5 CURVATURE OF SMOOTH AND DISCRETE SURFACES

After many chapters of preliminaries, we *finally* reach a topic that is a staple of pure and applied geometry: measuring the curvature of a surface. Here, we explore famous models characterizing local surface geometry, quantifying how a two-dimensional surface embedded in $\mathbb{R}^3$ cuts through space. Once again we turn to our discussion of curves in Chapter 3 for preliminaries and inspiration; we will show that the curvature of a surface at a point is closely linked to the curvature of curves along the surface cutting through that point.

While the basic measures of surface curvature date back to eighteenth century mathematics, we add a modern twist by considering algorithms for estimating curvature. After exploring the local theory of smooth surfaces, we show how curvature can be measured on a triangulated surface. We will see that there is no broad consensus for this task: While mathematicians agree that Gaussian and mean curvature are the most relevant measures of smooth bending and stretching and broadly use identical definitions of these quantities (up to a constant factor, at least), many models of curvature on triangulated surfaces are used in practice.

5.1 THINKING ABOUT CURVATURE

Imagine modeling a surface in 3D out of a thin sheet of plastic. Our sheet of plastic is thin enough to be malleable, allowing us to form it into any shape we wish by deforming it in different ways. As illustrated in Figure REF, there are roughly two ways we could mold the plastic into our desired shape:

1. First, we could *bend* the surface. As an extreme example, maybe replace the plastic with a strong ("inextensible") piece of paper, which does not tear but can fold. This bending motion does not affect the length of a curve drawn along the surface: If we draw a curve on a sheet of paper and then fold the paper up, the folded curve has the same arc length as the original. On the other hand, the normal vector to the surface changes considerably under bending motion; adjacent panels on what used to be the flat surface now have varying normal directions. Bending of course does not have to create a sharp crease, as evidenced by a flag smoothly flapping in the wind.

2. Second, we could *stretch* the surface. For example, if we wanted to form a (hyperbolic) potato chip or an (elliptic) egg out of our sheet of plastic, we would have to stretch or compress the material. This deformation—unlike bending—affects distances along the surface. Molding a flat sheet into a saddle-shaped potato chip requires stretching out the material wider and wider as we move farther from the saddle point, while molding the same sheet into an egg requires compressing points together.

These two modes of deformation, bending and stretching, are captured by the two most popular means of measuring the curvature of a surface at a given point, the *mean* and *Gaussian* curvature. Mean curvature measures how the normal of a surface changes as we move along from one point to another, while the Gaussian curvature measures whether a surface is stretched or compressed relative to a flat sheet.

As we will see, much like curvature and torsion for curves (see §3.6), the geometry of a surface is completely determined by its mean and Gaussian curvature functions. Our intuitive descriptions above also align with theorems in differential geometry. For instance, Gauss's Theorema Egregium[1] states that an ant crawling along a surface can measure Gaussian curvature, that is, it

1 Roughly translated from Latin, "Totally Awesome Theorem."

is an *intrinsic* function of distances along the surface without knowledge of bending or the normal vector.

These intuitions also play out in how we discretize curvature on a triangulated surface. As a preview, the simplest versions of mean and Gaussian curvatures on a triangle mesh are shown in Figure REF. Mean curvature along an edge adjacent to triangles is captured by the *dihedral angle* between the two triangles, treating the two triangles together as an origami sheet that folds along their shared edge; this flapped bending motion can be accomplished locally without modifying the lengths of the triangle edges. Gaussian curvature at a vertex, on the other hand, becomes *angle deficit*, that is, the difference between the sum of angles adjacent to the vertex and 2π. Negative Gaussian curvature indicates a hyperbolic surface with "too much" interior angle relative to a planar patch, while positive Gaussian curvature indicates a cone-shaped local neighborhood that points inward.

The theory of curvature is extremely involved and easily could fill a course worth of material. Here we focus only on two-dimensional surfaces in $\mathbb{R}^3$ and skip many of the technical and qualitative proofs that comprise most of the discussion in a more theoretically-oriented treatment.

5.2 DIFFERENTIAL OF A MAP

Our broad goal is to define ways to measure the shape of an orientable surface $\mathcal{M} \subseteq \mathbb{R}^3$ locally. As a stepping stone to that point, we first consider the problem of computing derivatives of functions along $\mathcal{M}$; then in §5.3 we will apply these derivatives to the normal vector of the surface.

The notion of a derivative can be extended to submanifolds using curves:

Definition 5.1 (Differential). *Suppose $\varphi : \mathcal{M} \to \mathcal{N}$ is a map from a submanifold $\mathcal{M} \subseteq \mathbb{R}^k$ into a submanifold $\mathcal{N} \subseteq \mathbb{R}^\ell$. Then, the* differential $d\varphi_{\mathbf{p}} : T_{\mathbf{p}}\mathcal{M} \to T_{\varphi(\mathbf{p})}\mathcal{N}$ *of φ at a point $\mathbf{p} \in \mathcal{M}$ is given by*

$$d\varphi_{\mathbf{p}}(\mathbf{v}) := (\varphi \circ \gamma)'(0), \tag{5.1}$$

where $\gamma : (-\varepsilon, \varepsilon) \to \mathcal{M}$ is any curve with $\gamma(0) = \mathbf{p}$ and $\gamma'(0) = \mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$.

This construction is illustrated in Figure REF. At $\mathbf{p} \in \mathcal{M}$, we take a tangent vector $\mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$ and trace out a curve γ along $\mathcal{M}$ that passes through $\mathbf{p}$ at $t = 0$, with velocity $\mathbf{v}$. The image $\varphi \circ \gamma$ traces out a curve on $\mathcal{N}$, whose tangent at $t = 0$ provides $d\varphi_{\mathbf{p}}(\mathbf{v})$.

Before we proceed, we should check that our definition makes sense and agrees with our notion of a derivative in $\mathbb{R}^n$:

Proposition 5.1. *Definition 5.1 is consistent, in that (5.1) does not depend on the particular choice of γ so long as it passes through $\mathbf{p}$ at $t = 0$ with velocity $\mathbf{v}$. Moreover, $d\varphi_{\mathbf{p}}$ is a linear map.*

Proof. **JS: Give expression in terms of Jacobian after choosing a local parameterization.** □

The second half of this proposition, that $d\varphi_{\mathbf{p}}$ is linear, connects differentials to the usual notion of a derivative in $\mathbb{R}^n$; exercise 5.1. shows that the differential of a map between Euclidean spaces simply encodes its Jacobian matrix as a linear operator.

We pause here to highlight our first instance of a pattern that will repeat in our brief treatment of differential geometry. In particular, the differential $d\varphi$ is a curious object: It attaches an operator $d\varphi_{\mathbf{p}}$ to every point $\mathbf{p} \in \mathcal{M}$. Making matters seemingly more complicated, the target of $d\varphi_{\mathbf{p}}$ is $T_{\varphi(\mathbf{p})}\mathcal{N}$, while the target of $d\varphi_{\mathbf{q}}$ for some $\mathbf{q} \neq \mathbf{p}$ is $T_{\varphi(\mathbf{q})}\mathcal{N}$—a potentially different space! This apparent inconsistency is needed to account for the fact that tangent spaces change as we move along manifolds and is one of the key sources of confusing notation in differential geometry. In exchange for this confusion, however, we successfully characterize how frames and coordinates change from point to point as we move along a surface or manifold. In more advanced treatments, the notion of a *bundle* accounts for the fact that the target of $d\varphi$ changes from point to point.

5.3 THE SECOND FUNDAMENTAL FORM

Now, assume $\mathcal{M}$ is an orientable surface (2-submanifold) in $\mathbb{R}^3$. In particular, we can assume that $\mathcal{M}$ admits a field of unit normal vectors $\mathbf{n} : \mathcal{M} \to \mathcal{S}^2$ as defined in Definition 4.5. Since we can think of the unit sphere $\mathcal{S}^2$ as a two-dimensional submanifold of $\mathbb{R}^3$, we can reasonably think of $\mathbf{n}$ as a map between submanifolds and hence compute its differential.

By definition, we know $\|\mathbf{n}(\mathbf{p})\|_2 = 1$ for all $\mathbf{p} \in \mathcal{M}$. Hence, applying Proposition 3.3 to Definition 5.1 immediately justifies the following property:

Proposition 5.2. *If $\mathbf{n} : \mathcal{M} \to \mathcal{S}^2$ is a field of unit normal vectors, then for all $\mathbf{p} \in \mathcal{M}$ we have $d\mathbf{n}_\mathbf{p} : T_\mathbf{p}\mathcal{M} \to T_\mathbf{p}\mathcal{M}$.*

The operator $d\mathbf{n}_\mathbf{p}$—or its counterpart with reversed sign, depending on the textbook—is known as the *shape operator* of $\mathcal{M}$ at $\mathbf{p}$.

The shape operator $d\mathbf{n}_\mathbf{p}$ takes in a vector $\mathbf{v} \in T_\mathbf{p}\mathcal{M}$ and outputs another vector $d\mathbf{n}_\mathbf{p}(\mathbf{v}) \in T_\mathbf{p}\mathcal{M}$; in local coordinates using Einstein notation, it would have one upper index and one lower index similar to (2.2). A convenient convention in differential geometry *lowers the index*—leading to a "dot product type" matrix as in (2.3)—by defining the *second fundamental form*:

Definition 5.2 (Second fundamental form)**.** *The* second fundamental form *at $\mathbf{p} \in \mathcal{M}$ is the bilinear operator $\mathrm{I\!I}_\mathbf{p} : T_\mathbf{p}\mathcal{M} \times T_\mathbf{p}\mathcal{M}$ given by*

$$\mathrm{I\!I}_\mathbf{p}(\mathbf{v}, \mathbf{w}) := -\mathbf{v} \cdot d\mathbf{n}_\mathbf{p}(\mathbf{w}). \tag{5.2}$$

Notice that $d\mathbf{n}_\mathbf{p}$ and $\mathrm{I\!I}_\mathbf{p}$ encode the same information; given the operator $\mathrm{I\!I}_\mathbf{p}$ we could recover $d\mathbf{n}_\mathbf{p}$ by evaluating $\mathrm{I\!I}_\mathbf{p}(\mathbf{e}_i, \mathbf{e}_j)$ on a basis $\{\mathbf{e}_1, \mathbf{e}_2\} \subset T_\mathbf{p}\mathcal{M}$.

The diagonal of $\mathrm{I\!I}_\mathbf{p}$ has an elegant interpretation. Suppose you are a passenger in a car is driving along a surface $\mathcal{M}$. As the car traces out its path, you experience forces whenever the driver hits the accelerator pedal or turns the steering wheel. But passengers in a car driving with constant (high) speed over a hill while wearing a seat belt have experienced another force, the acceleration of the car simply needed to keep it glued to $\mathcal{M}$ rather than flying along a straight line. This force has to do with the geometry of $\mathcal{M}$, while the forces due to steering and forward acceleration have more to do with the driver.

This "acceleration due to geometry" is formalized by the idea of *normal curvature*. Take $\gamma : (-\varepsilon, \varepsilon) \to \mathcal{M}$ to be a curve along the surface $\mathcal{M}$, parameterized by arc length. By definition, the tangent to γ is also tangent to $\mathcal{M}$, showing it is orthogonal to the normal $\mathbf{n}$:

$$\gamma'(s) \cdot \mathbf{n}(\gamma(s)) \equiv 0 \; \forall s \in (-\varepsilon, \varepsilon).$$

Differentiating both sides shows

$$\kappa(s)\mathbf{N}(s) \cdot \mathbf{n}(\gamma(s)) + \gamma'(s) \cdot d\mathbf{n}_{\gamma(s)}(\gamma'(s)) \equiv 0.$$

Here $\kappa(s)$ is the curvature of γ and $\mathbf{N}(s)$ is its unit normal vector—not to be confused with the unit normal $\mathbf{n}$ of the surface $\mathcal{M}$! Hence, we have verified the relationship

$$\mathbf{k}(s) \cdot \mathbf{n}(\gamma(s)) \equiv \mathrm{I\!I}_{\gamma(s)}(\mathbf{T}(s), \mathbf{T}(s)), \tag{5.3}$$

where $\mathbf{k}(s) := \kappa(s)\mathbf{N}(s) = \gamma''(s)$ is the *curvature vector* of γ and $\mathbf{T}(s) = \gamma'(s)$ is the unit tangent to γ. Hence, we have verified *Meusnier's Theorem*, which states that the component of $\mathbf{k}$ in the $\mathbf{n}$ direction is determined by the second fundamental form, without any reference to the geometry of γ beyond its tangent vector. Figure REF illustrates this property using normal sections to the surface.

Warning. *The discussion above highlights a potentially confusing linguistic point:* ***The curvature and normals of a curve along a surface along a surface are related to* but not the same as *the curvature and normals of the surface itself.*** *For example, as illustrated in Figure REF, if we draw a circle on a plane embedded in $\mathbb{R}^3$, the plane has a constant normal vector and zero curvature, while the circle has nonzero curvature and normal vectors that are tangent to the surface.*

5.4 PRINCIPAL CURVATURES

As we have written it, the relationship between $\mathrm{I\!I}_{\mathbf{p}}(\mathbf{v},\mathbf{w})$ and $\mathrm{I\!I}_{\mathbf{p}}(\mathbf{w},\mathbf{v})$ is unclear: The vectors $\mathbf{v},\mathbf{w} \in T_{\mathbf{p}}\mathcal{M}$ seem to play different roles in the definition of $\mathrm{I\!I}$. In reality, these two quantities actually agree:

Proposition 5.3. *For all* $\mathbf{v},\mathbf{w} \in T_{\mathbf{p}}\mathcal{M}$, $\mathrm{I\!I}_{\mathbf{p}}(\mathbf{v},\mathbf{w}) = \mathrm{I\!I}_{\mathbf{p}}(\mathbf{w},\mathbf{v})$.

Proof. JS: Is there some coordinate-free way to do this? □

JS: Draw a picture and give some intuition.

Thanks to Proposition 5.3, we can think of $\mathrm{I\!I}_{\mathbf{p}}$ like a symmetric matrix, an intuition that can be made concrete by evaluating $\mathrm{I\!I}_{\mathbf{p}}$ in a basis $\{\mathbf{e}_1,\mathbf{e}_2\} \subset T_{\mathbf{p}}\mathcal{M}$. Thanks to the Spectral Theorem (Theorem 2.1), it thus makes sense to compute eigenvectors and eigenvalues of $\mathrm{I\!I}_{\mathbf{p}}$.

One of many ways to define the eigenvectors of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is to find the critical points of the map $\mathbf{v} \mapsto \mathbf{v}^\top A\mathbf{v}$ subject to the constraint $\|v\|_2 = 1$. This formulation gives a geometrically intuitive definition of the *principal curvatures* at $\mathbf{p} \in \mathcal{M}$:

Definition 5.3 (Principal curvatures and directions)**.** *The* principal curvatures *of a surface* $\mathcal{M}$ *at a point* $\mathbf{p} \in \mathcal{M}$ *are given by*

$$\kappa_{\min} := \begin{cases} \min_{\mathbf{v}\in T_{\mathbf{p}}\mathcal{M}} & \mathrm{I\!I}(\mathbf{v},\mathbf{v}) \\ \textit{subject to} & \|\mathbf{v}\|_2 = 1 \end{cases} \tag{5.4}$$

$$\kappa_{\max} := \begin{cases} \max_{\mathbf{v}\in T_{\mathbf{p}}\mathcal{M}} & \mathrm{I\!I}(\mathbf{v},\mathbf{v}) \\ \textit{subject to} & \|\mathbf{v}\|_2 = 1. \end{cases} \tag{5.5}$$

That is, by (5.3)*,* $\kappa_{\min}$ *and* $\kappa_{\max}$ *are the minimum and maximum normal curvatures achievable at* $\mathbf{p} \in \mathcal{M}$*. We call unit vectors achieving these extrema* principal directions $\mathbf{v}_{\min},\mathbf{v}_{\max} \in T_{\mathbf{p}}\mathcal{M}$.

The principal directions are nonunique, since changing the sign of $\mathbf{v}$ does not affect the objective function.

Definition 5.3 is somewhat nonstandard. Instead, the more typical definition is to resort to a local basis. Take $\mathbf{e}_1,\mathbf{e}_2 \in T_{\mathbf{p}}\mathcal{M}$ to be an orthonormal basis to $T_{\mathbf{p}}\mathcal{M}$. Then, we can construct a symmetric matrix with entries $\ell_{ij} := \mathrm{I\!I}(\mathbf{e}_i,\mathbf{e}_j)$. The principal directions are given by the eigenvectors of ℓ_{ij}, and the principal curvatures are the corresponding eigenvalues. This construction provides one additional geometric observation as a corollary to the Spectral Theorem: The directions $\mathbf{v}_{\min}$ and $\mathbf{v}_{\max}$ are orthogonal to one another. This observation is trivial mathematically but somewhat surprising geometrically: The directions of minimum and maximum normal curvatures at $\mathbf{p} \in \mathcal{M}$ *must* be perpendicular to one another.

Figure REF shows example fields of principal directions on different surfaces. From an applications perspective, these directions can be extremely desirable: They align to local features of the surface, making them natural choices for a variety of tasks from hatching during nonphotorealistic rendering CITE to feature alignment for quadrilateral meshing CITE.

5.5 GAUSSIAN AND MEAN CURVATURES

We finally are in a position to define the two most famous measurements of surface geometry:

Definition 5.4 (Gaussian and mean curvature)**.** *The Gaussian curvature* K *and mean curvature* H *at a point* $\mathbf{p} \in \mathcal{M}$ *are given by:*

$$K := \kappa_{\min} \cdot \kappa_{\max} \tag{5.6}$$

$$H := \frac{1}{2}(\kappa_{\min} + \kappa_{\max}). \tag{5.7}$$

Based on the eigenvalue interpretation of principal curvatures provided in the last section, an alternative description is that the Gaussian curvature is the determinant of $\mathrm{I\!I}_{\mathbf{p}}$, and the mean curvature is half the trace of $\mathrm{I\!I}_{\mathbf{p}}$.

These two measures of curvature are *scalar fields* along the surface, that is, they each assign a real number to each point on $\mathcal{M}$ measuring its curvature. These two values are the best known geometric quantities in the local theory of surfaces and provide intuitive information about the geometry of $\mathcal{M}$ explained informally below.

MEAN CURVATURE. The mean curvature H inherits its name not just from the definition (5.7) but also a second relationship. If $\{\mathbf{e}_1, \mathbf{e}_2\}$ forms an orthogonal basis for $T_{\mathbf{p}}\mathcal{M}$, we can compute the normal curvature of $\mathcal{M}$ at every direction from $\mathbf{p} \in \mathcal{M}$ by evaluating

$$\kappa(\theta) := \mathrm{I\!I}(\mathbf{e}_1 \cos\theta + \mathbf{e}_2 \sin\theta, \mathbf{e}_1 \cos\theta + \mathbf{e}_2 \sin\theta).$$

Then, a straightforward argument from linear algebra shows

$$H = \frac{1}{2\pi} \int_0^{2\pi} \kappa(\theta)\, d\theta. \tag{5.8}$$

That is, the mean curvature H is truly the mean of the normal curvatures in all the possible outgoing directions.

A second interpretation of H formalized in exercise 5.3. is illustrated in Figure REF. Given an oriented surface $\mathcal{M}$, we can generate a set of surfaces $\mathcal{M}_t$ parameterized by scalar $t \in (-\varepsilon, \varepsilon)$ by displacing each surface point a distance t along the unit normal $\mathbf{n}$, via $\mathbf{p} \mapsto \mathbf{p} + t\mathbf{n}_{\mathbf{p}}$. When $t = 0$, we have $\mathcal{M}_0 = \mathcal{M}$. In a local neighborhood of each $\mathbf{p} \in \mathcal{M}$, we could compute the rate of change of surface area. As shown in Figure REF, if the surface is planar, surface area does not change, while if the surface is a sphere, displacing along the normal increases surface area. The local expansion or contraction rate of surface area at $t = 0$ is given by $H(\mathbf{p})$, and the rate of change of the total area of the surface is given by $\int_{\mathcal{M}} H(\mathbf{p})\, d\mathbf{p}$.

The $H \equiv 0$ case is particularly interesting. Surfaces with zero mean curvature are known as *minimal surfaces* and—as we will show in § REF—are local minimizers of the surface area functional. Physically, minimal surfaces are given by soap films stretched over wire-shaped fixed boundaries. Some minimal surfaces are shown in Figure REF; even though the surfaces change significantly as they are displaced along their normal vectors, the surface area remains constant to first order. Minimal surfaces are necessarily flat or hyperbolic, since $H = 0$ implies that κ_1 and κ_2 have opposite signs.

GAUSSIAN CURVATURE. While mean curvature H measures the bending of the normal vector, Gaussian curvature K measures stretching of the surface relative to the plane. Most importantly, recalling that K is the product of extremal normal curvatures $\kappa_{\min}$ and $\kappa_{\max}$, we can use the sign of Gaussian curvature to classify local surface geometry as follows:

- **Elliptic ($K > 0$):** Elliptic points are egg-shaped; since the two principal curvatures agree in sign, this case corresponds to points on a surface that bend the same way in all directions.
- **Hyperbolic ($K < 0$):** Hyperbolic points are saddle-shaped, reflecting the fact that $\kappa_{\min}$ and $\kappa_{\max}$ have opposite sign.
- **Parabolic ($K = 0$):** Parabolic points are flat in at least one direction. Surfaces where $K \equiv 0$ are *developable*, roughly corresponding to surfaces achievable by bending a flat sheet of paper without stretching.

Figure REF shows a surface $\mathcal{M}$ divided into regions based on the sign of K, providing a straightforward if noisy way to divide surface the surface into meaningful segments.

Figure REF provides an alternative intuition for K that lifts to definitions of scalar curvature on higher-dimensional manifolds and even metric spaces like graphs CITE. At a point $\mathbf{p} \in \mathcal{M}$, take a short piece of string of length $r > 0$ and draw a *geodesic circle* surrounding $\mathbf{p}$ consisting of all points distance r from $\mathbf{p}$ along $\mathcal{M}$; such distances along $\mathcal{M}$ are geodesic, defined more carefully in § REF. The resulting geodesic circle has arc length, or circumference, $C(r)$. Then, Gaussian curvature can be obtained as the limit

$$K = \lim_{r \to 0^+} 3\frac{2\pi r - C(r)}{\pi r^3}. \tag{5.9}$$

Recall that $2\pi r$ is the circumference of a circle of radius r on the plane. Hence, Gaussian curvature can be understood as a comparison between the circumference of a small circle on M relative to that of a small planar circle. Although a proof of (5.9) is out of the scope of our discussion, it is a consequence of the Bertrand–Diquet–Puiseaux Theorem and was originally introduced in JS: year.

Figure REF illustrates this formula at points with a range of K values. As the surface becomes more pointy—corresponding to large K—the circumference of the circle around $\mathbf{p}$ shrinks. In the opposite direction, making the surface more hyperbolic effectively adds more circumference at a fixed radius from $\mathbf{p}$; the surface buckles into a wavey hyperbolic shape to fit in the additional perimeter. As a hint for how we will go about discretizing Gaussian curvature, Figure REF shows that the same behavior can be observed in the origami-like scenario of gluing together equilateral triangles. As we add more triangles around a fixed shared vertex, the length of the one-ring grows and the piecewise-flat surface eventually becomes hyperbolic.

A related formula shows that we can also describe K in terms of areas:

$$K = \lim_{r \to 0^+} 12\frac{\pi r^2 - A(r)}{\pi r^4}. \tag{5.10}$$

Here, $A(r)$ is the area on the surface enclosed by the geodesic circle, and πr^2 is the area of a circle in the plane. Comparing this formula to our descriptions of mean curvature H yields some insight into the differences between these two quantities: We compute K by measuring how areas change as we expand radially in the tangential direction along a surface, while we compute H by measuring changes in area as we move orthogonally to the surface along its normal vectors.

Two theoretical descriptions of Gaussian curvature are pillars of the classical differential geometry literature:

- Gauss's *Theorema Egregium* is a key theoretical result about K as a function on a surface $\mathcal{M}$ stating that K is invariant under isometric deformation. That is, if we bend a surface without stretching—hence preserving distances along the surface—its Gaussian curvature remains the same at every point. This result is somewhat surprising since our definition of K is in terms of normal vectors, which certainly change as $\mathcal{M}$ undergoes nearly any deformation.

- The *Gauss–Bonnet Theorem*, similar to the winding number theorem in § REF, shows that the total Gaussian curvature over a surface without boundary is topologically invariant:

$$\int_{\mathcal{M}} K(\mathbf{p})\, dA(\mathbf{p}) = 2\pi\chi, \tag{5.11}$$

  where χ is the Euler characteristic of $\mathcal{M}$ defined in (4.2). This result links the local geometry of a surface to a global topological structure, even if a surface undergoes non-isometric deformation. Note that this formula needs to be adjusted with a boundary integral term if $\mathcal{M}$ has a boundary.

PUTTING THE TWO TOGETHER. Similarly to Proposition 3.2 for curves, we might ask whether curvature is sufficient to characterize a surface up to rigid motion. The analogous result for surfaces is somewhat harder to state but known as the *Bonnet Theorem* in classical differential geometry. The basic take-away of this result is that we can reconstruct a surface patch given both its first and second fundamental forms; although we have not defined it here, the first fundamental

form is simply the matrix of inner products of tangent vectors. For technical reasons, given only the Gaussian and mean curvatures it is not possible to reconstruct a surface, and furthermore the prescribed first and second fundamental forms must satisfy a certain set of compatibility conditions known as the Gauss and Mainardi–Codazzi equations. Perhaps the simplest corollary of the Bonnet Theorem is that if a map from one surface into another preserves dot products and the second fundamental form, it is necessarily an orthogonal transformation (composition rotation, translation, and reflection).

5.5.1 *Advanced Topic: Mean Curvature Normal*

Recall from §3.5.1 that another way we were able to define the curvature of a curve was by taking the first variation of the arc length functional. The two-dimensional analog of this calculation leads to the idea of the *mean curvature normal* to a surface, which is exactly what it sounds like: the normal to a surface weighted at each point by the mean curvature.

Suppose $f_t(u,v) : \mathbb{R}^2 \to \mathbb{R}^3$ parameterizes a *time-varying* set of surface patches; that is, at any fixed $t \in (-\varepsilon, \varepsilon)$ if we take a small open patch $U \subseteq \mathbb{R}^2$ then $f(U)$ gives a small piece of a surface in $\mathbb{R}^3$. Being extremely sloppy about the bounds on (u,v) for integration, by the determinant formula for surface area JS: add me the area of our patch is

$$A_t := \iint \underbrace{\left\| \frac{\partial f_t}{\partial u} \times \frac{\partial f_t}{\partial v} \right\|_2 du\, dv}_{dA}.$$

For convenience, define tangent vectors $\mathbf{t}_u := \frac{\partial f_t}{\partial u}$ and $\mathbf{t}_v := \frac{\partial f_t}{\partial v}$; notice that the cross product of these two vectors is parallel to the normal to the surface.

We can differentiate in the time variable t to understand how surface area changes as the geometry is perturbed:

$$\begin{aligned}
\frac{dA_t}{dt} &= \iint \frac{d}{dt} \|\mathbf{t}_u \times \mathbf{t}_v\|_2 \, du\, dv \text{ by differentiating under the integral} \\
&= \iint \|\mathbf{t}_u \times \mathbf{t}_v\|_2^{-1} (\mathbf{t}_u \times \mathbf{t}_v) \cdot \frac{d}{dt} (\mathbf{t}_u \times \mathbf{t}_v) \, du\, dv \text{ by exercise 3.4.} \\
&= \iint \|\mathbf{t}_u \times \mathbf{t}_v\|_2^{-1} (\mathbf{t}_u \times \mathbf{t}_v) \cdot \left(\frac{\partial \mathbf{t}_u}{\partial t} \times \mathbf{t}_v + \mathbf{t}_u \times \frac{\partial \mathbf{t}_v}{\partial t} \right) du\, dv \text{ by the product rule} \\
&= \iint \mathbf{n} \cdot \left(\frac{\partial \mathbf{t}_u}{\partial t} \times \mathbf{t}_v + \mathbf{t}_u \times \frac{\partial \mathbf{t}_v}{\partial t} \right) du\, dv \text{ by definition of the unit normal} \\
&= \iint \mathbf{n} \cdot \left(\frac{\partial \mathbf{w}_t}{\partial u} \times \mathbf{t}_v + \mathbf{t}_u \times \frac{\partial \mathbf{w}_t}{\partial v} \right) du\, dv \text{ if we define velocity } \mathbf{w}_t := \frac{\partial f_t}{\partial t} \\
&= \iint \left(\frac{\partial \mathbf{w}_t}{\partial u} \cdot (\mathbf{t}_v \times \mathbf{n}) + \frac{\partial \mathbf{w}_t}{\partial v} \cdot (\mathbf{n} \times \mathbf{t}_u) \right) du\, dv \text{ by the triple product identity } \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a}).
\end{aligned}$$

Now assume that our setup is similar to that shown in Figure REF, namely that $\mathbf{w}_t$ vanishes outside of a small neighorhood in (u,v) space; this will allow us to ignore boundary terms when integrating by parts. Then, we have

$$\begin{aligned}
\frac{dA_t}{dt} &= -\iint \mathbf{w}_t \cdot \left(\frac{\partial}{\partial u}(\mathbf{t}_v \times \mathbf{n}) + \frac{\partial}{\partial v}(\mathbf{n} \times \mathbf{t}_u) \right) du\, dv \text{ by exercise 5.4. (integration by parts)} \\
&= -\iint \mathbf{w}_t \cdot \left(\mathbf{t}_v \times \frac{\partial \mathbf{n}}{\partial u} - \mathbf{t}_u \times \frac{\partial \mathbf{n}}{\partial v} \right) du\, dv \text{ by the product rule and since } \frac{\partial \mathbf{t}_v}{\partial u} = \frac{\partial \mathbf{t}_u}{\partial v} \\
&= \iint \mathbf{w}_t \cdot (d\mathbf{n}(\mathbf{t}_u) \times \mathbf{t}_v + \mathbf{t}_u \times d\mathbf{n}(\mathbf{t}_v)) \, du\, dv \text{ by definition of } d\mathbf{n} \\
&= 2 \iint \mathbf{w}_t \cdot H\mathbf{n} \, dA, \text{ by exercise 5.5. (identity for mean curvature).}
\end{aligned}$$

In direct analogy to the computations we did for curves in §3.5.1, we have shown that the *first variation* of surface area is the *mean curvature normal* $H\mathbf{n}$. That is, if we wish to extremize surface area most efficiently, we should flow points on the surface along their normal vectors.

This idea leads to the idea of *mean curvature flow*, or gradient descent on surface area. Mean curvature flow, illustrated in Figure REF, is related to the physical processes under which soap films evolve. Beyond being the topic of considerable mathematical literature attempting to describe qualitative properties of mean curvature flow, it can be used in practice to smooth out noisy surfaces, as originally proposed in CITE. Figure REF shows an example of this application: Noise in a surface leads to spurious surface area that is reduced in the flow. On the other hand, if we run the flow too long, it can develop *singularities*, as illustrated in Figure REF. Development of algorithms "adjacent to" mean curvature flow that do not develop these spurious artifacts is a topic of research; one recent proposal is illustrated in Figure REF.

5.6 APPLICATIONS OF CURVATURE

Having established the definition and basic properties of surface curvature, we now turn our attention to the applied world, examining how and why it is useful to compute curvature algorithmically. Curvature was one of the earliest quantities from differential geometry to be adapted to computer vision, computer-aided design (CAD), and other numerical settings. Stable algorithms for computing curvature—which can be difficult to design, since curvature is a second-derivative quantity—find application in a diverse set of pipelines including the ones listed below:

SURFACE DESCRIPTOR. Probably the most natural application of computing principal curvatures, principal directions, and Gauss/mean curvatures is to generate a *descriptor* (or *feature*) for describing points on a surface. Computer vision pipelines often seek "salient" features of surface geometry to anchor the computations, and extrema of curvature or derived quantities are often suitable for this task. For instance, if we wish to align two scans of the same object automatically, we could find the rotation/translation that best aligns points with similar curvature values. Integral curves of the principal direction field also provide *feature curves* that align to the interesting geometric features of a surface. Unlike coordinates or vectors along edges, curvature values are invariant to rigid motions of a surface in space. Indeed, by the Theorema Egregium, Gaussian curvature is even invariant to bending motions, making it a suitable feature to compute for matching points on surfaces even if they undergo certain types of deformation; one key application is to matching scans of human bodies, whose deformations are largely bending rather than stretching thanks to our rigid underlying skeletons.

One abstract way to understand the role of curvature as a descriptor is shown in Figure REF. We can think of the function assigning a mean and Gaussian curvature value to every point on a smooth surface $\mathcal{M}$ as an *embedding* of sorts $\phi : (x, y, z) \in \mathcal{M} \to (H, K) \in \mathbb{R}^2$. As we move a surface around in 3D space or deform lengths and angles along the surface, the (x, y, z) coordinates of its features change considerably. But, the (H, K) "coordinates" of the points are less strongly affected. Tasks like matching points and features often are more straightforward in (H, K) space thanks to this property: There is not some unknown rotation or deformation acting on the coordinates that we must account for.

FAIRNESS MEASURE. A key consideration when designing surfaces for manufacturing applications is their *fairness*, or (roughly) smoothness. In this domain, surfaces are often built out of building blocks like quadric patches, joined along *seams* that transition from one patch to another. Meeting these pieces as smoothly as possible is desirable in industrial and mechanical design, for both physical (e.g. aerodynamics) and aesthetic reasons.

Examples of spline surfaces joined together are shown in Figure REF. In general, in computer-aided geometric design (CAGD) a number of curvature-related properties are considered when

understanding geometry along a seam, in terms of the parameterizations $f_1(u,v), f_2(u,v) : \mathbb{R}^2 \to \mathbb{R}^3$ of the joined patches:

- C^1 continuity: The gradients of the two functions f_1, f_2 agree along the seam.
- G^1 continuity: The two patches have the same tangent planes along the same.
- C^2 continuity: The second derivatives of f_1, f_2 agree along the seam.
- G^2 continuity: The second fundamental form varies continuously along the seam.

The G^k properties arguably are more valuable for manufacturing, since at the end of the day the physical object should not reveal the patches from which it is built. But, these can be challenging constraints to enforce in practice, so often the C^k constraints are used as proxies.

Figure REF shows an interesting visualization of curvature often used in user interfaces for design. Here, we show the *reflection* of a striped pattern off the side of a modeled surface. The traces of the stripes reveal curvature discontinuities as sharp bends and G^1 discontinuities as kinks in the reflected curves.

SMOOTHING AND RECONSTRUCTION. As described in §5.5.1, mean curvature flow—or equivalently, gradient descent on surface area—is a fundamental technique for surface smoothing, used e.g. for removing artifacts from meshes reconstructed from 3D scanning equipment. Curvature is also involved in other surface editing tasks. For instance, CITE proposes an algorithm for deforming surfaces by modifying discrete analogs of their fundamental forms and then reconstructing, and CITE proposes allowing a user to paint a "mean curvature half-density" function onto a surface to add bumps and bends to the geometry.

RENDERING. Curvature has a strong bearing on the appearance of physical objects when they are rendered or photographed. After all, at a local scale curvature gives a complete picture of surface geometry. Early techniques for surface reconstruction attempted to recover curvature values from two-dimensional visual content CITE, and some theoretical results show that such a procedure can succeed when surfaces are shaded using flat, Lambertian materials. While this early theoretical work showed intrinsic interest, however, practical application was challenging since real-world surfaces exhibit irregular, spatially-varying textures and reflectance models.

Turning this computer vision work on its head, however, we can use the observation that curvature is encoded in visual content to inspire *expressive* ways to render three-dimensional shapes. This idea is widely used in pipelines for *nonphotorealistic rendering*, where the goal is to render in a visually attractive or communicative fashion rather than accurately simulating the physics of light and materials. As an example, principal directions are used to guide the hatching algorithm described in CITE, which simulates pen-and-ink sketching of 3D surfaces; Figure REF shows an example.

(RE)MESHING. While our discussion has focused on triangle meshes, the reality is that a surface can be tiled with a variety of regular polygons other than triangles. One common choice is quadrilaterals (quads), which present a variety of advantages in graphics and scientific computing applications: Quad meshes locally resemble grids, facilitating adaptation of classical grid-based simulation techniques to these structures, and they can be used to define control points of spline patches and subdivision surfaces in case we wish to smooth out a surface from its meshed facets. An example of a quad mesh is shown in Figure REF.

A key task is conversion from one type of mesh to another. For instance, we may wish to take a triangle mesh produced using a 3D scanner and convert it to a quad mesh for use by an artist. Design of automatic algorithms for this *remeshing* task are a topic of research in the geometry processing community.

A common approach to quad remeshing is to align quads to the geometric features of a surface. This alignment tends to improve approximation quality when replacing a dense triangulated surface with a sparse set of quads, and it facilitates subsequent editing procedures that displace salient features of a mesh. To this end, algorithms like the one in REF compute principal curvatures as an initial step, and in a second discrete step tile the surface with quads that are aligned to these directions. Principal curvature directions can fit the bill for algorithms in this domain: Not only are they aligned to key surface features, but also they provide two orthogonal directions per point on a surface, differentially resembling the sides of a quad. An example of this procedure is shown in Figure REF. Foreshadowing our discussion of vector field topology in § REF, note the presence of certain *singular points*, or vertices on the quad mesh with fewer or more than four neighbors.

5.7 APPROXIMATING CURVATURE

Having established the basic mathematics of surface curvature, we now consider a discrete analog of the problem: estimation of curvature on a triangulated surface. This task can be interpreted in two (not necessarily incompatible) ways:

- Treating a triangle mesh like a discretization of a smooth surface, and approximating the derivatives involved in computing curvature using sampled points available on the mesh.
- Redefining curvature what curvature *is*, directly for a triangulated rather than surface.

The latter task seems almost political: What gives us the right to redefine curvature simply because we do not have access to a smooth surface approximated by a mesh? But a few observations make this approach seem less outrageous. First of all, we often are not in the situation where we can somehow refine our approximation of a surface by somehow "hallucinating" a denser sampling; rather, a mesh is given as input from a scan of a 3D object or an artist model, from which we have no opportunity to draw additional samples. Furthermore, treating a triangle mesh as a surface leads to a particularly boring treatment of curvature, with $\kappa \equiv 0$ in the interior of each triangle and $\kappa = \infty$ on non-flat edges and vertices. In any event, it is not unreasonable to expect some discrete measures of curvature to converge to the true analog on a smooth surface in the limit of refinement.

Less satisfyingly, we will see that there exist *many* models for approximating Gaussian curvature, mean curvature, principal directions, and other quantities on discrete surfaces, and that they do not necessarily agree with one another. Just as we found for the curvature of curves in § REF, there is "no free lunch" here—using different formulas for or properties of curvature to motivate a discrete computation can lead to different numerical values for curvature in the end.

What we provide below is a small sampling of the broad literature on curvature computation from discrete surfaces; a full survey of the many methods available and their shades of difference would take far more space.

5.7.1 *Local Tensor Methods*

We begin with some of the earliest methods for curvature computation to make it into the computer graphics and vision literatures. These methods typically aimed not just to compute Gaussian or mean curvature but rather to compute matrices resembling the full second fundamental form, which encode principal directions and curvature in a single tensor.

TAYLOR SERIES METHOD. One early technique introduced by G. Taubin in 1995 builds up a matrix related to—but not the same as—the second fundamental form [CITE]. On a smooth surface $\mathcal{M}$, a matrix M_p at a point $\mathbf{p} \in \mathcal{M}$ is defined via

$$M_{\mathbf{p}} := \frac{1}{2\pi} \int_{-\pi}^{\pi} \kappa_\theta \mathbf{t}_\theta \mathbf{t}_\theta^\top \, d\theta. \tag{5.12}$$

Here, $\kappa(\theta)$ is the normal curvature in direction θ and $\mathbf{t}_\theta = \mathbf{e}_1 \cos\theta + \mathbf{e}_2 \sin\theta$, as in (5.8). It is easy to see that $M_{\mathbf{p}}$ has the normal vector $\mathbf{n}_{\mathbf{p}}$ as an eigenvector with eigenvalue 0. More surprisingly, the remaining eigenvectors of $M_{\mathbf{p}}$ are the two principal directions $\mathbf{v}_{\min}$ and $\mathbf{v}_{\max}$ at $\mathbf{p}$, with eigenvalues $\lambda_1 = \frac{3}{8}\kappa_{\min} + \frac{1}{8}\kappa_{\max}$ and $\lambda_2 = \frac{1}{8}\kappa_{\min} + \frac{3}{8}\kappa_{\max}$; after computing the λ_i's using 3×3 eigenvector computation, these relationships can be inverted to find the principal curvatures at $\mathbf{p}$. Exercise 5.6. fills in details of these computations.

To complete our description of Taubin's curvature computation technique, we need one more formula for normal curvature. Take $\gamma(s) : (-\varepsilon, \varepsilon) \to \mathcal{M}$ to be a curve along a smooth surface $\mathcal{M}$ with $\gamma(0) = \mathbf{p}$, parameterized by arc length. If $\kappa_\gamma(s)$, $\mathbf{n}_\gamma$, and $\mathbf{t}_\gamma$ are the curvature, normal, and tangent of $\gamma(s)$ as a curve, then expanding a Taylor series about $s = 0$ shows

$$\gamma(s) - \mathbf{p} = \mathbf{t}_\gamma(0)s + \frac{1}{2}\kappa_\gamma(0)\mathbf{n}_\gamma(0)s^2 + O(s^3).$$

Two corollaries immediately follow from this formula:

$$\begin{aligned} \|\gamma(s) - \mathbf{p}\|_2^2 &= s^2 + O(s^3) \\ 2\mathbf{n}_{\mathbf{p}}^\top(\gamma(s) - \mathbf{p}) &= \kappa_{\theta(\mathbf{t})}s^2 + O(s^3), \end{aligned}$$

where we take $\mathbf{t} = \mathbf{t}_\gamma(0)$, $\mathbf{n}_{\mathbf{p}}$ is the normal to $\mathcal{M}$ (not γ!) at $\mathbf{p}$, and $\kappa_{\theta(\mathbf{t})}$ is the normal curvature of $\mathcal{M}$ at $\mathbf{p}$ in direction $\mathbf{t}$. Combining these two expressions shows

$$\kappa_{\theta(\mathbf{t})} = \frac{2\mathbf{n}_{\mathbf{p}}^\top(\gamma(s) - \mathbf{p})}{\|\gamma(s) - \mathbf{p}\|_2^2} + O(s) \tag{5.13}$$

Taubin's algorithm for per-vertex curvature on a triangle mesh first approximates $M_{\mathbf{p}}$ at each vertex using (5.13) and then uses its eigenvalues to obtain curvature. First, the algorithm approximates a per-vertex unit normal $\mathbf{n}_{\mathbf{v}}$ for each $\mathbf{v} \in V$; while this calculation is not crucial, the specific approximation proposed is to average the normals of triangles adjacent to each vertex, weighted by triangle area. Then, M from (5.12) is approximated using a sum

$$M_{\mathbf{v}} \approx \sum_{\mathbf{u} \sim \mathbf{v}} w_{\mathbf{vu}} \kappa_{\mathbf{vu}} \mathbf{t}_{\mathbf{vu}} \mathbf{t}_{\mathbf{vu}}^\top.$$

This sum is over vertices $\mathbf{u}$ adjacent to $\mathbf{v}$. The individual terms in this expression are as follows:

- $\mathbf{t}_{\mathbf{vu}}$ is an approximation of the tangent direction orthogonal to $\mathbf{n}_{\mathbf{v}}$ in the $\mathbf{u}$ direction:

$$\mathbf{t}_{\mathbf{vu}} := \frac{(I_{3\times 3} - \mathbf{n}_{\mathbf{v}}\mathbf{n}_{\mathbf{v}}^\top)(\mathbf{u} - \mathbf{v})}{\|(I_{3\times 3} - \mathbf{n}_{\mathbf{v}}\mathbf{n}_{\mathbf{v}}^\top)(\mathbf{u} - \mathbf{v})\|_2}.$$

- $\kappa_{\mathbf{vu}}$ is an estimation of the directional curvature using (5.13)

$$\kappa_{\mathbf{vu}} := \frac{2\mathbf{n}_{\mathbf{v}}^\top(\mathbf{u} - \mathbf{v})}{\|\mathbf{u} - \mathbf{v}\|_2^2}.$$

- $w_{\mathbf{vu}}$ is an averaging weight proportional to the sum of the two triangle areas adjacent to edge $\mathbf{uv}$, normalized to sum to 1.

Once $M_{\mathbf{v}}$ is approximated, the curvature and principal directions are obtained using standard eigenvalue computations on the resulting 3×3 matrix.

This algorithm is extremely simplistic and gives reasonable approximations of surface curvature, so long as the surface is fairly densely and smoothly sampled. This computation obtains curvature from the one-ring of each vertex, so local noise can obstruct obtaining a reasonable value—this property will be true for many of the techniques we consider, and it is somewhat reasonable in the sense that curvature is built out of derivatives. Since this curvature estimate is built from Taylor series, it is also somewhat difficult to characterize its behavior *before* the mesh is densely sampled.

DIRECT APPROXIMATION OF $\mathbb{II}$. A somewhat more recent method by S. Rusinkiewicz in 2004 directly attempts to build up approximations of the second fundamental form $\mathbb{II}$ CITE. While it is hard to compare the quality of this technique to the previous one from a theoretical perspective, it is interesting to see the alternative engineering decisions go into this alternative method, e.g. estimating curvature per-triangle rather than per-vertex.

Once again, we start with a smooth formula on a surface $\mathcal{M}$ and then discretize it term-by-term on a triangle mesh. For a point $\mathbf{p} \in \mathcal{M}$, take $\mathbf{u}, \mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$ to be two orthonormal tangent vectors at $\mathbf{p}$. In this case, we can express the second fundamental form as a matrix in the $(\mathbf{u}, \mathbf{v})$ coordinates as follows:

$$\mathbb{II}_{\mathbf{p}} = \begin{pmatrix} d\mathbf{n}_{\mathbf{p}}(\mathbf{u}) \cdot \mathbf{u} & d\mathbf{n}_{\mathbf{p}}(\mathbf{v}) \cdot \mathbf{u} \\ d\mathbf{n}_{\mathbf{p}}(\mathbf{u}) \cdot \mathbf{v} & d\mathbf{n}_{\mathbf{p}}(\mathbf{v}) \cdot \mathbf{v} \end{pmatrix}. \tag{5.14}$$

The orthonormality condition here simplifies working with $\mathbb{II}_{\mathbf{p}}$ directly, without having to account for change in coordinates. In this case, for a general vector $\mathbf{w} = c_1\mathbf{u} + c_2\mathbf{v}$ we can show

$$\mathbb{II}_{\mathbf{p}} \cdot \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = d\mathbf{n}_{\mathbf{p}}(\mathbf{w}). \tag{5.15}$$

This relationship is our starting point for discrete estimation of $\mathbb{II}$. In this technique, rather than estimating the second fundamental form per-vertex, it is estimated per-triangle. For every triangle, we first compute an arbitrary orthogonal basis $(\mathbf{u}, \mathbf{v})$ for the plane of the triangle; the final choice of principal directions and curvatures will not depend on this basis, so it can be chosen arbitrarily. A triangle with edges $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$ is shown in Figure REF; we assume the mesh has per-vertex unit normal vectors (e.g. using the strategy suggested in the previous section), restricted to this triangle as $(\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2)$. Then, starting from (5.15) we make the following approximation for the behavior of $\mathbb{II}$ on the triangle:

$$\begin{aligned}
\mathbb{II} \cdot \begin{pmatrix} \mathbf{e}_0 \cdot \mathbf{u} \\ \mathbf{e}_0 \cdot \mathbf{v} \end{pmatrix} &= \begin{pmatrix} (\mathbf{n}_2 - \mathbf{n}_1) \cdot \mathbf{u} \\ (\mathbf{n}_2 \cdot \mathbf{n}_1) \cdot \mathbf{v} \end{pmatrix} \\
\mathbb{II} \cdot \begin{pmatrix} \mathbf{e}_1 \cdot \mathbf{u} \\ \mathbf{e}_1 \cdot \mathbf{v} \end{pmatrix} &= \begin{pmatrix} (\mathbf{n}_0 - \mathbf{n}_2) \cdot \mathbf{u} \\ (\mathbf{n}_0 \cdot \mathbf{n}_2) \cdot \mathbf{v} \end{pmatrix} \\
\mathbb{II} \cdot \begin{pmatrix} \mathbf{e}_2 \cdot \mathbf{u} \\ \mathbf{e}_2 \cdot \mathbf{v} \end{pmatrix} &= \begin{pmatrix} (\mathbf{n}_1 - \mathbf{n}_0) \cdot \mathbf{u} \\ (\mathbf{n}_1 \cdot \mathbf{n}_0) \cdot \mathbf{v} \end{pmatrix}
\end{aligned}$$

This gives a noisy estimate of $\mathbb{II}$ per triangle; CITE then proposes averaging this per-triangle matrix to adjacent vertices after rotating from the triangle normal to the vertex normal.

OTHER DISCRETIZED APPROACHES. The two techniques above are representative of a large class of curvature computation algorithms that each identify a particular formula for curvature or related tensors and then attempt to approximate it on a mesh via divided differences. There are *many* formulas for curvature in the theory of differential geometry, any of which could be used as a starting point for this strategy. For example, CITE uses height functions as a starting point, and

JS: mention someone else.

5.7.2 *Structure-Preserving Methods*

The tensor-based approaches in the previous section would be considered *discretized*, i.e. they are approximations we might expect to converge as a mesh is sampled more and more densely but we have little understanding of behavior in the finite-sample regime. We now show two curvature approximations—one for Gaussian curvature and one for mean curvature—that are designed to preserve structure from the smooth case in a discrete model.

STRUCTURE-PRESERVING GAUSSIAN CURVATURE. For a surface with boundary, we need a slight extension of the Gauss–Bonnet Theorem (5.11):

$$\int_{\mathcal{M}} K(\mathbf{p})\, dA(\mathbf{p}) + \int_{\partial\mathcal{M}} \kappa_g(\mathbf{p})\, ds(\mathbf{p}) = 2\pi\chi(\mathcal{M}). \tag{5.16}$$

Here, κ_g denotes the *geodesic curvature* of the boundary, i.e. the curvature of the boundary curve $\partial\mathcal{M}$ projected onto the tangent plane. The "boundary-aware" Euler characteristic $\chi(\mathcal{M})$ is given by $2-2g-b$, where g is the genus and b is the number of boundary curves JS: check me.

Now, consider a cell V around a vertex on a triangle mesh, shown in Figure REF. We define V as the *Voronoi cell* around the vertex, whose boundary edges are defined to be the perpendicular bisectors of triangle edges; this structure will become critical for our discussion of JS: topic in § REF. Notice that V has disk topology, and hence we can use (5.16) to write

$$\int_V K(\mathbf{p})\, dA(\mathbf{p}) = 2\pi - \sum_j \varepsilon_j.$$

Here, the sum over exterior angles comes from our discussion of curves in § REF: The sum of the exterior angles gives the total geodesic curvature, since the only other facets in the curve are where triangles meet—at which point the change is in the normal rather than tangent direction. As shown in exercise 5.7., by a basic trigonometric argument we can write the same sum using interior angles at the center vertex:

$$\int_V K(\mathbf{p})\, dA(\mathbf{p}) = 2\pi - \sum_j \theta_j. \tag{5.17}$$

As originally proposed in CITE, we can *define* the integrated curvature K over cell V using the right-hand side of the expression above; pointwise curvature can be approximated by dividing by the area of V.

Why choose this definition of curvature per-vertex? Take M to be a triangle mesh without boundary (for simplicty), and define $\{V_i\}_{i=1}^{|V|} V|$ to be the set of per-vertex Voronoi cells. We can compute an integral over the mesh to reveal a conserved structure from smooth theory:

$$\begin{aligned}
\int_M K\, dA &= \sum_i \int_{V_i} K\, dA \text{ since } M = \cup_{i=1}^{|V|} V_i \\
&= \sum_i \left(2\pi - \sum_{j\sim i} \theta_{ij}\right) \text{ by our definition (5.17)} \\
&= 2\pi|V| - \sum_{j\sim i} \theta_{ij} \text{ by expanding the sum} \\
&= 2\pi|V| - \pi|F| = 2\pi\chi(M)
\end{aligned}$$

where $|V|$ is the number of vertices and $|F|$ is the number of triangles. Here, the final line is a byproduct of observing that the sum of all θ_{ij}'s is the same as the sum of all the interior angles of all the triangles. The observation that $2|V| - |F|$ equals the Euler characteristic is a byproduct of the counting arguments in § REF.

Just as we were able to preserve the turning angle theorem for curves when defining planar curvature in § REF, here we were able to preserve a topological invariant, the Euler characteristic χ, encoded in our integrated measure of curvature.

STRUCTURE-PRESERVING MEAN CURVATURE. While a simple topological invariant like the Euler characteristic is not easily encoded in mean curvature, we can choose a different structure to preserve when considering mesh-based measures of this curvature value. In particular, we will use the discussion in §5.5.1 to motivate a definition of mean curvature using derivatives of surface area.

Suppose we have a triangle mesh with vertices V and faces F. Assuming the connectivity of the mesh is fixed, we can think of surface area as a *function* taking the positions of all the vertices $X \in \mathbb{R}^{|V|\times 3}$ and outputting a scalar value:

$$A(X) := \sum_{f\in F} \text{Area}(f; X) = \sum_{(f_1,f_2,f_3)\in F} \frac{1}{2}\|(X_{f_2} - X_{f_1}) \times (X_{f_3} - X_{f_1})\|_2. \tag{5.18}$$

That is, $A(\cdot)$ takes in a list of vertex positions and outputs the sum of the areas of all the triangles on the mesh.

Recall from §5.5.1 that we obtained the mean curvature normal by differentiating surface area in the positions of points along the surface. Here we will do the same, but rather than differentiating in a parameterization of a smooth surface we differentiate in X.

We build up our calculation using a few propositions. Refer to Figure REF for notation:

Proposition 5.4. *The gradient of area for the triangle in Figure REF in vertex* $\mathbf{p}$ *is given by*

$$\nabla_{\mathbf{p}} A = \frac{1}{2}\mathbf{e}_{\perp}, \tag{5.19}$$

where $\mathbf{e}$ *is a vector along the edge opposite* $\mathbf{p}$ *and* $\perp$ *indicates rotation by* 90° *in the plane of the triangle.*

Proof. JS: see slides for now □

While this formula is a useful intuitive description of the gradient of triangle area, we need to massage it more before recognizing a famous formula in discrete differential geometry. For convenience, our next step is to prove a trigonometric identity:

Proposition 5.5. *The ratio of the base* b *to the height* h *of a triangle is*

$$\frac{b}{h} = \cot\alpha + \cot\beta \tag{5.20}$$

as notated in Figure REF.

Proof. JS: see slides for now □

Using this new formula, we can rewrite the formula in Proposition **??**:

Proposition 5.6. *The gradient of area for the triangle in Figure REF in vertex* $\mathbf{p}$ *is given by*

$$\nabla_{\mathbf{p}} A = \frac{1}{2}((\mathbf{p} - \mathbf{r})\cot\alpha + (\mathbf{p} - \mathbf{q})\cot\beta). \tag{5.21}$$

Proof. JS: see slides for now □

Proposition 5.6 gives the gradient of a single triangle's area in terms of quantities easily gathered from a triangle mesh. Recall, however, that the area functional in (5.18) is summed over the entire mesh. Hence, differentiating with respect to a vertex $\mathbf{p}$ on a mesh requires a sum around its one-ring, leading to the first appearance of a *cotangent Laplacian* formula in our treatment of mesh geometry:

Proposition 5.7. *The gradient of mesh area with respect to vertex* $\mathbf{p}$ *(using notation shown in Figure* REF*) is given by*

$$\nabla_{\mathbf{p}} A = \frac{1}{2} \sum_j (\cot \alpha_j + \cot \beta_j)(\mathbf{p} - \mathbf{q}_j), \tag{5.22}$$

where j *indexes the set of vertices adjacent to* $\mathbf{p}$*, each with position* $\mathbf{q}_j$ *and opposite angles* α_j, β_j*.*

This formula provides the gradient of surface area with respect to the positions of the mesh vertices, the discrete analog of the first variation formula derived in §5.5.1. We can regard this vector $\nabla_{\mathbf{p}} A$ in Proposition 5.7 as an *integrated mean curvature vector*. Similar to our discussion of integrated Gaussian curvature above, this is an *integrated quantity* because it corresponds to the full Voronoi region V around the center vertex $\mathbf{p}$. One way to see this is by taking a finer and finer mesh, the neighbors $\mathbf{q}_j$ will approach $\mathbf{p}$ and hence the expression will vanish, while mean curvature should approach a nonzero value: Dividing by the area of V addresses this issue.

ALTERNATIVE STRUCTURES. Stepping back, we should contrast the strategies we have used to compute curvature here with those in §5.7.2. In §5.7.2, we *discretized* assorted derivatives involved in curvature formulas using divided differences, leading to notions of curvature which likely converge in the limit of refinement but are hard to describe for a coarsely-sampled mesh. In this section, we instead chose a global property of a curvature measure—e.g. that it encodes the Euler characteristic or can be obtained as the gradient of surface area—and chose to preserve it *exactly* in discretizing curvature; in doing so, however, we made our link to a specific explicit expression for curvature somewhat less clear.

There are trade-offs between these and other notions of curvature that make it murky to choose one technique universally. For instance, both of our structure-preserving methods above involve only a vertex and its one-ring of neighbors and hence can be sensitive to noise. Also, there is "no free lunch," in the sense that if we chose to preserve a different structure in our discretization that it would agree with the formulas above; for instance, CITE uses turning angles of the normal vector to characterize mean curvature and ends up with an angle-weighted formula rather than cotangents.

5.7.3 *Alternative Strategies and General Advice*

The curvature computation methods here are just a few of the many algorithms for curvature computation available. Many other considerations can inform the choice of a curvature formula, from stability to local surface noise—in which case larger patches might be used to estimate $\mathbb{II}$) to resilience to incomplete data—in which case we may need to employ machine learning techniques to impute missing surface patches.

From a practical perspective, a reasonable piece of advice is to try multiple curvature computation techniques until identifying one that is best for a given application. The reality is that there is no single choice that is best in all scenarios, as each algorithm has its own tradeoff between stability, fidelity to smooth analogs, and structure preservation. Most methods for computing curvature are relatively straightforward, and often implementations are available online.

5.8 GENERALIZATIONS OF CURVATURE

We conclude by noting that scalar curvature—the generalized term for measures resembling Gaussian curvature—can be measured on much weaker, non-differentiable structures. Definitions of curvature in the weak case are often motivated by *comparison formulas* like (5.9) and (5.10), which measure curvature as the difference between the area of a flat shape like a circle and the area of the analogous shape on a curved domain.

JS: For later: Informal paragraph on Alexandrov space

JS: For later: Curvature of a vertex on a graph

5.9 EXERCISES

5.1. JS: Computing differential of map from $\mathbb{R}^n$ into $\mathbb{R}^m$ and showing it agrees with definitions from calculus

5.2. JS: Developable surfaces, rulings

5.3. JS: Mean curvature from area of displaced surface

5.4. JS: Integration by parts for mean curvature formula

5.5. Suppose $\mathbf{t}_u, \mathbf{t}_v \in T_{\mathbf{p}}\mathcal{M}$ for some point $\mathbf{p} \in \mathcal{M}$ of a surface $\mathcal{M}$. Derive the following identity:

$$d\mathbf{n}(\mathbf{t}_u) \times \mathbf{t}_v + \mathbf{t}_u \times d\mathbf{n}(\mathbf{t}_v) = 2H(\mathbf{p})\mathbf{t}_u \times \mathbf{t}_v.$$

5.6. JS: Derive Taubin formulas

5.7. JS: Exterior angle to internal angle