# Discrete Differential-Geometry Operators for Triangulated 2-Manifolds

Mark Meyer[1], Mathieu Desbrun[1,2], Peter Schröder[1], and Alan H. Barr[1]

[1] Caltech
[2] USC

**Summary.** This paper proposes a unified and consistent set of flexible tools to approximate important geometric attributes, including normal vectors and curvatures on arbitrary triangle meshes. We present a consistent derivation of these first and second order differential properties using *averaging Voronoi cells* and the mixed Finite-Element/Finite-Volume method, and compare them to existing formulations. Building upon previous work in discrete geometry, these operators are closely related to the continuous case, guaranteeing an appropriate extension from the continuous to the discrete setting: they respect most 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 also present applications of these operators, such as mesh smoothing, enhancement, and quality checking, and show results of denoising in higher dimensions, such as for 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 therefore 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 for many applications using meshes.

## 1.1 Previous work

Several expressions for 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 normals of the adjacent faces of a mesh. Thürmer and Wüthrich [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) tessellation 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. Even for a property as fundamental as the surface normal, we can see that several (often contradictory) formulæ exist.

Taubin proposed the most complete derivation of surface properties, 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 (invariants) on $\mathcal{C}^0$ surfaces such as polyhedral meshes.

In order to preserve fundamental invariants, we have followed a path initiated by Federer, Fu, Polthier, and Morvan to name a few [Fu93, PP93, PS98, Mor01, TM02]. This series of work proposed simple expressions for the total curvatures, as well as the Dirichlet energy for triangle meshes, and derived discrete methods to compute minimal surfaces or geodesics. We refer the reader to the overview compiled by Morvan [Mor01]. Note also the tight connection with the "Mimetic Discretizations" used in computational physics by Shashkov, Hyman, and Steinberg [HS97, HSS97]. Although it shares a lot of similarities with all these approaches, our work offers a different, unified derivation that ensures accuracy and tight error bounds, leading to simple formulæ that are straightforward to implement.
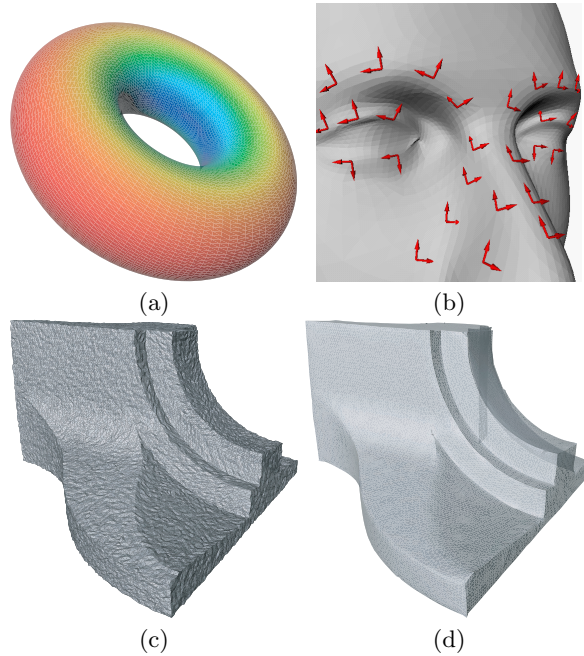
**Fig. 1.** *Some applications of our discrete operators: (a) mean curvature plot for a discrete surface, (b) principal curvature directions on a triangle mesh, (c-d) automatic feature-preserving denoising of a noisy mesh using anisotropic smoothing.*

### Contributions

In this paper we define and derive the first and second order differential attributes (normal vector $\mathbf{n}$, mean curvature $\kappa_H$, Gaussian curvature $\kappa_G$, principal curvatures $\kappa_1$ and $\kappa_2$, and principal directions $\mathbf{e}_1$ and $\mathbf{e}_2$) for piecewise linear surfaces such as arbitrary triangle meshes. We present a unified framework for deriving such quantities resulting in a set of operators that is consistent, accurate, robust (in both regular and irregular sampling) and simple to compute.

The remainder of this paper is organized as follows. 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 first 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 some of 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 triangle 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 the plane containing both $\mathbf{n}$ and $\mathbf{e}_\theta$. The two *principal curvatures* $\kappa_1$ and $\kappa_2$ 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)). The *mean curvature* $\kappa_H$ is defined as the average of the normal curvatures:

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

Expressing the normal curvature in terms of the principal curvatures, $\kappa^N(\theta) = \kappa_1 cos^2(\theta) + \kappa_2 sin^2(\theta)$, leads to the well-known definition: $\kappa_H = (\kappa_1 + \kappa_2)/2$. The *Gaussian curvature* $\kappa_G$ is defined as the product of the two principle curvatures:

$$\kappa_G = \kappa_1 \kappa_2. \tag{2}$$

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

$$2\kappa_H \ \mathbf{n} = \lim_{diam(\mathcal{A}) \to 0} \frac{\nabla \mathcal{A}}{\mathcal{A}}$$

where $\mathcal{A}$ is a infinitesimal area around a point $P$ on the surface, $diam(\mathcal{A})$ its diameter, and $\nabla$ is the gradient with respect to the $(x, y, z)$ coordinates of $P$. We will make extensive use of the mean curvature normal $\kappa_H \ \mathbf{n}$. Therefore, we will denote by $\mathbf{K}$ the operator that maps a point $P$ on the surface to the vector $\mathbf{K}(P) = 2\kappa_H(P) \ \mathbf{n}(P)$. $\mathbf{K}$ is also known as the Laplace-Beltrami

operator for the surface $S$. Note that in the remainder of this paper we will make no distinction between an operator and the value of this operator at a point as it will be clear from context. Gaussian curvature can also be expressed as a limit:

$$\kappa_G = \lim_{diam(\mathcal{A}) \to 0} \frac{\mathcal{A}^{\mathcal{G}}}{\mathcal{A}} \tag{3}$$

where $\mathcal{A}^{\mathcal{G}}$ is the area of the image of the Gauss map (also called the spherical image) associated with the infinitesimal surface $\mathcal{A}$. The above definitions, as well as many more details, can be found in various sources on Differential Geometry [Gra98, DHKW92].
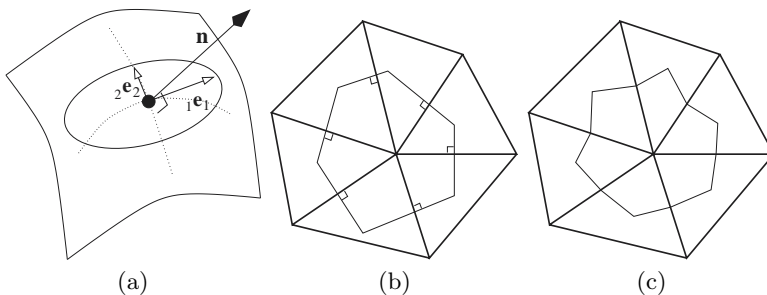


(a)                    (b)                    (c)

**Fig. 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

Most of the smooth definitions described above need to be reformulated for $\mathcal{C}^0$ surfaces. We can consider a mesh as either the limit of a family of smooth surfaces, or as a linear (yet assumedly "good") approximation of an arbitrary surface. We define properties (geometric quantities) of the surface at each vertex as *spatial averages* around this vertex. If these averages are made *consistently*, and given a few assumptions such as non-degeneracy of the triangle mesh, 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 as a vertex will be updated according to the average behavior of the surface around it. Therefore, the piecewise linear result of the flow 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 or star neighborhood. For example, we define the discrete Gaussian curvature, $\widehat{\kappa}_G$, at a vertex $P$ as:

$$\widehat{\kappa}_G = \frac{1}{\mathcal{A}} \iint_{\mathcal{A}} \kappa_G \ dA$$

where $\mathcal{A}$ is a properly selected area around $P$. Note however that we will not distinguish between the (continuous) pointwise and the (discrete) spatially averaged notation, except when 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 $\mathcal{A}_M$. 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 correspondences between the continuous and the discrete case. We then find the precise surface patch that optimizes the accuracy of our operators, completing the operator derivation. These steps will be explained in detail for the first operator, the mean curvature normal operator, $\mathbf{K}$, and a more direct derivation will be used for the Gaussian curvature operator $\kappa_G$, the two principal curvature operators $\kappa_1$ and $\kappa_2$, 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 common in practice, see Figure 2(b-c). In each case, their piecewise linear boundaries connect the midpoints of the edges emanating from the center vertex and a point within each adjacent triangle. For the point inside each adjacent triangle, we can use either the barycenter or the circumcenter. The surface area formed from using the barycenters is denoted $\mathcal{A}_{\mathrm{Barycenter}}$ while the

surface area using the circumcenters is recognized as the local Voronoi cell and denoted $\mathcal{A}_{\mathrm{Voronoi}}$. In the general case when this point could be anywhere, we will denote the surface area as $\mathcal{A}_{\mathrm{M}}$.

We now wish to compute the integral of the mean curvature normal over the area $\mathcal{A}_{\mathrm{M}}$. 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 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]:

$$\iint_{\mathcal{A}_{\mathrm{M}}} \mathbf{K}(\mathbf{x})\ dA = -\iint_{\mathcal{A}_{\mathrm{M}}} \Delta_{u,v}\mathbf{x}\ du\ dv. \tag{4}$$

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 thus reduces to the following simple form:

$$\iint_{\mathcal{A}_{\mathrm{M}}} \mathbf{K}(\mathbf{x})dA = \frac{1}{2} \sum_{j\in N_1(i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})\ (\mathbf{x}_i - \mathbf{x}_j), \tag{5}$$

where $\alpha_{ij}$ and $\beta_{ij}$ 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 equation was already obtained by minimizing the Dirichlet energy over a triangulation in [PP93]. Additionally, 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*:

$$\iint_{\mathcal{A}_{\mathrm{M}}} \mathbf{K}(\mathbf{x})dA = \nabla \mathcal{A}_{\text{1-ring}}. \tag{6}$$

where $\mathcal{A}_{\text{1-ring}}$ is the 1-ring area around a vertex $P$, and $\nabla$ is the gradient with respect to the $(x, y, z)$ coordinates of $P$.

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 since 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 in the next section that Voronoi cells provide provably tight error bounds under reasonable assumptions of smoothness.

### 3.2 Voronoi Regions for Tight Error Bounds

We now show that using Voronoi regions provides provably tight error bounds for the discrete operators by comparing the local spatial average of mean curvature with the actual pointwise value. Given a $\mathcal{C}^2$ surface tiled by small patches $\mathcal{A}_i$ 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:

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

where $C_i$ is the Lipschitz constant of the Beltrami operator over the smooth surface patch $\mathcal{A}_i$, and $C_{max}$ the maximum of the Lipschitz constants. The Voronoi region of each sample point *by definition* minimizes $\|\mathbf{x} - \mathbf{x}_i\|$ since they contain the closest points to each sample, thus minimizing the bound on the error $E$ due to spatial averaging [DFG99]. 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 $\epsilon$, we can actually guarantee that the Voronoi cell borders are less than $O(\epsilon)$ 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, determining these Voronoi cells requires few extra computations.

### 3.3 Voronoi Region Area

Given a non-obtuse triangle $P, Q, R$ with circumcenter $O$, as depicted in Figure 3(b), we must now compute the Voronoi region for $P$. 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 can write the non-obtuse Voronoi area for a vertex $\mathbf{x}_i$ as a function of the neighbors $\mathbf{x}_j$:
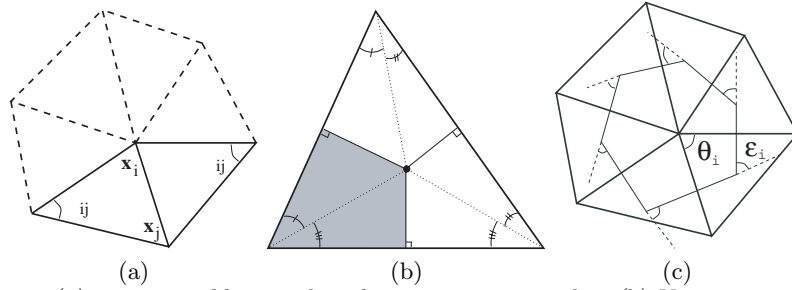
**Fig. 3.** *(a) 1-ring neighbors and angles opposite to an edge; (b) Voronoi region on a non-obtuse triangle; (c) External angles of a Voronoi region.*

$$\mathcal{A}_{\text{Voronoi}} = \frac{1}{8} \sum_{j \in N_1(i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})\ \|\mathbf{x}_i - \mathbf{x}_j\|^2. \qquad (7)$$

Since the cotangent terms were already computed for Eq. (5), the Voronoi area can be computed very efficiently. However, if there is an obtuse triangle among the 1-ring neighbors 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, the integral of the Laplace-Beltrami operator given in equation (6) 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 use a slightly more subtle area, to guarantee a perfect tiling of our surface, and therefore, optimized accuracy as each point on the surface is counted once and only once. We define a new surface area for each vertex $\mathbf{x}$, denoted $\mathcal{A}_{\text{Mixed}}$: for each non-obtuse triangle, we use the circumcenter point, and for each obtuse triangle, we use the midpoint of the edge opposite to the obtuse angle. Algorithmically, this area around a point $\mathbf{x}$ can be easily computed as detailed in Figure 4. 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 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.

```
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 either area(T)/4 or area(T)/2
        If the angle of T at x is obtuse
            𝒜_Mixed+ = area(T)/2
        Else
            𝒜_Mixed+ = area(T)/4
```

**Fig. 4.** *Pseudo-code for region $\mathcal{A}_{Mixed}$ on an arbitrary mesh*

### 3.5 Discrete Mean Curvature Normal Operator

Now that the mixed area is defined, we can express the mean curvature normal operator **K** defined in Section 2.1 using the following expression:

**Mean Curvature Normal Operator**

$$\mathbf{K}(\mathbf{x}_i) = \frac{1}{2\mathcal{A}_{\text{Mixed}}} \sum_{j \in N_1(i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})\ (\mathbf{x}_i - \mathbf{x}_j) \qquad (8)$$

From this expression, we can easily compute the mean curvature value $\kappa_H$ 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 of zero mean curvature (flat plane or local saddle point), we simply average the 1-ring face normal vectors to evaluate **n** appropriately.

It is interesting to notice that using the barycentric area as an averaging region results in an operator very similar to the definition of the mean curvature normal by Desbrun *et al.* [DMSB99], since $\mathcal{A}_{\text{Barycenter}}$ is a third of the whole 1-ring area $\mathcal{A}_{\text{1-ring}}$ used in their derivation - however, our new derivation uses *non-overlapping* regions and is therefore more accurate. At this time, we are not aware of a proof of convergence for this operator. However, our tests have shown no divergence as we refine a mesh, as long as we do not degrade the mesh quality (the triangles must not degenerate). We will give more precise numerical results in Section 6.1 showing the improved quality of our new estimate.

## 4 Discrete Gaussian Curvature

In this section, the Gaussian curvature $\kappa_G$ for bivariate (2D) meshes embedded in 3D is studied. We will demonstrate that a derivation similar to the above is easily obtained.

### 4.1 Expression of the Local Integral of $\kappa_G$

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 $\kappa_G$ over a finite-volume region on a piecewise linear surface. From Eq. (3), we could compute the integral over an area $\mathcal{A}_{\mathrm{M}}$ as the associated spherical image area (also called the image of the Gauss map). Instead, we use the *Gauss-Bonnet theorem* [DHKW92, Gra98, AZ67] which proposes a very simple equality, valid over any surface patch. Applied to our local finite-volume regions, the Gauss-Bonnet theorem simply states:

$$\iint_{\mathcal{A}_{\mathrm{M}}} \kappa_G \; dA = 2\pi - \sum_j \epsilon_j$$

where the $\epsilon_j$ are the external angles of the boundary, as indicated in Figure 3(c). Note that this simplified form results from the fact that the integral of geodesic curvature on the piece-wise linear boundaries is zero. If we apply this expression to 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 $\theta_j$, 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 is still valid for the mixed region and is proven using a similar geometric argument. This result was already proven by Polthier and Schmies [PS98], who considered the area of the image of the Gauss map for a vertex on a polyhedral surface. Therefore, analogous to Eq. (6), we can now write for the 1-ring neighborhood of a vertex $\mathbf{x}_i$:

$$\iint_{\mathcal{A}_{\mathrm{M}}} \kappa_G dA = 2\pi - \sum_{j=1}^{\#f} \theta_j$$

where $\theta_j$ 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 $\mathcal{A}_{\mathrm{M}}$ 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 good error bounds. In practice, we use the mixed area $\mathcal{A}_{\mathrm{Mixed}}$ to account for obtuse triangulations. Since the mixed area cells tile the whole surface without any overlap, we will satisfy the (continuous) Gauss-Bonnet theorem: the integral of the discrete Gaussian curvature over an entire sphere for example will be equal to $4\pi$ *whatever the discretization used* since the sphere is a closed object of genus zero. This result ensures

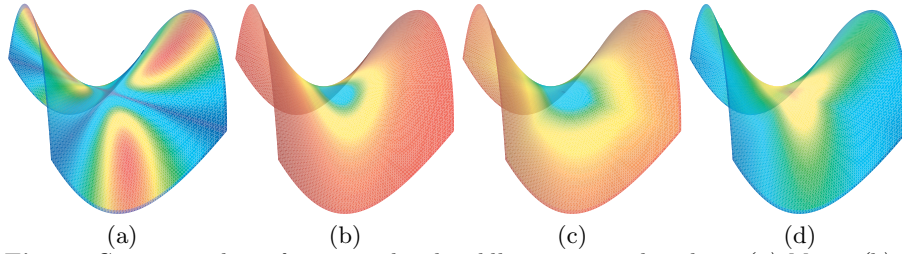(a)            (b)            (c)            (d)

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

a robust numerical behavior of our discrete operator. Our Gaussian curvature discrete operator can thus be expressed as:

### Gaussian Curvature Operator

$$\kappa_G(\mathbf{x}_i) = (2\pi - \sum_{j=1}^{\#f} \theta_j)/\mathcal{A}_{\mathrm{Mixed}} \tag{9}$$

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. Note also that *convergence conditions* (using fatness or straightness) exist for this operator [Fu93, TM02], proving that if the triangle mesh does not degenerate, the approximation quality gets better as the mesh is refined. We postpone 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 $\kappa_1$ and $\kappa_2$. Therefore, since both $\kappa_H$ and $\kappa_G$ have been derived for triangulated surfaces, we can define the discrete principal curvatures as:

## Principal Curvature Operators

$$\boxed{\kappa_1(\mathbf{x}_i) = \kappa_H(\mathbf{x}_i) + \sqrt{\Delta(\mathbf{x}_i)}} \tag{10}$$

$$\boxed{\kappa_2(\mathbf{x}_i) = \kappa_H(\mathbf{x}_i) - \sqrt{\Delta(\mathbf{x}_i)}} \tag{11}$$

with:  $\Delta(\mathbf{x}_i) = \kappa_H^2(\mathbf{x}_i) - \kappa_G(\mathbf{x}_i)$ and $\kappa_H(\mathbf{x}_i) = \dfrac{1}{2}\|\mathbf{K}(\mathbf{x}_i)\|.$

Unlike the continuous case where $\Delta$ is always positive, we must make sure that $\kappa_H^2$ is always larger than $\kappa_G$ to avoid any numerical problems, and threshold $\Delta$ to zero if it is not the case (an extremely rare occurrence).

### 5.2 Mean Curvature as a Quadrature

In order to determine the principal axes at a vertex, we will first show that the mean curvature from our previous expression can be interpreted as a quadrature of normal curvature samples:

$$
\begin{aligned}
\kappa_H(\mathbf{x}_i) &= \frac{1}{2}\left(2\kappa_H(\mathbf{x}_i)\mathbf{n}\right)\cdot\mathbf{n} = \frac{1}{2}\mathbf{K}(\mathbf{x}_i)\cdot\mathbf{n} \\
&= \frac{1}{4\mathcal{A}_{\text{Mixed}}} \sum_{j\in N_1(i)} \left(\cot\alpha_{ij} + \cot\beta_{ij}\right)\left(\mathbf{x}_i - \mathbf{x}_j\right)\cdot\mathbf{n} \\
&= \frac{1}{4\mathcal{A}_{\text{Mixed}}} \sum_{j\in N_1(i)} \left(\cot\alpha_{ij} + \cot\beta_{ij}\right)\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}(\mathbf{x}_i - \mathbf{x}_j)\cdot\mathbf{n} \\
&= \frac{1}{\mathcal{A}_{\text{Mixed}}} \sum_{j\in N_1(i)} \left[\frac{1}{8}(\cot\alpha_{ij} + \cot\beta_{ij})\,\|\mathbf{x}_i - \mathbf{x}_j\|^2\right]\kappa_{i,j}^N, \tag{12}
\end{aligned}
$$

where we define:

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

This $\kappa_{i,j}^N$ can be shown to be an estimate of the normal curvature in the direction of the edge $\mathbf{x}_i\mathbf{x}_j$. 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). Since we must have a right angle at the neighbor vertex $\mathbf{x}_j$, we have $(\mathbf{x}_i - \mathbf{x}_j)\cdot(\mathbf{x}_i - \mathbf{x}_j - 2R\,\mathbf{n}) = 0$. This implies:

$$R = \|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\,(\mathbf{x}_i - \mathbf{x}_j)\cdot\mathbf{n})$$

This proves that $\kappa_{i,j}^N$ is a normal curvature estimate in the direction of the edge $\mathbf{x}_i\mathbf{x}_j$ (as it is the inverse of the radius of the osculating circle). This expression was also used in the context of curvature approximation in [MS92] and [Tau95].

Therefore, Eq. (12) can be interpreted as a quadrature of the integral from Eq. (1), with weights $w_{ij}$:

$$\kappa_H(\mathbf{x}_i) = \sum_{j \in N_1(i)} w_{ij} \ \kappa_{i,j}^N,$$

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 Principal Directions

In order to find the two orthogonal principal curvature directions we can simply compute the eigenvectors of the curvature tensor. Since the mean curvature obtained from our derivation can be seen as a quadrature using each edge as a sample direction, we 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 $a, b, c$:

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

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:

$$\mathbf{d}_{i,j}^T \, B \, \mathbf{d}_{i,j} \ = \ \kappa_{i,j}^N,$$

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:

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

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

$$E(a, b, c) = \sum_j w_j \left( \mathbf{d}_{i,j}^T \, B \, \mathbf{d}_{i,j} - \kappa_{i,j}^N \right)^2$$

Adding the two constraints $a + b = 2\kappa_H$ 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 necessary (reducing the problem to a linear system). An example of these principal directions is shown in Figure 1(b).

Although we could actually determine the principal curvatures (and thus the mean and gaussian curvatures) using an unconstrained least squares procedure, we use our operators to compute the curvatures and only use the least squares for the principal directions as the curvature values computed from the least squares are often *less* accurate in practice while the directions are fairly robust. A plausible interpretation for the bad numerical properties of a pure least squares approach is the hypothesis of elliptic curvature variation: although this is perfectly valid for smooth surfaces, this is somewhat arbitrary for coarse, triangulated surfaces. It seems therefore more natural to use our previous operators that rely on differential properties still valid on discrete meshes.

# 6 Results & Applications

With robust curvature estimates at our disposal, we demonstrate some useful applications such as quality checking for surface design and tools for smoothing and enhancement of meshes. We first demonstrate the numerical quality of our operators.

## 6.1 Numerical Quality of our Operators

We performed a number of tests to demonstrate the accuracy of our approach in practice. First, we compared our operators to the well-known second-order accurate Finite Difference operators on several discrete meshes approximating simple surfaces such as spheres, or hyperboloids, where the curvatures are known analytically. In order to do so, we used special surfaces defined as height fields over a flat, regular grid so that the FD operators can be computed and tested against our results. The table below lists some representative results:

| $\%error$ | FD $\kappa_H$ | [DMSB99] $\kappa_H$ | our $\kappa_H$ | FD $\kappa_G$ | our $\kappa_G$ |
|---|---|---|---|---|---|
| Sphere patch | 0.20 | 0.17 | 0.16 | 0.4 | 1.2 |
| Paraboloid | 0.0055 | 0.0038 | 0.0038 | 0.01 | 0.02 |
| Torus (irregular) | - | 0.047 | 0.036 | - | 0.05 |

**Table 1.** *Comparison of our operators with Finite Differences. The error is measured in mean percent error compared to the exact, known curvature values. Dashes "-" indicate that the FD tests cannot be performed since the triangulation is irregular. The angles $\theta_j$ needed for the Gaussian curvature were computed using the C function* `atan2`, *instead of* `acos` *or* `asin` *since* `acos` *and* `asin` *would significantly deteriorate the precision of the results.*

Overall, the numerical quality of our operators is equivalent to FD operators for regular sampling. A major advantage of our new operators over FD operators is that these differential-geometry based operators can *still* be used on irregular sampling, with the same order of accuracy.

We also tested our operators against one of the most widely used curvature estimation techniques [Tau95]. We tested several simple surfaces (spheres, parametric surfaces, etc.) to determine the effect of sampling on the operators. The surfaces were created with 258 points, quadrisected and reprojected to create surfaces of 1026, 4098 and 16386 points. In all cases, the average percent error of our operators did not exceed 0.07% for mean curvature and 1.3% for gaussian curvature. The previous method had average errors of up to 1.8% for mean curvature and exceeding 10% in some instances for gaussian curvature.

Finally, we tested the effects of irregularity on the operators. In irregular areas of the surfaces (such as the area joining two regions of different sampling rates), our operators performed with the same order of accuracy as in the fairly regular regions (less than 0.2% average error for mean curvature and below 1.8% average error for gaussian curvature in regions of mild irregularity). The accuracy of our operators decreases as the irregularity (angle and edge length dispersion) increases, but, in practice, the rate at which the error increases is low.

### 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.

### 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.

**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
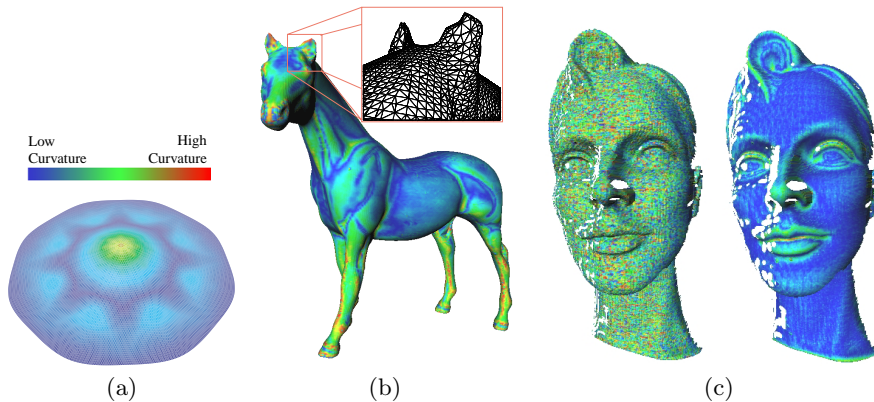
**Fig. 6.** *Mean curvature plots revealing surface details for: (a) a Loop surface from an 8-neighbor ring, (b) a horse mesh, (c) a noisy mesh obtained from a 3D scanner and the same mesh after smoothing. Our operator performs well on irregular sampling such as on the ear of the horse. Notice also how the operator correctly computes quickly varying curvatures on the noisy head while returning slowly varying curvatures on the smoothed version.*
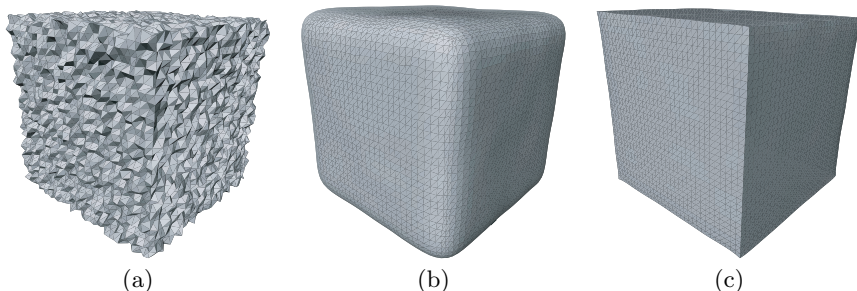


**Fig. 7.** *Cube: (a) Original, noisy mesh (±3% uniform noise added along the normal direction). (b) Isotropic smoothing. (c) Anisotropic smoothing defined in Section 6.3.*

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.

**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 noisy, yet homogeneous regions should be smoothed. Different forms of anisotropic diffusion have shown good results for this problem in image processing [PM90], in flow visualization [PR99], and more recently on meshes too [CDR00]. 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 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.

**An Anisotropic Smoothing Technique** Most of the meshes acquired 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 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:

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

The parameter $T$ is a user defined value determining edges. The general smoothing flow is then: $\partial\mathbf{x}_i/\partial t = -w_i\,\kappa_H(\mathbf{x}_i)\,\mathbf{n}(\mathbf{x}_i)$. As we can see, uniformly noisy regions (cases 1 and 5 in the weight definition given above) will be smoothed isotropically, while corners (case 2) will not move. For edges (cases 3 and 4), we smooth with a speed proportional to the minimum curvature, to be assured not to smooth ridges. The caveat is that this smoothing is no longer well-posed: we try to enhance edges, and this is by definition a very unstable process. Pre-mollification techniques have been reported successful in [PR99], and should be used in such a process. However, we have 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 types of algorithms (smoothing techniques, etc.) on datasets such as vector fields, tensor images, or volume data.

### 7.1 Operators for 2-Manifolds in $n$D

We now 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 $n$D. The area of a triangle formed by two vectors $\mathbf{u}$ and $\mathbf{v}$ in 3D is $2\mathcal{A} = ||\mathbf{u} \times \mathbf{v}||$. Being proportional to the sine of the angle between vectors, we can also express it as:

$$\mathcal{A} = \frac{1}{2}||\mathbf{u}||||\mathbf{v}||sin(\mathbf{u}, \mathbf{v}) = \frac{1}{2}||\mathbf{u}||||\mathbf{v}||\sqrt{1 - cos^2(\mathbf{u}, \mathbf{v})}$$
$$= \frac{1}{2}\sqrt{||\mathbf{u}||^2||\mathbf{v}||^2 - (\mathbf{u} \cdot \mathbf{v})^2}. \tag{13}$$

This expression is now valid in $n$D, 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. (5) in $n$D. We detail this proof in Appendix B, but the result is very simple: the previous cotangent formula is still valid in $n$D 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 $n$D space is straightforward and efficient, as dot products require little computation.

**Gaussian Curvature Operator**  The expression of the Gaussian curvature operator Eq. (9) still holds in $n$D. Indeed, the Gaussian curvature is an intrinsic attribute of a 2-manifold, and does not depend on the embedding.

## 7.2 Beltrami Operator for 3-Manifolds in $n$D

We also extend the previous mean curvature normal operator, valid on triangulated surfaces, to tetrahedralized volumes which are 3-parameter volumes in an embedding space of arbitrary dimension. This can be used, for example, on 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 as 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. To demonstrate the practical accuracy 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.
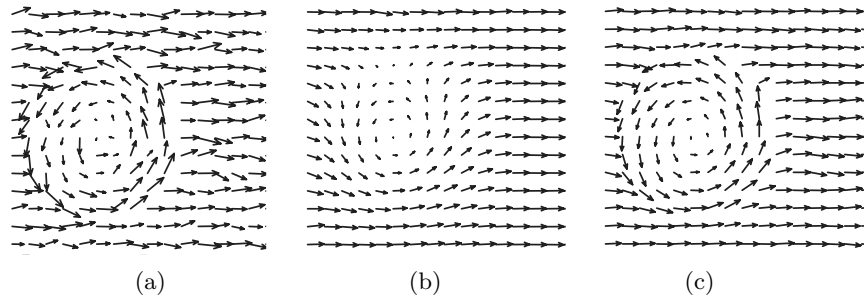


(a)                              (b)                              (c)

**Fig. 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.*

## 8 Conclusion

A complete set of accurate differential operators for any triangulated surface has been presented. We consistently derived estimates for normal vectors and mean curvatures (Eq. (8)), Gaussian curvatures (Eq. (9)), principal curvatures (Eq. (10) and (11)), 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. Our operators perform as well as established methods such as Finite Differences in the regular setting and degrade gracefully as irregularity is increased. 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 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 other surface-based applications. The mean curvature normal operator for instance 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, remeshing, geometry based subdivision schemes, and mesh simplification along the lines of [HG99].

Future work will try to answer some of the open questions. For instance, we are trying to determine what would be the minimum sampling rate of a continuous surface to guarantee that our discrete estimates are accurate within a given $\epsilon$ - laying the foundations for an irregular sampling theory. More generally, we would like to extend well known digital signal processing tools and theorems to digital geometry.

# References

[AZ67]      A. D. Aleksandrov and V. A. Zalgaller. *Intrinsic Geometry of Surfaces.* AMS, Rhode Island, USA, 1967.

[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.

[CDR00]     U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic Geometric Diffusion in Surface Processing. In *IEEE Visualization*, pages 397–405, 2000.

[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 Ortwin 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.

[Fu93]      J. Fu. Convergence of Curvatures in Secant Approximations. *Journal of Differential Geometry*, 37:177–190, 1993.

[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.

[HS97]      J. M. Hyman and M. Shashkov. Natural Discretizations for the Divergence, Gradient and Curl on Logically Rectangular Grids. *Applied Numerical Mathematics*, 25:413–442, 1997.

[HSS97]     J. M. Hyman, M. Shashkov, and S. Steinberg. The numerical solution of diffusion problems in strongly heterogenous non-isotropic materials. *Journal of Computational Physics*, 132:130–148, 1997.

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

[Mor01]     J.M. Morvan. On Generalized Curvatures. *Preprint*, 2001.

[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 Visualization*, 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.

[TM02]    B. Thibert and J.M. Morvan.  Approximations of A Smooth Surface with a Triangulated Mesh. *Preprint*, 2002.

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

## Appendix

## A Mean Curvature Normal on a triangulated domain

In this section, we derive the integral of the mean curvature normal over a triangulated domain. We begin by computing the integral of the Laplacian of the surface point $\mathbf{x}$ with respect to the conformal parameter space. Using Gauss's theorem, we can turn the integral of a Laplacian over a region into a line integral over the boundary of the region:

$$\iint_{\mathcal{A}_{\mathrm{M}}} \Delta_{u,v}\mathbf{x}\ du\ dv = \int_{\partial \mathcal{A}_{\mathrm{M}}} \nabla_{u,v}\mathbf{x} \cdot \mathbf{n}_{u,v}\ \ dl, \tag{14}$$

where 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 $\partial \mathcal{A}_{\mathrm{M}}$ 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 write:

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

where $^{\perp}$ denotes a counterclockwise rotation of 90 degrees.
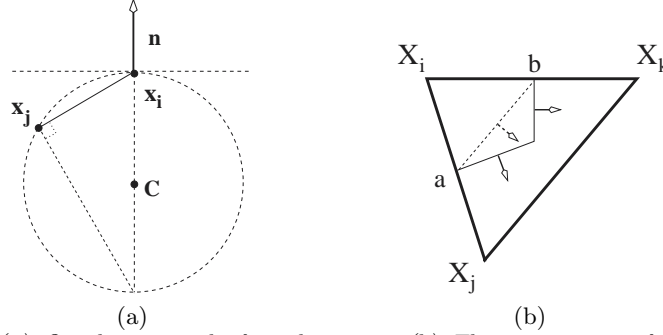


(a)                              (b)

**Fig. 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 $B_l$ over the triangle, it follows:

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

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 $\nabla_{u,v}\mathbf{x} = \frac{1}{2\mathcal{A}_T}[(\mathbf{x}_j-\mathbf{x}_i)\left([\mathbf{x}_i - \mathbf{x}_k]_{u,v}^{\perp}\right)^T + (\mathbf{x}_k-\mathbf{x}_i)\left([\mathbf{x}_j - \mathbf{x}_i]_{u,v}^{\perp}\right)^T]$, where $^T$ denotes the transpose. Note that $\nabla_{u,v}\mathbf{x}$ is an $n$ x 2 matrix - $n$ for the dimension of the embedding of $\mathbf{x}$ and 2 for the $(u,v)$ space. The previous integral can then be rewritten as:

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

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

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

Combining the previous equation with Eq. (4) and (14), using the current surface discretization as the conformal parameter space, and reorganizing terms by edge contribution, we obtain:

$$\iint_{\mathcal{A}_\mathrm{M}} \mathbf{K}(\mathbf{x}) dA = \frac{1}{2} \sum_{j \in N_1(i)} \left( cot\ \alpha_{ij} + cot\ \beta_{ij} \right) (\mathbf{x}_i - \mathbf{x}_j)$$

where $\alpha_{ij}$ and $\beta_{ij}$ 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

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

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:

$$\mathcal{A}^2 = \frac{1}{4} \left( AB_i AB_i AC_j AC_j - AB_i AC_i AB_j AC_j \right).$$

Straightforward term by term differentiation with respect to $A$ yields:

$$\begin{aligned}
4\frac{\partial \mathcal{A}^2}{\partial A_q} &= -\delta_{iq} AB_i AC_j AC_j - \delta_{iq} AB_i AC_j AC_j \\
&\quad -\delta_{jq} AB_i AB_i AC_j - \delta_{jq} AB_i AB_i AC_j \\
&\quad +\delta_{iq} AC_i AB_j AC_j + \delta_{iq} AB_i AB_j AC_j \\
&\quad +\delta_{jq} AB_i AC_i AC_j + \delta_{jq} AB_i AC_i AB_j \\
&= -2 AB_q AC_j AC_j - 2 AB_i AB_i AC_q + AC_q AB_j AC_j \\
&\quad + AB_q AB_j AC_j + AB_i AC_i AC_q + AB_i AC_i AB_q \\
&= 2 \left[ AB_q (AB \cdot AC - AC \cdot AC) + AC_q (AB \cdot AC - AB \cdot AB) \right] \\
&= 2 \left[ BA_q (BC \cdot CA) + CA_q (AB \cdot BC) \right].
\end{aligned}$$

Additionally, we have:

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

Using Eq. (13), and defining the cotangent of an angle between two $n$D 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. (5), elegantly extending the 3D case to $n$D.

## C Volume minimization in $n$D

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 $c$:

$$J = \big( a|b|c \big) \tag{15}$$

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

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

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

$$\det G = 2(a \cdot b)(a \cdot c)(b \cdot c) + (a \cdot a)(b \cdot b)(c \cdot c)$$
$$-(a \cdot b)^2(c \cdot c) - (a \cdot a)(b \cdot c)^2 - (a \cdot c)^2(b \cdot b).$$

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

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

and that we have, as a consequence of Eq. (15): $\frac{\partial J_{ij}}{\partial A_q} = -\delta_{iq}$, we finally get the following terms for the gradient:

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

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. (5), 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.