation becomes rapidly unreliable for sparse sampling. In surface modeling, a number of other techniques are designed to create very smooth surfaces from coarse meshes, and use discrete curvature approximations to measure the quality of the current approximation (for example, see [MS92]). Accurate curvature normals are also essential to the problem of surface denoising [DMSB99, GSS99] where good estimates of mean curvatures and normals are the key to undistorted smoothing. More generally, discrete operators satisfying appropriate discrete versions of continuous properties would guarantee reliable numerical behavior.

1.1 Previous work

Many expressions of different surface properties have been designed. For instance, we often see the normal vector at a vertex defined as a (sometimes weighted) average of the adjacent faces of a mesh. Thürmer and Würthrich [TW98] use the incident angle of each face at a vertex as the weights, since they claim the normal vector should only be defined very locally, independent of the shape or length of the adjacent faces. However, this normal remains consistent only if the faces are subdivided linearly, introducing vertices which are not on a smooth surface. Max [Max99] derived weights by assuming that the surface locally approximates a sphere. These weights are therefore exact if the object is a (even irregular) tesselation of a sphere. However, it is unclear how this approximation behaves on more complex meshes, since no error bounds are defined. Additionally, many meshes have local sampling adapted to local flatness, contradicting the main property of this approach.

Taubin proposed a more complete derivation, leading to a discrete approximation of the curvature tensors for polyhedral surfaces [Tau95]. Similarly, Hamann [Ham93] proposed a simple way of determining the principle curvatures and their associated directions by a least-squares paraboloid fitting of the adjacent vertices, though the difficult task of selecting an appropriate tangent plane was left to the user. Our paper is closely related to these works since we also derive all first and second order properties for triangulated surfaces. However, many of the previous approaches do not preserve important differential geometry properties on C^0 surfaces such as polyhedral meshes. In order to preserve such quantities, we have also followed a path similar to that of Polthier [PP93, PS98]. He and his coauthors proposed simple expressions for the total curvature, as well as the Dirichlet energy for triangle meshes, and derived discrete methods to compute minimal surfaces or geodesics. Our work offers a unified derivation that ensures accuracy and tight error bounds, leading to simple formulæ that are straightforward to implement.

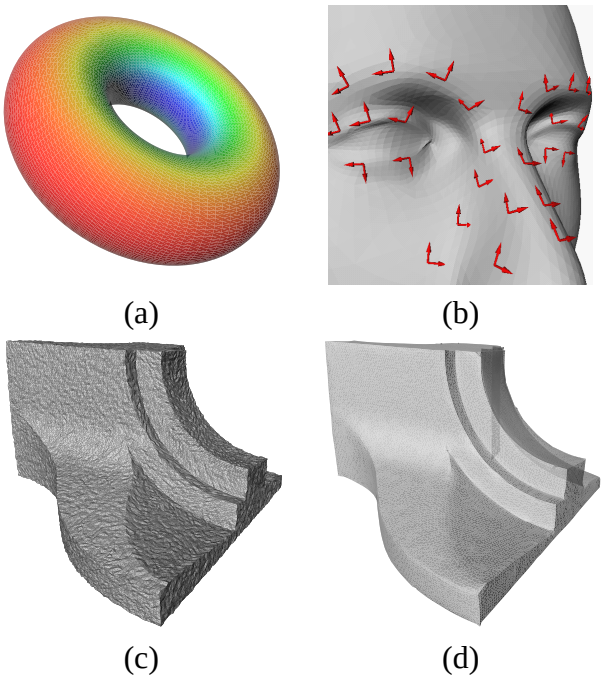


Figure 1: *Some applications of our discrete operators: (a) mean curvature plot for a discrete surface, (b) principal curvature directions on a triangle mesh, (c-d) automatic feature-preserving denoising of a noisy mesh using anisotropic smoothing.*

Contributions

In this paper we define and derive the first and second order differential attributes (normal vector $\mathbf{n}$, mean curvature $\overline{\kappa}$, Gaussian curvature κ_G, principal curvatures κ_1 and κ_2, and principal directions $\mathbf{e}_1$ and $\mathbf{e}_2$) for piecewise linear surfaces such as arbitrary triangle meshes. Details of why a local spatial average of these attributes over the immediate 1-ring neighborhood is a good choice to extend the continuous definition to the discrete setting is given in Section 2. We then present a formal derivation of these quantities for triangle meshes using the mixed Finite-Element/Finite-Volume paradigm in Sections 3, 4 and 5. The relevance of our approach is demonstrated by showing the optimality of our operators under mild smoothness conditions. We demonstrate the accuracy and the use of these operators in different applications, including the smoothing and enhancement of meshes in Section 6. In Section 7, we generalize these operators to any 2-manifold or 3-manifold in an arbitrary dimension embedding space, offering tools for smoothing vector fields and volume data. Conclusions and perspectives are given in Section 8.

2 Defining Discrete Operators

In this section, we describe a general approach to define a number of useful differential quantities associated with a surface represented by a discrete triangular mesh. We begin with a review of several important quantities from differential geometry. This is followed by a technique for extending these quantities to the discrete domain using spatial averaging. Concluding this section is a general framework, used in the remaining sections, for deriving first and second order operators at the vertices of a mesh.

2.1 Notions from Differential Geometry

Let S be a surface (2-manifold) embedded in $\mathbb{R}^3$, described by an arbitrary parameterization of 2 variables. For each point on the surface S, we can locally approximate the surface by its tangent plane, orthogonal to the *normal vector* $\mathbf{n}$. Local bending of the surface is measured by *curvatures*. For every unit direction $\mathbf{e}_\theta$ in the tangent plane, the normal curvature $\kappa^N(\theta)$ is defined as the curvature of the curve that belongs to both the surface itself and a perpendicular plane containing both $\mathbf{n}$ and $\mathbf{e}_\theta$. The two *principal curvatures* κ_1 and κ_2 of the surface S, with their associated orthogonal directions $\mathbf{e}_1$ and $\mathbf{e}_2$ are the extremum values of all the normal curvatures (see Figure 2(a)). We have the property: $\kappa^N(\theta) = \kappa_1 cos^2(\theta) + \kappa_2 sin^2(\theta)$. The *mean curvature* $\overline{\kappa}$ is then defined as the average of the normal curvatures:

$$\overline{\kappa} = \frac{1}{2\pi}\int_0^{2\pi} \kappa^N(\theta) d\theta. \tag{1}$$

Using the two previous equations, calculus leads to the well-known definition: $\overline{\kappa} = (\kappa_1 + \kappa_2)/2$. The *Gaussian curvature* κ_G is defined as the product of the two principle curvatures: $\kappa_G = \kappa_1 \kappa_2$.

These latter two curvatures represent important local properties of a surface. Lagrange noticed that $\overline{\kappa} = 0$ is the Euler-Lagrange equation for surface area minimization. This implies a direct relation between surface area minimization and mean curvature flow, which gave rise to a considerable body of literature on minimal surfaces:

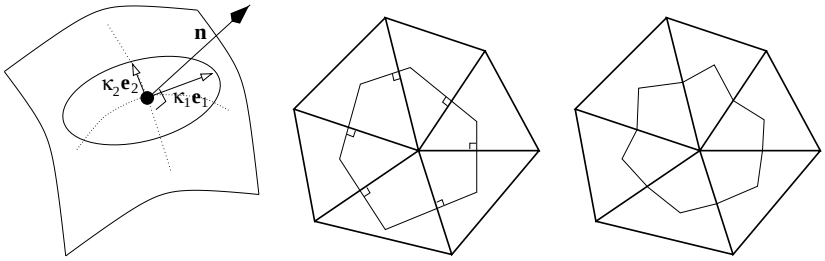
$$2\overline{\kappa}\,\mathbf{n} = \lim_{diam(A)\to 0} \frac{\nabla A}{A}$$

where A is a infinitesimal area around a point P on the surface, $diam(A)$ its diameter, and ∇ is the gradient with respect to P. We will make extensive use of the mean curvature normal $\overline{\kappa}\,\mathbf{n}$. Therefore, we will denote by $\mathbf{K}$ the operator that maps a point P on the surface to the vector $\mathbf{K}(P) = 2\overline{\kappa}_P\,\mathbf{n}_P$. $\mathbf{K}$ is the Laplace-Beltrami operator for the surface S. Gaussian

curvature can also be expressed as a limit:

$$\kappa_G = \lim_{diam(A)\to 0} \frac{A^G}{A} \tag{2}$$

where A^G is the area of the Gauss map (also called spherical image) associated to the infinitesimal surface A. The above definitions, as well as many more details can be found in various sources [Gra98, DHKW92].



(a) (b) (c)

Figure 2: *Local regions: (a) an infinitesimal neighborhood on a continuous surface patch; (b) a finite-volume region on a triangulated surface using Voronoi cells, or (c) Barycentric cells.*

2.2 Discrete Properties as Spatial Averages

A mesh can be considered either as a limit of a family of smooth surfaces, or as a linear (yet assumedly "good") approximation of an arbitrary surface. However, most of the continuous definitions described above have to be reformulated for C^0 surfaces. We choose to define properties (geometric quantities) of the surface at a vertex as *spatial averages* around this vertex. If these averages are made consistently, a property at a given vertex will converge to the pointwise definition as the local sampling increases. Thus, by using these spatial averages, we extend the definition of curvature or normal vector from the continuous case to discrete meshes. Moreover, this definition is appropriate when, for example, geometric flows must be integrated over time on a mesh: a vertex will be updated according to the average behavior of the surface around it. Therefore, the piecewise linear result will be a correct approximation of the smoothed surface if the initial triangle mesh was a good approximation of the initial surface. Since we make no assumption on the smoothness of the surface, we will restrict the average to be within the immediately neighboring triangles, often referred as the 1-ring neighborhood. As a consequence, for example the average Gaussian curvature at a vertex P will be defined in its discrete form $\widehat{\kappa}_G$ as:

$$\widehat{\kappa}_G = \frac{1}{A} \iint_A \kappa_G \; dA$$

for a given area A properly selected around P. Note however that we will not distinguish between the pointwise continuous and the spatially averaged notation, except if there may be ambiguity.

2.3 General Procedure Overview

The next three sections describe how we derive accurate numerical estimates of the first and second order operators at any vertex on an arbitrary mesh. We first restrict the averaging area to a family of special local surface patches denoted A_M. These regions will be contained within the 1-ring neighborhood of each vertex, with piecewise linear boundaries crossing the mesh edges at their midpoints (Figures 2(b) and (c)). We show that this choice guarantees strong analogies between the continuous and the discrete case. The precise surface patch that optimizes the accuracy of our operators is then found, completing the operator derivation. These steps will be explained in detail for the first operator, the mean curvature normal, $\mathbf{K}$, and a more direct derivation will be used for the Gaussian curvature operator κ_G, the two principal curvature operators κ_1 and κ_2, and the two principal direction operators $\mathbf{e}_1$ and $\mathbf{e}_2$. All these operators take a vertex $\mathbf{x}_i$ and its 1-ring neighborhood as input, and provide an estimate in the form of a simple formula that we will frame for clarity.

3 Discrete Mean Curvature Normal

We now provide a simple and accurate numerical approximation for both the normal vector, and the mean curvature for surface meshes in 3D.

3.1 Derivation of Local Integral using FE/FV

To derive a spatial average of geometric properties, we use a systematic approach which mixes finite elements and finite volumes. Since the triangle mesh is meant to visually represent the surface, we select a linear finite element on each triangle, that is, a linear interpolation between the three vertices corresponding to each triangle. Then, for each vertex, an associated surface patch (so-called finite volume in the Mechanics literature) over which the average will be computed is chosen. Two main types of finite volumes are usually used in practice, see Figure 2(b-c). In each case, their piecewise linear boundaries intersect the edges emanating from the center vertex at their midpoints. As for the point inside each adjacent triangle, we can either use the barycenter or the circumcenter. The surface area formed from using the barycenters is denoted $A_{\text{Barycenter}}$ while the latter surface area is recognized as the local Voronoi cell and denoted A_{Voronoi}. In the general case when this point could be anywhere, we will denote the surface area as A_{M}.

Since the mean curvature normal operator, also known as Laplace-Beltrami operator, is a generalization of the Laplacian from flat spaces to manifolds [DHKW92], we first compute the Laplacian of the surface with respect to the *conformal space* parameters u and v. As in [Dzi91] and [PP93], we use the current surface discretization as being the conformal parameter space, that is, for each triangle of the mesh, the triangle itself defines the local surface metric. With such an induced metric, the Laplace-Beltrami operator simply turns into a Laplacian $\Delta_{u,v}\mathbf{x} = \mathbf{x}_{uu} + \mathbf{x}_{vv}$ [DHKW92]:

$$\iint_{A_M} \mathbf{K}(\mathbf{x}) \; dA = \iint_{A_M} \Delta_{u,v}\mathbf{x} \, du \, dv. \tag{3}$$

Using Gauss's theorem as described in Appendix A, the integral of the Laplacian over a surface going through the midpoint of each 1-ring edge of a triangulated domain can be expressed as a function of the node values and the angles of the triangulation. The integral of the Laplace-Beltrami operator turns into the following simple form:

$$\iint_{A_M} \mathbf{K}(\mathbf{x}) dA = \frac{1}{2} \sum_{j \in N_1(i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})\ (\mathbf{x}_i - \mathbf{x}_j), \tag{4}$$

where α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles sharing the edge $(\mathbf{x}_i, \mathbf{x}_j)$ as depicted in Figure 3(a), and $N_1(i)$ is the set of 1-ring neighbor vertices of vertex i.

Note that this result is similar to results obtained by minimizing the Dirichlet energy over a triangulation [PP93, DCDS97]. More importantly, it is exactly the formula established in [DMSB99] for the gradient of surface area for the entire 1-ring neighborhood. This confirms, in the discrete setting, the area minimization nature of the mean curvature normal as derived by Lagrange. We can therefore express our previous result using the following general formula, valid for *any triangulation*:

$$\iint_{A_M} \mathbf{K}(\mathbf{x}) dA = \nabla A_{\text{1-ring}}. \tag{5}$$

Notice that the formula results in a zero value for any flat triangulation, regardless of the shape or size of the triangles of the locally-flat (zero curvature) mesh: the gradient of the area is zero for any locally flat region.

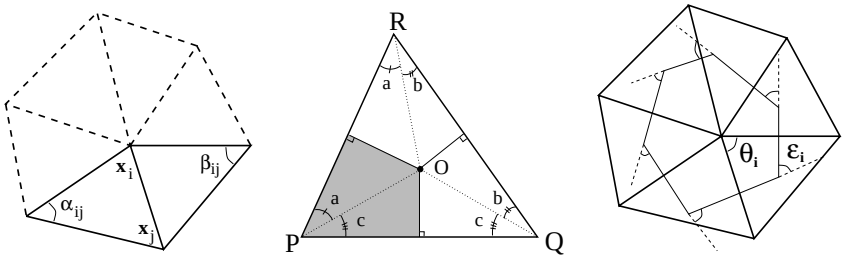
Although we have found an expression for the integral of the mean curvature normal independent of which of the two finite volume discretizations is used, one finite volume region must be chosen in order to provide an accurate estimate of the spatial average. We show next that the Voronoi cells provide provably tight error bounds under reasonable assumptions of smoothness.

3.2 Voronoi Regions for Tight Error Bounds

Given a C^2 surface let us compare the local spatial average of mean curvature with the actual pointwise value. If our surface is tiled by small patches A_i around n sample points $\mathbf{x}_i$, we can define the error E created by local averaging of the mean curvature normal compared to its pointwise value at $\mathbf{x}_i$ as:

$$\begin{aligned} E &= \sum_i \| \iint_{A_i} \mathbf{K}(\mathbf{x}) dA - A_i \mathbf{K}(\mathbf{x}_i)) \|^2 \\ &= \sum_i \| \iint_{A_i} (\mathbf{K}(\mathbf{x}) - \mathbf{K}(\mathbf{x}_i))\, dA \|^2 \\ &\leq \sum_i \iint_{A_i} C_i \|\mathbf{x} - \mathbf{x}_i\|^2\, dA \;\leq C_{max} \sum_i \iint_{A_i} \|\mathbf{x} - \mathbf{x}_i\|^2\, dA, \end{aligned}$$

where C_i is the Lipschitz constant of the Beltrami operator over the smooth surface patch A_i, and C_{max} the maximum of the Lipschitz constants. Therefore, the Voronoi region of each sample point *by definition* minimizes the bound on the error E due to spatial averaging [DFG99], since they contain the closest points to each sample. Furthermore, if we add an extra assumption on the sampling rate with respect to the curvature such that the Lipschitz constants from patch to patch vary slowly with a ratio ε, we can actually guarantee that the Voronoi cell borders are less than ε away from the optimal borders. As this still holds in the limit for a triangle mesh, we use the vertices of the mesh as sample points, and pick the Voronoi cells of the vertices as associated finite-volume regions. This will guarantee optimized numerical estimates and, as we will see, computing these voronoi cells requires few extra computations.



(a) (b) (c)

Figure 3: *(a) 1-ring neighbors and angles opposite to an edge; (b) Voronoi region on a non-obtuse triangle; (c) External angles of a Voronoi region.*

3.3 Voronoi Region Area

Suppose a non-obtuse triangle P,Q,R with circumcenter O, as depicted in Figure 3(b). Using the properties of perpendicular bisectors, we find : $a+b+c = \pi/2$, and therefore,

$a = \pi/2 - \angle Q$ and $c = \pi/2 - \angle R$. The Voronoi area for point P lies within this triangle if the triangle is non-obtuse, and is thus: $\frac{1}{8}(|PR|^2 cot\angle Q + |PQ|^2 cot\angle R)$. Summing these areas for the whole 1-ring neighborhood, we write the non-obtuse Voronoi area for a vertex $\mathbf{x}_i$ as a function of the neighbors $\mathbf{x}_j$:

$$A_{\text{Voronoi}} = \frac{1}{8} \sum_{j \in N_1(i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})\ \|\mathbf{x}_i - \mathbf{x}_j\|^2.$$

Since the cotangent terms were already computed for Eq. (4), the voronoi area can be computed very efficiently. However, if there is an obtuse triangle among the 1-ring neighborhood or among the triangles edge-adjacent to the 1-ring triangles, the Voronoi region either extends beyond the 1-ring, or is truncated compared to our area computation. In either case our derived formula no longer stands.

3.4 Extension to Arbitrary Meshes

The previous expression for the Voronoi finite-volume area does not hold in the presence of obtuse angles. However, Equation (5) holds even for obtuse 1-ring neighborhoods: the only assumption used is that the finite-volume region goes through the midpoint of the edges. It is thus *still valid even in obtuse triangulations*. Therefore, we could simply divide the integral evaluation by the barycenter finite-volume area in lieu of the Voronoi area for vertices near obtuse angles to determine the spatial average value. We opted for using a slightly more subtle area, to guarantee a perfect tiling of our surface, and therefore, an optimized accuracy.

We define a new surface area for each vertex $\mathbf{x}$, denoted A_{Mixed}, which can be easily computed as follows:

```
A_Mixed = 0
For each triangle T from the 1-ring neighborhood of x
   If T is non-obtuse,   // Voronoi safe
      // Add Voronoi formula (see Section 3.3)
      A_Mixed += Voronoi region of x in T
   Else                  // Voronoi inappropriate
      // Add area(T)/3
      A_Mixed += Barycenter region of x in T
```

Figure 4: *Pseudo-code for region A_{Mixed} on an arbitrary mesh*

Note that the derivation for the integral of the mean curvature normal is still valid for this mixed area since the boundaries of the area remain inside the 1-ring neighborhood and go through the midpoint of each edge. Moreover, these mixed areas tile the surface without overlapping. This new cell definition is thus equivalent to a local adjustment of the diagonal

mass matrix in a finite element framework in order to ensure a correct evaluation. The error bounds are not as tight when local angles are more than $\pi/2$, and therefore, numerical experiments are expected to be worse in areas with obtuse triangles. Finally, notice that other mixed areas can be defined. For instance, if our operator needs to be defined on a mesh moving over time, the current mixed area may create numerical difficulties since the mixed area can vary slightly when an angle just exceeds $\pi/2$. In such a case, we recommend defining the mixed area over obtuse triangles using neither the barycenter, nor the circumcenter, but the midpoint of the edge opposite the obtuse angle. This area does not suffer from any discontinuities around $\pi/2$, and is therefore more appropriate in some cases.

3.5 Discrete Mean Curvature Normal Operator

Now that the mixed area is defined, we can express the mean curvature normal operator $\mathbf{K}$ defined in Section 2.1 using the following expression:

Mean Curvature Normal Operator

$$\boxed{\mathbf{K}(\mathbf{x}_i) = \frac{1}{2A_{\text{Mixed}}} \sum_{j \in N_1(i)} (cot\ \alpha_j + cot\ \beta_j)\ (\mathbf{x}_j - \mathbf{x}_i)} \tag{6}$$

From this expression, we can easily compute the mean curvature value $\overline{\kappa}$ by taking half of the magnitude of this last expression. As for the normal vector, we can just normalize the resulting vector $\mathbf{K}(\mathbf{x}_i)$. In the special rare case where the mean curvature is zero (flat plane or saddle point), we simply average the 1-ring face normal vectors to evaluate $\mathbf{n}$ appropriately.

It is interesting to notice at this point that in the (extremely rare) case of only obtuse triangles, this new operator is very similar to the definition of the mean curvature normal by Desbrun *et al.* [DMSB99], since $A_{\text{Barycenter}}$ is a third of the whole 1-ring area $A_{\text{1-ring}}$ used in their derivation. We will give numerical results in Section 6.1 showing the improved quality of our new estimate.

4 Discrete Gaussian Curvature

In this section, the Gaussian curvature κ_G for bivariate (2D) meshes in 3D is studied. We will demonstrate that a similar derivation can be obtained easily.

4.1 Expression of the Local Integral of κ_G

Similar to what was done for the mean curvature normal operator, we first need to find an exact value of the integral of the Gaussian curvature κ_G over a finite-volume region on a piecewise linear surface. From Eq. (2), we could compute the integral over an area A_{M} as the associated spherical image area (also called the Gauss map). Instead, we prefer using the *Gauss-Bonnet theorem* [DHKW92, Gra98] that proposes a very simple equality, valid over any surface patch. Applied to our local finite-volume regions, and since the extra integral of geodesic curvatures is zero on the piece-wise linear boundaries of our regions, the Gauss-Bonnet theorem simply states:

$$\iint_{A_{\text{M}}} \kappa_G \, dA = 2\pi - \sum_j \varepsilon_j .$$

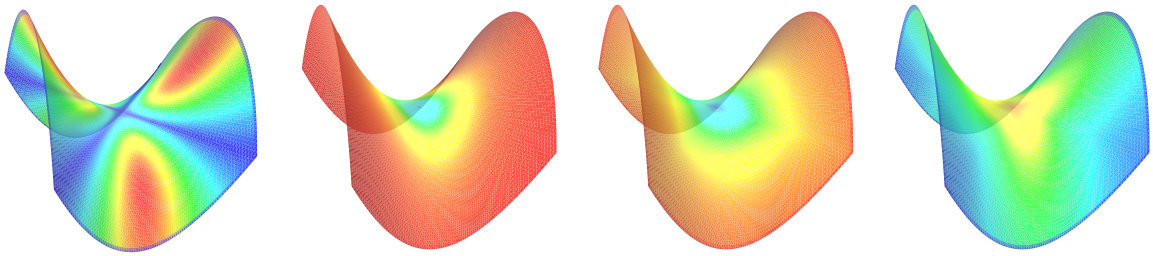
The ε_j are the external angles of the boundary, as indicated in Figure 3(c). If we use this expression on a Voronoi region, the external angles are zero across each edge (since the boundary stays perpendicular to the edge), and the external angle at a circumcenter is simply equal to θ_j, the angle of the triangle at the vertex P. Therefore, the integral of the Gaussian curvature (also called total curvature) for non-obtuse triangulations is: $2\pi - \sum_j \theta_j$. This result actually extends to the barycenter region and the mixed region using the same kind of simple geometric consideration. This result has already been proven by Polthier and Schmies [PS98], by considering the area of the Gauss map for a vertex on a polyhedral surface. Therefore, analogous to Eq. (5), we can now write for the 1-ring neighborhood of a vertex $\mathbf{x}_i$:

$$\iint_{A_{\text{M}}} \kappa_G dA = 2\pi - \sum_{j=1}^{\#f} \theta_j$$

where θ_j is the angle of the j-th face at the vertex $\mathbf{x}_i$, and $\#f$ denotes the number of faces around this vertex. Note again that this formula holds for any surface patch A_{M} within the 1-ring neighborhood whose boundary crosses the edges at their midpoint.

4.2 Discrete Gaussian Curvature Operator

To estimate the local spatial average of the Gaussian curvature, we use the same arguments as in 3.2 to claim that the Voronoi cell of each vertex is an appropriate local region to use for guaranteed error bounds. In practice, we use the mixed area A_{Mixed} to account for obtuse triangulations. Since the mixed area cells tile the whole surface without any overlap, we will satisfy the Gauss-Bonnet (continuous) theorem: the integral of the discrete Gaussian curvature over an entire sphere for example will be equal to 4π *whatever the discretization*



(a) (b) (c) (d)

Figure 5: *Curvature plots of a triangulated saddle using pseudo-colors: (a) Mean, (b) Gaussian, (c) Minimum, (d) Maximum.*

used since the sphere is a closed object of genus one. This result ensures a robust numerical behavior of our discrete operator. Our Gaussian curvature discrete operator can thus be expressed as:

Gaussian Curvature Operator

$$\kappa_G(\mathbf{x}_i) = (2\pi - \sum_{j=1}^{\#f} \theta_j)/A_{\text{Mixed}} \tag{7}$$

Notice that this operator will return zero for any flat surface, as well as any roof-shaped 1-ring neighborhood, guaranteeing a satisfactory behavior for trivial cases. We postpone the numerical tests until Section 6.1.

5 Discrete Principal Curvatures

We now wish to robustly determine the two principal curvatures, along with their associated directions. Since the previous derivations give estimates of both Gaussian and mean curvature, the only additional information that must be sought are the principal directions since the principal curvatures are, as we are about to see, easy to determine.

5.1 Principal Curvatures

We have seen in Section 2.1 that the mean and Gaussian curvatures are easy to express in terms of the two principal curvatures κ_1 and κ_2. Therefore, since both $\overline{\kappa}$ and κ_G have been derived for triangulated surfaces, we can define the discrete principal curvatures as:

Principal Curvature Operators

$$\boxed{\kappa_1(\mathbf{x}_i) = \overline{\kappa}(\mathbf{x}_i) + \sqrt{\Delta(\mathbf{x}_i)}} \tag{8}$$

$$\boxed{\kappa_2(\mathbf{x}_i) = \overline{\kappa}(\mathbf{x}_i) - \sqrt{\Delta(\mathbf{x}_i)}} \tag{9}$$

$$\text{with:} \quad \Delta(\mathbf{x}_i) = \overline{\kappa}^2(\mathbf{x}_i) - \kappa_G(\mathbf{x}_i) \text{ and } \overline{\kappa}(\mathbf{x}_i) = \frac{1}{2}\|\mathbf{K}(\mathbf{x}_i)\|.$$

Unlike the continuous case where Δ is always positive, we must make sure that $\overline{\kappa}^2$ is always larger than κ_G to avoid any numerical problems, and threshold Δ to zero if it is not the case (extremely rare occurrence).

5.2 Mean Curvature as a Quadrature

We must find the principal axes at a vertex to complete the analogy between smooth and polyhedral surfaces. Starting from our previous expression for the mean curvature normal, we show that more information about local curvatures is already available:

$$\begin{aligned}
\overline{\kappa}_i &= \frac{1}{4A_{\text{Mixed}}} \sum_{j \in N_1(i)} (cot\ \alpha_j + cot\ \beta_j)\ (\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{n} \\
&= \frac{1}{4A_{\text{Mixed}}} \sum_{j \in N_1(i)} (cot\ \alpha_j + cot\ \beta_j)\ \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\|\mathbf{x}_i - \mathbf{x}_j\|^2} (\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{n} \\
&= \frac{1}{A_{\text{Mixed}}} \sum_{j \in N_1(i)} \left[\frac{1}{8} (cot\ \alpha_j + cot\ \beta_j)\ \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right] \kappa^N_{i,j},
\end{aligned} \tag{10}$$

where we define:

$$\kappa^N_{i,j} = 2 \frac{(\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{n}}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}.$$

This $\kappa^N_{i,j}$ turns out to be an estimate of the normal curvature along the edge (the inverse of the radius of the osculating circle) $\mathbf{x}_i\mathbf{x}_j$ as defined in Section 2.1. The radius R of the osculating circle going through the vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ is easily found using the mean curvature normal estimate as illustrated in Figure 9(a). We must have a right angle at the neighbor vertex $\mathbf{x}_j$, yielding:

$$(\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_i - \mathbf{x}_j - 2R\,\mathbf{n}) = 0.$$

This implies $R = \|\mathbf{x}_i - \mathbf{x}_j\|^2 / (2\,(\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{n})$, proving that $\kappa^N_{i,j}$ is a normal curvature estimate in the direction of edge $\mathbf{x}_i\mathbf{x}_j$. This expression was already mentioned in the context of curvature approximation in [MS92].

Therefore, Eq. (10) can be interpreted as a quadrature of the integral from Eq. (1), with weights w_{ij}:

$$\overline{\kappa}_i = \sum_{j \in N_1(i)} w_{ij} \ \kappa^N_{i,j},$$

where the $w_{ij} = \frac{1}{A_{\text{Mixed}}} \left[\frac{1}{8} (cot\ \alpha_j + cot\ \beta_j) \ \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right]$ sum to one for each i on a non-obtuse triangulation.

5.3 Least-Square Fitting for Directions

To mimic the continuous case, we must find the two principal curvatures such that their associated directions are orthogonal. Since the mean curvature obtained from our derivation can be seen as a Gaussian quadrature using each edge as a sample, we decide to use these samples to find the best fitting ellipse, in order to fully determine the curvature tensor. In practice, we select the symmetric curvature tensor B as being defined by three unknowns a, b, and c:

$$B = \begin{pmatrix} a & b \\ b & c \end{pmatrix}.$$

This tensor will provide the normal curvature in any direction in the tangent plane. Therefore, when we use the direction of the edges of the 1-ring neighborhood, we should find:

$$\mathbf{d}^T_{i,j} \, B \, \mathbf{d}_{i,j} \ = \ \kappa^N_{i,j},$$

where $\mathbf{d}_{i,j}$ is the unit direction *in the tangent plane* of the edge $\mathbf{x}_i\mathbf{x}_j$. Since we know the normal vector $\mathbf{n}$ to the tangent plane, this direction is calculated using a simple projection onto the tangent plane:

$$\mathbf{d}_{i,j} = (\mathbf{x}_j - \mathbf{x}_i) - [(\mathbf{x}_j - \mathbf{x}_i) \cdot \mathbf{n}] \, \mathbf{n}.$$

A conventional least-square approximation can be obtained by minimizing the error E:

$$E(a,b,c) = \sum_j w_j \mid \mathbf{d}^T_{i,j} \, B \, \mathbf{d}_{i,j} - \kappa^N_{i,j} \mid.$$

Adding the two constraints $a + b = 2\overline{\kappa}$ and $ac - b^2 = \kappa_G$, to ensure coherent results, turns the minimization problem into a third degree polynomial root-finding problem. Once the three coefficients of the matrix B are found, we find the two principal axes $\mathbf{e}_1$ and $\mathbf{e}_2$ as the two (orthogonal) eigenvectors of B. In practice, all our experiments have demonstrated that the non-linear constraint on the determinant is not really necessary. An example of these principal directions can be found on Figure 1(b).

6 Results & Applications

With robust estimates at our disposal, we propose some applications such as quality checking for surface design and tools for smoothing and enhancement of meshes.

6.1 Numerical Quality of our Operators

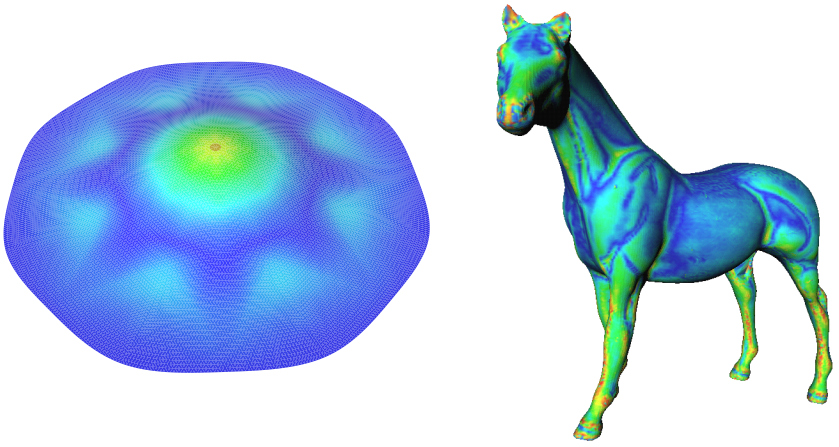
We performed a number of tests to confirm the validity of our approach. A first round was made on discrete meshes approximating simple surfaces like spheres, or hyperboloids, where the curvatures are known analytically. To be able to accurately gauge how well our operators perform, we also used special meshes defined as height fields over a flat regular grid: by proceeding this way, Finite Difference (FD) approximations can also be computed and tested against our results. The array below demonstrates the accuracy of our operators by giving the mean percent error for analytically defined surfaces compared to the exact, known curvatures:

$\%\overline{m}$	FD $\overline{\kappa}$	[DMSB99] $\overline{\kappa}$	our $\overline{\kappa}$	FD κ_G	our κ_G
Sphere	0.20	0.17	0.16	0.4	1.2
Paraboloid	0.0055	0.0038	0.0038	0.01	0.02
Torus	-	0.047	0.036	-	0.05

The dashes "-" indicate that the FD tests cannot be performed since the triangulation is arbitrary. We must indicate that the angles θ_j needed for the Gaussian curvature were computed using the C function `atan2`, much preferable over an `arcos` or an `arcsin` that would significantly deteriorate the precision of the results.

6.2 Geometric Quality of Meshes

Producing high quality meshes is not an easy task. Checking if a given mesh is appropriately smooth requires a long inspection with directional or point light sources to detect any visually unpleasant discontinuities on the surface. Curvature plots (see Figure 5), using false color to texture the mesh according to the different curvatures, can immediately show problems or potential problems since they will reveal the variation of curvatures in an obvious way. Figure 6 demonstrates that even if a surface (obtained by a subdivision scheme) looks very smooth, a look at the mean curvature map reveals flaws such as discontinuities in the variation of curvature across the surface. Conversely, curvature plots can reveal unsuspected details on existing scanned meshes, like the veins on the horse.



(a) (b)

Figure 6: *Mean curvature plots revealing surface details for: (a) a Loop surface from an 8-neighbor ring, (b) a horse mesh.*

6.3 Denoising and Enhancement of a Mesh

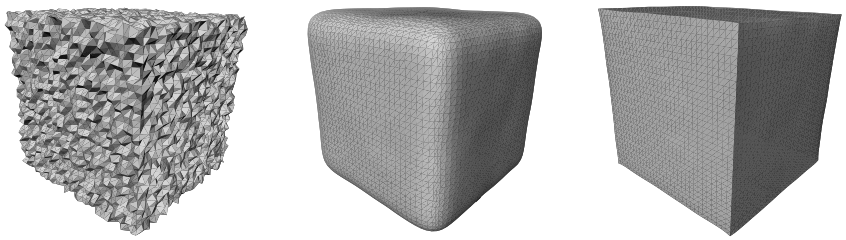
If the quality of a mesh is not sufficient (due to noise resulting from inaccurate scans for instance), denoising and enhancements can be performed using our discrete operators.

6.3.1 Isotropic Shape Smoothing

Just like Laplacian filtering in image processing, a mean curvature flow will disperse the noise of a smooth mesh appropriately by minimizing the surface area as reported in [DMSB99]. We implemented this implicit fairing technique using our new operators with success. However, since our mesh can represent a surface with sharp edges, we sometimes experienced a dilemma: how can one get rid of the noise by smoothing the surface, while preserving sharp edges to keep the underlying geometry intact? We would like to smooth a noisy cube, for example, without turning it into a cushion-like shape (Figure 7(a) and (b)). A possible solution is to manually spray-paint the desired value of smoothing on the vertices [DMSB99], making the preservation of sharp edges possible while suppressing noise. But it is a rather time-consuming task for big meshes, and it will leave ragged edges on the vertices forced to a low smoothing amount.

6.3.2 Enhancement of Meshes

We would like to automate the previous process, providing a way to smooth meshes while keeping clear features (like sharp edges) intact. This relates closely to the specific problem of image denoising, where clear features like object boundaries should be kept, while



(a) (b) (c)

Figure 7: *Cube: (a) Original, noisy mesh (±3% uniform noise added along the normal direction). (b) Isotropic smoothing. (c) Anisotropic smoothing defined in Section 6.3.3.*

noisy, yet homogeneous regions should be smoothed. Different forms of anisotropic diffusion have shown good results for this problem in image processing [PM90] and in flow visualization [PR99]. The underlying idea is to still diffuse the noise, but with an adaptive conductance over the image in order to preserve edges. We experimented with a simple technique to achieve similar results on meshes. Additionally, an enhancement procedure to help straighten up edges has been designed.

An isotropic implicit curvature flow on regions uniformly noisy is desired, while special treatment must be applied for obvious edges and corners to prevent them from being smoothed away. In our previous work [DMSB00], we proposed a weighted mean curvature smoothing, where the weights are computed using the first fundamental form to preserve height field discontinuities. However, even if such an approach is appropriate for height fields, it does not capture enough information to perform adequately on a general mesh. The second fundamental form, i.e., local curvature, provides more information on the local variations of the surface, and therefore, will be more accurate for the weighting.

6.3.3 An Anisotropic Smoothing Technique

Most of the meshes issued from real object scans contain corners and ridges, which will be lost if isotropic denoising is used. Therefore, if these sharp edges are necessary features of a noisy mesh, the noise should be only directionally diffused in order to keep the characteristics intact. Presence of such features can be determined using the second-order properties of the surface. Indeed, in the case of an edge between two faces of a cube mesh, the minimum curvature is zero along the edge, while the maximum curvature is perpendicular to this edge. An immediate idea is thus to perform a weighted mean curvature flow that penalizes vertices that have a large ratio between their two principal curvatures: this way, clear features like sharp edges will remain present while noise, more symmetric by nature, will be greatly reduced.

We define the smoothing weight at a vertex $\mathbf{x}_i$ as being:

$$w_i = \begin{cases} 1 & \text{if } |\kappa_1| \leq T \text{ and } |\kappa_2| \leq T \\ 0 & \text{if } |\kappa_1| > T \text{ and } |\kappa_2| > T \text{ and } \kappa_1\kappa_2 > 0 \\ \kappa_1/\overline{\kappa} & \text{if } |\kappa_1| = \min(|\kappa_1|, |\kappa_2|, |\overline{\kappa}|) \\ \kappa_2/\overline{\kappa} & \text{if } |\kappa_2| = \min(|\kappa_1|, |\kappa_2|, |\overline{\kappa}|) \\ 1 & \text{if } |\overline{\kappa}| = \min(|\kappa_1|, |\kappa_2|, |\overline{\kappa}|) \end{cases} .$$

The parameter T is a user defined value determining edges. The general smoothing flow is then: $\partial \mathbf{x}_i / \partial t = -w_i \, \overline{\kappa}_i \, \mathbf{n}_i$. As we can see, corners will not move, while uniformly noisy regions will be smoothed isotropically. For edges, we smooth with a speed proportional to the minimum curvature, to be assured not to smooth ridges. The caveat is that this smoothing is not well-posed anymore: we try to enhance edges, and this is by definition a very unstable process. Pre-mollification techniques have been reported successful in [PR99], and could be used in our case. However, we had good results by just thresholding the weights w_i to be no less than -0.1 to avoid strong inverse diffusion, and using implicit fairing to integrate the flow. As Figure 7 demonstrates, a noisy cube can be smoothed and enhanced into an almost perfect cube using our technique. For more complicated objects (see Figure 1(c-d)), a pass of curve smoothing (also using implicit curvature flow) has been added to help straighten the edges.

7 Discrete Operators in *n*D

Up to this point, we defined and used our geometric operators for bivariate (2D) surfaces embedded in 3D. We propose in this section to generalize our tools for 2D surfaces to any embedding space dimensionality, as well as extending the formulæ to 3-manifolds (volumes) in n dimensions. This will allow us to apply the same kind of smoothing techniques on datasets such as vector fields, tensor images, or volume data.

7.1 Operators for 2-Manifolds in *n*D

We here extend our operators for 2-parameter surfaces embedded in an arbitrary dimensional space, such as color images (2D surface in 5D), or bivariate vector field (2D surface in 4D).

Beltrami Operator

As we have seen in Sections 2.1 and 3.1, the Beltrami operator is in the direction of surface area minimization. In order to extend this operator to higher dimensional space, we must first derive the expression for a surface area in nD. The area of a triangle formed by two

vectors $\mathbf{u}$ and $\mathbf{v}$ in 3D is $2A = ||\mathbf{u} \times \mathbf{v}||$. Being proportional to the sine of the angle between vectors, we can also express it as:

$$\begin{aligned} A &= \frac{1}{2}||\mathbf{u}||||\mathbf{v}||sin(\mathbf{u},\mathbf{v}) = \frac{1}{2}||\mathbf{u}||||\mathbf{v}||\sqrt{1-cos^2(\mathbf{u},\mathbf{v})} \\ &= \frac{1}{2}\sqrt{||\mathbf{u}||^2||\mathbf{v}||^2-(\mathbf{u}\cdot\mathbf{v})^2}. \end{aligned} \tag{11}$$

This latter expression is now valid in nD, and is particularly easy to evaluate in any dimension.

We can now derive the gradient of the 1-ring area with respect to the central vertex to find the analog of Eq. (4) in nD. We detail this proof in Appendix B, but the result is very simple: the previous cotangent formula is still valid in nD if we define the cotangent between two vectors $\mathbf{u}$ and $\mathbf{v}$ as:

$$cot(\mathbf{a},\mathbf{b}) = \frac{cos(\mathbf{a},\mathbf{b})}{sin(\mathbf{a},\mathbf{b})} = \frac{\mathbf{a}\cdot\mathbf{b}}{\sqrt{||\mathbf{a}||^2||\mathbf{b}||^2-(\mathbf{a}\cdot\mathbf{b})^2}}.$$

With this definition, the implementation in nD space is straightforward and efficient, as dot products require little computation.

Gaussian Curvature Operator

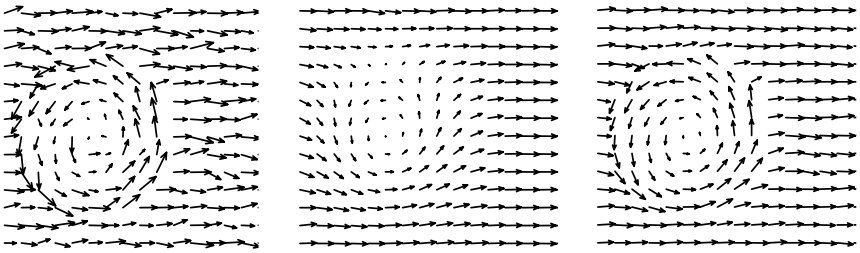
The expression of the Gaussian curvature operator Eq. (7) still holds in nD. Indeed, the Gaussian curvature is an intrinsic attribute of a 2-manifold, and does not depend on the embedding.

7.2 Operators for 3-Manifolds in nD

We also extend the previous operators, valid on triangulated surfaces, to tetrahedralized volumes which are 3-parameter volumes in an embedding space of arbitrary dimension. This will be used for example for any MRI volume data (intensity, vector field or even tensor fields). For these 3-manifolds, we can compute the gradient of the 1-ring volume this time to extend the Beltrami operator. Once again, the cotangent formula turns out to be still valid, but this time for the dihedral angles of the tetrahedrons. Appendix C details the derivation to prove this result. This Beltrami operator can still be used to denoise volume data as it minimizes volume just like we denoised meshes through a surface area minimization.

7.3 Denoising of Arbitrary Fields

The extension of our geometric operator to higher dimensional embedding spaces allows us to use the same smoothing technology used on meshes for vector fields or tensor images.



(a) (b) (c)

Figure 8: *Vector field denoising: (a) Original, noisy vector field; (b) Smoothed using Beltrami flow; (c) Smoothed using anisotropic weighted flow to automatically preserve the vortex region.*

To prove the validity of our operator, we performed different smoothings on higher dimensional spaces. For instance, Figure 8 demonstrates how our operators can smooth a vector field, with or without preservation of features. Anisotropic smoothing can indeed preserve significant discontinuities such as the boundary between the straight flow and the vortex, just as we preserved edges during mesh smoothing in 3D.

8 Conclusion

A complete set of accurate differential operators for any triangulated surface have been presented. We consistently derived estimates for normal vectors and mean curvatures (Eq. (6)), Gaussian curvatures (Eq. (7)), principal curvatures (Eq. (8) and (9)) , and principal directions (Section 5.3), and numerically showed their quality. Extended versions of our operator for surfaces and volumes in higher dimension embedding spaces have also been provided. Moreover, we described how to use these simple, local operators to denoise arbitrary meshes or vector fields, including preservation and/or enhancement of features. These methods form a family of robust tools to help with processing noisy data, or simply to build a scale space out of a dataset to offer an adaptive description of the data. With little user interaction to select (and direct) the appropriate tools, noisy scanned meshes can be turned into high-quality meshes, vector fields can be smoothed to later segment the general flow, or MRI multi-valued images can be denoised. However, smoothing techniques do not deal well with large amounts of noise. Multiplicative noise, for example, can create large dents in a dataset, that only statistical techniques using local averages of n neighbors can try to deal with [MS96], often without guarantee of success. Yet we believe that, as in image processing, our global framework can give rise to other anisotropic diffusion equations particularly designed for specific noise models.

We have confidence in the adequacy and efficiency of our simple discrete operators in numerous other surface-based applications. The mean curvature normal operator for in-

stance can be easily applied to function values on the surface, and it will define a Laplacian operator for the "natural" metric of the mesh. We are currently exploring other applications of these operators such as reparameterization, geometry based subdivision schemes, and mesh simplification along the lines of [HG99].

References

[Bar89] Alan H. Barr. The Einstein Summation Notation: Introduction and Extensions. In *SIGGRAPH 89 Course notes #30 on Topics in Physically-Based Modeling*, pages J1–J12, 1989.

[DCDS97] Tom Duchamp, Andrew Certain, Tony DeRose, and Werner Stuetzle. Hierarchical computation of PL harmonic embeddings. Technical report, University of Washington, July 1997.

[DFG99] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tesselations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, 1999.

[DHKW92] Ulrich Dierkes, Stefan Hildebrandt, Albrecht Küster, and Ortwi Wohlrab. *Minimal Surfaces (I)*. Springer-Verlag, 1992.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *SIGGRAPH 99 Conference Proceedings*, pages 317–324, 1999.

[DMSB00] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Anisotropic Feature-Preserving Denoising of Height Fields and Images. In *Graphics Interface'2000 Conference Proceedings*, pages 145–152, 2000.

[Dzi91] G. Dziuk. An Algorithm for Evolutionary Surfaces . *Numer. Math.*, 58, 1991.

[GH97] Michael Garland and Paul S. Heckbert. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH 97 Conference Proceedings*, pages 209–216, August 1997.

[Gra98] Alfred Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, 1998.

[GSS99] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution Signal Processing for Meshes. In *SIGGRAPH 99 Conference Proceedings*, pages 325–334, 1999.

[Ham93] Bernd Hamann. Curvature Approximation for Triangulated Surfaces. In G. Farin *et al.*, editor, *Geometric Modelling*, pages 139–153. Springer Verlag, 1993.

[HG99] Paul S. Heckbert and Michael Garland. Optimal Triangulation and Quadric-Based Surface Simplification. *Journal of Computational Geometry: Theory and Applications*, November 1999.

[Max99] Nelson Max. Weights for Computing Vertex Normals from Facet Normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.

[MS92] Henry P. Moreton and Carlo H. Séquin. Functional Minimization for Fair Surface Design. In *SIGGRAPH 92 Conference Proceedings*, pages 167–176, July 1992.

[MS96] R. Malladi and J.A. Sethian. Image Processing: Flows under Min/Max Curvature and Mean Curvature. *Graphical Models and Image Processing*, 58(2):127–141, March 1996.

[PM90] P. Perona and J. Malik. Scale-space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[PR99] T. Preußer and M. Rumpf. Anisotropic Nonlinear Diffusion in Flow Visualization. In *IEEE Visuazliation'99*, pages 323–332, 1999.

[PS98] Konrad Polthier and Markus Schmies. Straightest Geodesics on Polyhedral Surfaces. In H.C. Hege and K. Polthier, editors, *Mathematical Visualization*. Springer Verlag, 1998.

[Tau95] Gabriel Taubin. Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation. In *Proc. 5th Intl. Conf. on Computer Vision (ICCV'95)*, pages 902–907, June 1995.

[TW98] Grit Thürmer and Charles Wüthrich. Computing Vertex Normals from Polygonal Facets. *Journal of Graphics Tools*, 3(1):43–46, 1998.

Appendix

In this appendix, we will make heavy use of Einstein summation notation for conciseness. For an introduction, see [Bar89].

A Laplacian on a triangulated domain

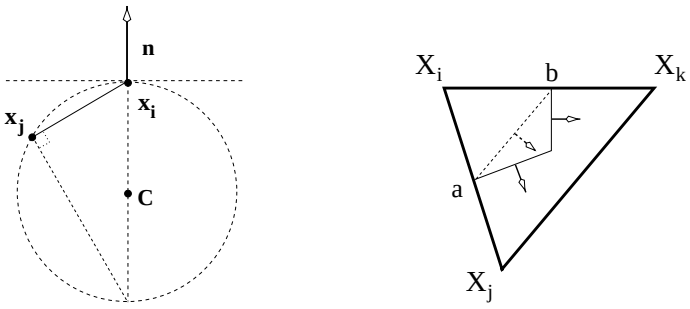
From Gauss's theorem, we can turn the integral of a Laplacian over a region into a line integral over the boundary of the region:

$$\iint_{A_M} \Delta_{u,v}\mathbf{x}\, du\, dv = \int_{\partial A_M} \nabla_{u,v}\mathbf{x}\cdot\mathbf{n}_{u,v}\; dl, \tag{12}$$

where the the subscript u,v indicates that the operator or vector must be with respect to the parameter space.

Since we assumed our surface to be piecewise linear, its gradient $\nabla_{u,v}\mathbf{x}$ is constant over each triangle of the mesh. As a consequence, whatever the type of finite-volume discretization we use, the integral of the normal vector along the border ∂A_M within a triangle will result in the same expression since the border of both regions passes through the edge midpoints as sketched in Figure 9(b). Inside a triangle $T = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$, we can thus write:

$$\int_{\partial A_M \cap T} \nabla_{u,v}\mathbf{x}\cdot\mathbf{n}_{u,v}\, dl = \nabla_{u,v}\mathbf{x}\cdot[\mathbf{a}-\mathbf{b}]^{\perp}_{u,v} = \frac{1}{2}\nabla_{u,v}\mathbf{x}\cdot[\mathbf{x}_j-\mathbf{x}_k]^{\perp}_{u,v}$$



(a) (b)

Figure 9: *(a) Osculating circle for edge $\mathbf{x}_i\mathbf{x}_j$. (b) The integration of the surface gradient dotted with the normal of the region contour does not depend on the finite volume discretization used.*

Since the function $\mathbf{x}$ is linear over any triangle T, using the linear basis functions over the triangle B_l, it follows:

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_i\, B_i(u,v) + \mathbf{x}_j\, B_j(u,v) + \mathbf{x}_k\, B_k(u,v) \\ \nabla_{u,v}\mathbf{x} &= \mathbf{x}_i\, \nabla_{u,v}B_i(u,v) + \mathbf{x}_j\, \nabla_{u,v}B_j(u,v) + \mathbf{x}_k\, \nabla_{u,v}B_k(u,v) \end{aligned}$$

Using the fact that the gradients of the 3 basis functions of any triangle T sum to zero and rearranging terms, the gradient of $\mathbf{x}$ over the triangle can be expressed as $\nabla_{u,v}\mathbf{x} =$

$\frac{1}{2A_T}[(\mathbf{x}_j-\mathbf{x}_i)[\mathbf{x}_i-\mathbf{x}_k]^{\perp}_{u,v}+(\mathbf{x}_k-\mathbf{x}_i)[\mathbf{x}_j-\mathbf{x}_i]^{\perp}_{u,v}]$. The previous integral can then be rewritten as:

$$\begin{aligned}\int_{\partial A\cap T}\nabla_{u,v}\mathbf{x}\cdot\mathbf{n}_{u,v}\,dl &= \frac{1}{4A_T}\left[([\mathbf{x}_i-\mathbf{x}_k]\cdot[\mathbf{x}_j-\mathbf{x}_k])_{u,v}\,(\mathbf{x}_j-\mathbf{x}_i)\right.\\ &\quad\left.+([\mathbf{x}_j-\mathbf{x}_i]\cdot[\mathbf{x}_j-\mathbf{x}_k])_{u,v}\,(\mathbf{x}_k-\mathbf{x}_i)\right].\end{aligned}$$

Moreover, the area A_T is proportional to the sine of any angle of the triangle. Therefore, we can use the cotangent of the 2 opposite angles to $\mathbf{x}_i$ to simplify the parameter space coefficients and write:

$$\int_{\partial A\cap T}\nabla_{u,v}\mathbf{x}\cdot\mathbf{n}_{u,v}\,dl=\frac{1}{2}[cot_{u,v}\,\angle(\mathbf{x}_k)(\mathbf{x}_i-\mathbf{x}_k)+cot_{u,v}\,\angle(\mathbf{x}_j)\,(\mathbf{x}_i-\mathbf{x}_j)].$$

Combining the previous equation with Eq. (3) and (12), using the current surface discretization as the conformal parameter space, and reorganizing terms by edge contribution, we obtain:

$$\iint_A\mathbf{K}(\mathbf{x})dA=\frac{1}{2}\sum_{j\in N_1(i)}(cot\,\alpha_{ij}+cot\,\beta_{ij})\,(\mathbf{x}_i-\mathbf{x}_j)$$

where α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles sharing the edge $(\mathbf{x}_j,\mathbf{x}_i)$ as depicted in Figure 3(a).

B Surface Area Minimization in *n*D

Consider 3 points A,B,C in a space of arbitrary dimension $n>2$. As mentioned in Section 7.1, we can write the area formed by the triangle (A,B,C) as follows:

$$A^2=\frac{1}{4}\left(AB_iAB_iAC_jAC_j-AB_iAC_iAB_jAC_j\right).$$

Straightforward term by term differentiation with respect to A yields:

$$\begin{aligned}4\frac{\partial A^2}{\partial A_q} &= -\delta_{iq}AB_iAC_jAC_j-\delta_{iq}AB_iAC_jAC_j\\ &\quad-\delta_{jq}AB_iAB_iAC_j-\delta_{jq}AB_iAB_iAC_j\\ &\quad+\delta_{iq}AC_iAB_jAC_j+\delta_{iq}AB_iAB_jAC_j\\ &\quad+\delta_{jq}AB_iAC_iAC_j+\delta_{jq}AB_iAC_iAB_j\\ &= -2AB_qAC_jAC_j-2AB_iAB_iAC_q+AC_qAB_jAC_j\\ &\quad+AB_qAB_jAC_j+AB_iAC_iAC_q+AB_iAC_iAB_q\\ &= 2\,[AB_q(AB\cdot AC-AC\cdot AC)+AC_q(AB\cdot AC-AB\cdot AB)]\\ &= 2\,[BA_q(BC\cdot CA)+CA_q(AB\cdot BC)].\end{aligned}$$

Additionally, we have:

$$\frac{\partial A^2}{\partial A_q} = 2A\frac{\partial A}{\partial A_q}.$$

Using Eq. (11), and defining the cotangent of an angle between two nD vectors $\mathbf{u}$ and $\mathbf{v}$ as:

$$cot(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{\sqrt{\|\mathbf{u}\|^2\|\mathbf{v}\|^2 - (\mathbf{u}\cdot\mathbf{v})^2}},$$

the gradient of the surface area can be expressed exactly as in Eq. (4), extending nicely the 3D case to nD.

C Volume minimization in nD

Let A,B,C, and D be four n-dimensional points. As mentioned in Section 7.2, we want to calculate the volume of the region (tetrahedron in 3D) formed by the three vectors originating at A:

$$a = AB \qquad b = AC \qquad c = AD.$$

We define a transformation of a 3D unit cube with axes u,v,w: $T(u,v,w) = au + bv + cw$. The Jacobian matrix J of this transformation is composed of three columns, a,b, and c:

$$J = \big(a|b|c\big) \tag{13}$$

The volume of the transformed unit cube is: $\iiint \sqrt{det\ G}\ \ dudvdw$, where $G_{ij} = J_{im}J_{jm}$ is the 3x3 metric tensor of the transformation.

The volume V we are looking for is therefore $\frac{1}{6}$ of the square root of determinant of G (ratio between the untransformed and transformed cube). We can obtain this latter term through the standard formulation:

$$det\ G = \varepsilon_{ijk}J_{1u}J_{2v}J_{3w}J_{iu}J_{jv}J_{kw}.$$

Expanding this expression, we find the following terms involving dot products:

$$\begin{aligned} det\ G = {} & 2(a\cdot b)(a\cdot c)(b\cdot c) + (a\cdot a)(b\cdot b)(c\cdot c) \\ & -(a\cdot b)^2(c\cdot c) - (a\cdot a)(b\cdot c)^2 - (a\cdot c)^2(b\cdot b). \end{aligned}$$

From now on, the rest of the derivation is very similar to the surface area minimization in nD, detailed in the previous Section. So, using the fact that:

$$\frac{\partial V^2}{\partial A_q} = 2V\frac{\partial V}{\partial A_q}$$

and that we have, as a consequence of Eq. (13): $\frac{\partial J_{ij}}{\partial A_q} = -\delta_{jq}$, we finally get the following terms for the gradient:

$$\begin{aligned}\frac{\partial V}{\partial A_q} = \frac{1}{V}\Big(\quad & a_q \quad && ((a\cdot c)(b\cdot b)+(b\cdot c)^2+(a\cdot b)(c\cdot c)\\ & && -(b\cdot b)(c\cdot c)-(a\cdot c)(b\cdot c)-(a\cdot b)(b\cdot c))\\ & +b_q && ((b\cdot c)(a\cdot a)+(a\cdot c)^2+(a\cdot b)(c\cdot c)\\ & && -(a\cdot a)(c\cdot c)-(b\cdot c)(a\cdot c)-(a\cdot b)(a\cdot c))\\ & +c_q && ((a\cdot c)(b\cdot b)+(a\cdot b)^2+(a\cdot a)(b\cdot c)\\ & && -(a\cdot a)(b\cdot b)-(a\cdot b)(b\cdot c)-(a\cdot b)(a\cdot c))).\end{aligned}$$

Although we can use this expression to compute the gradient of the volume, it turns out we can simplify it using Lagrange's identity to get a better insight of what these terms are. Lagrange's identity in 3D can be written as:

$$(s\cdot u)(t\cdot v)-(s\cdot v)(t\cdot u)=(s\wedge t)\cdot(u\wedge v).$$

The multiplicative term in front of c_q is then $(AB\wedge AC)\cdot(DC\wedge DB)$, representing (up to the product of the norm of these vectors) the cosine of the dihedral angle between the two opposite faces to the edge c. As the volume is proportional to the sine of this angle, we can see that we once again have the same formula as Eq. (4), this time with cotangents of the dihedral angles opposite to the edge. Note that there are generally more than two tetrahedra sharing the same edge.