Tensors in Geometry Processing

Eugene Zhang

Abstract Tensor fields have a wide range of applications outside scientific visualization. In this chapter, we review various types of tensors used in geometry processing, including their properties, application requirements, as well as theoretical and practical results. We will focus on the metric tensor and the curvature tensor, two of the most studied tensors in geometry processing.

1 Introduction

Tensor fields have been a major research topic in scientific visualization and medical imaging, due to their wide applicability in physics, chemistry, and biology. Examples of tensor fields in these domains include stress and strain tensors in solid mechanics, velocity gradient tensors in fluid dynamics, and diffusion tensors in medical imaging. In contrast, considerably less research effort has been given to tensors in geometry processing.

Fortunately, increasing attention has been given to tensor fields by the geometry processing community in recent years. Results in tensor field analysis and visualization have been borrowed from communities that traditionally deal with tensors (scientific visualization, medical imaging) and applied to geometry processing applications such as non-photorealistic rendering, surface parameterization, and geometry remeshing.

In differential geometry, there are three tensor fields describing the geometry of the surface. They are termed the first, second, and third *fundamental forms*, and are usually denoted by I, II, and III, respectively. However, these three forms are *not* independent as they are related by an Eq. [8]. Consequently, one often focuses on

E. Zhang (✉)
School of Electrical Engineering and Computer Science, 2111 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331, USA
e-mail: zhange@eecs.oregonstate.edu

C.-F. Westin et al. (eds.), *Visualization and Processing of Tensors and Higher Order Descriptors for Multi-Valued Data*, Mathematics and Visualization,
DOI 10.1007/978-3-642-54301-2_13, © Springer-Verlag Berlin Heidelberg 2014

the first two fundamental forms, which can be expressed in the language of *metric tensor* and *curvature tensor*, respectively. In the next sections, we will describe these tensors and their applications in geometry processing.

2 Metric Tensor

In this section we describe applications in geometry processing related to the metric tensor.

In the early days of computer graphics when processing speed and memory of computers are much lower than their counterparts today, modeling 3D geometry with complex details was often considered computationally prohibitive. Yet, the ability to render such details is essential to the realism of the synthesized images. To deal with this challenge, the idea of *texture maps* was employed. Basically, the geometry of a surface is modeled with two complementary components. The first component is a surface that approximates the target surface but has much lower geometric details. The second component is a texture map which is an image that contains fine geometry details. When *wrapping* the texture onto the surface appropriately, the resulting rendering has perceptually similar visual quality to images generated by directly modeling the surface with high geometry detail. Thanks to the hardware setup, the texture map approach is much faster than the geometry-only approach. The idea of using texture maps in representing high geometry details has inspired much research in image-based rendering [17].

To connect the two components in this approach, i.e., the surface S and the texture map I (represented as an 2D image), a correspondence between the two must be established. This correspondence, i.e., a map τ from S to I is referred to as the *surface parameterization* for S. One example of this is the world map for the Earth.

Generating a high-quality parameterization given an arbitrary surface is both important and challenging. For example, due to topological constraints, the surface must be cut open in order to be flattened [7]. The curves along which the surface is cut open are referred to as *seams*. Seams require special care as they lead to texture discontinuity when wrapping the texture back to the surface. However, an even more challenging problem is distortion. Distances can be distorted, so can angles and areas. In cartography, such problems can lead to false notions such as that Greenland is larger than Australia and that the South Pole (a point) is a line. In computer graphics, such distortions lead to uneven sampling rates over the surface, since regions of the same area in the surface may be given drastically different areas in the texture map. Consequently, geometry details in regions receiving less-than-average portion of the texture map typically go through more aggressive low-pass filters when the surface signal is stored in the texture map than regions receiving more-than-average portions of the map. This results in greater loss in the details of the surface signal for regions receiving less-than-average areas in the texture map. To ensure the visual realism for regions receiving the smallest share in the texture

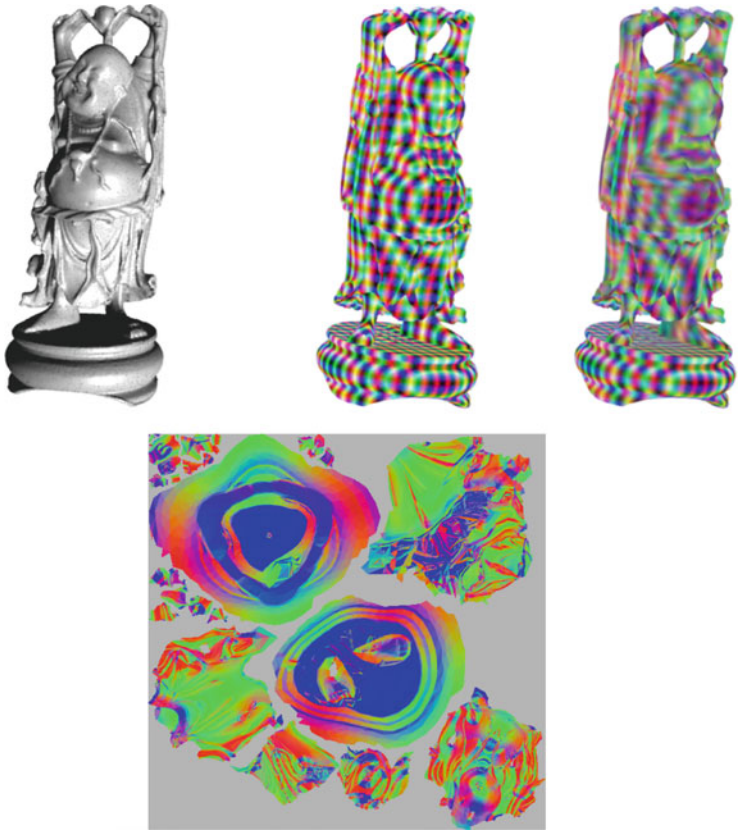

Fig. 1 Distortion in the parameterization (*bottom*) leads to uneven sampling rates on the surface and blurring for regions in the surface that have received less-than-average areas in the texture domain (e.g., Buddha's face and torso). This in turns leads to the loss of details in the texture patterns (*top: right*). Compare it with the true signal (*left*). Notice that the base on which the Buddha stands has a larger-than-average share of the texture map and therefore has preserved finer details in the texture map

domain, one has to use a texture map with a rather large size, which leads to slower performance during rendering. Given that regions in the surface receiving more than a fair share of the parameterization space can already achieve sufficiently high visual quality even with smaller-sized texture maps, increasing the size of the texture map is essentially a waste for such regions. See Fig. 1 for one such example. Next, we will review the cause of distortion and means to reduce distortion.

Consider a surface S represented as a triangular mesh and a parameterization τ that maps every triangle t_i in S to some triangle $\tau(t_i)$ in the plane. Note that sometimes the inverse map of τ, denoted by $\delta = \tau^{-1}$, is referred to as the parameterization for S. We will stay consistent with literature in computer graphics

in which τ is considered the parameterization (also known as texture coordinates). However, our discussion on measuring distortion will be based on δ.

In graphics hardware, a triangle is often the unit of processing. For texture mapping, the signals inside a triangle in the texture domain will be lifted to the corresponding triangle on the surface S through δ as follows:

$$\delta(f_a\tau(v_a) + f_b\tau(v_b) + f_c\tau(v_c)) = f_a(v_a) + f_b(v_b) + f_c(v_c) \tag{1}$$

in which v_a, v_b, and v_c are the vertices of the triangle t in S, and f_a, f_b, and f_c are the barycentric coordinates of points inside t. Notice that δ is piecewise linear which means the distortion is constant over each triangle. To measure distortion, let us assume the total area of the surface is equal to the total area in the texture map. Under this assumption, it is easy to see that a triangle $t \in S$ has a zero stretch only when t is isomorphic to $\tau(t)$. However, when the two triangles are not identical, it is not immediately clear how to measure their difference, i.e., the distortion. For example, the two triangles may be similar (same angle distributions and different areas), or have the same area but different angle distribution, or both. Should we penalize angle distortions more than area distortions, or the other way around? How much of area distortion would be considered equivalent to angle distortion? When mapping t to a similar triangle $\tau(t)$, how to measure distortion when $\tau(t)$ is larger than t and vice versa? All of these questions trace back to a fundamental problem: the distortion is not a scalar, but a tensor.

Let us examine Eq. 1 more closely. Given a triangle $t \in S$, δ_t, the restriction of δ on t, is a bijective linear map. Consider two mutually perpendicular, unit vectors w_1 and w_2 in $\tau(t) \subset \mathbb{R}^2$. The squared length of w_i is given by $|w_i|^2 = w_i \cdot w_i = 1$ for $i = 1, 2$. The angle between the vectors is related to the dot product $w_1 \cdot w_2 = 0$. Finally, the squared area of the parallelogram spanned by w_1 and w_2 is given by $|w_1 \times w_2|^2 = (w_1 \cdot w_1)(w_2 \cdot w_2) - (w_1 \cdot w_2)^2 = 1$.

We are interested in similar quantities for $\delta_t(w_i)$ because the difference in these quantities can give us the distortion in distance, angle, and areas, respectively. Note that δ_t can be represented as a 3×2 matrix M_t that maps a vector w in the plane to a vector $M_t w$ in $\mathbb{R}^3$. The squared length of $\delta(w_i)$ is

$$|\delta_t(w_i)|^2 = (M_t w_i \cdot M_t w_i) = w_i'(M_t' M_t) w_i \tag{2}$$

Notice that the difference between the lengths of w_i and $\delta_t(w_i)$ can only be attributed to the symmetric matrix $G_t = M_t' M_t$. It is straightforward to verify that G_t is also responsible for the angular and areal distortions. G, the tensor field whose restriction to a triangle t is G_t, is referred to as the *metric tensor*. In the ideal setting, i.e., when a triangle $t \in S$ and $\tau(t)$ are isometric, G_t is the identity matrix Id. A parameterization that satisfies this for every triangle is referred to as *isometric parameterization*. However, when distortion is present, how to measure distortion from the metric tensor is a challenging topic. Various measures have been proposed, with respect to which the parameterization algorithms have been optimized. A popular approach is to strive for *conformal parameterizations* which

preserve angles between any pair of vectors. Another possible criterion is *equiareal parameterizations* which preserve areas of the parallelograms spanned by any vector pair. The metric tensor G corresponding to the conformal parameterization and equiareal parameterization satisfies $G_t = k_t Id$ and $det(G_t) = 1$, respectively, for every triangle t. Note that G is isometric if and only if G is both conformal and equiareal. The eigenvalues of G_t correspond to the largest and smallest squared lengths of any unit planar vector under the map δ_t, while the eigenvectors of G_t correspond to directions in which these lengths are achieved. Usually only the eigenvalues are considered important for texture mapping purposes. In terms of the eigenvalues $\sigma_{t,1} \geq 0$ and $\sigma_{t,2} \geq 0$, G_t is isometric, conformal, and equiareal if $\sigma_{t,1} = \sigma_{t,2} = 1$, $\sigma_{t,1} = \sigma_{t,2}$, and $\sigma_{t,1}\sigma_{t,2} = 1$, respectively.

Various measures have been proposed based on $\sigma_{t,1}$ and $\sigma_{t,2}$, essentially providing a tradeoff between conformal and equiareal parameterizations. Example measures include

1. $\sqrt{(\sigma_{t,1} + \sigma_{t,2})(1/\sigma_{t,1} + 1/\sigma_{t,2})}$ [11]
2. $\frac{1}{2}(\sqrt{\sigma_{t,1}} - \sqrt{\sigma_{t,2}})^2$ [6, 20]
3. $\sqrt{\sigma_{t,1} + \sigma_{t,2}}$ [26]
4. $(\sigma_{t,1} - 1)^2 + (\sigma_{t,2} - 1)^2$ [32]

Note that the first two energy formulations from the above consider conformal parameterization as the ideal case, thus ignoring areal distortion. The third energy strikes a balance between conformal and equiareal parameterization. However, in some cases when there are distortions, this energy evaluates to the same as the isometry, such as when $\sigma_{t,1} = 1 + u$ and $\sigma_{t,2} = 1 - u$ for $0 < u < 1$. The fourth energy is minimized if and only if the map is isometric. There are other energy terms not defined directly in terms of the metric tensors, such as the angle-based flattening measure [27] which strive for conformal parameterization.

More recently, there has been work on parameterizing a genus zero triangular mesh surface over a sphere, with applications in remeshing. The general approach is to construct a compatible partition of the mesh surface and the sphere, such as an octahedral partitioning. This allows the sphere and the mesh surface to be mapped onto a common planar domain D. The spherical parameterization is then the composite of the parameterization of the sphere over D and the inverse map of the parameterization of the mesh surface. In this case the metric tensor from the sphere to the mesh surface is also the composition of the metric tensors of the sphere and the mesh surface with respect to D, and the distortions between the sphere and the mesh can again be measured using distortions in the planar parameterizations.

While the original and still primary use of the surface parameterization (and the metric tensor) is in texture mapping, additional applications have been identified, such as fluid simulation on surfaces [28], texture synthesis [24], surface compression [9], and triangular remeshing [2]. With surface parameterization, the computation on mesh surfaces can be transferred into similar (but typically simpler) computations in the plane. The ability to measure distortion in the parameterization is key to achieving desired results as it needs to be reversed when performing the computation in the plane.

Surface parameterization has received much attention from the geometry processing community. However, the focus has been on some derived scalar quantity from the eigenvalues of the metric tensor per triangle. Eigenvectors are typically not considered, and the use of the tensors are isolated, i.e., per triangle. In this sense it is hardly treated as a tensor field, which has structures unique to it. In the next section, we will examine another popular tensor in the geometry processing community, the curvature tensor.

3 Curvature Tensor

The curvature tensor describes the bending of the surface. It has been used in various graphics applications such as non-photorealistic rendering and geometry remeshing.

First, let us consider the curvature of a planar curve γ. Suppose a person is travelling along γ at a constant speed, e.g., the speed limit. Due to the bending of the curve, the person constantly changes his forward direction. However, since he is travelling at a constant speed, the change in his travel direction at any given moment must be perpendicular to the forward direction at the moment. The curvedness of the road can be measured by how sharply the traveler turns. More formally, let $\gamma(s)$ be parameterized by arc length s (equivalent to the milemarkers along the road), the forward travel direction is the tangent to the curve, i.e., $T(s)$. Since the traveler travels at a constant speed $C > 0$, we have $|T(s)|^2 \equiv C^2$. Differentiating both sides with respect to s results in

$$T(s) \cdot N(s) = 0 \tag{3}$$

where $N(s) = T'(s)$ is the normal to the curve. The *signed curvature* at a point $\gamma(s)$ is given by $\kappa(s) = T(s) \times N(s)$.

Let us now consider surfaces. Given a smooth surface $S \subset \mathbb{R}^3$, there are infinitely many curves passing through any point $\mathbf{p} \in S$. Moreover, they may have different curvatures at $\mathbf{p}$. Fortunately results from classical differential geometry state that the curvatures are not a function of individual curves, but of the tangent space at $\mathbf{p}$ [8]. More formally, two curves γ_1 and γ_2 in S have the same curvature at $\mathbf{p}$ if they have the same tangent vectors at $\mathbf{p}$. In addition, given a tangent vector v at $\mathbf{p}$, the curvature corresponding to v is a quadratic function $\kappa(v) = v'Kv$ where K is a symmetric matrix known as the curvature tensor. The eigenvalues of K are referred to as *principal curvatures*, and eigenvectors as *principal directions*. The larger eigenvalue κ_1 is referred to as the *major eigenvalue*, and the smaller eigenvalue κ_2 as the *minor eigenvalue*. Their corresponding eigenvectors are referred to as the major and minor principal directions, respectively.

Unlike the metric tensor which is positive definite, the curvature tensor can have negative eigenvalues. Applying the well-known isotropic-deviatoric decomposition to the diagonalized curvature tensor results in

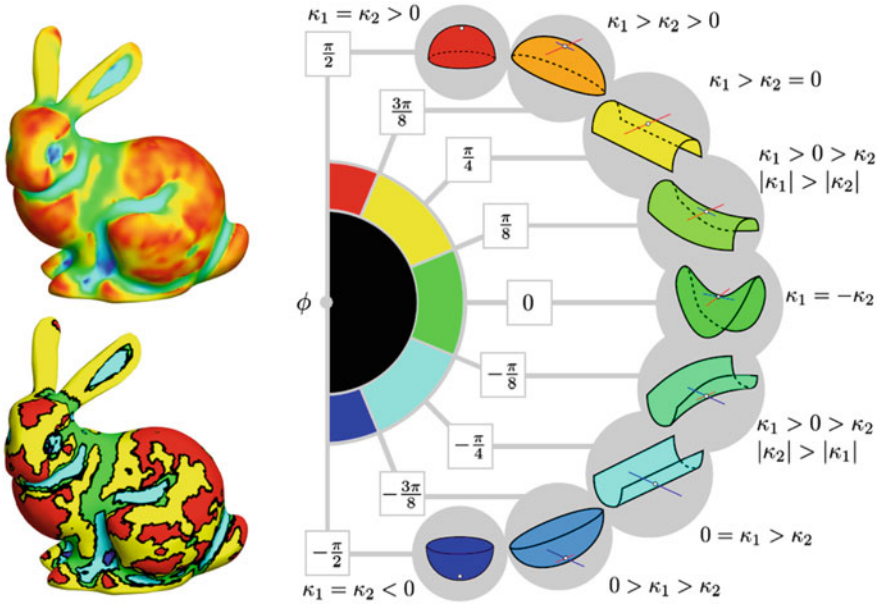

Fig. 2 Surface classification scheme based on the shape index $\phi \in [\pi/2, \pi/2]$ is color mapped to the (*blue, red*) arc in HSV color space: *Left top*: continuous mapping. *Bottom*: binned classification. The legend (*right*) shows surfaces patches which are locally similar to points with given values (This figure is a courtesy of [22], ©2012 IEEE)

$$\begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix} = \frac{\kappa_1 + \kappa_2}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{\kappa_1 - \kappa_2}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{4}$$

Let us consider the vector $\begin{pmatrix} \frac{\kappa_1 - \kappa_2}{2} \\ \frac{\kappa_1 + \kappa_2}{2} \end{pmatrix}$ and write it in the polar form:

$$\rho = \sqrt{(\frac{\kappa_1 - \kappa_2}{2})^2 + (\frac{\kappa_1 + \kappa_2}{2})^2} = \frac{\sqrt{\kappa_1^2 + \kappa_2^2}}{\sqrt{2}} \tag{5}$$

$$\phi = \tan^{-1}(\frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2}) \tag{6}$$

ρ is the tensor magnitude of the curvature tensor and is zero only when the surface is locally planar. It is referred to as the *curvedness*. Recall that $\kappa_1 \geq \kappa_2$. Consequently, ϕ is well defined and satisfies $-\pi/2 \leq \theta \leq \pi/2$. It is referred to as the *shape index* [18]. Figure 2 illustrates the power of this decomposition with the classification over the bunny surface. $H = \frac{\kappa_1 + \kappa_2}{2}$ is referred to as the *mean curvature*, which is not only the average between the principal curvatures, but the average of the curvatures corresponding to the set of all unit tangent vectors.

The quantity $\frac{\kappa_1-\kappa_2}{2}$ shows the anisotropy in the curvature. It achieves minimal value only in spherical regions in the surface. Note that in the discussion of the shape index we do not consider planar regions since it is where the curvedness is zero and can therefore be considered as having any shape index. The quantity $G = (\frac{\kappa_1+\kappa_2}{2})^2-(\frac{\kappa_1-\kappa_2}{2})^2 = \kappa_1\kappa_2$ measures the relative strength between the isotropic and anisotropic parts of the curvature tensor. It is referred as the *Gaussian curvature*. If positive, the point of interest is more isotropic than it is anisotropic, i.e., elliptical. If negative, the point is more anisotropic, i.e., hyperbolic. When zero, the point is cylindrical. Results from classical differential geometry states that the total Gaussian curvature over a closed two-dimensional manifold S with g handles is $2\pi\chi(S)$, where $\chi(S) = 2 - 2g$ is the *Euler characteristic* of S [8].

We now return to the discussion of the eigenvectors of the curvature tensor. Recall that the eigenvectors of the metric tensor do not play a prominent role in surface parameterization. This is not the case for the curvature tensor in graphics, as the principal curvature directions are important geometric characterization of the surface. In pen-and-ink sketching, Interrante [12] has shown that curves following the principal curvature directions can better illustrate a shape in visualization. Moreover, artists typically draw hatches along principal curvature directions despite not having necessary mathematical background in differential geometry.

We define a *major hyperstreamline* as a curve whose tangent coincides with the major principal curvature directions everywhere along its path. A *minor hyperstreamline* can be defined in a similar fashion. Major and minor hyperstreamlines must intersect perpendicularly, since the major and minor principal directions at a point in the surface are mutually perpendicular. However, eigenvectors are not well-defined at points where $\kappa_1 = \kappa_2$, i.e., spherical points. Such points are referred to as the *umbilical points*, which are the equivalent of singularities in vector fields. An umbilical point can be measured in terms of the local tensor field behavior around it. More specifically, consider an isolated umbilical point $\mathbf{p}_0$ which has a neighborhood inside which no other umbilical points exist. Assume this neighborhood N is a topological disk. When travelling along the boundary of the neighborhood ∂N, the normalized eigenvectors along the curve are also travelling on the Gauss circle S^1. Due to tensor field continuity, when one finishes travelling ∂N once, the eigenvectors must have also travelled the Gauss circle a number of times. However, due to sign ambiguity in the eigenvectors, it is possible that the eigenvectors have travelled only half of the circle instead of the full circle. It can be shown that in general the eigenvectors must have travelled $\frac{L}{2}$ times around the Gauss circle where L is an integer. Moreover, L is independent of the size and shape of the neighborhood N as long as N does not contain additional umbilical points beyond $\mathbf{p}_0$ in its interior or on its boundary. Consequently, $\frac{L}{2}$ is considered as the *index* of $\mathbf{p}_0$ and is denoted by $I(\mathbf{p}_0)$. Note that the index is zero if and only if $\mathbf{p}_0$ is not an umbilical point.

The two most fundamental types of umbilical points are wedges (index $1/2$) and trisectors (index $-1/2$). Interestingly, they correspond to the two simplest ways of reversing travel directions when driving a car: U-turn (wedge) and three-point-turn

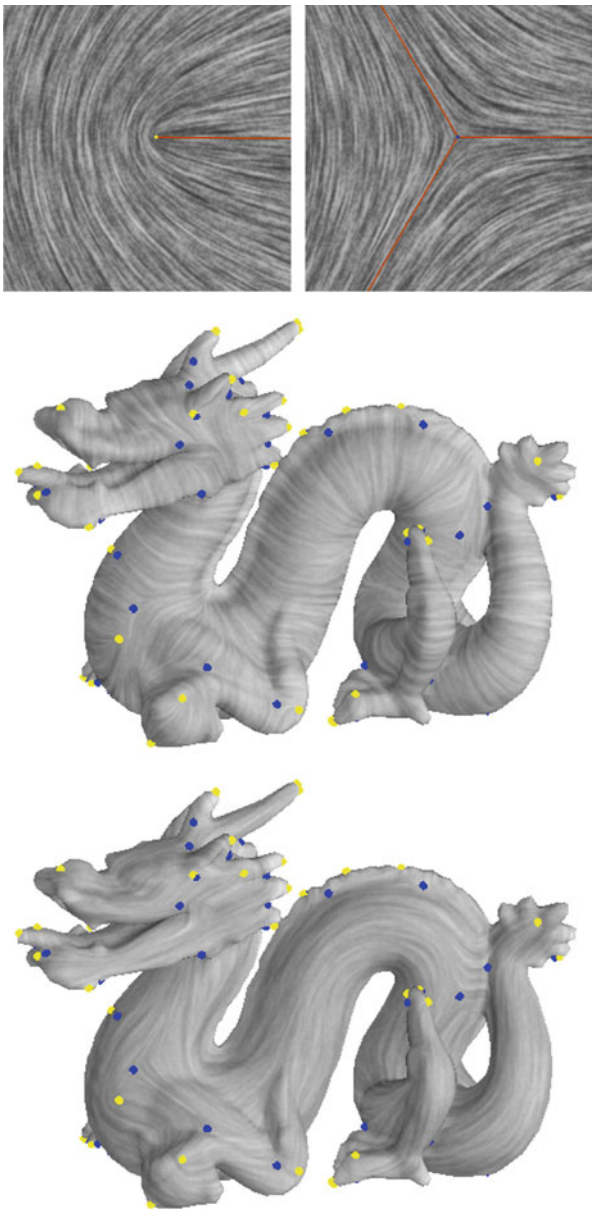

Fig. 3 Two most fundamental umbilical points in the curvature tensor: (*top-left*) wedge, and (*top-right*) trisector. Umbilical points appear in natural locations in shapes (*middle* and *bottom*): wedges in *yellow* and trisectors in *blue*. Shown in the *bottom* are also the major hyperstreamlines (*middle*) and minor hyperstreamlines (*bottom*)

(trisector). See Fig. 3 (top). In addition, umbilical points appear in natural locations in the surface (Fig. 3 (bottom)).

Given a closed, two-dimensional manifold S whose umbilical points are all isolated, Delmarcelle and Hesselink [5] show that

$$2\pi \sum_{J} I(\mathbf{p}_i) = \chi(S) \tag{7}$$

where J is the set of the umbilical points in S. It is interesting to note that the distribution of two seemingly unrelated quantities from the curvature tensor, i.e., the Gaussian curvature and the index of the umbilical points, are both constrained by the topology of the underlying surface. The curvature tensor can be computed using a number of methods [3, 21, 23, 25]. We refer interested readers to these papers for details.

Next we consider some graphics and geometry applications in which the curvature tensor plays a prominent role.

3.1 Non-photorealistic Rendering

Pen-and-ink sketching is a well-researched topic in Non-Photorealistic Rendering (NPR). In a typical setting, a set of lines (hatches), usually monochromatic (e.g., black) are placed against a background (typically white). The locations and densities of the lines are used to outline the shapes, present the main features in the objects, and convey the shading. An NPR system must determine the location, orientation, and density of hatches. Most existing automatic hatching algorithms differ in how they extract some or all of this information.

3.1.1 Feature Line Drawing

The first class of algorithms extract line features from the shapes and highlight these lines. These classes of algorithms have direct application in engineering illustration and CAD and medical data visualization. Consequently, the lines are typically chosen to reflect the underlying geometry without the effect of shading. The difference among this class of algorithms lies in how line features are defined and extracted. We will review some of line definitions that are most relevant to the curvature tensor.

The most commonly used lines in line drawing are perhaps *visible contours*. Line drawing based only on visible contours is sometimes referred to as *silhouette drawing*. Notice that contours are view-dependent. When the viewpoint is changed, the set of contour points is also changed. However, intrinsic features in a mesh, like sharp edges, are not guaranteed to be part of the contour for any appropriate viewpoint. This has led to view-independent line features, such as ridges and valleys which can be defined in terms of the curvature tensor.

A point $\mathbf{p} \in S$ is a *ridge point* if the following conditions are met:

1. $\kappa_1 > |\kappa_2|$, i.e., the absolutely maximal bending occurs in the major principal curvature direction.

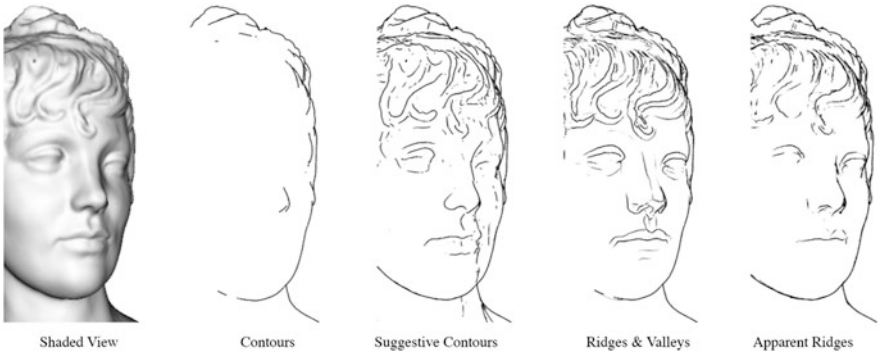

Fig. 4 A comparison of a number of feature-based drawing techniques (This figure is a courtesy of [14], ©2007 ACM)

2. $\mathbf{p}$ is a local maxima of κ_1 on the unique major hyperstreamline containing $\mathbf{p}$

Similarly, a point $\mathbf{p} \in S$ is a *valley point* if

1. $-\kappa_2 > |\kappa_1|$, i.e., the absolutely maximal bending occurs in the minor principal curvature direction, and
2. $\mathbf{p}$ is a local minima of κ_2 on the unique minor hyperstreamline containing $\mathbf{p}$

We will omit the details for computing ridges and valleys here and instead refer interested readers to [23, 25] (Fig. 4).

It has been noted that visible contours often do not produce sufficient lines that reveal the underlying geometry. On the other hand, ridges and valleys are not view-independent. To address these difficulties, the concepts of *suggestive contours* [4], *apparent ridges* [14], and *demarcating curves* [19] are introduced. Both suggestive contours and apparent ridges are view-dependent, while demarcating curves are view-independent.

Roughly speaking, points on suggestive contours given a viewpoint V are not part of the contours with respect to V, but contours with respect to some nearby viewpoint. More formally, a point $\mathbf{p} \in S$ is on the suggestive contour with respect to viewpoint V if

$$D_{\mathbf{w}}\kappa_r > 0 \tag{8}$$

where $\mathbf{w}$ is the projection of $V - \mathbf{p}$ onto the tangent plane at $\mathbf{p}$, κ_r is the curvature at $\mathbf{p}$ in the direction $\mathbf{w}$, and $D_{\mathbf{w}}\kappa_r$ is the derivative of κ_r in the direction of $\mathbf{w}$. Equivalently, a point is on the suggestive contour if it is a local minima of $N \cdot (V - \mathbf{p})$. The computation of suggestive contours can be performed in both object-space and image-space, leading to different tradeoffs between accuracy and numerical stability.

Another view-dependent feature measure is *apparent ridges*. Apparent ridges differ from ridges as the latter is derived from the curvature tensor, while the former

from the *projected curvature tensor* onto the image space. Due to the distortion in orthographic and perspective projections, the projected curvature tensor differs from the curvature tensor. Apparent ridges are therefore the ridges extracted from the projected curvature tensor. One of the main motivations behind apparent ridges is to capture places where luminance would change rapidly should the model be shaded. This notion is in a way related to the idea of suggestive contours as the latter also tries to capture lines that are not features in the current but nearby viewpoints.

Finally, demarcating curves are considered in addition to ridges and valleys. Demarcating curves are transition points between ridges and valleys, much in the same sense as that *inflection points* of a function f ($f'' = 0$) are the transition between local maxima ($f' = 0$ and $f'' < 0$) and local minima ($f' = 0$ and $f'' > 0$). More formally, the gradient of the curvature tensor, C_{ijk}, is a third-order tensor. Given $v = v^i$, a unit tangent vector, $C_{ijk}v^i v^j v^k$ gives the rate of curvature change in v. The direction in which this change is the largest is defined as the *direction for maximal normal curvature variation*. A point $\mathbf{p}$ is on the demarcating curve if $v_{\mathbf{p}}^T K v_{\mathbf{p}} = 0$ where $v_{\mathbf{p}}$ is the direction for maximal normal curvature variation at $\mathbf{p}$.

While feature lines have been a powerful tool in illustrating shapes, they are not often well-suited for surfaces that lack lines features, such as smooth surfaces like a cylinder. In addition, feature-based line drawing primarily aims to revealing geometric features rather than shading effects. Hatching is a more suitable alternative in these cases, which we review next.

3.1.2 Illustration of Smooth Surfaces

Drawing as a form of art often captures our attention in terms of the shading effect and varying geometric details, even when there are few geometric features (such as ridges and valleys) present in the shape. For example, consider a cylinder of an infinite height. Line features as described in Sect. 3.1.1 can only capture the silhouette of the cylinder, i.e., two straight lines. It would be difficult, without drawing the caps of the cylinder, to realize that it is a cylinder.

Hatching provides a nice alternative way of illustrating the shapes in this case. Hatches are used to present the shading effect as well as the internal bending in the geometry, thus making it possible to understand the geometry of an otherwise featureless surface. The key issues for hatching are:

1. How to use hatches to present geometric details?
2. How to use hatches to provide shading effects?

Interrante [12] shows that when hatches follow the principal curvature directions, the shape is best illustrated. Moreover, the lighting effect can be simulated with the density of hatches. Denser hatches indicates lower luminance, while lower or no hatches indicates bright spots or highlight under the viewing condition. Cross-hatching, i.e., drawing two families of mutually perpendicular lines can increase the darkness of a region without increasing the density of the hatches. Consequently,

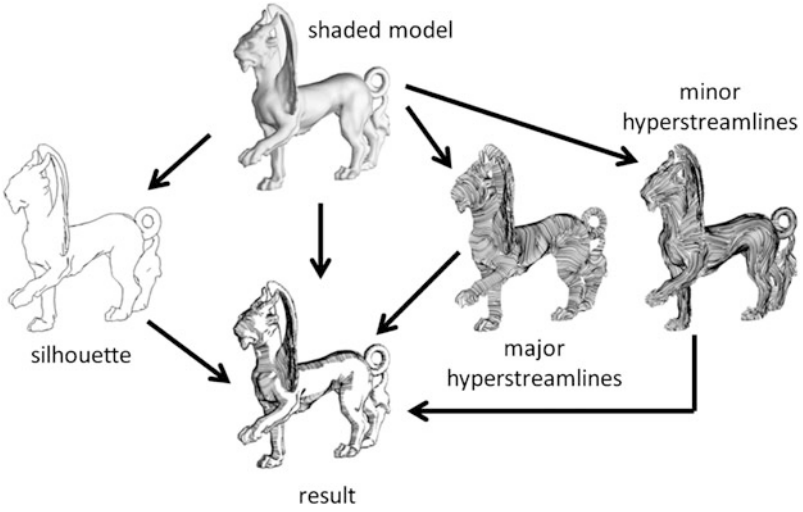

Fig. 5 The process of generating a hatch-based drawing from an input model

cross-hatched regions are often reserved for deep shadows, while single-hatched regions (only one family of hatches) are used for regions that are neither in deep shadows nor in the highlight (Fig. 5).

As mentioned earlier, the principal curvature directions can be computed using a number of methods [3, 21, 23, 25]. Once the curvature tensor field has been computed, a family of evenly spaced hyperstreamlines can be generated for the major principal curvature field and the minor principal curvature field, respectively. Generating evenly spaced hyperstreamlines can be achieved by adapting techniques generating evenly spaced streamlines in vector fields [13]. These two families of hyperstreamlines will be used to generate two images, one for the major and the other for the minor hyperstreamlines. We refer to these two images as I_1 and I_2, respectively. In addition, an image I_3 based on visible contours are also generated. Furthermore, a pixel with a value of 1 is white and a value of 0 is black. These three images will then be composed into a single image as follows:

$$I(p) = \begin{cases} 0 & \text{if } I_3(p)=0 \\ 1 & \text{if in highlight} \\ min(I_1(p), I_2(p)) & \text{if in shadow} \\ I_2(p) & \text{otherwise} \end{cases} \tag{9}$$

Note that in the above one can also choose to always use $I_1(p)$ for single-hatched regions. Another means of generating I_1 and I_2 is to first project the principal directions onto the image plane and trace hyperstreamlines in the image plane. This alternative is view-dependent but typically is fast enough for interactive applications. In contrast, the object-based approach requires much time for

pre-processing but is then ready for interactive display, except when the resolution is changed.

3.2 Quadrangular and Triangular Remeshing

The curvature tensor has also been used in geometry remeshing, which refers to generating a new mesh from an input mesh (typically triangular) subject to some optimization criteria. If the output mesh is also triangular, the process is referred to as *triangular remeshing*. On the other hand, if the output is a quad mesh, the process is referred to as *quadrangular remeshing*.

Classical triangular remeshing approaches place a set of points on the input mesh surface according to some density constraints [2,10,29,30]. These points will be the vertices of the output mesh. The density of the points is usually required to reflect the geometry in the input. Consequently, some forms of curvature (mean curvature, Gaussian curvature, etc.) are considered as part of the density function. Delauney triangulation or centroidal Voronoi tessellation can then be performed on the point set to generate the triangulation.

Alliez et al. [1] revitalize the topic of quadrangular remeshing. In their pioneering work, a quad-dominant mesh is generated by intersecting one family of evenly spaced major hyperstreamlines with one family of evenly spaced minor hyperstreamlines. Since the two families intersect at the right angle, the resulting quad-dominant mesh consists of mostly nice rectangles. Moreover, the edges in the mesh follow the principal curvature directions and therefore have low approximation error.

Due to numerical issues, there are usually an excessive number of umbilical points in the output mesh, leading to a large number of irregular vertices, i.e., the valency is not four. Irregular vertices, especially when misplaced, can lead to difficulties in subsequent mesh processing. Zhang et al. [31] introduce operations to control the number and location of umbilical points by cancelling a pair of umbilical points with opposite tensor indexes, or by moving an umbilical point to a more appropriate location. We refer to [31] for details.

To remove T-junctions, which occur when the tracing of a hyperstreamline stops without reaching an umbilical point, Kälberer et al. [15, 16] make use of the mathematical concept of *covering space* and convert the tensor field to a vector field in the covering space. The vector field is then made curl-free through the Hodge decomposition, and the quadrangulation is performed in the covering space which nicely maps to a pure quad mesh (no T-junctions) in the original input mesh.

Nieser et al. [22] use a similar idea (covering space) for triangulation. Unlike quadrangular remeshing, in which both the major and minor principal directions can be used as edges in the remeshed quad mesh, in triangular remeshing at most one can be used. To deal with this they make use of the shape index and ensure that the edges in the triangles will be aligned with the minor eigenvector directions in ridge-like regions and the major eigenvector directions in valley-like regions.

4 Conclusion

Tensor fields are important to not only scientific visualization and medical imaging, but also computer graphics and geometry processing. In this chapter we review two of the most popular tensors, the metric tensor and the curvature tensor, with applications in surface parameterization, non-photorealistic rendering, and remeshing. We expect that more graphics and geometry applications will be identified for tensor fields, and we believe that research in tensor fields can continue to benefit the visualization, image processing, and graphics communities.

References

1. Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., Desbrun, M.: Anisotropic polygonal remeshing. ACM Trans. Graph. (SIGGRAPH 2003) **22**(3), 485–493 (2003)
2. Alliez, P., Meyer, M., Desbrun, M.: Interactive geometry remeshing. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, San Antonio, pp. 347–354. ACM, New York (2002). doi:10.1145/566570.566588, http://doi.acm.org/10.1145/566570.566588
3. Cohen-Steiner, D., de Verdière, É.C., Yvinec, M.: Conforming delaunay triangulations in 3d. In: Symposium on Computational Geometry, Barcelona, pp. 199–208 (2002)
4. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. In: ACM SIGGRAPH 2003 Papers, SIGGRAPH'03, San Deigo, pp. 848–855. ACM, New York (2003). doi:10.1145/1201775.882354, http://doi.acm.org/10.1145/1201775.882354
5. Delmarcelle, T., Hesselink, L.: The topology of symmetric, second-order tensor fields. In: IEEE Visualization Conference, pp. 140–147 (1994)
6. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic parameterizations of surface meshes. Comput. Graph. Forum **21**(3), 209–218 (2002)
7. Floater, M.S., Hormann, K.: Surface parameterization: a tutorial and survey. In: Advances in Multiresolution for Geometric Modelling, pp. 157–186. Springer, Berlin (2005)
8. Gray, A.: Modern Differential Geometry of Curves and Surfaces with Mathematica, 1st edn. CRC, Boca Raton (1996)
9. Gu, X., Gortler, S.J., Hoppe, H.: Geometry images. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'02, Bristol, pp. 355–361. ACM, New York (2002). doi:10.1145/566570.566589, http://doi.acm.org/10.1145/566570.566589
10. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: SIGGRAPH, Anaheim, pp. 19–26 (1993)
11. Hormann, K., Greiner, G.: MIPS: an efficient global parametrization method. In: Laurent, P.J., Sablonnière, P., Schumaker, L.L. (eds.) Curve and Surface Design: Saint-Malo 1999, Innovations in Applied Mathematics, pp. 153–162. Vanderbilt University Press, Nashville (2000)
12. Interrante, V.: Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution. In: SIGGRAPH'97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, pp. 109–116. ACM/Addison-Wesley, New York (1997)
13. Jobard, B., Lefer, W.: The motion map: efficient computation of steady flow animations. In: IEEE Visualization, Phoenix, pp. 323–328 (1997)

14. Judd, T., Durand, F., Adelson, E.: Apparent ridges for line drawing. In: ACM SIGGRAPH 2007 Papers, SIGGRAPH'07, San Diego. ACM, New York (2007). doi:10.1145/1275808.1276401, http://doi.acm.org/10.1145/1275808.1276401
15. Kälberer, F., Nieser, M., Polthier, K.: Quadcover – surface parameterization using branched coverings. Comput. Graph. Forum **26**(3), 375–384 (2007)
16. Kälberer, F., Nieser, M., Polthier, K.: Stripe parameterization of tubular surfaces. In: Pascucci, V., Hagen, H., Tierny, J., Tricoche, X. (eds.) Topological Methods in Data Analysis and Visualization. Theory, Algorithms, and Applications., Mathematics and Visualization. Springer, Berlin/Heidelberg (2010)
17. Kang, S.B.: A survey of image-based rendering techniques. In: In Videometrics, SPIE, pp. 2–16. Digital, Cambridge Research Laboratory, Cambridge (1999)
18. Koenderink, J.J., van Doorn, A.J.: Surface shape and curvature scales. Image Vision Comput. **10**, 557–565 (1992)
19. Kolomenkin, M., Shimshoni, I., Tal, A.: Demarcating curves for shape illustration. In: ACM SIGGRAPH Asia 2008 papers, SIGGRAPH Asia'08, Singapore, pp. 157:1–157:9. ACM, New York (2008). doi:10.1145/1457515.1409110, http://doi.acm.org/10.1145/1457515.1409110
20. Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'02, Bristol, pp. 362–371. ACM, New York (2002). doi:10.1145/566570.566590, http://doi.acm.org/10.1145/566570.566590
21. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Proc. of the Int. Workshop on Visualization and Mathematics, pp. 35–57 (2002)
22. Nieser, M., Palacios, J., Polthier, K., Zhang, E.: Hexagonal global parameterization of arbitrary surfaces. IEEE Trans. Vis. Comput. Graph. **18**(6), 865–878 (2012). doi:10.1109/TVCG.2011.118, http://dx.doi.org/10.1109/TVCG.2011.118
23. Ohtake, Y., Belyaev, A., Seidel, H.P.: Ridge-valley lines on meshes via implicit surface fitting. ACM Trans. Graph. **23**(3), 609–612 (2004). doi:10.1145/1015706.1015768, http://doi.acm.org/10.1145/1015706.1015768
24. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: SIGGRAPH'00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, pp. 465–470. ACM/Addison-Wesley, New York (2000)
25. Rusinkiewicz, S.: Estimating curvatures and their derivatives on triangle meshes. In: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, 3DPVT'04, Thessaloniki, pp. 486–493. IEEE Computer Society, Washington (2004). doi:10.1109/3DPVT.2004.54, http://dx.doi.org/10.1109/3DPVT.2004.54
26. Sander, P.V., Snyder, J., Gortler, S.J., Hoppe, H.: Texture mapping progressive meshes. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'01, Los Angeles, pp. 409–416. ACM, New York (2001). doi:10.1145/383259.383307, http://doi.acm.org/10.1145/383259.383307
27. Sheffer, A., Lévy, B., Mogilnitsky, M., Bogomyakov, A.: Abf++: fast and robust angle based flattening. ACM Trans. Graph. **24**(2), 311–330 (2005). doi:10.1145/1061347.1061354, http://doi.acm.org/10.1145/1061347.1061354
28. Stam, J.: Flows on surfaces of arbitrary topology. ACM Trans. Graph. (SIGGRAPH 2003) **22**(3), 724–731 (2003)
29. Turk, G.: Re-tiling polygonal surfaces. In: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'92, New York, pp. 55–64. ACM, New York (1992). doi:10.1145/133994.134008, http://doi.acm.org/10.1145/133994.134008
30. Yan, D.M., Lévy, B., Liu, Y., Sun, F., Wang, W.: Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In: Proceedings of the Symposium on Geometry Processing, SGP'09, Berlin, pp. 1445–1454. Eurographics Association, Aire-la-Ville (2009). http://dl.acm.org/citation.cfm?id=1735603.1735629

31. Zhang, E., Hays, J., Turk, G.: Interactive tensor field design and visualization on surfaces. IEEE Trans. Vis. Comput. Graph. **13**(1), 94–107 (2007)
32. Zhang, E., Mischaikow, K., Turk, G.: Feature-based surface parameterization and texture mapping. ACM Trans. Graph. **24**, 1–27 (2005)