6 GEODESIC DISTANCES

We started our discussion of shape analysis by taking a magnifying glass to a surface and describing the neighborhood around a single point using curvature. We now step inch by inch away, considering relationships between points, then between shapes, and so on.

To consider more than one point at a time, it is useful to develop notions of *proximity* or distance. The distance between two points on a surface, manifold, or other geometric domain not only gives a simple piece of information about their relationship but also inspires alternative definitions of curvature, bending, stretching, and other finer geometric measurements. As suggested briefly in §5.8, we will also see that distance metrics provide a weaker notion of shape than the geometry inherited from $\mathbb{R}^n$ by a submanifold. Indeed, *metric spaces*—or spaces of points equipped with distance functions—appear all over the computational and data-oriented worlds, from graph theory to abstract machine learning.

In this chapter, we focus specifically on *geodesic distances*, or shortest paths along manifolds. We consider the problem of computing distances along triangle meshes and more general simplicial complexes, highlighting the computational challenges that distinguish this problem from computing distances along graphs—a classical problem in computer science. Then in the next chapter we consider more general problems in distance computation, and in inferring distance functions in spaces where they are missing.

6.1 MOTIVATION

Consider the two points marked on paws of a cat model in Figure REF. From an *extrinsic* standpoint, that is, when considering the three-dimensional space around the cat and using the geometry of $\mathbb{R}^3$, the points are close. But, from the *intrinsic* standpoint of an ant moving along the surface the two paws are extremely far apart; the ant must crawl all the way up one leg and then down the other. This latter notion of distance, the length of the shortest path constrained to move along a surface—that is, the geodesic distance—, is our key focus in this chapter.

Geodesic distances bring many challenges to differential geometry that do not appear in the simpler flat case. They are not computable using a closed-form formula—unlike the classic formula $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$ in $\mathbb{R}^n$—, and the shortest path between two points is rarely a line. Furthermore, geodesics can be non-unique, such as the shortest path along a sphere between two antipodal points. Related to this issue, as illustrated in Figure REF, it might be that the geodesics between a point $\mathbf{x}$ and two points $\mathbf{y}$ and $\mathbf{y}'$ are extremely different even if $\mathbf{y}$ and $\mathbf{y}'$ are close.

Computer scientists are familiar with the challenges of computing a shortest path, at least along structures like graphs. And since meshes are nothing more than collections of nodes and edges, it may be tempting to apply graph algorithms to approximating geodesic distances; indeed this is common practice and often leads to somewhat reasonable estimate values. But consider the counterexample in Figure REF, in which we wish to compute the geodesic distance between two corners of a square. No matter how much we refine our simple triangulation of the square, the distance between the corners of this square remains 2 rather than the correct value $\sqrt{2}$. To make matters worse, the graph estimate does not respect the symmetry of the square: The distance between the other pair of corners is the correct value $\sqrt{2}$.

There is a wide set of applications for geodesic distances across theoretical and applied geometry. In differential geometry, geodesic distances are the basic currency for countless other computations; for example, we can characterize scalar curvature by understanding the relation-

ship between radius and volume of a small ball on a manifold. In applications, geodesic distances (unsurprisingly) appear across geometrically-structured problems like the following:

- Perhaps the most obvious applications of geodesic distances appear in geographic information systems (GIS), where geodesic distances are used to evaluate distances along a topographical map, at least before taking into account roads and paths along which a traveler is permitted to move.
- In 3D computer vision, geodesic distances are used to relate features to one another on a surface; for example, after we detect the eyes and mouth of a face, the inter-feature distances can be used to help characterize a surface for recognition or correspondence.
- In machine learning, we often think of a dataset as cutting through a lower-dimensional submanifold of $\mathbb{R}^n$. This "manifold learning" perspective attempts to capture geodesic distances between points in a dataset as a better proxy for similarity or proximity than the standard metric on $\mathbb{R}^n$.

Notice just among the example problems above that the domain for computing geodesics changes considerably, from a grid to a mesh to a point cloud in high-dimensional space.

6.2 DEFINING GEODESIC DISTANCES

6.2.1 *False Equivalences*

Simply defining geodesic distance and geodesic curves along an m-dimensional submanifold $\mathcal{M}$ of $\mathbb{R}^n$ is a tricky matter. In particular, our intuition from Euclidean space can lead us astray thanks to conflation of a few different problems that are equivalent for straight lines in $\mathbb{R}^n$ but not necessarily the same in the general case:

- GLOBAL OPTIMIZATION PROBLEM: The basic definition of geodesic distance between $\mathbf{p}, \mathbf{q} \in \mathcal{M}$ will be—unsurprisingly—the infimum of the arc length functional (3.2) in Definition 3.3 over all curves γ connecting $\mathbf{p}$ and $\mathbf{q}$ along $\mathcal{M}$.
- LOCAL OPTIMIZATION PROBLEM: A related problem would be to take a curve γ connecting $\mathbf{p}$ and $\mathbf{q}$ and to perform an analog of gradient descent on the arc length functional $L[\cdot]$ until γ converges. Intuitively, this is similar to connecting $\mathbf{p}$ and $\mathbf{q}$ with a piece of string and pulling the ends until they are tight; extensions of the first variation formula we derived in §3.5.1 suggest how we might formalize this procedure mathematically. As shown in Figure REF, however, locally optimizing arc length cannot overcome topological differences between different initializers γ and can be prone to local optima. On the other hand, we can show that the global optimum from the previous bullet will satisfy our local conditions as well, providing some useful necessary (but not sufficient) ways to describe shortest paths on surfaces.
- INITIAL VALUE PROBLEM: Yet another way we can draw lines in $\mathbb{R}^n$ is to choose an initial point $\mathbf{p} \in \mathbb{R}^n$ and a vector $\mathbf{v} \in \mathbb{R}^n$ and walk along $\mathbf{v}$ starting at $\mathbf{p}$: $\gamma(t) := \mathbf{p} + t\mathbf{v}$. This construction suggests a notion of a "locally straightest" curve starting at $\mathbf{p}$ with initial direction $\mathbf{v}$, a construction we can attempt to imitate along $\mathcal{M}$. When we build shortest paths this way, we are taking the perspective of a car driving along $\mathcal{M}$ constrained to move along the manifold but unable to turn its steering wheel.

We will see that all three notions of shortest path agree for small distances but diverge when we consider $\mathcal{M}$ on the whole rather than small neighborhoods around a single point $\mathbf{p} \in \mathcal{M}$.

6.2.2 *Derivative Formulas*

In developing theory around the notion of geodesic distance, we can begin with a simple definition:

Definition 6.1 (Geodesic distance)**.** *The* geodesic distance *between two points* $\mathbf{p}, \mathbf{q} \in \mathcal{M}$ *on a submanifold* $\mathcal{M}$ *is given by*

$$d_{\mathcal{M}}(\mathbf{p}, \mathbf{q}) := \begin{cases} \inf_{\gamma:[0,1]\to\mathcal{M}} & L[\gamma] \\ \quad \textit{subject to} & \gamma(0) = \mathbf{p} \\ & \gamma(1) = \mathbf{q} \\ & \gamma \in C^1([0,1]). \end{cases} \tag{6.1}$$

Here, the curve γ *connects* $\mathbf{p}$ *to* $\mathbf{q}$*, and we are minimizing arc length as defined in* (3.2)*. A curve* γ *realizing this infimum is known as a* global (minimizing) geodesic *curve.*

Reversing and concatenating curves shows that $d_{\mathcal{M}}$ is symmetric and satisfies the triangle inequality, so geodesic distance is a metric in the sense we will formalize in Definition 7.1. Already this definition has some potential pitfalls. We are taking an infimum rather than a minimum and have not argued that the minimizing value is realized by a curve γ; indeed Figure REF illustrates some scenarios in which this may not be the case. This motivates the derivation of a *local* formula defining geodesic curves.

Recall from exercise 3.4. (or—more simply—the derivative $\frac{d}{dt}|t| = \text{sign}(t)$) that the derivative of the norm of a vector is a somewhat difficult object to work with algebraically: The norm is non-differentiable at $\mathbf{0}$, and regardless the expression for the derivative involves a fraction. Hence, a common trick in differential geometry is to work with an alternative measurement, the *energy* of a curve γ:

Definition 6.2 (Energy of a curve)**.** *The* energy *of a parameterized curve traced by* $\gamma : [a, b] \to \mathcal{M}$ *is given by*

$$E[\gamma] := \frac{1}{2}\int_a^b \|\gamma'(t)\|_2^2\, dt.$$

Unlike Definition 3.3, we have defined energy for a parameterized curve γ rather than a curve as a set of points in $\mathbb{R}^n$. Indeed, we can see that energy of a curve is dependent on reparameterization:

Example 6.1 (Energy of a curve)**.** *Suppose* $\gamma : [0, 1] \to \mathcal{M}$ *traces a curve of length* L *with constant speed, so that* $\|\gamma'(t)\|_2 \equiv L$ *and* $E[\gamma] = \frac{1}{2}L^2$*. The reparameterization* $\tilde{\gamma}(t) := \gamma(\phi(t))$ *with* $\phi(t) := \frac{1}{2}(t + t^2)$ *traces the same curve from* $t = 0$ *to* $t = 1$*, but it starts slowly and speeds up:* $\|\tilde{\gamma}'(t)\|_2 = L\phi'(t) = \frac{L}{2}(1 + 2t)$*. Hence,*

$$E[\tilde{\gamma}] = \frac{1}{2}\int_0^1 \frac{L^2}{4}(1 + 2t)^2\, dt = \frac{13}{24}L^2 > \frac{1}{2}L^2 = E[\gamma],$$

even though γ *and* $\tilde{\gamma}$ *have the same arc length* L*.*

Keeping this example in mind, however, we can still provide a useful lemma that provides the foundation for our derivation of local arc length variations:

Proposition 6.1. *For a curve* $\gamma : [a, b] \to \mathcal{M}$*, we have*

$$L[\gamma]^2 \leq 2(b - a)E[\gamma], \tag{6.2}$$

with equality when γ *is parameterized with constant speed.*

Proof. We define an inner product of functions $f, g : [a, b] \to \mathbb{R}$ as follows:

$$\langle f, g\rangle := \int_a^b f(t)g(t)\, dt. \tag{6.3}$$

This definition of an inner product extends dot products to the infinite-dimensional space of functions on $[a, b]$, where all we have done is replace the sum $\mathbf{v} \cdot \mathbf{w} = \sum_i v^i w^i$ with an integral. As such, it is subject to the famous Cauchy–Schwarz inequality for inner products:

$$|\langle f, g \rangle| \leq \|f\| \|g\|, \tag{6.4}$$

where we have defined the norm $\|f\| := \sqrt{\langle f, f \rangle}$.

We will start on the right-hand side—the expression for E—and work our way to the left:

$$\begin{aligned}
2(b-a)E[\gamma] &= (b-a) \int_a^b \|\gamma'(t)\|_2^2 \, dt \text{ by definition} \\
&= \left[\int_a^b 1 \, dt \right] \int_a^b \|\gamma'(t)\|_2^2 \, dt \text{ since the new term equals } b-a \\
&= \langle 1, 1 \rangle \langle \|\gamma'\|_2, \|\gamma'\|_2 \rangle \text{ by (6.3)} \\
&= \|1\|^2 \left\| \|\gamma'\|_2 \right\|^2 \text{ by definition of the norm of a function} \\
&\geq \left| \langle 1, \|\gamma'\|_2 \rangle \right|^2 \text{ by (6.4)} \\
&= \left[\int_a^b \|\gamma'\|_2 \, dt \right]^2 \text{ by (6.3)} \\
&= L[\gamma]^2.
\end{aligned}$$

Now, if $\gamma(t)$ is parameterized with constant speed, we know $\|\gamma'(t)\|_2 \equiv c$ for all t, for some $c \geq 0$. Then,

$$\begin{aligned}
L[\gamma]^2 &= \left[\int_a^b c \, dt \right]^2 = (b-a)^2 c^2 \\
2(b-a)E[\gamma] &= (b-a) \int_a^b c^2 \, dt = (b-a)^2 c^2,
\end{aligned}$$

verifying the equality case for our proposition. □

Unlike arc length $L[\cdot]$, the energy $E[\cdot]$ is relatively easy to differentiate:

Proposition 6.2. *Let $\gamma_t : [a, b] \to \mathcal{M}$ be a family of curves with fixed endpoints $\mathbf{p}, \mathbf{q} \in \mathcal{M}$ on a submanifold $\mathcal{M}$, and for convenience assume γ_t is parameterized by arc length at $t = 0$. Then,*

$$\frac{d}{dt} E[\gamma_t] = -\int_a^b \left(\frac{d\gamma_t(s)}{dt} \cdot \text{proj}_{T_{\gamma_t(s)}\mathcal{M}} [\gamma_t''(s)] \right) ds. \tag{6.5}$$

Here, we do not assume s is an arc length parameter when $t \neq 0$.

Proof. Since we do not move the endpoints, we have $\gamma_t(a) \equiv \mathbf{p}$ and $\gamma_t(b) \equiv \mathbf{q}$ for all t; hence, $\frac{d}{dt}\gamma_t(a) \equiv \frac{d}{dt}\gamma_t(b) \equiv \mathbf{0}$. With this in mind, we carry out a direct computation, recalling in this case that primes indicate differentiation with respect to s (not t!):

$$\begin{aligned}
\frac{d}{dt} E[\gamma_t] &= \frac{1}{2} \frac{d}{dt} \int_a^b \|\gamma_t'(s)\|_2^2 \, ds \text{ by definition of } E \\
&= \frac{1}{2} \int_a^b \frac{d}{dt} \|\gamma_t'(s)\|_2^2 \, ds \text{ by differentiation under the integral} \\
&= \int_a^b \gamma_t'(s) \cdot \frac{d}{dt} \gamma_t'(s) \, ds \text{ by the product rule} \\
&= \left[\gamma_t'(s) \cdot \frac{d}{dt} \gamma_t(s) \right]_a^b - \int_a^b \gamma_t''(s) \cdot \frac{d\gamma_t(s)}{dt} \, ds \text{ by integration by parts} \\
&= -\int_a^b \gamma_t''(s) \cdot \frac{d\gamma_t(s)}{dt} \, ds \text{ since the endpoints are stationary.}
\end{aligned}$$

By definition, the second term in the dot product is in the tangent space $T_{\gamma_t(s)}\mathcal{M}$, and hence any component of $\gamma_t''(s)$ in the normal direction is discarded in the dot product. Hence we can safely project $\gamma_t''(s)$ onto $T_{\gamma_t(s)}\mathcal{M}$ without affecting the integrand, providing the desired formula. □

This lemma is illustrated in Figure REF. Roughly, rather than connecting $\mathbf{p}$ and $\mathbf{q}$ with one curve, we connect them with a whole set of curves $\gamma_t(s)$, where t indexes which curve we are considering and s is a general parameter (not necessarily arc length); we assume all curves connect $\mathbf{p}$ to $\mathbf{q}$, that is, $\gamma_t(a) = \mathbf{p}$ and $\gamma_t(b) = \mathbf{q}$. Our computation is differentiating energy $E[\cdot]$ as γ_t evolves with t, essentially a one-dimensional plot shown in Figure REF. If γ_t is *not* a geodesic at $t = 0$, there will exist a way to flow it that will reduce E, and otherwise we know we are at a critical point in our plot.

As a corollary, under suitable differentiability conditions (which no doubt would be carefully stated in a less informal treatment of differential geometry) we have the following condition:

Proposition 6.3. *If a curve $\gamma : [a, b] \to \mathcal{M}$ parameterized by arc length is a global geodesic, then*

$$\text{proj}_{T_{\gamma(s)}\mathcal{M}}[\gamma''(s)] \equiv 0 \tag{6.6}$$

for $s \in (a, b)$.

Proof. This follows from Proposition 6.2; if this is not the case then there is a direction we can flow γ to decrease its energy E and hence its arc length by Proposition 6.1. In more detail, if the projection in (6.6) is nonzero, flowing γ in the direction of this projection (scaled to vanish at the endpoints) yields curves γ_t with $E[\gamma_t] < E[\gamma]$ for small $t > 0$. Since $\gamma = \gamma_0$ has constant speed, the equality case of Proposition 6.1 then shows $L[\gamma_t]^2 \leq 2(b-a)E[\gamma_t] < 2(b-a)E[\gamma] = L[\gamma]^2$, contradicting our assumption that γ is a shortest path. □

Curves that satisfy this derivative condition are known as *local geodesics*; they are critical points of arc length among nearby curves along $\mathcal{M}$ with the same endpoints, and sufficiently short pieces of a local geodesic are true shortest paths.

This proposition, which extends easily to the more general case of Riemannian manifolds, gives a clean intuition for what it means for a curve to be a geodesic locally. If we think of γ as tracing out the path of a car driving along $\mathcal{M}$, the second derivative γ'' corresponds to the force experienced by a passenger in the car. Similar to our discussion of geodesic curvature in §5.7.2, this force can be decomposed into two components: the force needed to glue the car to $\mathcal{M}$ rather than flying off the manifold into ambient space $\mathbb{R}^n$ and the force induced by the driver as he or she turns the steering wheel. The latter force—corresponding to tangential motion along $\mathcal{M}$—is zero if the car drives along a geodesic.

6.2.3 *Exponential Map*

We now flip our perspective on (6.6) on its head. We motivated this formula by supposing we are given a geodesic curve γ and try to perturb it. But, we can also think of (6.6) as a *differential equation*: It characterizes (locally) what it means for γ to be a geodesic curve in terms of its derivatives; in some sense we have taken the derivative of arc length (really energy—but we showed these lead to the same minima!) and set it equal to zero.

Hence, we could attempt to *solve* (6.6) for $\gamma(s)$ in one of two ways:

- BOUNDARY VALUE PROBLEM: Given $\gamma(a)$ and $\gamma(b)$, find $\gamma(s)$ connecting these two end points satisfying (6.6).
- INITIAL VALUE PROBLEM: Given $\gamma(a)$ and $\gamma'(a)$, extend γ to a function $\gamma(s)$ for $s \in [a, a + \varepsilon)$ obeying the given initial conditions. Under fairly innocuous regularity conditions, it is usually the case in the absence of a boundary that $\gamma(s)$ exists for $s \in [a, \infty)$.

These two perspectives distinguish the bulleted approaches given in §6.2.1. Note that these conditions do not assume γ is parameterized by arc length, although thanks to (6.6) they do require γ to have constant speed.

The initial value problem for a geodesic is easier to analyze; existence and uniqueness results for ordinary differential equations (ODEs) apply, showing when we can integrate the relationship (6.6) to generate a curve along $\mathcal{M}$. In particular, given any initial point $\mathbf{p} \in \mathcal{M}$ and tangent vector $\mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$, we can solve the initial value problem above to find a point $\gamma_{\mathbf{v}}(1)$ reached by traveling a distance $\|\mathbf{v}\|_2$ from $\mathbf{p}$ along $\mathcal{M}$ with initial tangent $\mathbf{v}$. This leads to the definition of the *exponential map*:

Definition 6.3 (Exponential map)**.** *Given a submanifold $\mathcal{M}$ and point $\mathbf{p} \in \mathcal{M}$, the* exponential map $\exp_{\mathbf{p}}(\cdot) : T_{\mathbf{p}}\mathcal{M} \to \mathcal{M}$ *takes tangent vectors $\mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$ to the point $\gamma_{\mathbf{v}}(1)$, where $\gamma_{\mathbf{v}}$ is the local geodesic satisfying $\gamma_{\mathbf{v}}(0) = \mathbf{p}$ and $\gamma'_{\mathbf{v}}(0) = \mathbf{v}$.*

By definition of the exponential map, a geodesic along $\mathcal{M}$ with tangent $\mathbf{v}$ is traced out by the curve $\gamma_{\mathbf{v}}(t) := \exp_{\mathbf{p}}(t\mathbf{v})$.

The definition above is somewhat sloppy in that geodesics are only guaranteed to exist for short time; in reality, the exponential map should only act on a neighborhood of $\mathbf{0}$ in the tangent plane $T_{\mathbf{p}}\mathcal{M}$. The celebrated Hopf–Rinow Theorem in Riemannian geometry, however, shows that $\exp_{\mathbf{p}}$ is well defined on all of $T_{\mathbf{p}}\mathcal{M}$ for every $\mathbf{p} \in \mathcal{M}$—a property known as *geodesic completeness*—exactly when $\mathcal{M}$ has no "missing" points, in the sense that a traveler cannot fall off an open edge of $\mathcal{M}$ or into a puncture after walking a finite distance. For example, a sphere and the entire plane are complete, while a plane with one point removed is not. In the complete case, a length-minimizing geodesic satisfying the local conditions (6.6) exists between every pair of points on $\mathcal{M}$. Figure REF shows examples when $\exp_{\mathbf{p}}$ is and is not defined globally.

Intuition for the exponential map is shown in Figure REF. We can think of the map as a local parameterization of $\mathcal{M}$ around $\mathbf{p}$, in that it takes points on a flat space $T_{\mathbf{p}}\mathcal{M}$ to the curved manifold $\mathcal{M}$. This particular parameterization is chosen so that around $\mathbf{0} \in T_{\mathbf{p}}\mathcal{M} \mapsto \mathbf{p} \in \mathcal{M}$, the map $\mathbf{v} \mapsto \exp_{\mathbf{p}}(\mathbf{v})$ has relatively low distortion. From a Taylor series perspective, $\exp_{\mathbf{p}}$ is an isometry to first order at $\mathbf{0}$, that is, its differential at $\mathbf{0}$ preserves lengths and angles; this is only an infinitesimal property, however, since a globally isometric map from a flat space $T_{\mathbf{p}}\mathcal{M}$ to a potentially curved submanifold $\mathcal{M}$ is only possible when $\mathcal{M}$ is flat. Generically $\exp_{\mathbf{p}}$ is differentiable, providing a reasonable coordinate system for the manifold known as *geodesic normal coordinates*.

Even when $\exp_{\mathbf{p}}$ is defined on all of $T_{\mathbf{p}}\mathcal{M}$, it is not necessarily one-to-one. This is a fairly reasonable property to expect: After all, mapping the infinitely-wide tangent plane of a sphere $\mathcal{S}^2$ onto the sphere itself certainly leads to overlap. In some neighborhood of $\mathbf{0}$, $\exp_{\mathbf{p}}(\cdot)$ is a diffeomorphism: The size of this neighborhood, known as the *injectivity radius*, corresponds to the distance one can travel along a surface before local and global geodesic curves are not the same, i.e., two geodesics starting at $\mathbf{p}$ meet.

Taking this reasoning one step farther, we can define the *cut locus* of $\mathbf{p} \in \mathcal{M}$ as follows:

Definition 6.4 (Cut locus)**.** *The* cut locus *of $\mathbf{p} \in \mathcal{M}$ is the set of points $\exp_{\mathbf{p}}(\mathbf{v})$ with the property that $\exp_{\mathbf{p}}(t\mathbf{v})$ is a global geodesic for $t \in [0, 1]$ but fails to be the globally shortest path for $t \in [0, 1 + \varepsilon)$ for any $\varepsilon > 0$.*

The cut locus of a point on a surface is shown in Figure REF. The function that assigns to every point on $\mathcal{M}$ its geodesic distance to $\mathbf{p}$ is smooth away from the cut locus and $\mathbf{p}$ itself.

JS: To do: Forward reference to manifold optimization

6.2.4 *Eikonal Equation*

Description of geodesic distances and curves is a critical topic in differential geometry that easily could fill a textbook's worth of material. We will largely defer discussion of other properties of

geodesics until they are needed. For instance, we could ask what vector fields (known as *Jacobi fields*) can be used to slide a geodesic curve along a manifold while preserving the property that it is a geodesic, and we could examine the relationships between geodesics and curvature.

One additional more advanced topic is worth mentioning, the *eikonal equation*. To do so, we will take a short detour to define the gradient of a function on a submanifold $\mathcal{M}$. Recall Definition 5.1, which provides a way to define the *differential* of a function; Proposition 5.1 checked that the differential is indeed a linear map. As a special case of that construction, consider a function on $\mathcal{M}$, i.e., a map $f : \mathcal{M} \to \mathbb{R}$; we can think of $\mathbb{R}$ as a one-dimensional manifold ($\mathbb{R} = \mathcal{N}$ in the notation of our earlier discussion). In this case, $df_{\mathbf{p}}$ is a linear map from $T_{\mathbf{p}}\mathcal{M}$ into $T_{f(\mathbf{p})}\mathbb{R} \cong \mathbb{R}$, that is, it takes tangent vectors to real numbers. A direct consequence of linearity made explicit in exercise 6.1. is that $df_{\mathbf{p}}$ encodes a dot product: $df_{\mathbf{p}}(\mathbf{v}) \equiv \mathbf{v} \cdot \mathbf{w}_{\mathbf{p}}$ for some vector $\mathbf{w}_{\mathbf{p}} \in T_{\mathbf{p}}\mathcal{M}$. We denote this vector $\mathbf{w}_{\mathbf{p}}$ as the *gradient* of f:

Definition 6.5 (Gradient). *The* gradient *of a function $f : \mathcal{M} \to \mathbb{R}$ at a point $\mathbf{p} \in \mathcal{M}$ is the vector $\nabla f(\mathbf{p}) \in T_{\mathbf{p}}\mathcal{M}$ so that $df_{\mathbf{p}}(\mathbf{v}) = \mathbf{v} \cdot \nabla f(\mathbf{p})$ for all $\mathbf{v} \in T_{\mathbf{p}}\mathcal{M}$.*

The collection of gradient vectors ∇f at all $\mathbf{p}$ gives a vector field tangent to $\mathcal{M}$, as illustrated in Figure REF.

Given a point $\mathbf{p} \in \mathcal{M}$, we can construct a function $d_{\mathbf{p}} : \mathcal{M} \to \mathbb{R}_+$ such that $d_{\mathbf{p}}(\mathbf{q})$ is the geodesic distance from $\mathbf{p}$ to $\mathbf{q}$ along $\mathcal{M}$. The eikonal equation is a nonlinear relationship about the gradient of $d_{\mathbf{p}}$ and other "distance-like functions." In particular, a function $u : \mathcal{M} \to \mathbb{R}$ satisfies the eikonal equation if

$$\|\nabla u(\mathbf{q})\|_2 = 1 \; \forall \mathbf{q} \in \mathcal{M}. \tag{6.7}$$

As a sanity check, it is easy to check that this relationship holds for the function $u(\mathbf{x}) := \|\mathbf{x} - \mathbf{x}_0\|_2$ for $\mathbf{x} \in \mathbb{R}^n$. The eikonal equation is simply a reflection of the fact that distance functions grow with constant speed radially away from the center point. It is a relationship that holds *almost everywhere*: The gradient ∇u for a geodesic distance function does not exist on the cut locus or at the source point itself.

Yet another way to understand the problem of computing geodesic distances (but not paths)—in this case from a single source point to *all* target points on $\mathcal{M}$—is that we are attempting to "solve" the eikonal equation for u. We will see that this is the starting point of the fast marching algorithm. While we will tolerate some imprecision here, describing the relationship between (6.7) and the single-source geodesic problem accurately is challenging due to boundary conditions and differentiability; Figure REF shows some functions in one dimension that satisfy the eikonal equation almost everywhere but likely would be unexpected outputs of geodesic distance code. As an aside, the term *eikonal* comes from the Greek word for "image" due to a link between a generalization of (6.7) and optics.

6.3 INITIAL VALUE PROBLEM ON MESHES

Having laid out the basic formulas for geodesic distance in theory, we can now consider the problem of computing geodesic distances between points on manifold simplicial complexes like triangle and tetrahedral meshes. Whereas our treatment of curvature in the previous chapter applied only to (two-dimensional) surfaces, here most of our algorithms will extend without serious adjustment to higher-dimensional structures.

Our discussion of how to characterize geodesic distance worked from the easiest *definition* of geodesic distance—Definition 6.1—to differential formulas that are easier to manipulate in practice. Now that we are deriving algorithms, we will proceed in approximately reverse order, since the problem of finding a globally-optimal geodesic curve is more difficult computationally than finding a locally-shortest path.

To start, we can consider the problem of evaluating the exponential map, or, equivalently, solving a version of the ODE (6.6). This straightforward construction is studied in detail by Polthier

and Schmies [50], who carefully lay out the relationships and differences between *discrete straightest geodesics* and their smooth counterparts. We will focus on the case of discrete surfaces although this construction can be extended to higher dimensions.

Rather than discretizing the ODE directly, a simpler approach is to return to its intuitive definition. Recall that the local geodesic curve γ is obtained by driving with a straight steering wheel, that is, without inducing any bend in the tangent plane. In the interior of a triangle, this implies an intuitive and somewhat obvious corollary: The discrete geodesic should be a straight line. As illustrated in Figure REF, this implies that discrete geodesics on triangle meshes can be represented as collections of line segments, which turn when the line segment reaches an edge or vertex of a triangle.

Given the reasoning above, our remaining task is to decide what should happen when a geodesic crosses from one triangle into another. Taking the philosophy of the ODE (6.6), we will derive a condition based on what it means for the piecewise curve to be a local minimizer of distance.

Consider a piecewise flat geodesic crossing from one triangle into another along an edge, as shown in Figure REF. We can hinge the two triangles about their shared edge—which forms the dihedral angle between the triangles—until the two triangles together are in the same plane. Locally, we can hinge the two triangles without changing their edge lengths or affecting the lengths of the two line segments on the two sides. After unfolding, if the two-segment curve is a geodesic then the unfolded curve must be a straight line.

We might wish to derive a condition for being a geodesic that does not involve this unfolding motion. To do so, we define the left and right angles θ_ℓ and θ_r to be the angles shown in Figure REF; as we unfold these angles are preserved, and they are 180° when the geodesic is flattened to a plane. Hence, a poly-line geodesic on a triangle mesh must have $\theta_\ell = \theta_r = 180^\circ$ when crossing an edge.

There is one remaining rare case to check, shown in Figure REF, in which the curve crosses a vertex of the mesh. Treating the one-ring of the vertex like a folded sheet of paper, we can still unfold to the plane after cutting an edge, shown in Figure REF. Once again our two line segments get unfolded along with the curve, meeting at the flattened vertex. In this case, the left and right angles sum to the total angle around the vertex, which generally is not 360°; hence, they cannot both equal 180° unless the vertex is flat. The best we can do is to ask that the two angles be equal, so that the curve splits the angle around the vertex evenly. This leads us to the following definition:

Definition 6.6 (Straightest geodesic on a triangle mesh)**.** *A piecewise-linear curve on a triangle mesh is a* straightest geodesic *if at every point we have* $\theta_\ell = \theta_r$.

As shown in Figure REF, considering the $\theta_\ell \neq \theta_r$ case also gives a reasonable discrete definition of geodesic curvature for a curve on a surface that extends the planar definition of discrete curvature for a curve from §3.8.3.

Surprisingly, straightest geodesics and shortest geodesics are *not* the same, even locally! Recall from (5.17) that we can define integrated Gaussian curvature at a vertex as $K := 2\pi - \sum_j \theta_j$, where here the θ_j's are the interior angles of the triangles at the vertex. Figure REF illustrates two curious properties of locally straightest geodesics on meshes:

- When $K > 0$, that is, when the surface is locally spherical at a vertex, a straightest geodesic is *never* locally shortest.
- When $K < 0$, that is, when the surface is locally hyperbolic at a vertex, there are multiple shortest geodesics through the vertex but only a single straightest geodesic.

These properties illustrate the perils in working directly on a mesh as a discrete object. Local existence and uniqueness results that help us understand (6.6) no longer apply: While a straightest geodesic can always be extended uniquely through a vertex, a *shortest* geodesic arriving at a vertex

may have no extension or many extensions that remain locally shortest, even in small neighborhoods. In a sense, dealing with these issues is a matter of convention; straightest geodesics and shortest-path geodesics—which locally agree on smooth surfaces—are simply not the same in this regime.

6.4 FAST MARCHING

Algorithms for finding locally-straightest geodesics on meshes are interesting theoretically because they highlight the difference between smooth and discrete geodesics. But from a practical perspective, these methods are of limited value: A more common task is to find the shortest path between two points on a mesh rather than to "speculatively" compute a geodesic emanating from a single vertex.

Thanks to the drawbacks illustrated in Figure REF, geodesics are poorly approximated by shortest paths along mesh edges. At the same time, computing the *exact* geodesic distance between points on a mesh is challenging; we will outline a famous (and famously complex!) algorithm for doing so in §6.5. Before we get to that point, however, we will introduce a well-known approximation using the fast marching algorithm [56] that is straightforward to implement and often works well in practice, originally introduced in [36].

In addition to being a standard method for computing geodesic distances on simplicial complexes, fast marching is also exemplary of an algorithm that takes the discretized rather than discrete approach to computing geodesic distances. In particular, fast marching can be considered a numerical algorithm for approximating solutions to the eikonal equation (6.7); as the triangle mesh is refined, these approximations (often) converge to the true geodesic distance, but before that point they have fewer guarantees. In particular, geodesic distances approximated using fast marching may not satisfy the triangle inequality or other axioms used to define true shortest paths.

6.4.1 *Dijkstra's Algorithm on Graphs*

Although we have established that shortest paths on graphs are bad proxies for geodesics, a particular shortest path algorithm known as *Dijkstra's algorithm* can be extended to triangle meshes to yield the fast marching algorithm.

Suppose we have a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_+ \backslash \{0\}$, and that we have marked a source vertex $v_0 \in V$. Dijkstra's algorithm computes a function $d : V \to \mathbb{R}_+$ giving the length of the shortest path along G from v_0 to each vertex in V, with $d(v_0) = 0$.

Dijkstra's algorithm maintains a set of vertices $S \subseteq V$ for which the shortest path length is known. We start by initializing the distance to v_0 as zero, the distance to the remaining vertices as ∞, and the set S as empty:

$$\begin{aligned} d(v_0) &\leftarrow 0 \\ d(v) &\leftarrow \infty \; \forall v \in V \backslash \{v_0\} \\ S &\leftarrow \{\}. \end{aligned}$$

Then, each step of Dijkstra's algorithm chooses the vertex with the closest distance to v_0 that has not yet been visited and updates the distances to its neighbors using the triangle inequality:

$$\begin{aligned} v &\leftarrow \arg\min_{v \in V \backslash S} d(v) \\ S &\leftarrow S \cup \{v\} \\ d(u) &\leftarrow \min\{d(u), d(v) + w(e)\} \; \forall e = (u, v) \in E. \end{aligned} \tag{6.8}$$

In words, the final line looks among all the edges $e = (u, v)$ leaving v and makes sure that the length of the shortest path to u is updated in case it is shorter to travel to u through v. In each

iteration of Dijkstra's algorithm, S increases its size by 1; the algorithm terminates when $S = V$. The runtime for Dijkstra's algorithm is $O(|E| + |V| \log |V|)$ if a priority queue is used to find the next v, effectively storing the set $V \backslash S$.

Proving correctness of Dijkstra's algorithm is straightforward. Since $w > 0$, after a vertex v is added to S we must have $d(v)$ constant for the remainder of the algorithm. Suppose Dijkstra's algorithm fails. Take v to be the *first* vertex added to S for which $d(v)$ is not the length of the shortest path from v_0 to v; $v \neq v_0$ since $d(v_0)$ was set to zero. Let $v_0 \to v_1 \to \cdots \to v_k \to v$ be a shortest path from v_0 to v. Immediately before v is added to S, we have $v_0 \in S$ but $v \notin S$; hence, somewhere along the shortest path there is an edge $v_j \to v_{j+1}$ with $v_\ell \in S \; \forall \ell \leq j$ and $v_{j+1} \notin S$. The edge $v_j \to v_{j+1}$ was updated according to (6.8) when v_j was added to S; since any subpath of a shortest path is itself a shortest path, we must have $d(v_{j+1})$ equal to the length of the shortest path from v_0 to v_{j+1}. If $v_{j+1} = v$, we have a contradiction since we assumed $d(v)$ was not the length of the shortest path; otherwise we have a contradiction since $d(v_{j+1}) < d(v)$ and hence v_{j+1} would be chosen instead of v for an update in the v iteration.

Figure REF illustrates an intuition for Dijkstra's algorithm that will carry over to our treatment of geodesic distances. In each iteration, the set S contains vertices for which the shortest path length from v_0 is known; in each iteration S expands by one vertex on its "fringe," the interface between S and $V \backslash S$. A typical way to describe this behavior is that Dijkstra's algorithm maintains an *advancing front* of vertices whose shortest path is known, roughly corresponding to concentric circles outward from v_0.

We have not explicitly given a method for computing the shortest path—rather than the shortest path length—from v_0 to another vertex v, but doing so is fairly straightforward. One option is to keep track of the predecessor of each vertex in the chain that led to $d(v)$ while running Dijkstra's algorithm. Alternatively, as shown in Figure REF, we can trace backward from v to v_0 given the distance vector $d(\cdot)$.

6.4.2 *Planar Front Approximation*

Fast marching for geodesics, introduced in [36], is extremely similar to Dijkstra's algorithm, simply with a few adjustments to account for paths that cut through the interiors of simplices. We still keep track of an advancing front of vertices on which we know the distance to a source point; the main difference is the update formula used when adding a new vertex to S.

Figure REF illustrates the simplest approximation used to compute geodesics in fast marching. In $\mathbb{R}^2$, the level sets of the distance function d from a source point are concentric circles; far away from the source point, these level sets are reasonably well-approximated by straight lines. On a triangle mesh, the same is true: Level sets of the (true) distance function are composed of circular arcs, which look less and less curved the farther we move from the source point. We will approximate geodesics by assuming that within a single triangle the distance function is well-approximated by one whose level sets are straight lines. This *planar front approximation* is effective far away from the source point, although locally it can make some mistakes.

Following [36], we can work out explicit formulas for a local planar front approximation. Suppose for simplicity that $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^2$ are on the plane with distances d_1, d_2 to some far-away center point; our goal is to find the distance d_3 to a third point on the plane $\mathbf{x}_3$. Under the planar front approximation, we approximate the distance function using the linear formula

$$d(\mathbf{x}) = \mathbf{n}^\top \mathbf{x} + c. \tag{6.9}$$

By the eikonal equation (6.7), we should assume $\|\mathbf{n}\|_2 = 1$.

Without loss of generality, suppose we shift $\mathbf{x}_1, \mathbf{x}_2$ so that $\mathbf{x}_3 = \mathbf{0}$; this can be accomplished by mapping $\mathbf{x}_1 \mapsto \mathbf{x}_1 - \mathbf{x}_3$ and $\mathbf{x}_2 \mapsto \mathbf{x}_2 - \mathbf{x}_3$. Then, our problem is as follows:

$$\begin{aligned} \text{Given } \mathbf{x}_1, \mathbf{x}_2, d_1, d_2 \text{ satisfying} \quad & d_1 = \mathbf{n}^\top \mathbf{x}_1 + c \\ & d_2 = \mathbf{n}^\top \mathbf{x}_2 + c \\ \text{and assuming} \quad & \|\mathbf{n}\|_2 = 1, \\ \text{find} \quad & d_3 = \mathbf{n}^\top \mathbf{x}_3 + c = \mathbf{n}^\top \mathbf{0} + c = c. \end{aligned}$$

We can stack our variables into matrices as follows

$$\underbrace{\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}}_{\mathbf{d}} = \underbrace{\begin{pmatrix} - & \mathbf{x}_1^\top & - \\ - & \mathbf{x}_2^\top & - \end{pmatrix}}_{X^\top} \mathbf{n} + c \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\mathbf{1}} \implies \mathbf{d} = X^\top \mathbf{n} + c\mathbf{1} \implies \mathbf{n} = X^{-\top}(\mathbf{d} - c\mathbf{1}).$$

Recalling $\|\mathbf{n}\|_2 = 1$, we find:

$$\begin{aligned} 1 &= \mathbf{n}^\top \mathbf{n} \\ &= (\mathbf{d} - c\mathbf{1})^\top X^{-1} X^{-\top} (\mathbf{d} - c\mathbf{1}) \\ &= [\mathbf{1}^\top (X^\top X)^{-1} \mathbf{1}] c^2 + [-2\mathbf{1}^\top (X^\top X)^{-1} \mathbf{d}] c + [\mathbf{d}^\top (X^\top X)^{-1} \mathbf{d}]. \end{aligned}$$

This expression is quadratic in c, and hence we can solve it for $c = d_3$.

The end result is *two* roots that could be used to find c. Geometrically, as illustrated in Figure REF, this issue can be explained by the fact that the values d_1, d_2 determine the component of $\mathbf{n}$ along the edge from $\mathbf{x}_1$ to $\mathbf{x}_2$, while the condition $\|\mathbf{n}\|_2 = 1$ determines the component of $\mathbf{n}$ perpendicular to this edge only up to sign. Given the advancing front interpretation of fast marching, we choose the larger root, which corresponds to the front crossing the edge before reaching $\mathbf{x}_3$, and verify that it satisfies $c = d_3 \geq \max(d_1, d_2)$.

Figure REF illustrates one additional issue, which occurs because the planar front approximation is inexact. In this scenario, the planar front derived above indicates that the front should not come from the current triangle under consideration. This can occur when the triangle $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is obtuse. As explored in exercise 6.2., the front arrives at $\mathbf{x}_3$ from within the triangle exactly when both entries of $(X^\top X)^{-1} X^\top \mathbf{n} = X^{-1}\mathbf{n}$ are negative, so we can detect this situation by checking the signs of these two values. When this test fails, we can resort to one of two possible methods:

- Resort to an edge-based update similar to Dijkstra's algorithm on graphs:

$$d_3 \leftarrow \min\{d_3, d_1 + \|\mathbf{x}_3 - \mathbf{x}_1\|_2, d_2 + \|\mathbf{x}_3 - \mathbf{x}_2\|_2\}.$$

- As explored in [36] and illustrated in Figure REF, subdivide the obtuse triangle into two acute triangles and do the calculation above twice.

Putting it all together, the fast marching algorithm maintains a set $S \subseteq V$ of the vertices of a mesh; it initializes $d(v_0) = 0$ and $d(v) = \infty$ for all $v \in V \backslash \{v_0\}$. In each iteration, it chooses the vertex $v \in V \backslash S$ with the smallest value $d(v)$ and adds v to S. Then, for all neighboring triangles (u, v, w) containing v, we update $d(u)$ and $d(w)$ according to the discussion above and falling back on graph edge updates. This technique easily extends to higher-dimensional simplices. Once we have the per-vertex function d, we can trace backward along the per-triangle gradient $\mathbf{n}$ to recover an approximate geodesic curve.

The planar front approximation implies that we did not compute true geodesic distances, but the approximation converges as the underlying mesh is refined REF. Many improvements to this algorithm have been proposed over the years, such as approximating the planar front using concentric circles rather than lines CITE or adapting the technique to grid structures CITE.

6.5 EXACT GEODESICS: MMP ALGORITHM

JS: will waffle on this one in 6.838; it's hard to describe precisely. Should eventually cover assorted attempts to accelerate it as well.

6.6 OTHER METHODS FOR SINGLE-SOURCE

Numerous other techniques have been proposed for single-source-all-targets geodesic computation, similar to MMP and fast marching. For instance, [20] uses a property of the heat diffusion operator to approximate geodesic distances; we will explain this method in detail in § REF after showing how to approximate heat diffusion along a meshed domain. [60] uses methods from optimal transport to approximate geodesics, covered in § REF.

6.7 STABILIZED DISTANCE FUNCTIONS

6.7.1 *Fuzzy Geodesics*

JS: not covered carefully in 6.838, skip for now

6.7.2 *Walking on Broken Meshes*

JS: not covered carefully in 6.838, skip for now

6.8 ALL-PAIRS GEODESIC DISTANCES

A different use case for geodesic distances appears when we have a fixed geometric object, e.g. the topography of a land mass, and expect to need to compute many shortest paths between arbitrary pairs of points. In this case, it might make sense to precompute a data structure that can accelerate general distance queries; the cost of constructing the data structure is amortized over the number of queries.

6.8.1 *Embedding Techniques*

JS: Not covered in 6.838 (for now): sample a few pairs and extrapolate using low-rank methods (Kimmel)

6.8.2 *Pathwise Subsampling*

JS: Not covered in 6.838: Use geodesics along a simplified mesh to guess geodesics on the dense one

6.9 HIGHER DIMENSIONS

6.9.1 *Challenges with k-NN Distances*

JS: Not covered in 6.838: Random graph negative result in [1]

6.9.2 *Geodesics in Point Clouds*

JS: Not covered in 6.838: Justin isn't sure what people do for this, if anything

6.10 EXPONENTIAL MAPS

6.10.1 *Solving the Geodesic ODE*

JS: Not covered in 6.838: Geodesics in the space of shells, Benamou–Brenier, medical imaging shape manifold methods

6.10.2 *Manifold Gradient Descent*

JS: Using closed-form geodesics for optimization

6.11 EXERCISES

6.1. Suppose $f : \mathcal{M} \to \mathbb{R}$ is a differentiable map from a submanifold $\mathcal{M}$ into the real numbers. For a given $\mathbf{p} \in \mathcal{M}$, show that $df_{\mathbf{p}}$ can be written $df_{\mathbf{p}}(\mathbf{v}) = \mathbf{v} \cdot \mathbf{w}_{\mathbf{p}}$ for a unique $\mathbf{w}_{\mathbf{p}} \in T_{\mathbf{p}}\mathcal{M}$. Argue that this vector agrees with the classical definition of a gradient ∇f for $f : \mathbb{R}^n \to \mathbb{R}$.

6.2. **JS: failure of planar front**